A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology

Ping Zhou, Bo Zhao, Jun Yang, Youtao Zhang‡
Electrical and Computer Engineering Department
‡ Department of Computer Science
University of Pittsburgh, Pittsburgh, PA 15261
{piz7, boz6, juy9}@pitt.edu, ‡zhangyt@cs.pitt.edu

ABSTRACT

Using nonvolatile memories in memory hierarchy has been investigated to reduce its energy consumption because nonvolatile memories consume zero leakage power in memory cells. One of the difficulties is, however, that the endurance of most nonvolatile memory technologies is much shorter than the conventional SRAM and DRAM technology. This has limited its usage to only the low levels of a memory hierarchy, e.g., disks, that is far from the CPU.

In this paper, we study the use of a new type of nonvolatile memories – the Phase Change Memory (PCM) as the main memory for a 3D stacked chip. The main challenges we face are the limited PCM endurance, longer access latencies, and higher dynamic power compared to the conventional DRAM technology. We propose techniques to extend the endurance of the PCM to an average of 13 (for MLC PCM cell) to 22 (for SLC PCM) years. We also study the design choices of implementing PCM to achieve the best tradeoff between energy and performance. Our design reduced the total energy of an already low-power DRAM main memory of the same capacity by 65%, and energy-delay^2 product by 60%. These results indicate that it is feasible to use PCM technology in place of DRAM in the main memory for better energy efficiency.

Categories and Subject Descriptors

B.3.0 [**Hardware**]: Memory StructuresGeneral

General Terms

Design

Keywords

Phase Change Memory, Endurance, Low Power

1. INTRODUCTION

The energy consumption of main memory is becoming a dominant portion of the total system energy due to the trend in increasing the memory capacity. This is mainly driven by the growing memory requirement of new applications, and the increasing number of processing cores in a single chip. Recent studies have shown that memory energy conservation should focus on leakage energy reduction since leakage grows with the memory capacity, and the main memory can dissipate as much leakage energy as dynamic energy [25]. Fortunately, several new nonvolatile memory technologies have emerged as potential solutions due to their exceptionally low leakage. Examples include NAND flash, Phase-Change Memory (PCM), and Spin-Transfer Torque RAM (STT-RAM). In addition, these nonvolatile memories are resilient to single event upsets. Therefore, they are more reliable than conventional DRAMs.

Among the three promising nonvolatile memories, the NAND flash has very limited number of write/erase cycles: 10^5 rewrites [31] as opposed to 10^{16} for DRAM. NAND flash also requires a block to be erased before writing into that block, which introduces considerably extra delay and energy. Moreover, NAND flash is not byte-addressable. Therefore, NAND flash has been proposed as a disk cache [3, 11] or a replacement for disks [26] where writes are relatively infrequent, and happen mostly in blocks. PCM and STT-RAM are probably the two most promising candidates for next-generation memory technology for both standalone and embedded applications. Both have been backed by key industry manufacturers such as Intel, STMicroelectronics, Samsung, IBM and TDK [8, 28]. Both memories are byte-addressable. STT-RAM is faster than PCM, and has nearly the same endurance as DRAM (10^{15} [24] vs. 10^{16} rewrites). PCM, on the other hand, is denser than STT-RAM. The cell area for DRAM, PCM and STT-RAM are $6F^2$ [25], $5\sim8F^2$ [14], and $37\sim40F^2$ [7] respectively, where F is the feature size. Moreover, phase change material has excellent scalability within current CMOS fabrication methodology [5, 13, 14, 22, 23], which is a key advantage because DRAM scaling will be clamped by the limitation in cell-bitline capacitance ratio.

In this paper, we propose to use PCM technology in the *main memory* of a 3D die stacked chip multiprocessor. 3D integration provides an opportunity to stack cache or main memory on top of a die that contains multiple processor cores [2, 12, 18, 25]. This can reduce the memory access latency, and help improve the memory bandwidth provision in CMPs. However, a 3D integrated chip is subject to tight power and thermal constraint. Hence, using PCM with near zero cell leakage can greatly reduce the energy consumption in main memory. However, integrating PCM as the main memory for a 3D chip faces the following two key challenges: (1) Although PCM has much higher endurance than

ISCA'09, June 20–24, 2009, Austin, Texas, USA.

flash memory, it is is still quite limited, compared to the conventional DRAM. PCM can sustain $10^8 \sim 10^9$ [9, 27, 31] rewrites per cell, while for a DRAM cell this number is virtually unlimited. (2) PCM cells require high write energy due to the nature of the phase change material. This will increase the dynamic energy consumption of the memory, which may surpass the leakage energy we saved. To tackle these problems, we propose a technique to remove significant amount of redundant writes from the memory. This not only removes substantial amount of write energy, but also plays a leading role in extending the lifetime of the PCM to 13∼22 years. We compared a 4GB PCM with a low-leakage commodity DRAM main memory. Experimental results shows that a PCM with low operating power devices for its peripheral circuit performs best in terms of both energy and energy-delay product. Compared to a low-leakage commodity DRAM based main memory, we reduced its total energy by 37%, and improved the energy-delay product by 34% by using a PCM based main memory.

The remainder of the paper is organized as follows. Section 2 introduces the basics of PCM. Section 3 explains our techniques to prolong the lifetime of a PCM. Section 4 described how we modeled DRAM and PCM. Section 5 discusses our experimental results. Finally, Section 6 concludes the paper.

2. BACKGROUND

A conventional DRAM cell uses a capacitor to store a bit information. Analogously, a PCM cell uses a special material, called phase change material, to remember a bit. The phase change material can exist in two different but stable structural states: amorphous and crystalline. Each of which has drastically different resistivity, which can be used to represent logic '0' or '1'. The phase change material is one type of alloy, such as $Ge_2Sb_2Te_5$ (GST), that can switch between its two states with the application of heat. When heated above the crystallization temperature ($\sim 300^\circ$C) but below the melting temperature ($\sim 600^\circ$C) over a period of time, GST turns into the crystalline state which corresponds to a logic '1' (a.k.a. SET state). When heated above the melting point and quenched quickly, GST turns into the amorphous state which corresponds to a logic '0' (a.k.a. RESET state). Though writing a PCM cell incurs high operating temperature, the thermal cross-talk between adjacent cells at 65nm is shown to be negligible even without thermal insulation material [22]. Similar to the multi-level flash memory, the phase change material can also be heated to four or more distinct states, forming a multi-level PCM cell that can represent four or more values [1, 9].

