

Characterizing and Mitigating the Impact of Process Variations on Phase Change based Memory Systems

Wangyuan Zhang and Tao Li

Intelligent Design of Efficient Architecture Lab (IDEAL)

Department of Electrical and Computer Engineering, University of Florida

zhangwy@ufl.edu, taoli@ece.ufl.edu

Abstract

Dynamic Random Access Memory (DRAM) has been used in main memory design for decades. However, DRAM consumes an increasing power budget and faces difficulties in scaling down for small feature size CMOS processing technologies. Compared to conventional DRAM, emerging phase change random access memory (PRAM) demonstrates superior power efficiency and processing scalability as VLSI technologies and integration density continue to advance. Nevertheless, using nano-scale fabrication technologies will unavoidably introduce design parameter variability in the manufacturing stage. In the past, the impact of process variation (PV) on conventional transistor-based storage cells and combinational logic has been studied extensively. However, the implication of PV on non-volatile memory design using emerging phase change techniques has not been well understood. In this paper, we take the first step toward characterizing the effect of process variation on PRAM and explore PV-aware design techniques. We show that process variation increases the PRAM programming power by 96% and degrades PRAM endurance by 50X. Our proposed circuit and two microarchitecture techniques with system-level support reduce PRAM power by 44%, 59% and 57% and improve PRAM endurance by 27X, 277X and 268X, relative to PV-affected PRAM design. Moreover, we show that the synergy of the proposed cross-layer approaches, which achieve an average 63% power savings and 13050X endurance improvement over the conventional case, provide an attractive design solution to mitigate the deleterious impact of PV for non-volatile memory in the upcoming nano-scale processing technology era.

Categories and Subject Descriptors

C [Computer System Organization]: General – modeling of computer architecture

General Terms

Performance, Design, Experimentation

Keywords

Phase Change Memory, Process Variation, Memory System

1. Introduction

Dynamic Random Access Memory (DRAM) has been used as main memory for decades due to its high-density, high-performance and low-cost. However, DRAM technologies are

facing both power and scalability issues: DRAM-based memory is consuming an increasing proportion of the power budget and has been reported to account for as much as 40% of the total system power [1]. DRAM is also difficult to scale down due to various limitations associated with storage interface, device leakages and challenges in integrating high aspect ratio capacitors within tight space. Currently, the DRAM manufacturing industries use 60-70nm processing technologies in their productions and there are substantial challenges to scale down the processing nodes to sub-50nm and beyond [2]. In contrast to conventional DRAM, cutting-edge Phase Change Random Access Memory (PRAM) is attracting increasing attention as a promising candidate for next generation memories [3, 4, 5]. Phase change memory is a type of non-volatile memory that uses the unique behavior of chalcogenide glass, which can be switched between two states (crystalline and amorphous) with the application of heat. Section 2 provides a brief background on PRAM. Among the desirable characteristics of PRAM (e.g. non-volatility and very low standby power), superior scalability is the most attractive. As memory density increases with each smaller generation, the volume of PRAM's phase change material (e.g. $\text{Ge}_2\text{Sb}_2\text{Te}_5$) shrinks as well, providing a truly scalable solution. This will allow PRAM to scale down for the next several generations of processing technology and have no physical limits until at least the 20nm technology node [6].

Although advanced process technologies provide smaller and faster devices (through aggressive feature size scaling), difficulties in manufacturing control have resulted in significant variability in the fabricated devices. At the chip-level, process variation can lead to performance discrepancies between dies. Process variation also exists at fine-grain levels and manifests as correlations among within-die devices. Transistor variability significantly reduces chip operating frequency and increases power consumption [7, 8, 9]. Recently, there have been increased interest in characterizing the impact of PV on processor performance/power and developing circuit- and microarchitecture- level techniques to mitigate the deleterious effect of PV [10,11,12,13]. These studies, however, all focus on conventional, CMOS-transistor based storage cells and combinational logic. As PRAM is becoming a promising technology for nano-scale non-volatile memory system design, architects need to understand the system-level implication of PV on PRAM memory and develop cost-effective PV mitigation techniques. Towards this goal, we provide the first study on the types and sources of process variations in PRAM design and analyze their impact on PRAM endurance and programming power. We further propose circuit-, microarchitecture- and system-level optimizations to overcome the deleterious impact of PV on PRAM. The synergy of the proposed cross-layer approaches yields cost-effective PV-aware design methodologies for nano-scale non-volatile memory using emerging phase change technologies.

The contributions of this paper are:

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, or 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

- We identify the major sources of process variation in PRAM design and quantify the impact of PRAM parameter variability on its programming current. We observe that variations in the PRAM Bottom Electrode Contact Diameter, heater thickness, and GST layer thickness lead to a variation coefficient (σ/μ) of 15%, 19% and 1% in programming current respectively. In conventional design, PV-affected PRAM is managed by tuning programming current for the worst case, resulting in over-programming which negatively impacts power and endurance. We show that this worst-case-driven design methodology results in 96% write power overhead and a 50X lifespan degradation, which compromises the power benefit and exacerbates the wear-out. This suggests that PV-aware PRAM design is imperative in light of nano-scale fabrication technologies.

- We explore cross-layer approaches to mitigate PV-induced PRAM power and endurance overhead. Our circuit-level design adapts PRAM programming currents based on device variability statistics. Our microarchitecture techniques use data-comparison-write and memory compression to reduce the number of bit writes to PRAM cells that suffer negative PV effects. To minimize the performance penalty introduced by the proposed techniques while maximizing their benefit, we use hot- and cold-modified page classification to selectively trigger and adapt PV mitigation schemes. We show that these techniques can cost-effectively mitigate the deleterious PV effect while yielding negligible performance and power overhead. Moreover, we show that the synergy of the proposed cross-layer approaches, which achieves 63% power savings and 13050X endurance improvement over the conventional design, provides an attractive solution for non-volatile memory in the upcoming nano-scale processing technology era.

The rest of this paper is organized as follows. Section 2 provides a brief background on PRAM and our process variation modeling methodology. Section 3 describes PRAM design parameters subject to process variation and quantifies the effect of PV on PRAM programming power and endurance. Section 4 proposes cross-layer PV-aware PRAM design and optimization techniques. Section 5 describes our experimental methodologies including machine configuration, simulation framework and workloads. Section 6 presents our evaluation results. Section 7 discusses related work and we summarize our work in Section 8.

2. Background: PRAM and Process Variation

2.1 PRAM

Phase-change Random Access Memory (PRAM) is a class of non-volatile memory devices that employ a reversible phase change in materials to store information. PRAM has many advantages such as random access, non-volatility, superior scalability, fast read-cycle and compatibility with CMOS process. Currently, PRAM is being considered as one of the most promising technologies for next generation non-volatile memory. As a result, PRAM has been moved to the forefront of memory industry R&D activity and is being commercialized by a number of semiconductor manufacturers [14, 15, 18, 21].

Figure 1 shows the basic structure of PRAM memory cell, consisting of a standard NMOS transistor and a phase change device. The key component in a phase change device is a small volume of phase change material, GST ($\text{Ge}_2\text{Sb}_2\text{Te}_5$), which can exist in either a crystalline or amorphous state. PRAM exploits the differences in the electrical resistivity of GST between the two

states to store information. The amorphous, high resistance state is used to represent a binary “0”, while the crystalline, low resistance state represents a “1”. The phase change in GST can be achieved by heating a region of phase change material to a high temperature threshold using electrical-pulse generated Joule heat [23]. Data stored in a cell is read by sensing the cell’s resistance when a small read current I_{Read} (less than $100\mu\text{A}$ [21]) passes through it. Similar to DRAM, the PRAM is hierarchically organized as sub-arrays, arrays and banks. As shown in Figure 2, a PRAM array consists of a number of cells, decoders for row/column addresses, Sense Amplifiers (S/As) and Write Drivers (W/Ds). Note that the data operation bandwidth in PRAM is smaller than that in DRAM. Unlike DRAM, the current sense amplifiers in PRAM are shared and multiplexed across bit-lines due to their large circuit size (compared to the PRAM cell array). For read and write operations, a column selector circuit selects a number of bit-lines (e.g. 64 bit-lines shown in Figure 2) and connects them to S/As for reading or W/Ds for writing. For DRAM, the entire row is activated and read into sense amplifiers and data is then taken from the sense amplifiers, selected by the column address. In the case of PRAM write operations, the write current flows from W/D to the cell ground line through column selector, bit-line, GST and access NMOS transistor. For read operations, the read current follows a similar path except that it originates from the sense amplifier.

