Enhancing the security of memory in cloud infrastructure through in-phase change memory data randomisation

Xianzhong Zhou¹ | Ying Wang²

¹School of Information Engineering, Guangdong University of Technology, Guangzhou, China
²State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

Abstract

As a promising alternative to dynamic RAM, phase change memory (PCM) suffers from limited write endurance. Therefore, many research proposals on PCM security or reliability have focussed on the possible threat of wear-out attacks from malicious applications. However, it is also found that the non-volatile nature and the programming behaviour of PCM bring other security challenges to the memory system. The authors examine the potential risk of information leakage and theft in memory management for PCM-based cloud server or multitenant systems. By observing the influence of process variation (PV) on PCM cell programming, they propose a fast and efficient in-memory data obfuscation mechanism to defend against memory attacks or information leakage during page reallocation mandated by OS. With the capabilities of in-memory data randomisation, the proposed SecuRAM avoids the long write latency of PCM cells to erase the content, and achieves higher data initialisation efficiency than conventional software solutions. Second, the proposed SecuRAM also provides a novel solution of fast in-memory hardware fingerprinting and random number generation, which are common and essential security functions in encryption or access authentication to protect confidential memory data from attackers. Two novel techniques are proposed to generate signatures and random numbers: the first is based on partial programming, which works in the same way as bulk data randomisation; the second is loop-counting, which is an overhead-free method by reusing the cell programming mechanism in iterate-write PCM devices. Through evaluation, SecuRAM shows a much better performance and energy-efficiency than conventional measures for PCM memory.

1 | INTRODUCTION

In contrast to conventional dynamic random access memory (DRAM), phase-change memory (PCM) is considered as one of the most promising memory technologies in next-generation computing systems. Prototypes and products of PCM-based memory or storage systems have emerged recently [1]. Because of its low-power and non-volatile features, PCM-based memory or storage systems are especially attractive to cost-sensitive data centres and embedded platforms. For these systems, security is a fundamental and important design goal, which is also the concern of researchers on PCM, because PCM cells suffer from limited write endurance compared to DRAM. Typically, each memory cell can only be written a limited number of times ($10^7$–$10^9$ as reported). Lots of studies have been done to prevent malicious attacks that exploit PCM’s limited write endurance [2].

Unlike all prior work studying secure PCM, the authors here observe that the non-volatile nature and write characteristics of PCM other than cell endurance are also likely to complicate memory protection or even expose vulnerable channels to memory attacks, which is detrimental to multi-tenant systems with security requirements. They discuss one of the key security issues that threaten PCM, retention-based memory attack, and exploit the nature of PCM to deal with it in an inexpensive way. Specifically, there are some scenarios of such memory attacks that are launched against the feature of PCM memory:

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. IET Computers & Digital Techniques published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.
Page de/reallocation: In data centres, virtual machines (VMs) rentable by users are forced to share common physical memory. When non-volatile PCM is used as the main memory or large-scale storage caching, it is important to carry out security-aware space allocation as there are multiple media-sharing untrusted VMs. There will be critical information leakage if no page initialisation is conducted before real-locating it to a malicious application. However, initialisation of bulk data in PCM is far more expensive than DRAM due to its long write latency. When frequent page exchange occurs in many-VM machines, the burst of page initialisation activities in large capacity working or storage memory will severely harm system performance. Prior research [3] has already shown that OS spends a significant portion of machine time on performing data initialisation, even with DRAM main memory. When it comes to PCM memory, such operations are subject to the low write bandwidth constrained by long RESET/SET latency and power budget, thereby causing even higher overhead.

Memory attack: Non-volatility is a fascinating feature for low-power and reliable systems. However, a long retention time makes it also easier to gain access to critical data through information interception or cold-boot attacks [4]. In contrast to DRAM cells that lose their state without refreshing at a fixed rate (64 ms for DDR3 standard), PCM is able to hold the values as long as days or years, even when it is deprived of power supply, which makes it easier to recover the critical data for those with access to the memory without complex and costly encryption protection [5]. In such cases, a fast page initialisation mechanism is necessary to obfuscate the data layout in memory after usage and helps fend off physical memory attacks [5].

Therefore, how to circumvent the problem of limited write bandwidth and deliver a fast page initialisation mechanism in PCM-based memory (PRAM) is significant for system security. To achieve the design goal, the authors revisit the concept of processing in-memory [3] or so-called near-data computing [6], and leverage the intrinsic randomness of process variation (PV) in devices to realise high-throughput bulk data randomisation directly in the proposed architecture of SecuRAM.

In addition to secure memory initialisation, SecuRAM also provides another layer of protection by providing high-throughput random data and hardware signature generation. For systems with data encryption ability, true random numbers that cannot be manipulated by attackers or hardware Trojans are necessary to generate seeds for confidential data encryption. Otherwise, hardware signatures are used for authentication and prevent malicious attackers and disguisers from accessing the critical memory regions. In general, a cost-effective source of random data and hardware fingerprints is very critical for secure PCM-based systems. These authors have found that the PV-induced randomness and noises can also be exploited to generate a high volume of hardware fingerprints and random numbers. Different from prior SRAM- or DRAM-based random number generators (RNGs) [7], the authors propose a novel fingerprinting and random number generation mechanism based on the existing cell programming logics that are unique in write-iterative and other PCM devices [8].

In brief, compared to conventional DRAM, the unique characteristics of PCM not only bring in new security risks that should be aware of in system design, but also creates opportunities to improve the performance of important security primitives in system and enhance security. The above two key findings lead to a novel and simple protection mechanism for PCM, SecuRAM, which supports fast in-PCM bulk data initialisation, randomisation, and fingerprinting. Specifically, the following contributions are made herein:

- The characterised write curve of PV-affected PCM by modelling is profiled by multiple random PV sources in PCM. The authors further propose to use partial-programming pulse to conduct fast bulk data randomisation, which is fast and less power-intensive than normal RESET/SET pulses, so that information leakage and memory attacks in PCM-based systems can be prevented.
- The technology of processing in-memory is revisited and bulk data randomisation and hardware fingerprint generation directly in the PCM device within the framework of standard memory access protocol is proposed, which eliminates most of the expensive CPU-to-memory activities.
- An overhead-free fingerprinting mechanism based on the write-iterative feature of PCM devices is also proposed, in addition to partial programming that provides an active randomisation function.

Different from all prior studies focussed on endurance attacks, the authors comprehensively discuss the implications of PCM on memory security. Some of the conclusions and techniques could also be extended to other emerging non-volatile memories (NVMs) application. Section 2 presents the background and motivation, and Section 3 describes the architecture of SecuRAM. Section 4 shows the experimental mechanism, evaluation results, and analysis, and finally, Section 5 provides the conclusions.