PCM arrays can be fabricated in a similar way as DRAM arrays except that the cell now uses the phase change material. Fig. 1 illustrates a typical structure of a 2×2 PCM cell array. It can use the same peripheral logic such as decoders, row buffers, request/reply networks etc. as the DRAM array. The PCM core, indicated in circle, consists of a thin layer of phase change material which is contacted from left and right by electrodes; a heater which is surrounded by thermal insulator; and a pass transistor. The read and write (SET and RESET) operations of a PCM cell require different current and voltage levels on the bitline, and take different amount of time to complete. The RESET operation brings the PCM cell to the amorphous state. It requires the highest voltage levels, in order to melt the phase change material, but only for a short amount of time. The SET operation brings the cell to the crystalline state and requires lower voltage for a longer period of time. The read operation requires the lowest voltage for the shortest amount of time. In Section 4, we will show that PCM memory can be modeled using different resistors corresponding to different material states in place of the core in the circle of the figure.

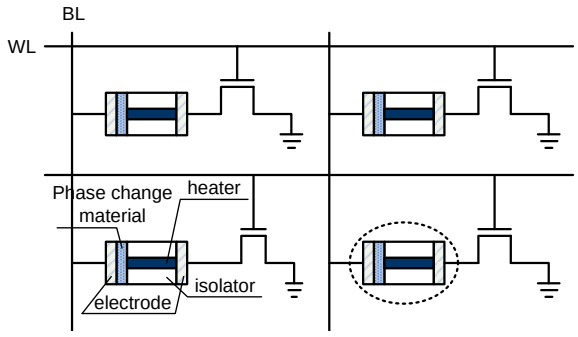

Figure 1: PCM cell array [10, 15, 19].

Due to repeated heat stress applied to the phase change material, the PCM has limited number of write cycles. A single cell can sustain $10^8 \sim 10^9$ [9, 27, 31] writes before a failure can occur. As we will show later, a main memory directly using PCM can last merely ∼100 days running a typical SPEC CPU program. Here we refer the lifetime of a PCM to the duration before the first cell starts to fail. Hence, we must manage to reduce the write cycles so as to make a PCM main memory practical. In addition, the write energy and power for a PCM cell are much higher than for a DRAM cell. From the measurements we obtained, for the same DRAM and PCM capacity (experimental setting given in Section 5), writing a single bit in a DRAM is 86.1fJ while for a PCM cell, writing a '1' takes 13733fJ and writing a '0' takes 26808fJ. Therefore, although PCM can bring in leakage energy savings, it is also imperative to suppress its dynamic energy consumption in order to achieve positive total energy savings. The last concern of PCM is that the read and write operation of a single cell are slower than that of a DRAM cell. We will show that PCM has faster burst reads and the performance of a workload is relatively insensitive to the latency of the main memory.

3. IMPROVING THE ENDURANCE OF PCM

The limited write cycles per PCM cell impairs the lifetime of the PCM memory. We tested the unprotected lifetimes of a PCM main memory using a variety of benchmarks including SPEC2K, SPEC2006, and SPECWeb. Many of these benchmarks such as `art`, `mcf`, `lucas`, and `milc` are memory intensive. The choice of these benchmarks will be further explained in Section 5. The memory lifetime running a benchmark is estimated assuming the PCM main memory is constantly accessed at a rate generated by this benchmark. In reality, such rate may vary with different workloads running in the system. The number of rewrite cycles for a PCM cell is assumed to be 10^8. We used a 4-core CMP with two levels of cache as configured in Table 2, Section 5. As shown in Figure 2, the results ranges from 25 days for `mcf` to 777 days for `specweb-banking`, and the average is only 171 days. In this section, we introduce a sequence of techniques that

can prolong the lifetime of the PCM main memory to an average of over 20 years.

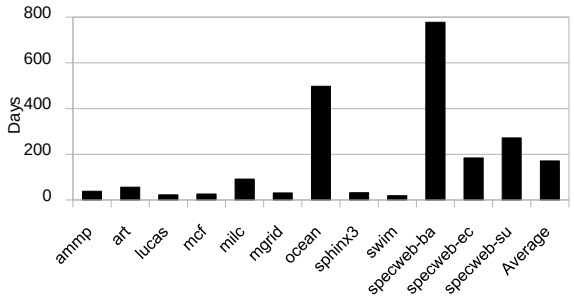

Figure 2: Raw lifetime of PCM main memory.

3.1 Redundant Bit-Writes

The design. To improve the endurance of a PCM, the first step we take is to reduce the write frequency of its cells. In a conventional DRAM access, a write updates the content of an *entire* row (also called a page) of a memory bank. Every bit in the row is written once. However, we observed that a great portion of these writes are redundant. That is, in most cases, a write into a cell did not change its value. These writes are hence unnecessary, and removing them can greatly reduce the write frequency of the corresponding cells. Fig. 3 shows the percentages of redundant writes for different benchmarks. They are calculated as the number of redundant bit-writes over the total number of bits in write accesses. The 'SLC' series represents redundant bit-writes in a single level PCM cell, i.e., each cell stores either '0' or '1'. The 'MLC-2' and 'MLC-4' series represent multi-level PCM cells of 2 and 4-bit width. That is, each cell stores 4 (MLC-2) or 16 (MLC-4) binary values. The number of rewrite cycles for both SLC and MLC's are assumed to be 10^8. Other detailed experimental settings are explained in Section 5.

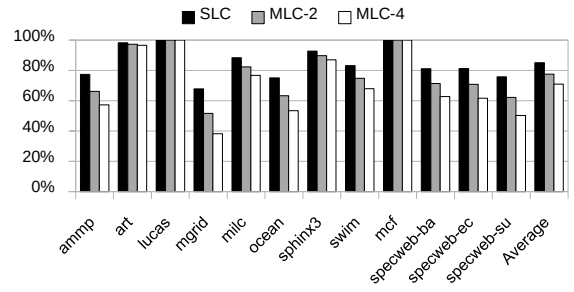

Figure 3: Percentage of redundant bit-writes for single-level and multi-level PCM cells.

We can clearly see from the results that all benchmarks exhibit high percentages of bit-write redundancy. For single-level cells, the statistical bit-write redundancy is 50% if writing a '0' and '1' is equally likely. For MLC-2 and MLC-4 cells, the redundancy probabilities are 25% and 6.25% ($(\frac{1}{2})^4$) respectively. However, the measured redundancies for real workloads are much higher than the theoretic values, showing interesting *value locality*. The redundancy ranges for SLC, MLC-2, and MLC-4 cells are 68∼99%, 52∼99%, and 38∼99%, with an average of 85%, 77% and 71% respectively. Similar value locality has been observed before at word level, and techniques exploiting the "silent stores" have been developed for L1 cache and multiprocessors [16, 17]. Notice that the redundancies for MLC's are higher than the statistical results: $77\% > (0.85^2 = 72\%)$ and $71\% > (0.85^4 = 52\%)$. The conclusions we draw from this study are: 1) there is a great opportunity that we can exploit to reduce the write frequency for PCM cells; and 2) this opportunity exist abundantly not only for SLC, but also for MLCs. After removing these redundant bit-writes, the lifetime of the memory increased by 4.5/3.5/3.0 times for SLC/MLC-2/MLC-4 on average, as shown in Fig. 4. These result in a lifetime of 770/592/510 days, or 2.1/1.6/1.4 years.