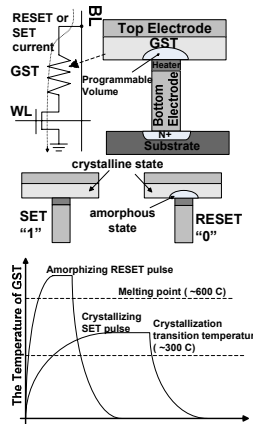


Figure 1. The basic structure of a PRAM cell

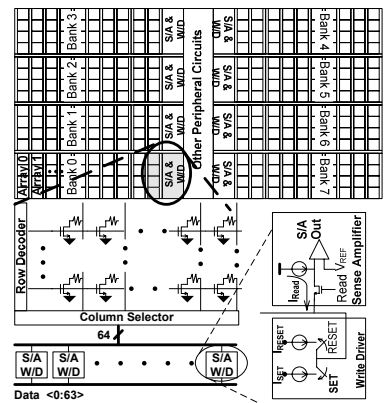


Figure 2. PRAM layout and schematic drawings of memory array, S/A and W/D

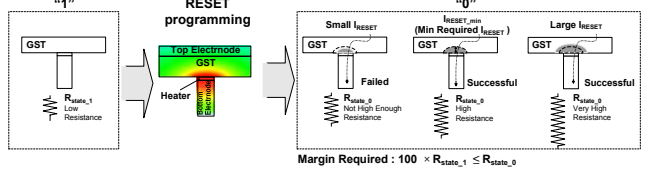


Figure 3. A minimum programming current is required for successful PRAM RESET

The most critical electrical characteristics of a PRAM cell are the required minimum RESET and SET currents ($I_{\text{RESET_min}}$ and $I_{\text{SET_min}}$). Taking RESET as an example, as shown in Figure 3, a temperature rise due to the heat generated by the RESET current is required to melt a small region of phase change material that is adjacent to the heater to achieve a successful RESET operation, which will produce a sufficiently large ratio (i.e. ≥ 100 [23]) between the resistance of RESET and SET state. A programming current that exceeds $I_{\text{RESET_min}}$ can ensure a successful programming operation. $I_{\text{RESET_min}}$ is affected by several factors including the physical dimensions of phase change devices and the parameters of current driving transistors (detailed in Section 3.1).

In the presence of process variations, $I_{\text{RESET_min}}$ varies across PRAM cells due to variability in physical dimensions, transistor gate length and threshold voltage. In conventional design, programming current has to be tailored to the worst case so that the write current satisfies the minimum requirement across all cells.

2.2 Process Variation

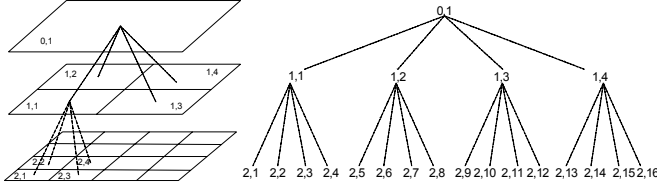


Figure 4. Multi-level quad-tree partitioning to model systematic variation Process variation is a combination of random effects and systematic effects. Random variation refers to random fluctuations in parameters from die to die (D2D) and device to device (WID). Systematic variation, on the other hand, refers to the layout-dependant variation through which nearby devices share similar parameters. Devices exhibit spatial correlations within a given die. D2D variation mainly exhibits as a random variation, whereas WID variation is composed of both random and systematic variation. For example, the gate length (L) of each transistor under process variation can be represented as:

$$L = L_{\text{nom}} + \Delta L_{D2D} + \Delta L_{WID} \quad (\text{Eq. 1})$$

where L_{nom} represents the normal value of gate length; ΔL_{D2D} and ΔL_{WID} denotes the die-to-die and within-die variations respectively. ΔL_{D2D} is shared by all transistors within a chip. In this work, we model both random and systematic effects of the PV. For ΔL_{D2D} , we generate random variables that follow a normal distribution through Monte-Carlo simulation and assign one random variable for each individual chip. ΔL_{WID} differs across transistors within a chip. We use the multiple-level quad tree [19] approach to model the correlated within-die variation effect, as is illustrated in Figure 4. With this approach, a chip is covered by several layers and each layer has a number of quadrants. Each quadrant is assigned a random variation following a normal distribution. All devices in the same quadrant share the same value. Lower-level squares represent localized spatial correlations while higher-level squares represent larger-scale spatial correlations. The systematic variation of each device within a chip is calculated by summing the random variables of the quadrants, through all the layers, to which it belongs. Therefore, the closer two devices are, the more quadrants they will share. For example, devices in quadrant (2,1) and (2,2) share random variables in quadrant (0,1) and (1,1), but devices in quadrant (2,1) and (2,16) only share quadrant (0,1). We chose an area of 16K-bit PRAM cells as the granularity of the smallest quadrant; this is sufficient to capture systematic variation. The WID variation follows a normal distribution with a standard deviation $\sigma = \sqrt{\sigma_{\text{rand}}^2 + \sigma_{\text{sys}}^2}$ where σ_{rand} and σ_{sys} represent the standard deviation for random and systematic variation respectively. In this study, we assume 45 nm process technology with $\sigma/\mu = 12\%$ for transistor length ($\sigma_{\text{rand}} = \sigma_{\text{sys}} = \sigma/\sqrt{2}$) and $\sigma V_{\text{th}}/V_{\text{th_normal}} = 10\%$ for transistor threshold voltage based on variability projections from [20].

3. Process Variations in PRAM: The Source and Impact

In this section, we first describe several PRAM design parameters that are subject to process variation and present analytical models to quantify how process variation affects $I_{\text{RESET_min}}$. We then characterize the impact of PV on PRAM power and endurance due to the variability of $I_{\text{RESET_min}}$. Due to space limitations, we only present our analysis for the RESET operation in this paper, but a similar characterization procedure has been applied to the SET operation. Our analysis shows that $I_{\text{SET_min}}$ is about 70% of $I_{\text{RESET_min}}$ (consistent with the conclusion draw from experiments in [21]). Note that the variation in $I_{\text{RESET_min}}$ will affect $I_{\text{SET_min}}$ proportionally as SET is the reverse operation of RESET.

3.1 PRAM Design Parameters subject to Process Variation

(1) Variations in Physical Dimensions of PRAM Devices

Process variation affects various PRAM physical dimension parameters (e.g. the thickness of the phase change material layer, the height of the heater, the contact size etc.), resulting in variability in PRAM electrical characteristics such as $I_{\text{RESET_min}}$. Among these, the impact of the Bottom Electrode Contact Diameter (BECD) size on $I_{\text{RESET_min}}$ has been experimentally demonstrated in [22]. The decreased BECD increases local current density and joule heating in GST, resulting in a melting down of the adjacent crystalline material more efficiently. This allows the use of a smaller $I_{\text{RESET_min}}$ (e.g. below the nominal value) to program a memory cell without causing a failure. On the contrary, a larger BECD reduces current density and joule heating, requiring a higher magnitude of current to reach the desired temperature. As the device feature size continues to scale down, it becomes increasingly difficult to fabricate PRAM cells with uniform contact size. This makes achieving a uniform programming current across all PRAM cells increasingly challenging.

Another design parameter that affects PRAM write current is the thickness of phase change material (ThickGST). Unlike BECD, the variability of ThickGST affects $I_{\text{RESET_min}}$ indirectly. When the ThickGST exhibits a positive variation (i.e. a thicker phase change material layer), the SET resistance is higher than the average because it is proportional to the thickness of GST. The RESET resistance, on the other hand, is primarily determined by the volume of crystalline material melted at RESET. Note that the PRAM read operation is performed by sensing the resistivity of the cell. The reduced SET and RESET resistance margin, if smaller than a minimum threshold, is likely to lead to a false read. To avoid this, the difference between SET and RESET resistance needs to be enlarged for correct sensing. To achieve this goal, a higher $I_{\text{RESET_min}}$ (i.e. above nominal value) is required to transform a larger volume of GST material from crystalline state to amorphous state. On the other hand, a lower $I_{\text{RESET_min}}$ is sufficient when ThickGST is reduced.

Variations in the thickness of heater (ThickHEATER), which result in variability in its resistance, also have a strong influence on $I_{\text{RESET_min}}$. As a high resistance is desirable for heat generation ($P=I^2R$), a thinner layer of heater is likely to demand a higher $I_{\text{RESET_min}}$ to reach the desired temperature for PRAM phase transition.

(2) Variation in Transistor Parameters within PRAM cells

Transistors in PRAM peripheral circuits control the amount of programming energy that can be delivered to memory cells. Both the magnitude and width of the write current is controlled by transistors (shown in Figure 5). Since PV in the transistor gate length can change the effective driving capability of these transistors, PV affects PRAM write energy by changing the magnitude of the write current. Precise control of the write energy is crucial for successful programming operations. This is because the actual resistance of a programmed PRAM device is sensitive to the write energy. Moreover, the variation in V_{th} , which affects the speed of transistors, will affect PRAM programming energy by changing the width of write current pulse. However, V_{th} does not affect the amplitude of the programming current.

3.2 Characterizing the Effect of Process Variations on PRAM Programming Current

We modeled a 2GB PRAM chip consisting of 8 banks (the layout is the same as the one shown in Figure 2) and used SPICE simulations and analytical models to quantitatively evaluate the effect of process variation on PRAM programming current I_{RESET_min} . We assume an I_{RESET_min} of 0.3mA when there is no PV. To model the impact of transistor parameter variation on the magnitude and width of PRAM programming current pulses, we built SPICE models for PRAM write electrical path. The physical MOSFET model we used is BSIM [17]. In addition, we used a one-dimensional heat conduction model [23] to quantify the effect of variation in the physical dimensions of phase change material on PRAM programming operations.