2 | BACKGROUND AND MOTIVATION

2.1 | Programming phase change memory

Phase-change random access memory (PRAM) has the same hierarchical organisation as DRAM. As shown in Figure 1, a PRAM bank contains multiple arrays, and each array consists of a large number of cells. The address decoders are used to select the target column/rows, and read out data via SAs or program the dirty lines into the cells through write drivers (W/DS). The standby and read power of PRAM are very small, and the read latency of PRAM is comparable to current DRAM. However, to write “1”/“0” into PRAM takes a longer time and costs much higher power than that in DRAM, which is a rather expensive operation in PRAM.
Motivation: the security threats and opportunities in PCM

2.2.1 | Case 1

Suppose a cloud server is partitioned into two separate security domains: SD-1 and SD-2, which accommodate multiple tenants in different security levels. However, user applications in the two domains have an information leakage channel through the physical page allocation unit in OS unless the shared physical memory space is strictly partitioned and never released. Since most systems use a buddy system that recycles and reallocates the used pages between different applications in LRU or pseudo-LRU, VMs in separate security domains have the possibility to snoop the content of each other by stealth, if they are allocated with uninitialised pages. Therefore, erasing the data in obsolete pages is very important to block the information leakage channel. However, the traditional way to initialise the physical pages is very expensive in terms of system performance for PCM-based memory. To prove it, the authors use the full system simulator to illustrate the essentiality of the lightweight data initialisation mechanism in PCM-based system. They run eight VMs on the octo-core multiprocessor that works as the host, where the shared memory space allowed for user application is 2 GB. Details about the VM workloads and the system configuration are described in the evaluation section. Each VM runs an application from SPEC2006 and Parsec suite. In the system, frequent physical page exchanges occur between the eight VMs. Figure 2 shows the measured performance degradation of eight applications over the initialisation-free system. The performance metric used is the total stall time spent on memory access and memory initialisation, which is measured as total memory access time (TMAT). Figure 2 shows the proportion of TMAT increase induced by PCM initialisation. It is shown that page initialisation in PCM memory degrades dramatically the memory access performance of real workloads, severely hurts the service quality experienced by the users, and also causes more energy and endurance cost to the PCM because of the intensive write operations. As proved in this experiment, the traditional method for delivery of secure memory reallocation, that is straightforwardly letting CPU write zeros into main memory after page recycling, is not suitable for PCM due to its programming overhead, which is very high in terms of performance, power, and endurance cost. To contain the overhead, the authors chose to borrow the capability of processing in-memory and migrate the function of “bulk data randomisation” totally into PCM devices, so that the huge overhead of bandwidth and power can be mitigated by executing a single “memory instruction”. Typically, PCM relies on a programmable charge pump to exert deterministic different SET/RESET pulse patterns on Ge2Sb2Te5 (GST) material under the control of a finite state machine. The intuitive idea is to generate a “partial” pulse properly shaped that transforms the GST into an ambiguous state of either SET or RESET depending on the parameters of manufactured cell.

2.2.2 | Case 2

It is proved in [4] that attackers with physical or remote access to DRAMs can snoop the front side bus or directly access the memory to recover the encryption data residual and acquire critical system images in there. For non-volatile memory, things get worse since it is more practical to launch such an attack on memory modules to access the persistent information than in dynamic RAM. Therefore, bulk data randomisation or encryption is important for critical systems with non-volatile memory. Memory initialisation after use helps reduce the chances of information leakage in these scenarios. Otherwise, data encryption or access authentication can be combined as countermeasures to fend off such memory attacks. Generating reliable hardware fingerprinting is an attractive way to identify and authenticate hardware components and generate device-unique keys, especially when no cryptographic mechanism is
provided or independent keys/signatures are desired. Moreover, the fingerprinting technique can be used in authentication applications such as identifying individual nodes in sensor networks or even users. In these situations, the proposed in-memory data randomisation mechanism in SecuRAM is also able to enhance the PCM security by offering a high-throughput and robust randomness source. For example, the generated data trunk could be used as hardware fingerprints or random numbers since the cell states programmed by “partial” pulses are influenced by the unpredictable PV distribution.

3 RELATED WORK

3.1 Memory initialisation and protection

In fact, the x86 ISA recently introduced instructions to provide enhanced performance for bulk copy and initialisation (ERMSB). Such software methods cannot be directly employed in a PCM system since they induce considerable overhead as was proved in the previous section. For PCM memory, WAPTM (Write-Activity-aware Page Table Management) is proposed to add a flag bit to page table entry (PTE) to identify if a page is initialised instead of zeroing all the bits in the page [9]. However, malicious attackers still can obtain the original data in the pages. As an improvement, Write-aware Random Page Initialisation (WRPI) is proposed to reduce writing to anonymous pages by randomly flipping some bits in the pages [10]. However, it relies on a reliable random data source and cannot hide all the information from attackers. Comparative, the authors’ SecuRAM is much safer than prior work. It also significantly improves initialisation performance by utilising the random PV distribution in PCM arrays.

3.2 Security in phase change memory

In phase change memory, write-induced performance and endurance issues are the major weaknesses that threaten the reliability and security of the system. A lot of researchers focus on optimising the write operation in PCM. Lee et al. proposes to use multiple row-buffers to increase the opportunities of write coalescing and partial write in PCM for endurance improvement [11]. Many prior schemes have been proposed to reduce the overheads of state change in PRAM cells [12]. Some techniques exploit the asymmetric characteristics of RESET and SET operations for performance and power improvement [13]. To defend PCM from endurance risk, some techniques use wear-levelling or write remapping to balance the write intensity in PCM, in order to alleviate the write-induced endurance problem [14]. Zhou et al. argue that a PCM design has to consider the malicious wear-out attacks. They proposed an integrated wear-levelling mechanism with both fine-grained and coarse-grained remapping [15]. Seong et al. show the scenario that a malicious application succeeds to the detriment of a PCM main memory. Then they proposed an inexpensive dynamic randomised address mapping scheme to prevent deliberate malicious code from wearing out the weakest point in PCM [2]. Most of the work focuses on preventing wear-out attacks by means of wear-levelling or hybridising PCM with DRAM memory. The authors’ work goes beyond wear-out attacks and discusses other aspects of security threats in a PCM-based system, and it is the first work to discuss the information leakage issue in phase change memory.