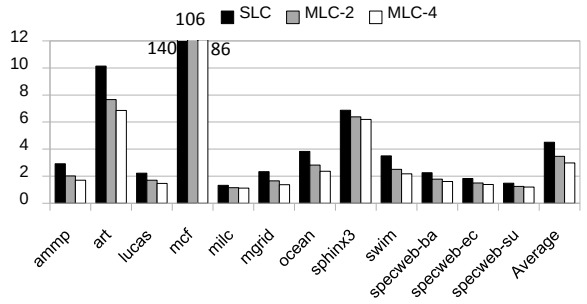

Figure 4: Lifetime increase factor after redundant bit-writes removal.

Implementation. Removing the redundant bit-write can be implemented by preceding a write with a read. In PCM operations, reads are much faster than writes, so the delay increase here is less than doubling the latency of a write. Also, write operations are typically less critical than read operations, so increasing write latency has less negative impact on the performance of the workload. The results we obtained show that removing redundant bit-writes brings forth far higher improvements in endurance than the losses in performance. Hence, it is beneficial to perform an extra read before a write.

The comparison logic can be simply implemented by adding an XNOR gate on the write path of a cell, as illustrated in Fig. 7. The XNOR output is connected to a pMOS which can block the write current when the write data equals the currently stored data. The XNOR gate is built based on pass-transistor logic, whose simple structure guarantees both small delay and negligible power dissipation. Nevertheless, we considered both its delay and power in our final experimental results.

Discussion. It appears from the first sight that we could apply the same design to regular DRAM arrays to reduce the energy in redundant bit-writes. However, as we will show in Section 5 that the read and write operation in DRAM arrays take about the same time, and consume about the same energy. It obviously does not pay off to perform 1.x operations (1 read and .x irredundant writes) in place of 1 write from energy standpoint. In PCM arrays, reads are much faster and consume much less energy than writes. It is this asymmetry can we benefit from to improve the PCM endurance, and reduce the write energy as we will show later.

3.2 Wear Leveling

Even though redundant bit-write removal achieved up to 5 times lifetime extension, the resulting 1.4∼2.2 years of lifespan is still too short for main memory. The reason is that the memory updates happen too locally: the bulk of writes are destined to only a small number bits, creating an extremely unbalanced write distribution. Therefore, those "hot" cells fail much sooner than the rest of the cells. For this reason, the next step we need to perform is wear leveling.

3.2.1 Row shifting

The design. After redundant bit-writes removal, the bits that are written most in a row tend to be localized, rather than spread out. Hence, we apply a simple shift mechanism to even out the writes in a row to all cells instead of a few cells. Simulation results indicate that it is not beneficial to shift on bit granularity, because hot cells tend to cluster together. For example, least significant bits are written more often than most significant bits. Shifting on too coarse a granularity is also not helpful since cold cells might be left out. For those reasons, we found that shifting by one byte at a time is most effective.

In addition, our simulation results also indicate that it is not advantageous to shift on every write because once a line is shifted, writing it back may incur more bit changes than without the shift. Hence, we should perform the shift periodically to amortize its cost. Moreover, a workload does not have equal accesses to every memory pages. Some pages are "hotter" than others, and some pages are accessed more balanced for every line than others. The best shift interval varies significantly from page to page. Hence, we select two representative memory pages from each category: hot, medium hot, medium balanced and unbalanced pages for a workload. They are selected based on the write counts of the pages, and the standard deviations of writes among all lines in a page. We do not consider cold pages as their lifetimes are much longer and tuning the shift interval should not be disturbed by statistics from them.

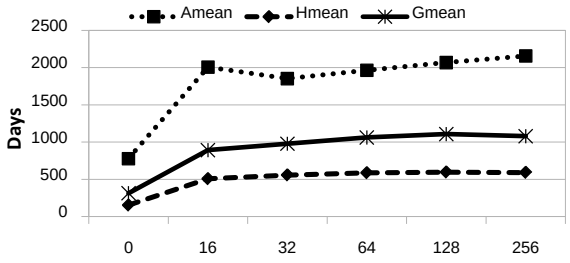

Figure 5: Lifetime with different row shift interval averaged over all benchmarks.

We varied the row shift interval from 0 (no shift) to 256 (shift one byte on every 256 writes), and collected the resulting lifetimes averaged from all selected sample pages. We found that the results from each individual benchmark varies greatly. For example, the `specweb-banking` favors an interval of 16 writes while this interval generated the lowest lifetime for `mcf`. Hence, we summarized the lifetime for different shift intervals averaging over all benchmarks. We used not only arithmetic mean, but also geometric mean and harmonic mean to give more weight to low-lifetime workloads. The results are plotted in Fig. 5. As we can see, the best shift interval is 256 writes, as it generates the highest lifetime for all means. However, there is no need to increase the interval as 1) both geometric and harmonic means have leveled off, and 2) longer interval incurs more counter bits and hardware overhead corresponding to each line. With such a wear leveling for each row, the lifetime of the main memory increased by another factor of 2.77/2.77/2.79 for SLC/MLC-2/MLC-4 as shown in Fig. 6, compared with using only redundant bit-write removal. The new lifetimes are now 5.9/4.4/3.8 years for SLC/MLC-2/MLC-4 PCM main memories, which are still not sufficient for commodity systems.

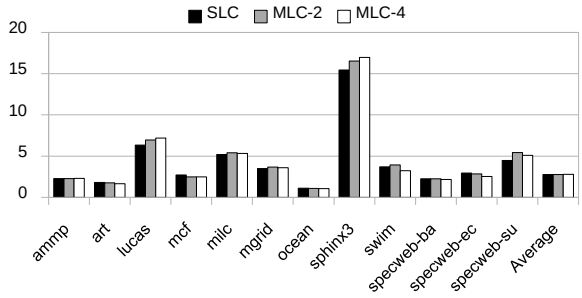

Figure 6: Lifetime increase factor after row shifting.

Implementation. We modeled in HSPICE the critical circuit of a memory array with an additional row shifter, along with a shift offset register, as shown in Fig. 7. The shifter's main part is designed based on pass-transistors. The shifter is the data interface between the cell subarray and peripheral circuitry. The shifter, together with the column muxes and the offset, performs the complete shift of a row. On a read access, the data is reset to its original position before being sent out. On a write access, data is first shifted in the pre-write read, and then shifted back again during the write. We measured the delay and power for a 1KB (our row size) shifter are 400ps and 795μW respectively. Both these overheads are considered in our final simulation results.