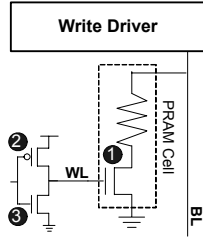


Figure 5. The electrical paths of PRAM write operations

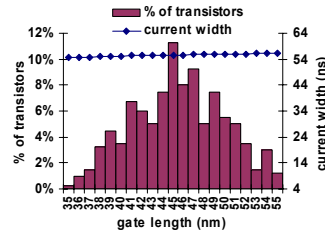


Figure 6. Variation in the width of I_{RESET_min} due to transistors' length variation

while variations between transistor 1 and transistors 2 and 3 are independent. The results presented in Figures 6 and 7 show the PV effect when a transistor parameter (e.g. gate length) of one group varies and the other remains constant. As shown in Figure 6, the widths of the write current pulses vary with the transistor gate length. The variation is around 1ns, which is only 2% of a typical current width. We also observed that the width of write current is affected by the variation of V_{th} , but the resulting variation in current width is very small, within 5%. However, the magnitude of the current shows a considerable variation as the transistor gate length varies. As shown in Figure 7, the magnitude of current pluses delivered to the PRAM cells decreases as the gate length increases because of the increased transistor on-resistance. Consequently, transistors with longer gate length may not be able to deliver sufficient current to successfully program a cell, while transistors with shorter gate length are likely to over program the PRAM cell.

To model the impact of PRAM physical dimension parameter variability on I_{RESET_min} , we use a heat conduction model [23], which links temperature to the minimal programming currents required for performing successful RESET operations. This heat conduction model captures the flow of heat in the PRAM device that is insulated everywhere except at the two ends, which connect to bit-line and access transistor. The model takes both device physical dimensions and properties of PRAM material as inputs and produces an estimated I_{RESET_min} . This allows us to quantify the impact of variation in the physical dimensions of PRAM devices on I_{RESET_min} .

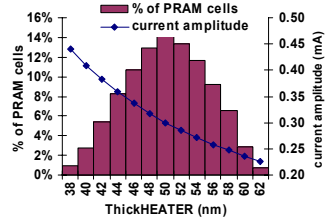


Figure 9. Variation in the amplitude of I_{RESET_min} due to ThickHEATER variation

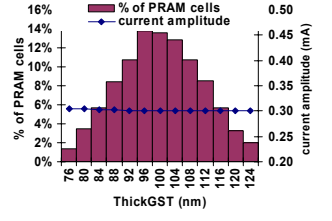


Figure 10. Variation in the amplitude of I_{RESET_min} due to ThickGST variation

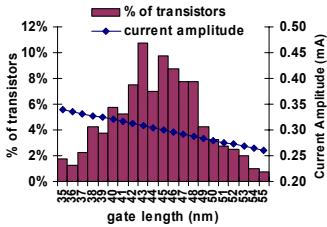


Figure 7. Variation in the amplitude of I_{RESET} delivered to PRAM cell due to transistors' length variation

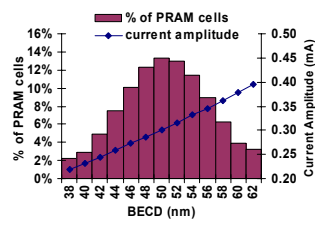


Figure 8. Variation in the amplitude of I_{RESET_min} due to BECD variation

Figure 5 illustrates the electrical path of PRAM cell write operations. Transistors that have a direct impact on the magnitude and width of the programming current are marked. Among these transistors, transistor 1 (located in PRAM cell array) determines the magnitude and transistors 2 and 3 affect the width of the programming current. As transistors 2 and 3 are physically located close to each other (both in the PRAM peripheral circuit), they are likely to share a similar variation due to correlation. Therefore, we assume variations exhibit correlation between transistors 2 and 3,

We characterize the variation in I_{RESET_min} by varying each physical dimension parameter individually. The results are shown in Figures 8, 9 and 10. These figures also plot the RESET current distribution across memory cells in the modeled PRAM chip. As can be seen, among all of the parameters, PRAM write current is the most sensitive to variation in BECD and ThickHEATER. As shown in Figure 8, the deviation of I_{RESET_min} can be as much as 32% due to the variation of BECD. Smaller contact area allows lower current and this observation is consistent with [22], which designs low power PRAM using a ring type contact to achieve a minimum programming current. Moreover, the variation in ThickHEATER can affect I_{RESET_min} by as much as 46%, shown in Figure 9. A thinner heater layer results in a higher I_{RESET_min} . This is because thermal energy generated by a low resistance Joule heater (i.e. $q_{Joule} = I_F^2 R t$) is limited. In addition, a thinner heater layer is unable to provide a high temperature environment under the GST layer. This results in a large temperature gradient between the GST layer and the material below it. Instead of concentrating on heating the GST material, the generated thermal energy dissipates. As shown in Figure 10, I_{RESET_min} is not sensitive to the

thickness of the GST layer (primarily consist of crystalline material) and the variation of $I_{\text{RESET_min}}$ is only about 1%. As discussed in Section 2, the volume of amorphous material melted by the RESET current is the primary contributor to the high RESET resistance, while the crystalline state is a very small contributor. As the thickness of the GST layer varies, the RESET resistance is affected slightly. Therefore, the resistance margin still satisfies the requirement and $I_{\text{RESET_min}}$ doesn't exhibit a substantial variation.

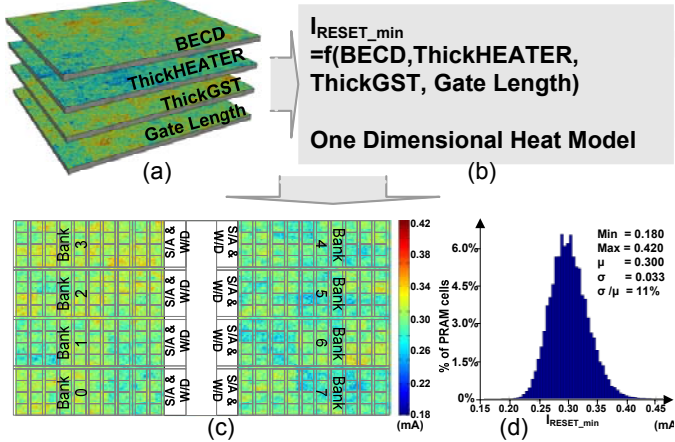


Figure 11. The color map and distribution of minimum required programming current for a PRAM chip

To quantify the aggregated impact of parameter variation on the PRAM programming current, we model each variation source using the methods described above and generate its statistical distribution for each PRAM chip, as is shown in Figure 11-a. We then use the generated PRAM physical dimension parameter statistics as inputs to the one dimensional heat conduction model to compute the $I_{\text{RESET_min}}$ for each memory cell (Figure 11-b). The variability of $I_{\text{RESET_min}}$ across the PRAM chip is shown in Figure 11-c and its histogram is shown in Figure 11-d. Note that Figure 11-c and 11-d only show the current profile for one of 400 PRAM chips that were simulated (detailed in Section 5). The PRAM RESET current distribution shows a 40% variation when the variation of all parameters are taken into account. This variation of $I_{\text{RESET_min}}$ is caused by the overall parameter variability and is smaller than the one due to varying the ThickHEATER alone. This is because the variation effect contributed by other sources is small and the overall impact of parameter variability on programming current is reduced.

3.3 PV Induced PRAM Programming Power Overhead and Reliability Degradation

Due to the variation in the PRAM design parameters, the required minimal programming current varies across the memory cells. In a conventional design, to ensure the successful programming of a PRAM cell, the current generated for write operations has to be tailored to the worst case. Consequently, the write driver functional block has to raise its voltage to a maximum level in order to deliver sufficient current for proper operation of all of the PRAM cells. This not only consumes 96% more write power (shown in Figure 18 in Section 6.1) than what is necessary, but also exposes 36% of the PRAM cells (e.g. the areas shown in red and yellow in Figure 11-c) to a much higher current than needed. Such over programming can physically degrade the endurance of

phase change material, leading to reduced reliability. The endurance test performed in [24] shows that the lifetime of a PRAM device is a function of the programming current pulse energy. The experimental data (Figure 12, obtained from [24]) shows a power law relationship between cycle lifetime and programming energy. The dependency of endurance on the magnitude of the RESET current indicates that over heating the cell with higher than necessary current leads to a reduced life cycle. As shown in Figure 11, to tolerate PV, the RESET current increases from 0.3mA (the mean value) to 0.42mA (the maximum value) due to conventional design. Assuming the energy value of 1 corresponds to the 0.3mA RESET current, the energy of RESET increases by 96% due to conventional design and this energy increase results in a 50X reduction in endurance. Note that we assume an endurance of $1E08$, as projected by ITRS 2007[25], for PRAM cells that have a nominal RESET current. The write endurance requirement for a computer system can be estimated using the following equation [3], $E = T_{\text{life}} \times B / (\alpha \times C)$ (Eq. 2), where E is the cycling endurance, T_{life} is life expectancy of the system, B is memory bandwidth, α is wear-leveling efficiency and C is the system memory capacity. Assuming a typical server with 1GB/sec bandwidth, 0.1 wear-leveling efficiency and 4GB PRAM, the estimated lifetime of PRAM memory is about 8 years, when there is no PV. This estimated lifetime can be significantly reduced, to as little as 2 months ($8 \text{ years} / 50 \approx 2 \text{ months}$), due to PV-induced over programming. PV-aware PRAM design is highly desirable as a means to overcome the abovementioned power and reliability disadvantages. Note that although the power and endurance of PRAM is affected by PV, the write latency of a PRAM cell is largely independent of PV. This is because the delay in transition between crystalline and amorphous states largely depends on the property of phase change material [26].