3.3 RNG and hardware fingerprints physical unclonable functions

PV-induced physical unclonable functions (PUFs), which can be used to generate special hardware signature, are not new to integrated circuits. Suh et al. introduced a new PUF circuit design based on ring oscillators, and used it for device identification and key generation [16]. Researchers have also exploited the bi-stable storage devices of SRAMs or flash cells to generate unique PUF fingerprints [17]. There is also work that leverages the volatility nature of dynamic RAM to generate random numbers with the help of on-chip generated random seeds [18]. This method relies on the timing randomness of memory requests, so that it is very slow to use [19]. In addition, such random number or signature generators resort to costly circuit operations like powering on and powering off the devices. They even more need a specific circuit to convert the analogue random source or random seed with circuit states [7,20]. Che et al. proposed to use a special voltage-to-digital converter and a median-finding algorithm to digitise the entropy of PV of Memristor for PUFs [20].

The authors are the first to use the PV feature in phase change memory to generate random number or signature without using specialised entropy digitising circuits. In addition, the problem of security-aware initialisation is unique in PCM-based memory or storage. Firstly, because a high-performance mechanism is needed that is available and affordable by on-line system calls and routines, there should be an awareness of the poor write performance and endurance of PCM, and the aim should be to offer a fast but inexpensive in-memory randomisation mechanism. Secondly, the authors are the first to utilise the unique programming mechanism in normal and write-iterative PCM devices. Specifically, they propose two methods to generate random data in PCM: partial programming for common devices and loop counting for write-iterative PCM. Particularly, loop counting generates random data silently in memory without interrupting normal memory requests as it does in proactive solutions like partial programming. In Table 1, the state-of-the-art PUFs technologies in different memory is compared with partial programming (PP) and loop counting (LC) using SecuRAM. Table 1 clearly shows that both PP and LC have both short latency and low hardware overheads compared to Flash, SRAM, DRAM and NVM. On the other hand, PP has high throughput and supports page initialisation, while LC is proactive. Therefore, SecuRAM can show better performance than the other techniques.
### Table 1: Comparison of PUF technologies

| Latency | Throughput | Hardware overhead | Page initialisation | Proactive |
|---------|------------|-------------------|---------------------|-----------|
| Flash [21] | Long | High | Low | × | × |
| SRAM [7] | Medium | Low | Low | × | × |
| DRAM [22] | Medium | Low | No | × | × |
| NVM [21] | Short | Low | Medium | × | × |
| PP | Short | High | Low | ○ | ○ |
| LC | Short | Low | Low | × | ○ |

Abbreviations: DRAM, dynamic random access memory; LC, loop counting; NVM, non-volatile memory; PP, partial programming; PUF, physical unclonable functions; SRAM, static random access memory.

---

**Figure 3**

(a) Initialising a large data trunk in phase-change memory (PCM): a baseline view; (b) in-memory bulk data randomisation: a SecuRAM view

---

### 4 | THE ARCHITECTURE OF SecuRAM

#### 4.1 | The framework of SecuRAM

For comparison, Figure 3a,b illustrates the procedure of bulk data initialisation in a conventional way and with the proposed in-PCM approach, respectively. In SecuRAM, a lightweight and calculated partial programming pulse is employed to obscure the GST state at the command of processor-initiated memory instruction. Such a concept of processing-in-memory helps reduce considerable off-chip data movement compared to traditional mechanisms. As illustrated in Figure 3, it can be easily seen that the memory system benefits from two key aspects of SecuRAM:

- Instead of moving initialisation data from CPU to memory, SecuRAM moves the data generation into PCM devices to utilise the excessive internal bandwidth in devices, in order to save the precious external bus bandwidth and energy.
- A full programming pulse ($I_{SET/RESET}$) takes longer time and more energy to change the GST state, while SecuRAM is able to use a short “partial” pulse to achieve the goal.
4.2 | Randomness in PCM

There is a plethora of literature studying the influence of PV on PCM programming. It is demonstrated in [18] that there are several major contributors to the variation of programming current $I_{set,\text{sat}}$. Among these, the bottom electrode contact diameter (BECED), the thickness of mental heater ($T_{heater}$), the thickness of phase change material GST ($T_{GST}$) and the length of switch gate ($L_{gate}$), also strongly affect the programming current patterns of the manufactured PCM. These design parameters are subject to both die-to-die (D2D) and within-die (WID) variations, and eventually determine the variability of SET/RESET current of the manufactured PCM. In other words, PCM cells in the same chip are likely to manifest distinct resistance after they are injected with the same programming current pattern. Both WID and D2D variations follow a normal distribution with a standard deviation of $\sqrt{\sigma_{\text{rand}}^2 + \sigma_{\text{py}}^2}$, which is a combination of systematic and random variation deviation. To profile PV’s influence on PCM operation, the authors derive the probability distribution of the essential design parameters from [21], and use them to construct the model of resistance variation of RESET/SET operation. For simplicity, they only focus on the width of the programming pulse and assume that the amplitude is fixed.

Although the minimum pulse width needed to convert the cell into a 100% crystalline/amorphous state is tracked by this model, it is still insufficient to get a feature write curve for the modelled PCM cells. Resort is made to the programming mechanism of the PCM cell to analytically quantify the variation of the $W/R$ curve, where $W$ is the width of current and $R$ is the resistance of GST material. In prior studies, the erasing model assumed phenomenological relationships for GST’s resistance is described as [22]:

$$R_{\text{GST}} = R_{\text{SET}} + (R_{\text{RESET}} - R_{\text{SET}})(1 - C(t)) \quad (1)$$

Equation (1) describes the change of GST resistance with the growth of crystal or amorphous interface. Where $t$ is the write time, $R_{\text{set}}$ and $R_{\text{RESET}}$ are the resistance of stable RESET and SET state, respectively, the crystalline fraction parameter $C$, that indicates the proportionality between the crystalline part and the amorphous part of GST is given by [20]:

$$C(t) = \frac{A_{\text{set}}}{A} = \frac{\pi r_{\text{sat}}^2}{A} \left(1 - \exp\left(-\frac{t - t_0}{\tau}\right)\right)^2 \quad (2)$$

$$R_{\text{GST}} = \frac{R_{\text{SET}}R_{\text{RESET}}}{R_{\text{SET}} + C(t)(R_{\text{RESET}} - R_{\text{SET}})} \quad (3)$$

where $r_{\text{sat}}$ is the radius of GST material in a cylinder shape, $A_{\text{set}}$ is the cross-sectional area of crystallised GST in time $t$, $A$ is the saturate area of GST, $t_0$ is the time to shunt formation (delay time), and $\tau$ is a time constant governing the kinetics of conductive shunt growing process. The authors have deduced the variation model of $A_{\text{set}}$ and $A$ from [21], so that the reversible write curve of any cell can be derived after injecting process variation into the simulated PCM arrays through a Monte Carlo approach. Figure 4 shows the write curves of three sampled PCM cells generated by this model, which is consistent with the results in [23]. According to [23], the transformation from a crystalline to an amorphous state is reversible by controlling the Jules of heat generated by the pulse. Thus, a correctly shaped partial pulse is set to heat the GST up to a proper annealing temperature that transforms the GST into a hybrid state regardless of its initial resistance.