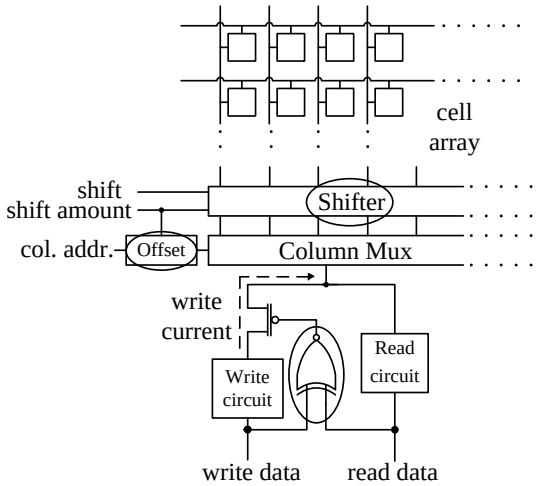

Figure 7: Implementation of redundant bit-write removal and row shifting. Added hardware are circled.

3.2.2 Segment swapping

The second (and last) step of wear leveling is at a coarser granularity. The row shift mechanism helps to extend the lifetime of each row. However, this has only limited effect on hot pages that have significant amount of writes than others. Those pages will still fail sooner even though each row has balanced writes. Therefore, we develop a coarse granularity wear leveling mechanism that periodically swaps memory segments of high and low write accesses.

The design. The main parameters we need to determine for segment swapping are segment size and swap interval. To select a proper segment size, we experimented with small sizes such as one, or several pages. The main difficulty is that the metadata to keep track of the page writes would be too big. For example, for a 4GB memory with 4KB page size, 1M page write counters need to be maintained. This is not only a big storage overhead but also it requires long latency at runtime for sorting the counters to find cold pages for swapping. This may also impose performance concerns. Therefore, we enlarged segment sizes and experimented with a number of options. For each segment size, we varied the the swap intervals in number of writes, and collected the resulting lifetime as depicted in Fig. 8.

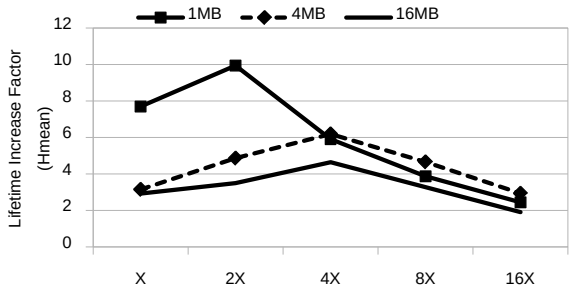

Figure 8: Effect of segment size and swap interval on lifetime (Hmean).

The x-axis shows different swap intervals in unit of a base interval ('X'). This is because larger segments should use larger intervals so different segment sizes have different base intervals. The lifetimes are averaged over all benchmarks under test. We used harmonic mean in order to find a solution that is not biased by an out lier that has particular long lifetime (which is likely to happen if arithmetic mean is used). The results in Fig. 8 clearly show that larger segment sizes do not benefit wear leveling. This is because swapping larger segments introduces higher overhead in terms of extra writes. For example, the overhead for 1MB, 4MB, and 16MB segments on their base swap intervals are 2.8%, 5.6% and 5.2% respectively. Therefore, the best option is the 1MB segment size with 2X swap interval which corresponds to 2×10^6 writes. We remark that segment swaps bring overhead mostly in performance. Endurance wise, each cell involved in the swapping is written only for *one more time*, which has neglig the lifetime.

Implementation. While the previous lifetime extension methods can be implemented in circuit, swapping large size memory segments should be implemented in the memory controller. The controller needs to keep a mapping table between the "virtual" segment number generated by the core and the "true" segment number for the actually location of the requested segment. The size of this table depends on the capacity of the main memory. For a 4GB main memory, the 1MB segment size results 4K entries in the table, and each entry stores $A - 20$ bits where A is the width of the physical memory address.

For each segment, we keep two control data: 1) *write_count* which counts the number of writes to the segment; and 2) *last_swapped* which remembers when this segment was swapped last time. A cold segment may be picked multiple times for swapping in a short period of time. Keeping *last_swapped* can prevent the segment from being selected again too soon. In addition, the segment swap should not happen too frequently within a short amount of time. If there are many swap requests, the controller can delay them because the time a swap should happen is not very critical. Such a design can even out segment swaps so that they do not have significant impact on system performance. We set a global constant *swap_throttle* as the threshold of the number of swaps that can happen in a fixed period of time. A *swap_queue* is used to record the awaiting swap requests if they cannot be serviced promptly.

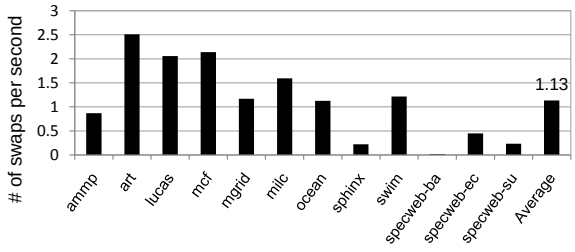

Figure 9: Frequency of segment swapping.

For the best swap configuration we selected — 1MB segment swapped on every 2×10^6 writes, we found the average number of swaps occurred per second is 1.13, as shown in Fig. 9. When a swap occurs, the memory has to stall for 2×1MB/1KB row-writes, assuming each row has 1KB. As we will show in Section 5, writing a row of 1KB in PCM takes at most 156.55ns (36.28ns for reads + 120.27ns for writes). Therefore, the memory is unavailable for 1.13 × 156.55ns × 2 × 1MB/1KB = 0.36ms per second. This amounts to 0.036% performance degradation to the running workloads in the worst case. This overhead can be decreased by using two row buffers in the memory controller to read and write the exchanging rows in parallel, reducing the overhead by half to 0.018%.

3.2.3 Final PCM main memory lifetime

With a hierarchy of techniques — redundant writes (RW) removal at bit level, byte shifting (BS) at row level, and swapping at segment level (SS), the lifetime of a PCM main memory can be extended greatly. We used a 4GB main memory to test the final lifetime with all three techniques applied. The results in number of years are listed in columns labeled 'SLC', 'MLC-2', and 'MLC-4' of Table 1. We showed both arithmetic mean and harmonic mean for the lifetime. As we can see, the lifetime of all PCM cell type is extended to 13∼22 years.

We remark that RW, BS and SS are all critical to extending the lifetime. We have shown in Fig. 4 and 6 that applying

Table 1: Final lifetime (years) with redundant bit-write removal, row shifting, and segment swapping.

Benchmarks	SLC(segment swap only)	SLC	MLC-2	MLC-4
ammp	4.9	32.5	22.7	19.3
art	1.3	24.8	18.2	15.4
lucas	1.1	15.0	12.7	11.2
mgrid	1.8	26.7	20.0	16.2
milc	2.3	36.3	32.8	31.5
ocean	52.8	224.4	161.2	133.1
sphinx3	5.5	586.2	583.3	580.9
swim	0.3	3.9	3.0	2.1
mcf	3.3	1285.4	889.0	719.5
specweb-banking	4.4	22.1	17.5	15.2
specweb-ecommerce	8.0	42.9	34.1	27.9
specweb-support	10.2	83.9	69.1	62.7
Amean	8.0	198.7	155.3	136.3
Hmean	1.7	**22.1**	**17.1**	**13.4**

RW and BS can prolong the lifetime to up to 5.8 years. Let us now see the effect of applying SS alone on lifetime. The results are shown in the 2^{nd} column of Table. 1. As we can see, without RW and BS, SS alone can only improve the lifetime to 1.7 years by harmonic mean. These results prove that every technique is an indispensable component of our scheme.