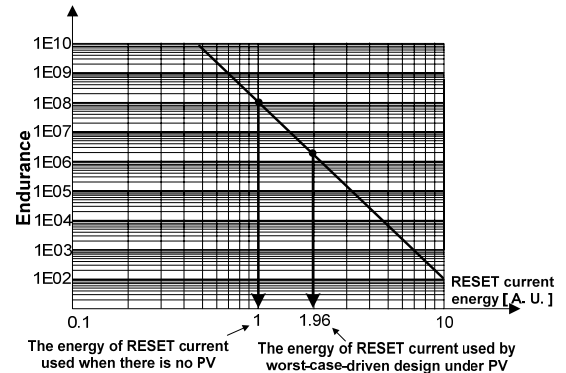


Figure 12. The cycle lifetime as a function of programming pulse energy [24]

4. Mitigating Process Variation Impact on PRAM Power and Reliability

In this Section, we propose cross-layer techniques that span circuit-, microarchitecture- and system- levels to cost-effectively mitigate the deleterious impact of PV on PRAM power and reliability.

4.1 Variation-aware Programming Current Provision

Our proposed variation-aware PRAM programming current provision technique addresses the variability issue by adaptively tuning the voltage level, which makes the programming current

adjustable. Instead of using a uniform current for programming all PRAM cells, our scheme provides multiple current magnitudes that can be chosen. The entire PRAM array is divided into multiple domains and each domain can select the lowest current magnitude that is sufficient to successfully program all cells within that domain. To achieve this goal, we use both post-fabrication tuning and run-time adaptation mechanisms.

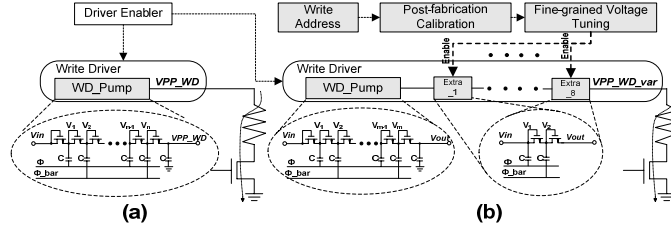


Figure 13. Fine-grained voltage tuning for write driver circuits

A typical PRAM write driver circuit, shown in Figure 13 (a), adopts Dickson Charge pump design [27] to generate a voltage level that is high enough to ensure successful programming on any cell across the entire memory. When the write enable signal is activated, the charge pump functional block starts pumping and continues until the target voltage level is reached. After that, the memory cell transistor is turned on and a write current flows through the cell, leading to the phase change in the cell. We propose breaking the charge pump down into multiple steps, as shown in Figure 13 (b). The voltage produced after the initial pump node, *WD_Pump*, is high enough to provide current for programming some of the cells. Other cells may demand higher voltage/current due to process variation. Functional blocks such as *Extra_1* and *Extra_2* can provide additional charge to further increase the voltage level at $\Delta V = \frac{C}{C + C_s} V_\phi - \frac{I_{out}}{(C + C_s) \cdot f_{osc}}$ per step

(C_s : stray capacitance, V_ϕ : the magnitude of anti-phase voltage, f_{osc} : operating frequency of charge pumps, I_{out} : driving capability). In this study, we use 8 voltage levels (detailed later in this Section) with $\Delta V = 0.4V$, which provides a current difference of 0.03mA between the two tuning levels. We assume $V_\phi = 1.1V$, $f_{osc} = 1GHz$, and $I_{out} = 20mA$. The overall delay introduced by the additional 8 charge pump stages is less than 8ns ($1 / 1GHz \times 8$ stages = 8 ns). To employ the proposed current provision technique, we rely on post-fabrication calibration to identify the minimum voltage boost required for successful write operation. This post-fabrication tuning information is then stored in a small flash memory as part of the peripheral circuit and is used to guide the run-time adaptation of voltage boosting. Given a write address, the write driver block performs a lookup in the storage containing post-fabrication (area and power overhead discussed later in this section) tuning information to find an appropriate voltage level. Then a number of charge pumps are enabled according to the level of output voltage.

An important design decision of the proposed technique is the tradeoff between tuning resolution (i.e. the number of domains) and the introduced overhead. If the tuning is very fine-grained (e.g. per bit), the circuit overhead (e.g. the memory space due to store tuning information) may offset the achieved benefit. On the other hand, if the tuning is very coarse-grained (e.g. per die); it will be unable to mitigate process variation induced power and reliability overheads. Process variability is a type of layout-

dependant variation through which nearby devices exhibit spatial correlations. Current memory design usually adopts a hierarchical organization consisting of sub-arrays, sub-banks and banks. As a result, memory cells residing in the same sub-array are likely to exhibit similar variabilities and $I_{RESET_min} / I_{SET_min}$. In this study, we chose a tuning resolution of one memory sub-array, which is 4-MB in size (sensitivity analysis results are shown in Section 6.3). This design choice requires a 1KB storage for post-fabrication tuning information with a die area overhead of 0.01mm², an additional power consumption of 0.01mW and an access latency of 2ns at 45nm technology. Another critical design choice is the voltage difference between two adjacent steps. Fine-grained voltage tuning with a large number of steps allows for a more accurate current provision. However, it will also incur significant die area overhead. This is because the charge pump is broken into many stages and this demands more circuits to implement the entire charge pump block. After performing sensitivity analysis and estimating the die area overhead, we found that a number of 8 voltage levels is sufficient to achieve the goal with a die area overhead of 0.23mm² (less than 0.1% of total PRAM chip area).

4.2 Adaptive Data Comparison Write using Page Classification

The scheme we propose in Section 4.1 allows the tuning of programming current for individual domains. However, it is incapable of reducing PV-induced power and endurance overhead on cells whose programming currents are larger than the nominal value within each domain. In this section, we further explore novel techniques to mitigate PV-induced overhead by taking advantage of PRAM's non-destructive read feature (discussed in Section 2). Our goal is to reduce the number of writes to cells that are negatively affected by PV within each domain. Unlike DRAM, a read to a PRAM cell does not destroy the stored data. Therefore, no write back to PRAM cell is needed after a read operation. PRAM writes are required to update the memory array only if a dirty cache line is evicted. A partial writes scheme is proposed in [28] to eliminate unnecessary word writes via tracking dirty data in the cache system and only writing back dirty data in the evicted cache lines to the PRAM-based memory. Note that we assume the size of each word is equal to 4B in our study.

In this paper, we propose an alternative technique (i.e. adaptive data comparison write), which can effectively eliminate redundant bit-writes to PRAM cells that require substantially higher programming due to PV. Our proposed adaptive data comparison write (ADCW) scheme takes the advantage of asymmetric power characteristics on PRAM reads and writes as well as the non-destructive PRAM read. The ADCW performs a read operation before writes and only executes write operations on bits that are different from the previously stored data. Moreover, it uses OS page level memory access characteristics to dynamically enable/disable data comparison write operations. As shown in Figure 14(a), the implementation of ADCW involves a slight modification in the PRAM peripheral circuitry and memory controller. For the memory controller, we add a new memory command *RD_WD_XOR*, which performs a bit-level comparison between the previously stored data (which is read out and stored in read latches) and the data to be written into the cells (which is stored in write FIFO). Therefore, two additional memory commands (i.e. *read* and *RD_WD_XOR*) will be generated upon a write memory transaction. The outcome of the XOR operation is

used to update *Write_Driver_Enable_Register*, which provides control for PRAM peripheral circuits to enable/disable the write operations on a per-bit basis so that only the bits whose states differ from the input bits will be updated.

Note that another implementation of data comparison write is proposed in [29] to extend PRAM lifespan. Our proposed ADCW is different with [29] in that our scheme can significantly reduce DCW-induced performance overhead in typical PRAM design scenarios. In fact, due to the increased write latency in DCW, applying DCW to every PRAM write will degrade performance by up to 23% (detailed in Section 6). [29] claims that the increased write latency does not have a significant impact on performance. The reasons for this discrepancy are twofold: (1) [29] assumes that the PRAM peripheral circuit is similar to the one used for conventional DRAM, which is capable of reading/writing the entire cache line simultaneously at a cost of considerably increased die area. With this assumption, the increased delay due to DCW for writing back a dirty cache line is the same as the latency of a single-bit read. To reduce the die area occupied by the peripheral circuit and improve the effective capacity, the majority of recently announced PRAM prototypes adopt design that has a reduced number of bit-line S/As, leading to a narrower width of parallel bit-read/write. For example, the prototype presented in [21] has 64 S/As per bank and only 64-bit read/write can be performed simultaneously. In this case, the increased delay due to DCW is 8X ($64 \times 8\text{-bit} / 64\text{-bit} = 8$) of a bit-read latency for a 64-byte cache line write back. (2) Writes can be performance-critical when the utilization of the transaction queue is high. Increasing write latency is likely to increase the probability of this queue filling up during program executions that exhibit burst write access patterns. Consequently, the following read requests will be blocked due to the full queue and result in performance penalty. In our work, we model a transaction queue of 32 entries.