Besides PV, the thermal drift effect provides another level of randomness in PCM memory. The thermal drift phenomenon in amorphous GST materials has been explained as a structural relaxation (SR) process [21]. Compared to parameter fluctuation that is basically fixed after manufacture, thermal drift provides a controllable source of randomness, which can be combined with the speed of access time and expands the space of randomness in PCM for RNG.

4.3 | Partial programming: creating chaos

PCM is not a strict bi-state device, because the GST state in PCM cells can be continuously programmed. Exerting a “partial” program pulse could shift the state of GST material and push it to the direction of “crystalline” or “amorphous” depending on the initial state of the cell. However, whether the change is sufficient to flip the original state depends on the process variation of chip. Depending on the PV condition, some of the cells will be flipped but some will only experience a change unidentifiable by the read circuit, if only a “partial” pulse is properly chosen. Since process variation is a common phenomenon that is relevant to chip yield in nanotechnologies, typically a conservative SET/RESET pulse with sufficient margin is generated by the write driver in order to guarantee the phase change of the worst-case PCM cell. Compared to the worst-case RESET/SET pulse, a “partial” pulse of low amplitude can also flip many PCM cells and cause sufficiently chaotic data maps to obscure the critical pages before assigning them to an untrusted VM. Such a claim will be proved by this derived PV model. Compared to a full RESET/SET pulse, a
“partial” programming pulse is able to achieve two goals with one operation. First, a “partial” pulse has a lower amplitude and width, so it consumes less power and cycles to fulfill the operation, which increases the programming throughput dramatically under the tight power cap. Second, a “partial” pulse is able to change the GST of the PCM cell into an intermediate state, which is possibly identified as “1” or “0” by the current sense amplifier in the read circuit, and its result depends on the unpredictable distribution of process variation in memory arrays.

4.4 The mechanism of fast randomisation in PCM

Memory is accessed and controlled as a slave unit by the memory controller through bus. According to the standard DDRx protocol, a typical read or write operation is composed of serialised commands such as activate, pre-charge, column read, I/O gating and so on. The proposed new operation of bulk data initialisation initiated by SecuRAM can be easily supported by these commands. For example, randomising one memory row takes the following consecutive steps:

1. Using the row address to latch the target row into row-buffers.
2. Decoding column address to select a block, and partial-programming the cells.
3. Selecting the next column in the row and repeating partial-program until the whole row has been evacuated by the command.

Setting continuous columns in a row needs the designation of burst signal that selects a column at a pre-determined interval. For example, in DDR3 protocol, an 8 burst is needed to transmit the 64 kB block through the 64-bit bus. The mode register in the DRAM device even allows it to have programmable CAS latency, burst length, and burst order. However, the command of partial-programming that controls the write driver inside the PCM chip through the cmd bus is not supported in the current protocol. Fortunately, there are reserved codes in the cmd bus for PCM that can be used to convey the command of partial-programming (cmd_rand). For example, the deprecated command of REFRESH is not necessary for non-volatile PCM memory, so it can be replaced by cmd_rand.

4.4.1 ISA support

To enable the software to have control over the bulk randomisation circuit, a new instruction is introduced to the ISA: memblott. Note that similar memory instructions are already present in some of the instructions sets in modern processors, for example, rep movsd, rep stosb, ermsb in x86 and movl in IBM S/390. The new memory instruction is embedded into system call deallocate or MMU, so that in-PCM bulk data randomisation call is exposed to a high level of system and applications.

4.5 Hardware fingerprinting with partial programming

Generating a large volume of hardware signatures directly in memory is useful as security primitives in a system. With SecuRAM, the memblott instruction is reusable as a fingerprint generation primitive since the unpredictable distribution of process variation provides a good source of randomness. Besides, such a mechanism can also guarantee the three important features of hardware fingerprints, which are also described in the experimental section:

1. Uniqueness: PCM cells manufactured in nanotechnology and high density tend to possess different write and read curves from one another [21], which means fingerprints generated in different memory regions are prone to have different values.
2. Robustness: Robustness means that fingerprints from the same memory block are robust over multiple measurements, so it can be easily distinguished from fingerprints of a different chip or block.
3. Security: An attacker could attempt to store the fingerprints of a PCM module and replay the fingerprints to fake a memory in trust. However, if the attacker cannot predict which block or word will be fingerprinted, it is impossible to use brute force to hack the gigabytes random sources.

4.6 Random data extraction from write-iterate MLC devices

Although it is very efficient to use a memory instruction to generate random data, it still needs to occupy the memory time and bandwidth at processor run-time. For some PCM devices with an iterative-write mechanism, the authors propose a lightweight method to generate random data silently without the performance overhead of partial programming. To tolerate process variation, some multi-level cell (MLC) or single-level cell (SLC) PCM devices use iterative writes to achieve accurate cell programming. According to the flow depicted in Figure 5a, write-iterative devices employ multiple rounds of short pulses to program the cell instead of a constant long pulse. Figure 5b shows an optimised write-iterative circuit that eliminates the expensive cell reading operation employed in the previous method [8]. Instead, a fixed number of short pulses are enabled by the written bit (WDQ) to transform the state of cells until the written bit equals the output of the cell (when $V_{\text{cmp}}$ is reset). Hereafter, the overhead of reading out the cell value is removed for comparison. In the write driver, each round of short pulse will be followed by a read-after-write in the verify-read circuit to generate the WD control signals. In the verify-read circuit as shown in Figure 5c, WDQ that carries the
written value will be compared to the read value to verify if the cell is correctly SET/RESET, which is also called write-and-verify. In this procedure, PbL and PbH are respectively set depending on WDQ. After that, the sense amplifier (SA) generates the comparison result, $V_{cmp}$, which enables the write driver to generate incremental SET/RESET pulses. At the same time, WDQ is also carried to the write driver (WD) circuit to enable cell programming. Such a procedure ensures that each individual cell is precisely programmed according to its parameter variation, avoiding the worst-case SET/RESET pattern for all cells. Therefore, the round it takes to SET/RESET a PCM cell depends strictly on its parameter variation, which makes a perfect source of randomness. In this case, it is only needed to find the actual number of rounds it takes to fully set or reset the cell and keep it in the loop counter added in the verify-read circuits as illustrated in Figure 5c.