4. MODELING THE PCM MAIN MEMORY

Having solved the endurance problem, we now proceed to PCM design for good performance and low energy. In this section, we describe how we model DRAM and PCM main memory, and how we make design choices for PCM in order to achieve the best energy-delay product.

4.1 DRAM Modeling

For DRAM modeling, we leveraged the recent power-performance cache/memory modeling tool CACTI-D that has been significantly enhanced with modeling of commodity DRAM technology and support for main memory DRAM chip organization [25]. CACTI-D provides three types of devices defined by ITRS: high performance (HP), low standby power (LSTP), and low operating power (LOP). These devices are used in the peripheral and global support circuitry such as input/output network, decoders, sense amplifiers etc. The HP devices have the highest operating speed. They have short gate length, thin gate oxide, and low V_{th}. The downside is that HP devices have high dynamic and leakage power. The LSTP devices on the other hand are designed for low leakage power. They have longer gate length, thicker gate oxides, and higher V_{th}. Naturally, these devices are slower than for HP devices. The LOP devices are between HP and LSTP devices in terms of performance. The advantage is that LOP devices provide lowest dynamic power among the three.

For the access transistor in each DRAM cell, there are commodity DRAM (COMM-DRAM) and logic process based DRAM (LP-DRAM) technology. The latter is used in embedded memories. The studies performed earlier [25] showed that using COMM-DRAM with LSTP peripheral circuitry generates the best energy-delay product for 3D stacked DRAM. We will use this configuration for our DRAM as a baseline to compare with.

4.2 PCM Modeling

PCM has been implemented using the same architecture as DRAMs [10,15,20]. They share similar peripheral circuits but differ in the implementation of cells. Hence, the methodology we used is to simulate the essential circuits such as the cell, bitlines, wordlines, read/write circuits etc. in HSPICE, and then replace the CACTI results related to those essential circuits with our HSPICE results. In other words, we only use the skeleton of the CACTI DRAM model, and fill in the contents with HSPICE PCM model. This method also provides a fair comparison because PCM and DRAM will use the same floorplan. We used 45nm feature size in our model. The memory capacity is 4GB.

4.2.1 Selecting peripheral device type

The DRAM uses LSTP transistors for the peripheral circuitry to lower the leakage energy in the entire memory. We could also opt for the same device for PCM. However, PCM cell's read/write latency are much longer (especially for write) than DRAM cells. For example, if everything else being equal, the access latency on the bitline, which dominate the memory access latency, for DRAM is ∼0.17ns, while it is ∼20ns for PCM. Though LSTP transistors have the lowest leakage, its performance is the worst among the aforementioned three options. The memory access latency relies heavily on the speed of the peripheral circuits. Therefore, we need to use performance advantageous devices such as HP or LOP. However, they both have much higher leakage energy than the LSTP devices (90× for HP and 2× for LOP).

Fortunately, using them for PCM does not raise this concern because PCM is persistent. When the memory is idle, nearly all peripheral circuitry can be powered down without losing the stored data. This is the most distinct benefit of using PCM with performance advantageous peripheral circuitry. Between HP and LOP, HP transistors are faster but consume more energy, in both dynamic and leakage energy. LOP transistors are slower but consume less total energy. Our experiments show that HP devices have too high leakage at runtime to be acceptable for PCM. Therefore, we chose LOP devices for the peripheral circuits in our design.

4.2.2 The cell

Since a write operation to a PCM cell incurs state changes, it is impractical to model the transient behavior of the phase change material during a write. We leveraged a recent work on 32nm PCM study [23] and extracted critical measures from its results. For write delay, the SET and RESET pulse widths are 90ns and 60ns respectively. For write power, the SET and RESET dissipate 152.6μW and 446.8μW which correspond to 13.733pJ and 26.808pJ respectively.

Since PCM works with a resistance sensing scheme, the read operation is highly related to the resistance of the SET and RESET state. More importantly, the resistance will greatly determine the read latency, size of the access device (the transistor shown in Fig. 1), stability and data retention. We used the results in [23] as our basis and applied scaling rules for a 45nm cell. This is because this work performed a comprehensive and detailed study on the scaling of phase change material (doped GeSb) into 32nm, covering our desired feature size (45nm). In their work, the device of cross-sectional area of 60nm^2 have a resistances of 95KΩ and 500KΩ for SET and RESET states respectively. They also reported that a cross-sectional area of ∼300nm^2 device can be projected to the 45nm feature size. In [22], the authors studied the RESET current and the thermal proximity cross talk for scaled PCM, from which they provided a simple linear relationship between resistance and geometric dimension. Based on this linear scaling rule, we came up with 19KΩ and 100KΩ (resistance is smaller when the cross-sectional area is bigger) for SET and RESET states at 45nm feature size. However, there is only ∼5× difference between the SET and RESET resistance, which is considered small compared with 10∼100X difference between the two for better reliability [4,27]. Hence, we increased the RESET resistance to 300KΩ (15× difference), referring to the parameters given in [27].

Access device selection. There are two main options for the access device of each cell: a transistor or a diode. A diode has a simpler structure, and hence it is good for cell density. However, a diode cannot satisfy the high write current requirement beyond sub-100nm technology [6]. Also the scaling rule of a diode is not so clear as NMOS [22]. Lastly, the diode-based cell has been reported to be more vulnerable to errors induced by writing data to adjacent cells because of bipolar turn-on of the nearest-neighbor cells [21]. Taking all of the above, especially the scalability, into consideration, we selected transistor-based cell for our PCM model.

5. EXPERIMENTAL RESULTS

5.1 Settings

We evaluated our proposed scheme with a set of programs that have *dense* memory write accesses, meaning that the number of writes per unit of memory region is high. Here the density of writes is more important than the memory footprint of the benchmark because the former is more hazardous to the endurance of PCM and varies significantly among different benchmarks. For example, Fig. 10 and Fig. 11 show the memory write histogram of two workloads: *mcf* and *raytrace* in 10 seconds of simulated time. The x-axis denotes the memory space in unit of 1MB memory segment, and the y-axis denotes the number of writes in logarithmic scale in each segment. These two benchmarks represent two extremes of memory write behaviors. *Mcf* has very high write density while *Raytrace* has low write density. *Raytrace* has naturally relatively balanced writes across the memory space, and are less of a concern from the endurance standpoint. We therefore chose those workloads that are more analogous to *mcf*.