Instead of applying DCW to every PRAM writes, we propose an adaptation scheme that dynamically enables/disables DCW based on OS-level page classification. Since variations are layout dependent, how the PRAM cells located at different regions are accessed by programs can have a significant impact on the PV-induced overhead. For example, when programs write frequently to the memory regions that require higher write current than the nominal value, the PV-induced overhead will be exacerbated. This motivates us to use OS paging to map hot-modified pages (e.g. pages #1000, #1001, #1002 in Figure 14(b)) to PRAM regions that are positively affected by PV and allocate PRAM regions that are negatively affected by process variation to cold-modified pages in which data is infrequently updated. Our scheme dynamically applies DCW on writes to the cold-modified pages (as shown in Figure 14(a)), which allows us to minimize PV-induced overhead while yielding minimum impact on performance. This is because the low frequency writes are unlikely to fill up the transaction queue and block the following read requests. To identify hot- or cold- modified pages, we use the modification counters to track the frequency of page updates and employ the Multi Queue Algorithm [16] for page classification. Using circuit-level design tools and a modified DRAMsim simulator, we estimate that the hardware-implemented algorithm and modification counters require a total die area of 0.35mm^2 and consume an approximate 11.1mW power. Our simulation results show that the additional power overhead induced by ADCW offsets less than 9% of power savings achieved by DCW.

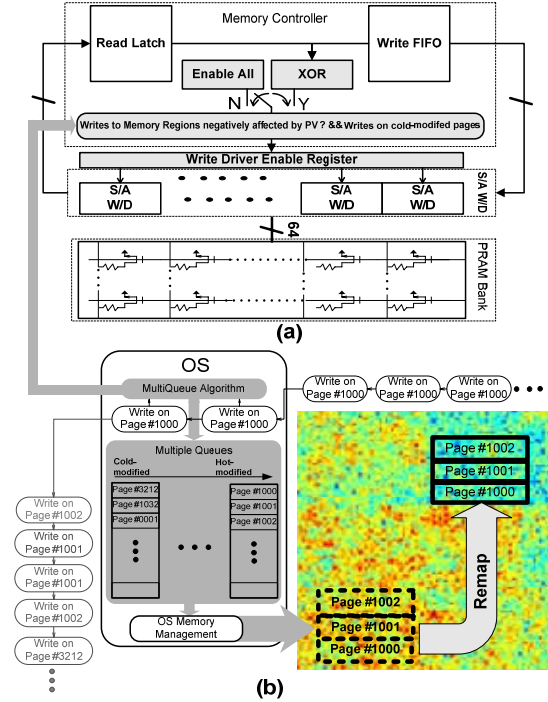


Figure 14. (a) The implementation of adaptive data comparison write (ADCW) and (b) OS-level page classification

4.3 Adaptive Memory Compression with PRAM Cell Refreshing

As mentioned earlier, partial writes [28] reduce the number of bit-writes by eliminating the unnecessary write back of clean words and can be used to mitigate PV-induced overhead. Partial writes perform well when a majority of the words in a cache line are unmodified, and its benefit diminishes as the number of dirty words increases. As a worst case scenario, no benefit is achieved by partial writes when all words in the entire cache line are dirty. By profiling the number of dirty words in the evicted cache lines across all 18 benchmarks from various benchmark suites (discussed in Section 5), we observed that on average equal to or more than 15 out of 16 words are modified in 62% of cache lines that are evicted from the last level cache (i.e. L2 cache with 64 Byte line size), as shown in Figure 15. This observation indicates that the worst case or near-worst case scenario occurs frequently.

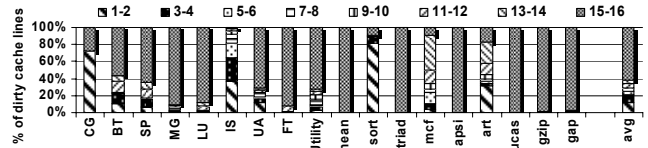


Figure 15. A profiling of the number of dirty words in the evicted cache lines

To address PV-induced overhead on writes dominated by dirty words, we developed a novel adaptive memory compression technique with cell refreshing. Our technique has two major components. The first component is OS-guided adaptive memory compression, which selectively compresses the evicted dirty cache lines, prior to writing them back to PRAM. Figure 16 illustrates the scheme we proposed. Similar to the adaptive DCW described in Section 4.2, the OS-level paging scheme is employed to perform page classification and remapping. In the memory controller, the

current provision mechanism performs a lookup in the storage containing post-fabrication tuning information upon receiving an evicted cache line. If the voltage level required for writes is smaller than the average level, it implies a write-back of dirty cache lines to a region that is positively affected by PV and we apply default partial writes on these write-backs. Otherwise, the write-back will update a memory region that is negatively affected by PV. In such cases, we selectively compress the cache line if compression outperforms partial writes in terms of bit-writes reduction. Note that compression and partial writes are mutual exclusive schemes and they cannot be applied simultaneously, because compression changes the form of data in memory and thereby requires writing back the entire compressed cache line.

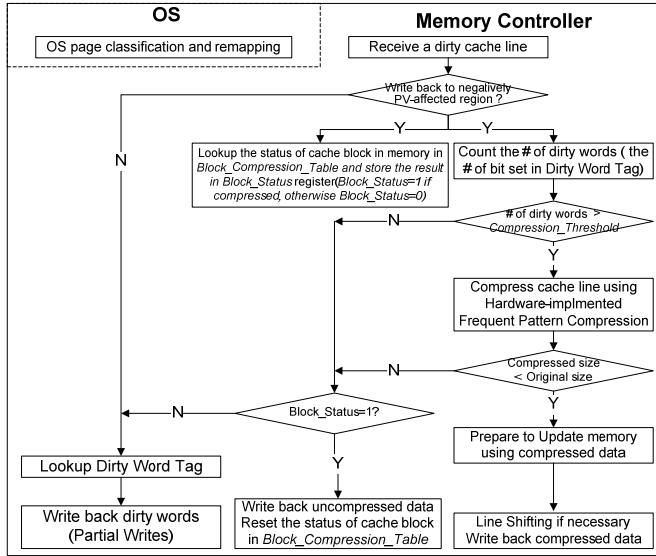


Figure 16. Adaptive memory compression

To exploit adaptive compression, we use a table, *Block_Compression_Table*, which assigns one bit for each cache line to indicate the status of data block in main memory (e.g. “1” : compressed, “0” : uncompressed). Then a lookup in this table is performed and the result is loaded to the *Block_Status* register. In parallel with *Block_Compression_Table* lookup, we determine whether the compression should be invoked by monitoring the number of dirty words present in the evicted cache line. The number of dirty words is tracked by a fine-grained dirty bit proposed in partial writes [28]. One important design parameter is the threshold of the number of dirty words that will trigger compression. Compression can either greatly reduce or considerably increase PV-induced overhead, depending on two factors: the number of dirty words and data compressibility. As compressibility can only be obtained after compression, we use the number of dirty words as an indicator to trigger the compression and update memory with compressed data only when it is smaller than the original size. We performed a sensitivity analysis and observed that choosing a threshold of 14 (e.g. *Compression_Threshold* = 14) provides the maximum benefit across all simulated workloads. When the number of dirty words is smaller than this threshold, compression is not invoked. Then only dirty words are written back if the cache line is in uncompressed form (i.e. *Block_Status* = 0), or the entire cache line (in uncompressed form) is written back if the data in memory is in compressed form (i.e. *Block_Status* = 1). When the number of dirty words reaches the threshold, compression is invoked. If the

compressed cache line has a larger size than the uncompressed one due to extra bits required by prefix, the uncompressed data are used for updating memory. The *Block_Status* register is updated and its data is written back to *Block_Compression_Table*. Otherwise, compressed data is written back to memory.

As compression is not used for improving effective memory capacity in our work, the same amount of memory space (64Byte in our case) is used to hold compressed or uncompressed data. Instead of writing compressed data starting at a cache line aligned boundary, we apply a simple line shifting mechanism to even out the writes within a PRAM block, as shown in Figure 17. The shifting is performed on a byte granularity and the first byte is reserved for storing *Head_Index* (highlighted in dark grey), which records the offset of the first byte. Each update on the memory block will result in shifting the compressed data right by one byte and *Head_Index* is incremented by one, as shown in Figure 17. The byte shifted off the right end wraps around to the second byte at the left end (byte “2B” in Figure 17-c). The power savings may be offset by the power overhead of the compression procedure. We estimate that the overall power overhead of compression/decompression only accounts for 17% of power saved due to compression mechanism. Moreover, we observed a noticeable performance improvement (detailed in Section 6.1) because the reduced number of bit-read/write due to compression can decrease the number of PRAM accesses, whose latency (i.e. 250ns) is 600X higher than compression/decompression delay (i.e. 0.41ns).