In the authors’ proposal, $V_{cmp}$ is used to increment the loop counter that records how many loops it takes to set or reset the written cell. For each SA, there is a counter keeping the counts it takes to reset the cell bit. The address is compared to guarantee that only the designated block is used as a source of randomness. Because of the content of write-back data issued by real applications, it might take multiple write operations to figure out the loop counts of all bits for the block that is used to generate random data. The tag and DCW bit ensure that the loop counts of cells are only recorded once. When all the loop counts of bits in block have been recorded for one time, the AND result of all tags becomes “1” and reset tag registers, indicating that the loop count is ready. The loop count for the written block is generated whenever the program issues write operations to the selected regions of PCM memory. Therefore, its generation is a byproduct of normal write operations issued by applications, which induces no performance overhead.

The size of the loop counter is the same with a data block designated for random data generation if the loop counts of all bits in a block are directly used as a data source. On the other hand, the counter can be associated with multiple predetermined blocks, so that it generates a set of bits as a random combination of loop counts belonging to different blocks. Suppose that there are only 512 3-bit counters associated with an arbitrary 512-bit PCM block. They are able to record the SET loop counts of all cells in that block and use it as a deterministic signature. However, if 512 counters are used to keep the loop counts of a larger PCM block, that is 4 MB, the counters will saturate at any time when the “ready” signals are set and only keep the loop counts of random 512 cells in that block depending on the actual stream of write-back data issued to the memory devices. Therefore, changing the number of loop counters and the profiled PCM trunk will change the

---

**Figure 5** The workflow of iterative-write and key circuits in write-iterative device. (a) The procedure of iterative write, (b) Iterative write driver and (c) Verify-read and loop-keeping circuit.
characteristics of random data streams. It is reconfigured to fit the requirement of generating signatures and random numbers with write-iterative devices. Hardware signature should be robust and consistent, so it must loyally record the loop counts of all cells within the same block instead of mixed loop counts from different blocks, to generate a block of consistent signature (256 or 512 counters). Thus, the number of loop counters has to be the same or close to the number of cells in the profiled PCM block. However, generating random data does not put restrictions on whether a loop counter profiles the cells within a whole PCM block or mixed cells from multiple arbitrary blocks. Compared to partial programming, loop counting serves as a light-weight signature or random number generator with negligible performance overheads, but it cannot generate high-throughput streams due to the limited width of loop counters in devices.

4.7 Overhead analysis

- **Partial programming implementing**: In-PCM bulk data randomisation involves almost no hardware overheads. Partial programming is naturally supported in PCM, because the write driver inside PCM devices generally uses a charge pump to output multiple pulse patterns by controlling their width and amplitude. Particularly for those memory devices using a program-and-verify programming mode that takes multiple iterations of small pulses to change one bit, it is easy to deploy partial write by constraining the number of iterations through the program counter [24]. Besides, memblott are created from existing micro-operations and protocol strobes in PCM. It only needs to extends the protocol and expose it to the upper layer of the system through ISA extension, but induces no hardware overhead.

- **Iteration counting**: Iteration counting records how many rounds it takes to set/reset the cells and uses it as a randomness or signature source, which occurs simultaneously with the write operations issued by normal workloads. Thus it induces no performance overheads. The hardware overhead it entails is the block-size counters that keep the iteration count per cell, the AND gates and tag registers. The counter is not necessarily the same size as the block designated for random data generation. In evaluation, it is assumed that 1024 3-bits counters are used to suit the width of PCM arrays. According to the authors’ synthesis results with Design Compiler, all the additional hardware adds up to less than 23K transistors, which amounts to less than 0.01% of the area of the 4 gigabyte PCM implemented in 45 nm technology as simulated with NVsim [25].

5 EVALUATION

5.1 Experimental setups

The simulation framework includes the full-system simulator Sniper-6.0 [23] and the authors’ integrated DRAMsim2.0 [26] that are modified to include the model of PCM-based main memory [23, 26]. The key timing and power parameters about PCM are obtained from NVSim [27]. The detailed parameters of the PCM can be found in Table 2. The authors’ simulator models the entire system, processor cores, L1 cache, L2 caches, network on chip (NoC), and PCM main memory cube. To faithfully simulate the state-of-the-art PCM memory using token-based power budgeting [28], it is assumed that at most 2048 PCM cells can be programmed due to the power delivery capability constraint in the evaluated PCM. By weighing the effects of value obfuscation according to the model in [24], the authors chose to use the partial write pattern of 75 ns in experiments if not otherwise specified, which is 60% shorter than the normal RESET pulse, so that the throughput of partial programming is significantly higher than normal write within the power cap.

Table 3 describes the constructed workloads mixed with benchmarks from SPEC2006 and Parsec suite. Each of the benchmarks represents an independent VM running on the multi-core single-chip cloud computer. Each of the VMs is mapped onto one consolidated core and shares the other resources such as the network and main memory controller.

| TABLE 2 | Network, processor, memory hierarchy configurations |
|---|---|
| Processor | In-order, Two Issues, 1.5 GHz, 8 Cores |
| L1 cache | 1-cache and D-cache, 32 kB per core, 64B line, 1-cycle hit |
| L2 cache | Private, 1 MB per core, 4-way pseudo-LRU, 32B line, 6 cycle hit |
| Base memory hierarchy parameters |
| Memory controller | Scheduling reads first, issuing writes when there is no read, assuming write burst when write queue is full |
| Memory | 4G PRAM, 260 cycles access, 45 nm process [21], 1 channel, 2 ranks/channel, 4 banks/rank |

| TABLE 3 | Multiprogrammed workloads |
|---|---|
| Mix-0 | mcf, Radiosity, FFT |
| Mix-1 | barnes, Radiosity, water-n |
| Mix-2 | Ocean-c, FFT, LU |
| Mix-3 | gcc, Ocean-c, cholesky, FFT |
| Mix-4 | radix, cholesky, gcc, LU |
| Mix-5 | gzip, gcc, ocean-n, LU, barnes |
| Mix-6 | twolf, gzip, LU, ocean-n, cholesky |
| Mix-7 | sjeng, gobmk, asar, braves, sphinx, hminer |
| Mix-8 | mcf, gcc, radiosity, radix, FFT, ocean-n |
| Mix-9 | raytrace, gzip, gcc, ocean-c, LU, water-s, barnes |
5.2 | Performance and power statistics

Figure 6a shows the system performance of bulk data initialisation in the simulated full system, which is measured in processor cycles. The baseline shows the performance of a system-level page deallocation and initialisation method that relies on successive store instructions to “zero” all the memory bits. PartialS shows the performance of software zeroing also, but it uses a partial programming pulse instead of full RESET operations. SecuRAM uses both partial pulse and in-memory instruction of memblott to fulfil the initialisation task. It is shown that SecuRAM initialisation outperforms the baseline by almost 2.3 times. In contrast, the average performance of PartialS is about 15% higher than that of the baseline.