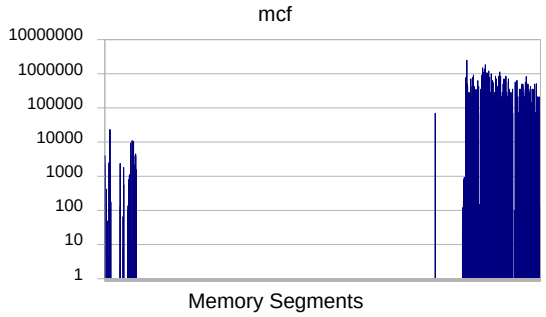

Figure 10: Memory write histogram of mcf.

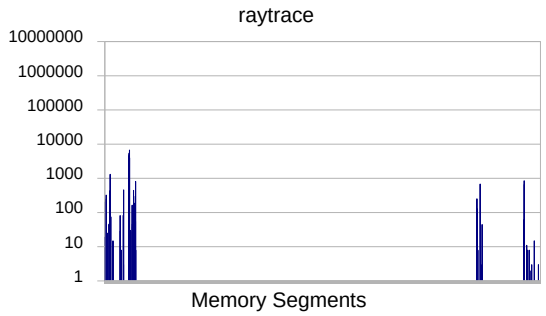

Figure 11: Memory write histogram of raytrace.

In our experiments we used (1) *ammp, art, lucas, mcf, mgrid* and *swim* from SPEC2K, and *milc, sphinx3* from SPEC2006; (2) *ocean* from SPLASH2; (3) *SPECweb-Banking*, *SPECweb-Ecommerce* and *SPECweb-Support* from SPECweb-2005 benchmark suites. Standard input sets are used for those benchmarks. We set the checkpoints in the middle of their executions to skip the warmup phase.

We simulated a 4-core CMP (with parameters listed in Table 2) in Simics [30] simulation environment. We set the core frequency as 1GHz because we assumed a 3D architecture (more below) which is subject to tight thermal constraint. Trace-driven simulation was used for lifetime analysis and execution-driven simulation was used for studying energy and performance. For the latter, we chose the GEMS [29] simulator with both Ruby (detailed cache and memory simulator) and Opal (out-of-order core simulator) activated. We enhanced the memory module to model both the latency and energy consumption of PCM and DRAM.

We assumed that the memory is stacked right onto the 4-core chip, similar to the PicoServer architecture [12]. We chose a 3D architecture to evaluate PCM because if it was used as an off-chip memory, the latency on the CPU-memory bus would diminish the latency problem of PCM. The memory is organized as one DIMM with 4 ranks, assuming each rank is one layer in a 3D stack. We implemented burst read mode in the row buffer. The parameter of PCM and DRAM are taken from HSPICE and rounded to memory cycles.

For lifetime analyses, we processed the traces as follows:

Table 2: Hardware parameters of a 4-core CMP.

Processor core	4-OOO-core, each core runs at 1GHz
L1 Cache	Private L1 cache (32K I-cache and 32K D-cache), 64-byte lines, 4-way set associative, 3 cycles access time
L2 Cache	Shared L2 cache, 4MB, 64-byte lines, 16-way set associative, 6 cycles access time
Memory controller	one controller, next to core 0
Memory size	4GB memory
Memory organization	1 DIMM, 4 Ranks/DIMM, 16 Banks/Rank; use top 16 bits as row number; each rank is a layer of 3D stacking of on-chip memory
Interconnect Network	2×2 mesh network

(1) Run Simics for a short time interval: e.g. 10 seconds of simulated time, and gather statistics on write counts for each 64-byte line; (2) Analyze the statistics of each line and select 6∼8 pages with different characteristics: hot pages, unbalanced pages, balanced pages and medium-hot pages; (3) Run long-time (20 minutes of simulated time) simulation with the workload again, and record trace for the selected pages. Each trace record contains the type (R/W), timestamp and physical address of the memory access. For write requests, it also contains the original data and modified data of the line.

To study the effectiveness of segment swapping, we ran simulations for longer time (5∼10 minutes of simulated time), and recorded the write accesses to each segment. Each trace record remembers the segment number of the segment being written to. To perform execution-driven simulation for studying energy consumption and performance, we skipped the initialization phase, warmed up for 5M instructions of each run and simulated the next 5M instructions. Energy and latency data are then collected from the output of the simulator.

5.2 Results

The results for lifetime extension have been presented in Section 3. In this section, we focus on the energy and delay comparisons of PCM and DRAM.

5.2.1 Per access latency and energy

Table 3 shows the latency and energy for a single read and write operation between PCM and DRAM main memory. The read access is for reading an entire row of 64B from a memory bank. The write access for PCM is for a single bit since we removed 85% of bit-writes on average. The DRAM write access is for an entire row.

We can see that PCM access latencies are longer than DRAM accesses, except for burst reads that hit the same bank of the memory. These hits do not necessarily hit the row buffer. This is mainly because of two reasons. First, the peripheral logic of PCM (LOP devices) are faster than that of the DRAM (LSTP devices). We have discussed the tradeoffs of these choices earlier. Second, the DRAM reads are destructive. DRAM needs to restore (write back) the data to cells before the next access to the same bank. This is a process to regenerate full-rail values on the high capacitive bitlines, which takes some time. These operations increase the memory's random access time. In contrast, PCM-based memory can easily handle successive same-bank accesses by nature because reads are not destructive, much like an SRAM array.

In addition to performance gains in burst reads, PCM also does not require refreshing cycles. We found that this saving results in 4∼5% of latency improvement.

Next, it is encouraging to see that our PCM design with LOP transistors generate lower dynamic energy than the DRAM using LSTP transistors. More distinctively, the write energy, which is the biggest concern in the literature due to the high write current, has been greatly reduced. The table only shows the per-bit write energy. The per access write energy is calculated as follows:

$$E_{pcmwrite} = E_{fixed} + E_{read} + E_{bitchange}$$

where E_{fixed} is the "fixed" portion of energy charged for each PCM write including row selecting, decoding, XNOR gates, etc. In our model, this part is 4.1nJ per access. E_{read} is the energy to read out the wordline for comparison. This part is approximately 1.075nJ which includes the energy spent in the row shifter as shown in Fig. 7. The $E_{bitchange}$ part depends on the proportion of updated bits ($0 \rightarrow 1$ or $1 \rightarrow 0$):

$$E_{bitchange} = E_{1\rightarrow 0}N_{1\rightarrow 0} + E_{0\rightarrow 1}N_{0\rightarrow 1}$$

As listed in Table 3, $E_{1\rightarrow 0}$ and $E_{0\rightarrow 1}$ are 0.0268nJ and 0.013733nJ respectively. Therefore, per access write energy for PCM (nJ) can be expressed as:

$$E_{pcmwrite} = 5.175 + 0.0268 \times N_{1\rightarrow 0} + 0.013733 \times N_{0\rightarrow 1}$$

We can see that removing redundant bit-writes can significantly reduce PCM write energy. For example, from the statistics in Fig. 3, we removed 85% of bit-writes on average (SLC). If we assume writing a '0' or '1' is equally likely, the total write energy per "row" is 6.73nJ including the energy on the circuit illustrated in Fig. 7. This is significantly lower than the per access write energy of DRAM.