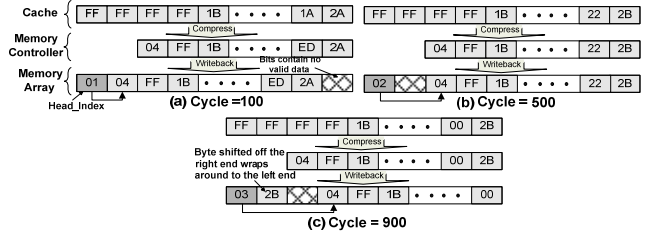


Figure 17. Line shifting mechanism

The second component of our proposed technique is a new programming approach that can refresh long-cycled PRAM cells. A recent work [31] which presents experimental evidences shows that reversing the programming current direction can refresh a degraded, long-cycled PRAM cell. The physical mechanism behind reverse programming is that field-induced atomic migration is a principle mechanism of endurance degradation. By reversing programming current, the direction of migration is reversed and therefore the endurance is extended. To identify long-cycled PRAM cells, we rely on built-in ECC as an indicator to locate memory blocks that contain degraded cells. In this study, we assume an off-chip memory bus with a width of 64-bit (8-byte) and a single-error correction code, Harming-based ECC. Upon a bit-error detected and corrected by ECC, the 8-byte aligned physical address that contains the error bit is inserted into a queue, *refresh_queue*. Before each write-back of dirty cache line, the *refresh_queue* is scanned to discover whether any address stored in the queue is covered by the write-back. If an address is found, reverse programming will be applied on the 8 bytes starting from the matched memory address. Then this address is removed from *refresh_queue*. To perform reverse programming, the memory controller sends a *reverse-programming* signal to the PRAM peripheral circuits to switch the input voltage of the charge pump from a positive voltage to a negative one, resulting in write current

flowing in the opposite direction. After that, the entire cache line is written back to memory since reverse programming destroys the data previously stored in memory.

To implement the proposed scheme, we use a DRAM-based storage for *Block_Compression_Table* and employ a hardware-implemented Frequent Pattern Compression (FPC) [30] design. For a 2GB PRAM, *Block_Compression_Table* requires a 4MB storage capacity with an area overhead of 0.2% of total die area and an additional power consumption of 8.4mW. The FPC can compress and decompress a cache line in 0.41ns [32]. The area overhead and dynamic power consumption of compressor and decompressor modules at 45nm technology are 0.183mm² and 0.273W respectively [32]. Our simulation results show an average compression ratio (i.e. compressed size/uncompressed size) between 0.47-0.98 across all simulated benchmarks, as shown in Table 3. For the reverse programming technique, we estimate an area overhead of 1% with a 0.02mW power overhead and additional 2ns delay. All of the power and performance overhead has been taken into account in our simulation results.

5. Experimental Methodology

Table 1. Baseline machine configuration

Parameter	Configuration
Frequency	3GHz
Width	4-wide fetch/decode/issue/commit
IQ	64 entries
ITLB	128 entries, 4-way
BranchPred	2K entries Gshare, 10-bit global history
BTB	2K entries, 4-way
RAS	32 entries RAS
L1 ICache	64KB, 4-way, 64 Byte/line, 2 ports, 3 cycle
ROB Size	128 entries
LDQ	48 entries
STQ	32 entries
Int ALU	4 I-ALU, 2 I-MUL/DIV, 1 Load/Store
FP ALU	2 FP-ALU, 2 FP-MUL/DIV/SQRT
DTLB	256 entries, 4-way
L1 D-Cache	64KB, 4-way, 64 Byte/line, 2 ports, 3 cycle
L2 Cache	shared 2MB, 8-way, 64 Byte/line, 12 cycle
Memory	PRAM, 2GB, 8 banks
Write Buffer	4 entries, 512B per entry

Table 2. PRAM timing and power parameters

Timing Parameters (in ns)	
tRCD	60
tCAS	12
tWR	55/250 for RESET/SET
tCMD	6
Power Parameters	
RESET current	0.3mA
SET current	0.2mA

In this section, we describe our experimental methodology for evaluating the benefits of the proposed variation-aware PRAM design. We simulate a quad-core system comprised of four out-of-order processors with the parameters given in Table 1 and a PV-affected off-chip 2GB PRAM memory. We assume a 45nm technology with a supply voltage of 1.1V. We use Monte-Carlo simulation in our variation analysis, which takes into account both die-to-die and within-die variations. A 3-level quad tree method (detailed in Section 2.3) is used to model variation correlations related to layout geometrics. We generate 400 PRAM chips and each chip has a unique write current profile obtained through Monte-Carlo simulation and one-dimensional heat model introduced in Section 3.2. We then perform a number of 400 architecture simulations to capture die-to-die variation. The characterization of PV impact on power and endurance is

performed for each individual PRAM chip. The results are reported as an average value or statistical distribution. To evaluate our techniques, we use a diverse set of memory intensive applications from various suites to cover a wide range of compressibility properties, miss rates and working set size. These applications are listed in Table 3 along with their average data compression ratio and memory reference characteristics including memory footprint size and Miss Per-Kilo Instruction (MPKI), when they run in standalone mode on a single core with a 2MB L2 Cache. We select six programs (mcf, art, apsi, lucas, gzip and gap) that exhibit the highest MPKI among all SPEC2000 benchmarks with reference inputs. We use eight programs (*IS*, *UA*, *BT*, *CG*, *FT*, *LU*, *MG* and *SP*) from NAS Parallel Benchmarks Version 3.2 with Class “C” input data set, featuring a gigabyte working set. We also choose two programs (*utility_mining* and *kmean*) from MineBench [33] with real-world data as input datasets. *Qsort* is a Unix utility and *Triad* is a streaming benchmark derived from the STREAM suite[34]. We use a random number generator to generate a sequence of 50M integers as inputs to these two benchmarks. All benchmarks are compiled on an x86 platform using GCC or FORTRAN compiler with optimization level -O3. To form 4-threaded multiprogramming workloads, we first categorize all benchmarks into: high-miss (MPKI>190), moderate-miss (190<MPKI<60), low-miss (MPKI<60) groups. In Table 4, the High-, Moderate- and Low- miss workloads (H1-H3, M1-M3, L1-L3) consist of four benchmarks exclusively from each category. The High-Moderate- and Moderate-Low miss workloads (HM1-HM3, ML1-ML3) are formed by using two benchmarks from each category. For all experiments, we use the SimPoint toolset to choose representative execution intervals for all benchmarks and perform detailed cycle-level simulation until at least 1 Billion instructions are committed on every core.

Table 3. Benchmarks used to form workloads

Benchmarks		MPKI (2MB L2)	Memory Footprint (MB)	Average Compression Ratio
NAS	<i>CG</i>	441	914	0.65
	<i>BT</i>	252	691	0.92
	<i>SP</i>	241	726	0.91
	<i>MG</i>	214	432	0.70
	<i>LU</i>	176	601	0.74
	<i>IS</i>	192	1056	0.66
	<i>UA</i>	123	486	0.55
	<i>FT</i>	120	1284	0.93
Mine Bench	<i>utility_mining</i>	93	547	0.83
	<i>kmean</i>	78	402	0.98
Unix utility	<i>qsort</i>	75	722	0.97
STREAM	<i>triad</i>	67	938	0.63
SPEC2000	<i>mcf</i>	37	98	0.47
	<i>apsi</i>	30	196	0.63
	<i>art</i>	30	4	0.65
	<i>lucas</i>	21	91	0.88
	<i>gzip</i>	17	75	0.90
	<i>gap</i>	14	192	0.62

For performance and power evaluation, we developed a framework based on a heavily extended full-system simulator, PTLSim/X[36], integrated with a modified memory model, DRAMSim[37]. PTLSim/X is a cycle accurate simulator supporting the x86 instruction set architecture. We extended PTLSim/X to model a 2-level write back cache, contention for memory buses, bus traffic, memory controller request queue (including a 4-entry write buffer to optimize write operation to PRAM) and memory compression. To model the latency and energy of PRAM memory, we enhanced the DRAMsim timing module to model PRAM-specific peripheral

structures (current sense amplifier and write driver blocks) and extended its power module for PRAM energy estimation. For the timing and power parameters, we find the range of values for a given PRAM parameter through an extensive literature search and use the median value for that parameter. The parameter values in our study are listed in Table 2 and they are similar to the ITRS 2007 projection for PRAM technologies [25]. The variations in RESET/SET currents are obtained through our analytical model and SPICE simulation using Predictive Technology Modes, known as BSIM[17] for a 45 nm technology. We assume a disk access latency of 4.2ms [38] on an IDE disk. The harmonic IPC of quad-core processor is used as our performance metric. To implement the OS paging scheme in Section 4.2, we modified the custom memory manager in PTLsim to support page migrations. This modified memory manager is responsible for maintaining TLB coherence, copying the page to its new home (we emulate this migration by invoking a `bcopy()` routine) and flashing the cache lines belonging to the pages to be migrated. The associated performance and power overhead have been taken into account in our reported simulation results.