Figure 6b compares the energy consumption of the evaluated schemes. SecuRAM consumes only 24% of baseline energy on average, whilst PartialS drains less than 49% of baseline power on average. The saved energy partly comes from the shortened system operation time. In the same time, the reduced memory activities including repetitive read-and-write command, bus transmission and I/O performance, which are all eliminated by in-PCM operation and can also reduce power consumption. Also shown is the performance and energy overhead of DRAM memory that replaces the PCM module in baseline in Figure 6a,b, respectively.

The authors also use the full-system simulator to evaluate SecuRAM in the PCM-based cloud processor. Table 2 describes the basic configuration of the multi-core processor system, while Table 3 describes the workload mixes as VMs running on the multi-core chip. In experiments, we select the benchmarks, then change the number of applications and combine them to generate 10 mixes of VMs as the target workloads. The VMs in each mix are randomly mapped onto the 8t core multiprocessor that works as the host. The VMs in a mix are forced to share the 4 GB memory in user space. Frequently, physical page exchanges and page initialisation occur between the concurrent VMs. Figure 6c shows a comparison of the total execution time of completing the same task by a set of mixed VMs with different initialisation schemes, which mimic the scenario of hosting multiple VMs of different users with one over-committed machine. The total execution time of the workload is normalised to the initialisation-free cases. In Figure 6c, SecuRAM shows the performance improvement of workloads when the software data initialisation method is used to erase the data, which reduces up to 7.52% execution time overhead. Comparatively, SecuRAM runs 4.3% faster than baseline on average and it delivers better service quality to users. Currently, there are many PIM (processing in memory) architectures for bulk bitwise operations, such as Pinatubo (Processing In Non-volatile memory ArchiTecture for bUlK Bitwise Operations) for NVM [29] or Ambit (Accelerator-in-Memory for bulk bitwise operations) [30], that could be used for page initialisation just by simple bulk bit-wise operations and offer rather competitive performance.

Compared with these PIM architectures, SecuRAM can offer unique security as well as performance improvement. For example, SecuRAM could initialise the page data without moving data, so that SecuRAM could save some time and energy.

5.3 | Sensitivity study

Figure 6(d) shows the performance of parallelised initialisation. Modern processor scales by integrating more cores, whilst the performance of SecuRAM is able to scale with the memory-level parallelism defined by the rank/bank/channel number. More ranks/banks means more independent write drivers as well as stronger randomisation capability. Four-bank SecuRAM are able to generate four streams of randomisation data in parallel if the pages to be obfuscated are evenly
interleaved across multiple banks. Figure 6(d) shows that SecuRAM of multiple banks and ranks are at least 2 times faster than the baseline when they process the same amount of data blocks (12 MB). However, when rank/bank increases to a certain degree, the performance of both PartialS and SecuRAM seems to level off, because the concurrency is limited by the fixed power delivery capability in the PCM package.

Multicore processors are hitting the “memory wall” because the scaling speed of memory bandwidth lags behind that of the computation capability. Fortunately, SecuRAM is able to save precious bandwidth by migrating the task from processor to memory. Figure 6c shows the memory traffic saved by SecuRAM. Compared to the baseline, SecuRAM eliminates more than 30% of the memory traffic on average that is harvested and utilised by the concurrent bandwidth-demanding workloads.

### 5.4 | Partial programming: in-PCM fingerprinting

Generating a large volume of hardware signatures directly in memory is useful as security primitives in a system. The signatures could be delivered from memory to storage via DMA or transmitted to the network via the network interface efficiently, because in-memory fingerprinting eliminates the data moving across the deep processor-to-memory hierarchy. With SecuRAM, memblot instruction is reusable as a fingerprint generation primitive since the unpredictable distribution of process variation provides a good source of randomness. Besides, such a mechanism can also guarantee the three important features of hardware fingerprints, which are also described in the experimental section.

#### 5.4.1 | Uniqueness

Hardware fingerprint should be unique, which means that fingerprints from different PCM devices or different locations of the same device must be different. Therefore, the correlation coefficient should be low, which is calculated using the defined Pearson correlation coefficient [17]. To test the uniqueness, the authors compared the fingerprints of each block to the fingerprints of the same address PCM block on different chips, and recorded their correlation coefficients. A total of 6240 pairs were compared in the experiment, where (24 chips choose 2) × 24 blocks measurements are taken. The correlation coefficients are very low, with an average of about 7%.

#### 5.4.2 | Robustness

The major concern about the robustness of the PCM signature is the dynamic noises, particularly the thermal fluctuation, that may change the state of initialised cells. It has been proved that the most critical dynamic noise is thermal fluctuation which could gradually shift the resistance of GST material. The thermal drift phenomenon in amorphous GST material has been explained as a structural relaxation (SR) process [21]. SR is depicted as a thermally activated spatial rearrangement of the atomic structure occurring shortly after programming in the presence of temperature disturbance. The characterisation methods are mostly based on an empirical model, in which the resistance of amorphous chalcogenide follows a power-law increase over time [25, 31].

\[
R(t) = R(t_0) \left( \frac{t}{t_0} \right)^\nu
\]

where \(R(t)\) denotes the resistance at time \(t\), \(R(t_0)\) is the initial resistance at time \(t_0\), and \(\nu\) is the drift coefficient. \(\nu\) varies with the temperature and the size of the amorphous fraction in the GST material, that is \(C(t)\) and temperature \(T\), which are unpredictable values, as is the access time \(t\) [32]. The authors adopt a power law empirical resistance model for the thermal drift effects [25, 31–33]. Then they set the memory data lifetime observed in the workloads randomly selected from the benchmark suites. Afterwards, they resample the value of signatures generated from the same cells (1000) for many times (100). Each of 100 samples is profiled after a random drift duration after partial programming. The results show that the average percentages of flipped bits for the same block (512-bit) after varied durations (100–1000 s) of thermal drift are less than 0.87%. The drift is negligible for the legitimate operating range of bit sensing in PCM. After all, there is enough data redundancy due to the large capacity of PRAM, which could be used to enhance the robustness through existing techniques such as correction coding, code-offset techniques or compression [33].