5.2.2 Dynamic energy savings

From the previous per access energy results, we can see that using PCM can save substantial amount of dynamic energy in main memory, when compared with traditional 3D stacked DRAM. We show in Fig. 12 the dynamic energy of DRAM (left bar) and PCM (right bar) broken down to initial reads, burst reads, writes and refreshes. PCM's dynamic energy is only 47% of the DRAM's dynamic energy, achieving a 53% of reduction. The greatest reductions come from 1) the write energy because the majority of the bit-writes are removed; and 2) the burst read energy, as we explained in the previous section;. Moreover, PCM-based memory does not require refreshing logic and energy because of the non-volatile nature and negligible cell array leakage due to the predischarged bitlines.

Table 3: Latency and energy comparison for a single access between PCM and DRAM accesses.

	Latency (ns)		Energy (nJ)	
	PCM	DRAM	PCM	DRAM
Read	36.28 (initial)	20.04 (initial)	10.68 (initial)	12.17 (initial)
	6.47 (burst)	9.33 (burst)	3.77 (burst)	12.06 (burst)
Write	90.27 (0)	20.04	0.0268 (0)	14.48 per row (64B)
	120.27 (1)		0.013733 (1)	

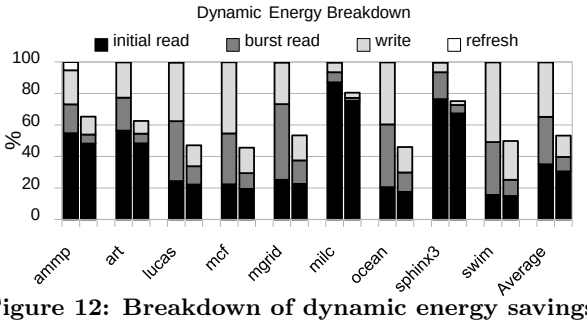

Figure 12: Breakdown of dynamic energy savings.

5.2.3 *Leakage energy savings*

The leakage savings come from two sources: 1) cell leakage reduction due to the non-volatility of PCM cells; and 2) the power gating of peripheral circuits when the memory is idle. During this time we power down the peripheral circuits because we will not loose the contents in the memory. This can save significant energy in the peripheral circuits because we used LOP devices which have higher leakage when active than the LSTP devices for the DRAM peripheral. Note that even for memory intensive workloads such as *mcf*, the leakage saving is nearly 40%. The combined effect results in great leakage energy reductions, as illustrated in Fig. 13. On average, we achieved 74% of savings, when compared with a DRAM-based memory that has already been optimized for low leakage.

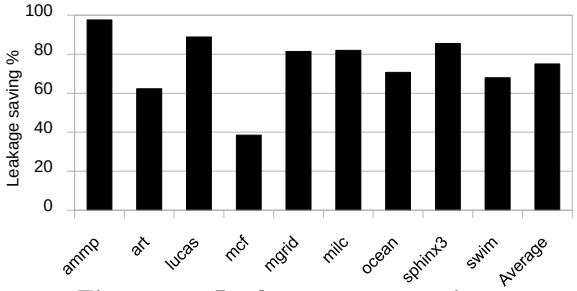

Figure 13: Leakage energy savings.

With dynamic and leakage energy savings combined, the total energy savings we achieved is 65% averaged over all programs, as shown in Fig. 14.

5.2.4 *Latency results*

Although PCM has slower read and write operations, their impact on program performance is quite mild. Similar observations have also been made in [25] where the long latency of the last-level cache implemented using commodity DRAM and LSTP peripheral did not have much weight on the performance of the chip. That is, programs are relatively insensitive to memory hierarchies that are far from the CPU. Note that our platform is 3D stacked chip, meaning that the memory is already closest to the core. The sensitivity of the performance to the memory latency should be higher than having an off-chip main memory. In other words, if our PCM design is used as an off-chip main memory, its latency effect on the overall system performance would be even smaller, indicating that reducing energy carries is more important than reducing the latency. Our results show that the CPI increase ranges from 0.3% for *ammp* to 27% for *art* with an average of 5.7%.

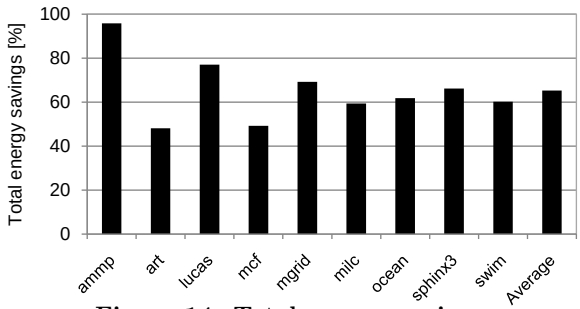

Figure 14: Total energy savings.

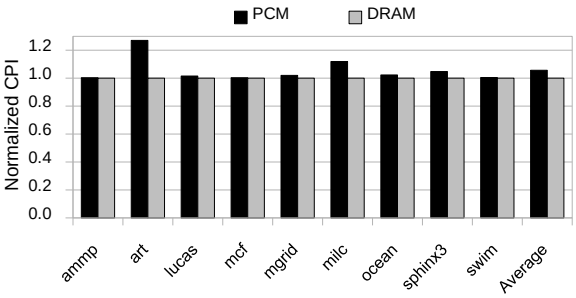

Figure 15: Memory latency impact on CPI.

5.2.5 *Energy-Delay^2 savings*

Finally, we present the results for ED^2 in Fig. 16. Due to the great savings in the total energy, and mild increase in total execution time, the ED^2 all resulted in positive savings, ranging from 96% for *ammp* to 16% for *art*. The average savings we achieved is 60%. These results show that using PCM-based main memory in 3D stacked architecture has a great advantage in achieving energy-efficiency.

6. CONCLUSION

We have proposed a suite of hierarchical techniques to prolong the lifetime of PCM-based main memory: redundant bit-write removal, row shifting, and segment swapping. The PCM lifetime with our proposed techniques is increased from 176 days to 22 years for single-level PCM cells. We performed detailed modeling of PCM main memory, and studied its energy-performance tradeoffs. Our conclusion is

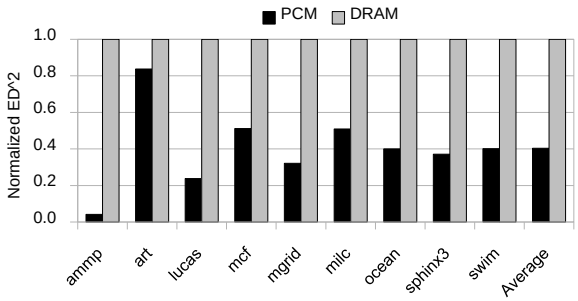

Figure 16: Energy-Delay^2 reduction.

that using low operating power devices for the peripheral of PCM main memory can achieve the highest energy and ED^2 reduction, when compared with commodity DRAM-based main memory using low standby power support circuit. Our experiments showed that 65% of energy saving and 60% of ED^2 reduction can be achieved.