Table 4. Workloads

High	H1(<i>CG,BT,SP,MG</i>)
	H2(<i>SP,MG,LU,IS</i>)
	H3(<i>CG,BT,LU,IS</i>)
High & Moderate	HM1(<i>CG,BT,Sort,Triad</i>)
	HM2(<i>SP,MG,Utility,kmean</i>)
	HM3(<i>LU,IS,UA,FT</i>)
Moderate	M1(<i>UA,FT,Utility,kmean</i>)
	M2(<i>Utility,kmean,sort,triad</i>)
	M3(<i>UA,FT,Sort,Triad</i>)
Moderate & Low	ML1(<i>UA,FT,gzip,gap</i>)
	ML2(<i>Utility,kmean,art,lucas</i>)
	ML3(<i>Sort,Triad,mcf,apsi</i>)
Low	L1(<i>mcf,apsi,art,lucas</i>)
	L2(<i>art,lucas,gzip,gap</i>)
	L3(<i>mcf,apsi,gzip,gap</i>)

6. Evaluation

In this section, we present the power and endurance benefit of using our proposed variation-aware PRAM design techniques. All results reported are computed as an average over the 400 simulated PRAM chips. Our experimental evaluation includes: 1) power savings achieved by variation-aware schemes and their performance overhead; 2) life span extension by adopting variation-aware techniques; 3) sensitivity analysis of tuning resolution.

6.1 Power Savings

Figure 18 shows the average power savings benefit of the proposed PV-aware PRAM optimization techniques. The bar labeled “gmean” denotes the geometric mean. We use the “NoPV” case, which is a PRAM without process variations, as our baseline. All other results are normalized with respect to the “NoPV” case. The “PV” represents the PRAM implementation that conservatively tolerates process variations based on the worst case scenario. The “CP” scheme uses the proposed process-variation aware Current Provision (CP) technique to adapt programming current.

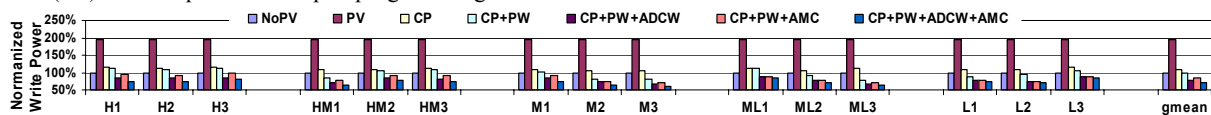


Figure 18. A comparison of write power saving

“CP+PW” is a design that combines the Current Provision with Partial Writes (PW). This “CP+PW” design can be further coupled with OS-guided DCW and OS-guided Adaptive Memory Compression (AMC) to form “CP+PW+ADCW” and “CP+PW+AMC”. The last bar, “CP+PW+ADCW+AMC” represents the case where all of the proposed techniques are applied simultaneously. As shown in Figure 18, the worst-case-scenario-driven PRAM increases write power by 96% in geometric mean. While this power overhead can be significantly reduced to 10% by PV-aware current provision. With partial writes employed, “CP+PW” achieves an additional 13% write power savings over “CP”. This additional benefit is primarily contributed by benchmarks *sort*, *FT*, *Utility*, *mcf* and *art*, which have a small number of dirty words in the evicted dirty cache lines as shown in Figure 15. Moreover, we find that a 22% and 17% write power savings can be achieved by “CP+PW+ADCW” and “CP+PW+AMC” respectively because of the fewer bit-writes on memory regions that are negatively affected by PV. Note that all mix workloads benefit more from OS-guided DCW than OS-guided adaptive memory compression, except for low miss workloads. This is because statistically DCW can remove bit-writes by 50% due to the equal possibility of writing a “0” and “1”, while compression is not as effective as DCW in bit-write reduction because of an average 74% compression ratio observed across our simulated benchmarks. For Low-Miss workloads, DCW and Compression are not invoked frequently. This is due to the fact that the total dataset resident in main memory for Low-Miss workloads is far less than the total memory capacity and therefore OS-paging can effectively re-map hot-modified pages to free memory regions that are positively affected by PV, while other workloads require the full space of main memory and limit the opportunities for page remapping. DCW can provide a larger power savings than adaptive compression, but at the expense of substantial performance degradation due to extra memory read accesses. Figure 19 shows IPC values for the different techniques. DCW incurs a 7% harmonic IPC penalty, while adaptive memory compression improves performance by 16%. Among all techniques, “CP+PW+ADCW+AMC” achieves the best tradeoff between power reduction and performance degradation. It achieves 63% power savings and 11% performance improvement that is similar to “CP+PW” over the “PV” case.

Figure 20 presents the normalized power consumption distribution in the presence of PV across the simulated 400 PRAM chips as well as a scenario after applying all proposed PV-aware optimization scheme. A power value of 1 corresponds to the power consumption without PV effect. Of PRAM chips not using PV optimization schemes, all of the chips exhibit power consumption greater than that of an ideal design, while 31% of the chip exhibit equal or greater than 2X power consumption. In contrast, with our optimization schemes, 9% of the chips consume power that is greater than an ideal design. No chip exhibits greater than 2X power consumption.

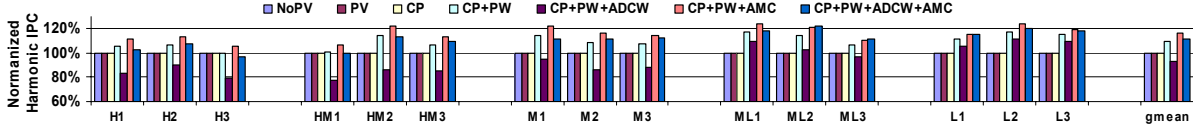


Figure 19. The performance impact of variation-aware PRAM design

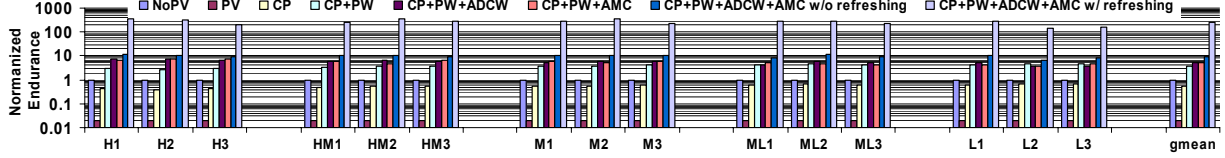


Figure 21. A comparison of endurance improvement

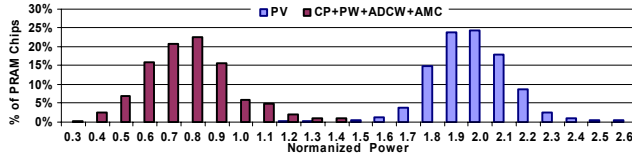


Figure 20. Power consumption distribution

6.2 Endurance Enhancement

To estimate wear-out induced failures, we run each workload repeatedly and track the number of writes to each bit. We use 1×10^5 instead of 1×10^8 writes in our lifetime estimation due to the extremely long simulation time to collect 1×10^8 writes to any PRAM cell. With our accelerated estimation method, when a bit has more than 1×10^5 writes on it, we mark it as a failed cell. We define a memory access failure when the number of failed bits accessed in a memory reference is larger than the number of errors that can be corrected (we assume ECC with single error detection/correction). The lifetime of PRAM-based memory is estimated as the number cycles elapsed before the first memory access failure occurs. Figure 21 shows the lifetime improvement achieved by our proposed techniques. All results are normalized to the lifetime of the “NoPV” case. Compared to the “NoPV” case, the endurance of PV-affected memory is substantially shortened by a factor of 50X because of the considerably large magnitude of current applied to program memory cells. By adapting the current used for writes, “CP” can significantly increase the endurance by 27X with respect to the “PV” case. Furthermore, “CP+PW” extends the lifetime by another 7X in average, relative to “CP”. The “CP+PW+ADCW” and “CP+PW+AMC” allow a geometric mean of 277X and 268X boost in endurance respectively over the “PV” case and they make the lifespan of PRAM memory system 5.5X and 5.3X better than that of “NoPV” case. Note that the lifetime extension for high miss workloads is larger than that for low miss workloads. This is because the applications in the high- and moderate- miss categories stress memory more intensively and also exhibit a large memory footprint than others, leading to a shorter lifetime on the baseline design. By combing all techniques together, “CP+PW+ADCW+AMC” has the ability of achieving 13050X (with cell refreshing) and 488X (without cell refreshing) endurance respectively over the “PV” case. Note that we assume the refresh technique is able to extend a cell lifetime by 35 times, which has been experimentally demonstrated in [37].

6.3 Sensitivity Analysis of Tuning Resolution

Although our post-fabrication tuning allows using fine-grained current control to mitigate the PV-induced overhead, this tuning optimization is sensitive to the selection of tuning resolution.

Moreover, the die area overhead and access latency required for storing and retrieving post-fabrication information should be considered. Due to the spatial effect of PV, using a very fine-grained tuning resolution may not be able to provide significant benefit, but incur more area and complexity overhead. We perform a sensitivity analysis and Figure 22 shows the power and endurance benefit of “CP” as the tuning resolution increases from one bit to one memory region of 40MB in size. Note that the results are normalized to the per-bit level tuning. We find that the benefits decrease dramatically when the tuning resolution drops below 4MB. The per-bit level tuning achieves only about 15% more power savings and 24% longer lifespan, while the die area overhead is 100% and the performance degradation is about 14% due to the longer latency in fetching post-fabrication data from a large flash storage. Thus we select a tuning resolution of 4MB that provides a good trade-off between power/endurance benefit and area/complexity overhead.