#### 5.4.3 | Security

Since the information in PCM and other related NVM will not be lost even after the power is off, one can physically access the memory system and just scan the memory content. Then an attacker can extract information from the memory system whatever they want [34]. An attacker could attempt to store the fingerprints of a PCM module and replay the fingerprint to fake a memory in trust. However, if the attacker cannot predict which block or word will be fingerprinted, they have to track all cells of all devices to ensure success. Characterising memory arrays of multiple gigabytes size will take a lot of effort, even if the attacker can physically access the memory systems, not to mention the difficulty in retrieving the shape of the partial-program pulse. Compared to the other works on in-memory encryption based on logic operation capability of NVMs, such as AIM (AES [Advanced Encryption Standard] In-Memory) [35], SecuRAM could offer a unique fingerprint generated by process variation that is different from chip to chip.
5.4.4 | Evaluating loop-counting

Similarly, the authors use the same method to evaluate the uniqueness of an LC-based signature generator. In this experiment, they assume there are 1K loop counters that can be flexibly mapped into a 1 kB PCM block for profiling. They measure the 1 kB cells of the same address in 12,480 PCM devices generated when the memory traces of workloads mix-0 to mix-9 are input to DRAMsim-2.0 simulator in sequence. In the simulator, the loop counters of the same 1K region in the 1280 devices are compared as their fingerprints. The coefficients are even smaller than is exhibited with partial programming, which is a good sign of uniqueness. Loop counting is more robust to dynamic factors than partial programming, because the generated signatures are directly stored in the counter implemented in CMOS technology, which almost never changes to the factor of thermal drift.

To test the robustness with different workloads, the authors compared the same 4 kB block’s loop counts generated under 10 workloads. The histogram of results for all blocks is shown in Figure 7a. The correlation coefficient for fingerprints from the same block under different workloads is very high, with an average of 0.9873. The minimum observed coefficient is 0.9622. The results show that fingerprints from the same page are robust over multiple measurements, and can be easily distinguished from fingerprints of a different chip or page. For use in an authentication scheme, one could set a threshold correlation coefficient $t$. When comparing two fingerprints, their correlation coefficient is above $t$, and then the two fingerprints are considered to come from the same page/chip. If their correlation coefficient is below $t$, then the fingerprints are assumed to be from different blocks/devices.

To use the loop counts as the random number, the authors input the memory traces of 10 workloads to the PCM memory simulator and record the iteration counts of all rows touched by the traces. Generally, entropy is a measure of random bits that are produced by a random source. To generate good random bits, it is needed to know how much entropy the output stream from the PCM memory contains. To do so, the loop counts of 4096 rows of cells have been recorded. The authors use the PV model to generate 1024 devices with 4096 rows. Each row contains 4K bytes, and its loop counts are kept in 64 counters, which means the counters only record the loop counts of 64 random cells in the page. Then the recorded LC sequences are compared between two randomly picked rows. On average, the number of different bits between pages for the sampled devices was calculated to be 9.3. Therefore, the average difference was calculated to be 9.3 bits per 4 kB block. For the required 512-bit of entropy according to the NIST recommendations, blocks of 256 kB should be hashed into one 512-bit digest as a random output. Generating the random bits with 10 workloads leads to energy dissipation in the loop counters and the relevant logics. Figure 7(b) shows the relative power overhead caused by loop keeping, which is normalised to the total memory power dissipated in the PCM module. The simulator shows that the power overhead of loop counters measures less than 0.0245%.

5.5 | Discussion on the granularity of bulky data randomisation

A physical page in the system is often consistent with a row inside the main memory device, which is at the size of 4 kB to 16 kB. When a read/write command is sent to the main memory, the address signal is used to index the row first and then select the column to get the target block. Considering the process of memory read/write, when cmd_rand is issued to PCM and activates the target, a whole row of PCM cells is activated and the blocks within the row could be serially programmed by an increment in the column offsets in the internal counter without the interference of the requesting processor. Therefore, the authors chose the granularity of 4 kB to be consistent with a page size. SecuRAM also supports reconfiguring the granularity of memblot operation through the burst counter added to the FSM controller. The column burst counter is controllable by the state machine and incremented in every interval of Twrite. Such a counter controls the burst length just like the burst length register in typical DRAM

![Figure 7](image-url)
memory that supports adaptive burst [36]. Intuitively, a coarser granularity is more beneficial for the sake of startup cost. Therefore, a larger initialisation size should be offered that spans multiple rows in PCM. In this experiment, the varying performance of memblott instruction using different operation granularity when SecuRAM processes bulky data is shown. The bulk data is 64 MB in total. The performance of 64 MB data initialisation is measured in execution time normalised to the result of baseline. In Figure 7c, there is an obvious trend that the memory system performance benefits more from the coarse-grained initialisation even when the total data trunk size is comparatively small.

5.6 | Applicability to MLC PCM

SecuRAM architecture is also compatible with MLC PCM, though the authors use SLC PCM as an illustration of the techniques in experiments. Actually, the effects of state obfuscation and partial programming in MLC will be even better than SLC PCM demonstrated herein. Comparatively, the MLC cell has to support more legitimate states than two, so the probability of state obfuscation between two adjacent states is higher than the same dynamic resistance range of a cell which has to represent and hold more values than an SLC cell. For example, a single MLC cell has to accurately change between four resistant states to represent “00”, “01”, “10” and “11”, and often takes more than one type of pulse. If a proper partial program pulse is selected to replace any of the valid pulses in a write driver, middle states between any of the four legitimate states and also random data can be generated in the same way.

As to loop counting, most MLC devices assume write-and-verify to defy the impacts of process variation on read/write reliability, so they can be easily extended to support loop counting. For MLC devices, the random numbers generated by an MLC cell have a higher bit-width than that in SLC cells, because an MLC cell has a wider dynamic range and needs additional rounds of iteration to complete SET/RESET. Therefore, the throughput of random data generation in MLC is to be increased over SLC.

6 | CONCLUSIONS

Researchers have been studying PCM optimisation to make it a more secure main memory for servers or embedded systems. Different from all prior work focussed on the wear-out attacks, the authors go beyond to investigate the other potential security threats in PCM memory, and also the opportunities in PCM-based system to enhance system security. It is shown that the proposed architecture, SecuRAM, succeeds to exploit the characteristics of PCM and the capability of processing in-memory to deliver security-aware bulk data initialisation in the computing system. The proposed architecture-level solution of in-PCM randomisation combined with partial programming, can boost the performance of memory obfuscation. In addition, SecuRAM provides a very efficient in-memory solution of high-throughput hardware fingerprinting and random number generation for PCM-based systems. For iterative-write, loop counting makes a small-footprint but overhead-free random data source as an efficient complement to partial programming.

ACKNOWLEDGEMENT

Xianzhong Zhou is grateful for financial support from NSFC (Grant No. 61704032) and Open Fund of State Key Laboratory of Computer Architecture (Grant No. CARCH201814).