Acknowledgement

This work is supported in part by NSF under CNS-0720595, CCF-0734339, CAREER awards CNS-0747242 and CCF-0641177, and a gift from Intel Corp. The authors thank anonymous reviewers for their constructive comments.

7. REFERENCES

[1] F. Bedeschi, et al. "A Multi-Level-Cell Bipolar-Selected Phase-Change Memory," *2008 IEEE International Solid-State Circuits Conference,* pp. 428-429, 2008.

[2] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCaule, P. Morrow, D. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, C. Webb, "Die Stacking (3D) Microarchitecture," *International Symposium on Microarchitecture,* pp. 469-479, 2006.

[3] F. Chen, S. Jiang, X. Zhang, "SmartSaver: Turning Flash Drive Into a Disk Energy Saver for Mobile Computers," *The 11th International Symposium on Low Power Electronics and Design,* pp. 412-417, 2006.

[4] Y. C. Chen, et al. "Ultra-Thin Phase-Change Bridge Memory Device Using GeSb," *International Electron Devices Meeting,* pp. 777-780, December 2006.

[5] S. L. Cho, et al., "Highly Scalable On-axis Confined Cell Structure for High Density PRAM beyond 256Mb," *Symposium on VLSI Technology Digest of Technical Papers,* pp. 96-97, 2005.

[6] W. Y. Cho, et al. "A 0.18-um 3.0-V 64-Mb Nonvolatile Phase-Transition Random Access Memory (PRAM)," *IEEE Journal of Solid-State-Circuits,* Vol. 40, No. 1, pp. 293-300, 2005.

[7] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, Y. Chen, "Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement," *Design Automation Conference,* pp. 554-559, 2008.

[8] M. Kanellos, "IBM Changes Directions in Magnetic Memory," August, 2007, http://news.cnet.com/IBM-changes-directions-in-magnetic-memory/2100-1004_3-6203198.html

[9] D. H. Kang, et al., "Two-bit Cell Operation in Diode-Switch Phase Change Memory Cells with 90nm Technology," *IEEE Symposium on VLSI Technology Digest of Technical Papers,* pp. 98-99, 2008.

[10] S. Kang, et al. "A 0.1-μm 1.8-V 256-Mb Phase-Change Random Access Memory (PRAM) with 66-MHz Synchronous Burst-Read Operation," *IEEE Journal of Solid-State Circuits,* pp. 210-218, Vol. 42, No. 1, Jan. 2007.

[11] T. Kgil, D. Roberts, T. Mudge, "Improving NAND Flash Based Disk Caches," *The 35th International Symposium on Computer Architecture,* pp. 327-338, 2008.

[12] T. Kgil, A. Saidi, N. Binkert, R. Dreslinski, S. Reinhardt, K. Flautner, T. Mudge, "PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor," *International Conference on Architecture Support for Programming Languages and Operating Systems,* pp. 117-128, 2006.

[13] S. Kim, H.-S. P. Wong, "Generalized Phase Change Memory Scaling Rule Analysis," *Non-Volatile Semiconductor Memory Workshop,* 2006.

[14] S. Lai, T. Lowrey, "OUM - A 180nm Nonvolatile Memory Cell Element Technology for Standalone and Embedded Applications," *International Electron Devices Meeting,* pp. 36.5.1-36.5.4, 2001.

[15] K.-J. Lee, et al. "A 90nm 1.8V 512Mb Diode-Switch PRAM with 266MB/s Read Throughput," *IEEE Journal of Solid-state Circuits,* pp. 150-162, Vol. 43, No. 1, January 2008.

[16] K. M. Lepak and M. H. Lipasti, "On the Value Locality of Store Instructions," *International Symposium on Computer Architecture,* pp. 182-191, 2000.

[17] K. M. Lepak and M. H. Lipasti, "Silent Stores for Free," *International Symposium on Microarchitecture,* pp. 22-31, 2000.

[18] G. Loh, "3D-Stacked Memory Architecture for Multi-core Processors," *The 35th International Symposium on Computer Architecture,* pp. 453-464, 2008.

[19] M. G. Mohammad, L. Terkawi, M. Albasman, "Phase Change Memory Faults," *the 19th International Conference on VLSI Design,* pp.108-112, 2006.

[20] H. Oh, et al. "Enhanced Write Performance of a 64-Mb Phase-Change Random Access Memory," *IEEE Journal of Solid-State-Circuits,* Vol. 41, No. 1, pp. 122-126, 2006.

[21] J. H. Oh, et al. "Full Integration of Highly Manufacturable 512Mb PRAM Based on 90nm Technology," *International Electron Devices Meeting,* pp. 49-52, 2006.

[22] A. Pirovano, A. Lacaita, A. Benvenuti, F. Pellizzer, S. Hudgens, R. Bez, "Scaling Analysis of Phase-Change Memory Technology," *IEEE International Electron Devices Meeting,* pp. 29.6.1-29.6.4, 2003.

[23] S Raoux, et al., "Phase-Change Random Access Memory: A Scalable Technology," *IBM Journal of Research and Development,* vol. 52, No. 4/5, pp. 465-479, 2008.

[24] F. Tabrizi, "The Future of Scalable STT-RAM as a Universal Embedded Memory," *Embedded.com,* February 2007.

[25] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. Brockman, N. P. Jouppi, "A Comprehensive Memory Modeling Tool and its Application to the Design and Analysis of Future Memory Hierarchies," *The 35th International Symposium on Computer Architecture,* pp.51-62, 2008.

[26] M. Wu, W. Zwaenepoel, "eNVy: A Nonvolatile, Main Memory Storage System," *International Conference on Architecture Support for Programming Languages and Operating Systems,* pp. 86-97, 1994.

[27] F. Yeung, et al., "$Ge_2Sb_2Te_5$ Confined Structures and Integration of 64Mb Phase-Change Random Access Memory," *Japanese Journal of Applied Physics,* pp. 2691-2695, 2005.

[28] Intel, "Intel, STMicroelectronics Deliver Industry's First Phase Change Memory Prototypes," *Intel News Release,* Feb. 6, 2008, http://www.intel.com/pressroom/archive/releases/20080206corp.htm

[29] Milo M. K. Martin, et al., "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset," *Computer Architecture News (CAN),* 2005.

[30] P. S. Magnusson, et al., "Simics: A full system simulation platform," *Computer,* 35(2):50–58, 2002.

[31] "The International Technology Roadmap for Semiconductors, Process Integration, Device and Structures," *http://www.itrs.net/links/2007itrs/2007_chapters/2007_PIDS.pdf* 2007.