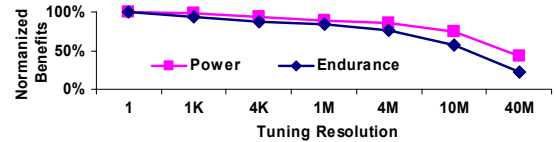


Figure 22. Sensitive analysis of tuning resolution

7. Related Work

As process variation significantly affects performance and leakage power consumption of microprocessor, there have been many techniques proposed in literature to mitigate the effect of process variation. For example, Body Biasing is widely applied to mitigate PV effect [8] and make a tradeoff between performance and power [10]. [11] proposes cycle time stealing on pipeline stages for frequency improvement. [12] adopts a novel 3T1D DRAM to implement memory for tolerating PV. [13] applies voltage interpolation and variable latency tuning techniques. However, existing work on analyzing and mitigating the impact of process variation has been largely focus on CMOS devices and circuits. In this work, we study the impact of PV on the novel phase change memory design, which is based on a new emerging memory technology.

Due to superior scalability and low power features, phase change memory is attracting increasing attention as one of the most promising technologies for next generation memory. To our knowledge, [28, 29, 39, 40] are among the first architecture level studies on using PRAM to implement main memory. [28] examines PRAM buffer organization and propose partial writes to tolerate long latency and high energy of PRAM writes. [39]

explores a hybrid PRAM/DRAM memory with the latency benefits of DRAM and capacity benefit of PRAM. [29] presents a set of techniques to extend the lifetime of PRAM-based memory, such as redundant bit-write removal, row shifting and segment swapping. Our study differs from these works in two ways: 1) we focus on process variation of PRAM, while their work doesn't take variability into consideration. 2) our proposed techniques address process variation issues at the circuit-, microarchitecture- and OS-level, whereas their schemes are built only on the architecture layers.

8. Conclusion

Phase change memory is a promising memory technology with superior scalability that has recently experienced a growing amount of research and development interest. As technology continues scaling down, PRAM is becoming increasingly susceptible to process variation due to the high density and small-size of built-in devices. In this study, we characterize the impact of PV on PRAM programming power and endurance. Our analysis shows that process variation can significantly affect the power consumption and endurance of PRAM. Methodologies addressing PV-induced issues are highly desired. We propose cross-layer PV optimizations that can effectively mitigate the impact of design parameter variations while significantly improving power and endurance. Our experimental results show that the aggregated effect of the proposed mechanisms has the capability of saving 63% power and improve lifespan by 13050X compared to the case without any optimization.

9. Acknowledgement

This work is supported in part by NSF grants 0937869, 0916384, 0845721(CAREER), 0834288, 0811611, 0720476, by SRC grants 2008-HJ-1798, 2007-RJ-1651G, by Microsoft Research Trustworthy Computing, Safe and Scalable Multi-core Computing Awards, by NASA/Florida Space Grant Consortium FSREGP Award 16296041-Y4, and by three IBM Faculty Awards.

10. References

- [1] C. Lefurgy, K. Rajamani, F. L. Rawson III, W. Felter, M. Kistler, and T. W. Keller, Energy Management for Commercial Servers, IEEE Computer, Vol. 36, 2003.
- [2] K. Kim, Technology for sub-50nm DRAM and NAND Flash Manufacturing, IEDM 2005.
- [3] C. Lam, Cell Design Considerations for Phase Change Memory as a Universal Memory, VLSI-TSA 2008.
- [4] K. Lee et al., A 90nm 1.8V 512Mb Diode-Switch PRAM with 266MB/s Read Throughput, ISSCC 2007.
- [5] Intel, STMicroelectronics Deliver Industry's First Phase Change Memory Prototypes, <http://www.intel.com/pressroom/archive/releases/20080206corp.htm>
- [6] G. Atwood and R. Bez, Current Status of the Phase Change Memory and its Future, IEDM 2003.
- [7] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, Parameter variations and impact on circuits and microarchitecture, DAC 2003.
- [8] K. Bowman, S. Duvall, and J. Meindl, Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration, Journal of Solid-State Circuits 2002.
- [9] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu, Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2002.
- [10] J. Tschanz, J. Kao, and S. Narendra, Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage, Journal of Solid-State Circuits 2002.
- [11] A. Tiwari, S. R. Sarangi, and J. Torrellas, ReCycle: Pipeline Adaptation to Tolerate Process Variation, ISCA 2007.
- [12] X. Liang, R. Canal, G.Y. Wei and D. Brooks, Process Variation Tolerant 3T1D-Based Cache Architectures, MICRO 2007.
- [13] X. Liang, G.Y. Wei, and D. Brooks, ReViVaL: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency, ISCA 2008.
- [14] M. Gill, T. Lowrey and J. Park, Ovonic Unified Memory - A High-Performance Non-volatile Memory Technology for Stand-alone Memory and Embedded Applications, ISSCC 2002.
- [15] L. Burcin, S. Ramaswamy, K. K. Hunt, J. D. Maimon, T. J. Conway, B. Li, A. Bumgarner, G. F. Michael and J. Rodgers, A 4-Mbit Non-Volatile Chalcogenide Random Access Memory, IEEE Aerospace Conference 2005.
- [16] Y. Zhou, P. M. Chen and K. Li, The Multi-Queue Replacement Algorithm for Second-Level Buffer Caches, USENIX 2001.
- [17] Y. Cao, T. Sato, D. Sylvester, M. Orshansky and C. Hu, New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design, CICC 2000.
- [18] S. Lai and T. Lowrey, OUM - A 180 nm Nonvolatile Memory Cell Element Technology for Stand-alone and Embedded Applications, IEDM 2001.
- [19] A. Agarwal, D. Blaauw, S. Sundareswaran, V. Zolotov, M. Zhou, K. Gala, and R. Panda, Path-based statistical timing analysis considering inter and intra-die correlations, TAU 2002.
- [20] A. Kahng, How much variability can designers tolerate? Design & Test of Computers 2003.
- [21] S. Kang et. al, A 0.1- μ m 1.8-V 256-Mb Phase-Change Random Access Memory (PRAM) with 66-MHz Synchronous Burst-Read Operation, IEEE Journal of Solid-State Circuits 2007.
- [22] Y.J. Song, J. H. Park, S. Y. Lee, Jae-Hyun Park, Y. N. Hwang, S. H. Lee, K.C. Ryoo, S.J. Ahn, C.W. Jeong, J.M. Shin, W.C. Jeong, K.H. Koh, G.T. Jeong, H.S. Jeong, and K.N. Kim, Advanced Ring Type Contact Technology for High Density Phase Change Memory, ESSDERC 2005.
- [23] D. H. Kang et. al., One-Dimensional Heat Conduction Model for an Electrical Phase Change Random Access Memory Device with an 8f2 Memory Cell ($f=0.15^{\circ}$ m), Journal of Applied Physics 2003.
- [24] K. Kim and S.J. Ahn, Reliability investigation for manufacturable high density PRAM, IRPS 2005.
- [25] ITRS 2007, Emerging Research Device, http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_ERD.pdf
- [26] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, Overview of candidate device technologies for storage-class memory, IBM Journal of Research and Development 2008.
- [27] Louie Pylarinos, Charge Pumps: An Overview, <http://www.eecg.utoronto.ca/~kphang/ece1371/chargepumps.pdf>
- [28] B.C. Lee, E. Ipk, D. Burger and O. Mutlu, Architecting Phase Change Memory as a Scalable DRAM Alternative, ISCA 2009.
- [29] P. Zhou, B. Zhao, J. Yang and Y. Zhang, A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology, ISCA 2009.
- [30] A. R. Alameldeen and D. A. Wood, Frequent Pattern Compression: A Significance-Based Compression Scheme for L2 Caches, Technical Report 1500, Computer Sciences Department, University of Wisconsin-Madison, 2004.
- [31] S. Lee, J. Jeong, T. Lee, W. Kim and B. Cheong, A novel programming method to refresh a long-cycled phase change memory cell, NVSMW/ICMTD 2008.
- [32] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif and C. R. Das, Performance and Power Optimization through Data Compression in Network-on-Chip Architectures, HPCA 2008.
- [33] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik and A. Choudhary, MineBench: A Benchmark Suite for Data Mining Workloads, IISWC 2006.
- [34] <http://www.cs.virginia.edu/stream/>
- [35] Micron DDR2 SDRAM 2Gb and 4Gb data sheet. <http://download.micron.com/pdf/datasheets/>
- [36] M.T. Yourst, PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator, ISPASS 2007.
- [37] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, DRAMsim: A memory-system simulator, SIGARCH Computer Architecture News 2005.
- [38] T. Kgil, D. Roberts, T. Mudge, Improving NAND Flash Based Disk Caches, ISCA 2008.
- [39] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, Scalable High Performance Main Memory System Using Phase-Change Memory Technology, ISCA 2009.
- [40] W. Zhang and T. Li, Exploring Phase Change Memory and 3D Die-Stacking for Power/Thermal Friendly, Fast and Durable Memory Architectures, PACT 2009.