ORCID

Xianzhong Zhou @ https://orcid.org/0000-0003-4311-7287

REFERENCES

1. Kim, H., et al.: Evaluating phase change memory for enterprise storage systems: a study of caching and tiering approaches. In: Proceedings of the 12th USENIX Conference on File and Storage Technologies. FAST ’14. (USA: USENIX Association. 33–45 (2014)

2. Seong, N.H., Woo, D.H., Lee, H.H.S.: Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In: Proceedings of the 37th Annual International Symposium on Computer Architecture. ISCA ’10. New York: Association for Computing Machinery, 383–394 (2010)

3. Patterson, D., et al.: A case for intelligent ram. IEEE Micro. 17(2), 34–44 (1997)

4. Halderman, J.A., et al.: Lest we remember: old-boot attacks on encryption keys. Commun. ACM. 52(5), 91–98 (2009)

5. Chhabra S., Solihin Y.: I-NVMM: a secure non-volatile main memory system with incremental encryption. Proceedings of the 38th annual international symposium on Computer architecture (ISCA ’11) pp. 177–188. Association for Computing Machinery, New York, NY (2011)

6. Balasubramonian, R., et al.: Near-data processing: insights from a micro-46 workshop. IEEE Micro. 34(4), 36–42 (2014)

7. Holcomb, D.E., Burleson, W.P., Fu, K.: Power-up state as an identifying fingerprint and source of true random numbers. IEEE Trans. Comput. 58(9), 1198–1210 (2009)

8. Lee, K., et al.: A 90nm 1.8v 512mb diode-switch ram with 266m/s read throughput. In: 2007 IEEE International Solid-State Circuits Conference pp. 472–616. Digest of Technical Papers, San Francisco (2007)

9. Liu D., Wang Y., Shao Z. Towards Write-Activity-Aware Page Table Management for Non-volatile Main Memories. ACM Transactions on Embedded Computing Systems. 14(2), 1–23 (2015). https://doi.org/10.1145/2679394

10. Xia, F., et al.: Write-aware random page initialisation for non-volatile memory systems. In: 2014 IEEE 32nd International Conference on Computer Design (ICCD), Seoul 208–215 (2014)

11. Lee, B.C., et al.: Architecting phase change memory as a scalable dram alternative. In: Proceedings of the 36th Annual International Symposium on Computer Architecture. ISCA ’09. New York: Association for Computing Machinery, 2–13 (2009)

12. Cho, S., Lee, H.: Flip-n-write: a simple deterministic technique to improve dram write performance, energy and endurance. In: 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), New York, NY 347–357 (2009)

13. Qureshi, M.K., et al.: Preset: improving performance of phase change memories by exploiting asymmetry in write times. In: 2012 39th Annual International Symposium on Computer Architecture (ISCA), Portland 380–391 (2012)

14. Hu, J., et al.: Reducing write activities on non-volatile memories in embedded CMPS via data migration and recomputation. In: Design Automation Conference, Anaheim, California 350–355 (2010)

15. Zhou, P., et al.: A durable and energy efficient main memory using phase change memory technology. In: Proceedings of the 36th Annual International Symposium on Computer Architecture. ISCA ’09. New York: Association for Computing Machinery, 14–23 (2009)
16. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: 2007 44th ACM/IEEE Design Automation Conference, 9–14 (2007)
17. Prabhu, P., et al.: Extracting device fingerprints from flash memory by exploiting physical variations. In: Proceedings of the 4th International Conference on Trust and Trustworthy Computing. TRUST’11. Berlin: Springer-Verlag. 188–201 (2011)
18. Zhang, W., Li, T.: Characterising and mitigating the impact of process variations on phase change based memory systems. In: 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 2–13 (2009)
19. Pyo C., Pae S., Lee G. DRAM as source of randomness. Electronics Letters, 45(1), 26 (2009). https://doi.org/10.1049/el:20091899
20. Chimenton, A., Zambelli, C., Olivero, P.: A new analytical model of the erasing operation in phase-change memories. IEEE Electron. Device Lett. 31(3), 198–200 (2010)
21. Oh, S., et al.: The impact of resistance drift of phase change memory (PCM) synaptic devices on artificial neural network performance. IEEE Electron. Device Lett. 40(8), 1325–1328 (2019)
22. Ventrice D., et al.: A Phase Change Memory Compact Model for Multilevel Applications. IEEE Electron Device Letters, 28(11), 973–975 (2007). https://doi.org/10.1109/LED.2007.907288
23. Wang, D., et al.: Dramsim: A memory system simulator. SIGARCH Comput. Archit. News. 33(4), 100–107 (2005)
24. Wang, J., et al.: Energy-efficient multi-level cell phase-change memory system with data encoding. In: 2011 IEEE 29th International Conference on Computer Architecture (ICCA). 175–182 (2011)
25. Provaro, A., et al.: Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials. IEEE Trans. Electron. Devices. 51(5), 714–719 (2004)
26. Carlson, T.E., Heirman, W., Eekhout, L.: Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In: SC ’11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. 1–12 (2011)
27. Dong, X., et al.: Nvsim: a circuit-level performance, energy, and area model for emerging nonvolatile memory. IEEE Trans. Comput. -Aided Design Integr. Circ. Syst. 31(7), 994–1007 (2012)
28. Hay, A., et al.: Preventing pen banks from seizing too much power. In: 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 186–195 (2011)
29. Li, S., et al.: Pinatubo: a processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. In: 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC). 1–6 (2016)
30. Seshadri, V., et al.: Ambit: in-memory accelerator for bulk bitwise operations using commodity dram technology. In: 2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Boston, MA 273–287 (2017)
31. Ielmini, D., et al.: Physical interpretation, modelling and impact on phase change memory (PCM) reliability of resistance drift due to chalcogenide structural relaxation. In: 2007 IEEE International Electron Devices Meeting, Washington, DC 939–942 (2007)
32. Ielmini, D., et al.: Physical mechanism and temperature acceleration of relaxation effects in phase-change memory cells. In: 2008 IEEE International Reliability Physics Symposium. 597–603 (2008)
33. Maes, R., Tuyls, P., Verbauwhede, I.: A soft decision helper data algorithm for SRAM PUFs. In: 2009 IEEE International Symposium on Information Theory, Phoenix, AZ 2101–2105 (2009)
34. Chhabra, S., Solihin, Y.: I-nvmm: secure non-volatile main memory system with incremental encryption. Proceedings of the 38th Annual International Symposium on Computer Architecture. ISCA ’11, 177–188. New York: Association for Computing Machinery (2011). https://doi.org/10.1145/2000064.2000086
35. Xie, M., et al.: Aim: fast and energy-efficient aes in-memory implementation for emerging non-volatile main memory. In: 2018 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany: 625–628 (2018)
36. Jacob, B., Ng, S., Wang, D.: Memory systems: cache, dram, disk. Morgan Kaufmann Publishers Inc, San Francisco, CA (2007)