# Decongest: Accelerating Super-Dense PCM Under Write Disturbance by Hot Page Remapping

Rujia Wang[1], Sparsh Mittal, Youtao Zhang, and Jun Yang

**Abstract**—At small feature sizes, phase change memory (PCM) shows write disturbance (WD) error (WDE) and this issue can eclipse the density and energy efficiency advantage of PCM. We propose 'Decongest', a technique to address WD errors in main memory designed with super-dense ($4F^2$ cell size) PCM. Decongest works by identifying and remapping write-intensive hot pages to a WD-free spare area, which avoids WD to nearby pages *due to* writing these hot pages, and WD *to* these hot pages from writing nearby pages. Compared to a WD-affected super-dense PCM baseline, Decongest improves the performance by 14.0 percent, and saves 21.8 percent energy.

**Index Terms**—Phase change memory, main memory, write disturbance, reliability, page remapping, energy saving

## 1 Introduction

THE energy efficiency targets of next-generation systems, along with scaling challenges of conventional memory technologies have motivated the researchers to explore novel memory technologies. PCM is a promising memory technology due to its attractive properties, e.g., high density, non-volatility and scalability. Due to this, design of PCM based main memory has received significant attention in recent years [1], [2]. Several vendors such as IBM and Intel-Micron have developed PCM or PCM-like prototypes [3], [4] which indicates that mass-scale production of PCM may soon become commercially feasible.

A crucial challenge in use of PCM, however, is that at small feature sizes (e.g., sub-20 nm), a write to one PCM cell can disturb the value stored in nearby cells which is referred to as WDE [5]. In fact, the heat produced in writing one PCM cell can alter the value stored in several nearby cells (e.g., up to 11 cells in a 64 B block [6]). With decreasing feature size, inter-cell spacing reduces which will aggravate WDE issue even further. Furthermore, WDE can lead to security vulnerability since an attack-like write sequence can disturb selected memory locations by altering the data stored in adjacent locations.

A naive approach to address WDE is to increase the inter-cell separation along wordline and/or bitline [7]. However, this approach reduces memory density significantly and may nullify the advantages of feature-size scaling. Another approach, termed verify-and-correct (VnC) reads the old data of neighboring lines before and after writing the actual line. Then, if required, a RESET (i.e., a write) operation is performed to correct the error. However, this operation itself can lead to errors which would require cascading VnC operations, leading to large performance and energy loss [8]. It is clear that addressing WDE is extremely important for improving performance and ensuring security of PCM.

1. R. Wang and S. Mittal are co-first authors.

- *R. Wang, Y. Zhang, and J. Yang are with the University of Pittsburgh, Pittsburgh, PA 15260. E-mail: {rujia.w, youtao, juy9}@pitt.edu.*
- *S. Mittal is with IIT Hyderabad, Kandi, Telangana 502285, India. E-mail: sparsh0mittal@gmail.com.*

*Manuscript received 14 Nov. 2016; revised 15 Feb. 2017; accepted 23 Feb. 2017. Date of publication 1 Mar. 2017; date of current version 5 Jan. 2018.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/LCA.2017.2675883*

In this paper, we present a technique, named Decongest, to address WDE in PCM based main memory. We observe that since most writes to main memory are directed to few pages, they are primarily responsible for WD. Decongest works on the key idea that by redirecting such hot pages in a spare area, (1) WD in neighboring pages due to remapped hot pages and (2) WD in hot pages due to neighboring pages can both be avoided. Thus, page remapping approach of Decongest leads to twofold reduction in WDE. Further, by reducing the writes to PCM main memory, it also improves the PCM lifetime.

Microarchitectural simulations have shown that compared to a WD-affected ($4F^2$) PCM baseline, Decongest technique improves performance by 14.0 percent, and saves 21.8 percent energy. By comparison, an iso-capacity WD-free $8F^2$ PCM memory which occupies *nearly double the area* provides performance improvement of 23.9 percent and energy saving of 39.8 percent. Clearly, Decongest provides more than half the improvement achievable from $8F^2$ PCM, while reducing memory area by half. Additional experiments have shown that Decongest allows trading performance improvement with implementation overhead.

## 2 Background

### 2.1 Write Disturbance Error

PCM uses GST material for data storage [8]. In this paper, the (super-dense) PCM is assumed to be designed with crossbar structure with diode as the selector and this provides an ideal cell size of $4F^2$. [8]. In PCM, application of electric pulse generates heat which can change the state of GST between crystalline (SET) and amorphous (RESET) states that have low and high resistance, respectively. To SET a PCM cell, the temperature is kept higher than crystallization temperature ($\sim 300$ °C) but lower than melting temperature ($\sim 600$ °C) for a long time duration. To RESET a cell, it is heated above melting temperature and rapidly quenched. Hence, the heat produced for writing one cell can propagate and change the resistance states of adjacent cells which is referred to as WDE [9].

If a cell is being RESET, an adjacent *idle* cell with *value 0* is vulnerable [8] and thus, WDE is a data-value dependent error. Fig. 1 shows how WDE may affect neighboring cells along both wordline and bitline directions. Since the magnitude of SET current is nearly half of that of RESET current, WDE due to SET operation is negligible [6]. In this paper, we focus on the bitline direction WDE problem, and adopt the DIN design from [6] to mitigate the WDE problem along wordline. Our WDE error model is adopted from [6], [8], which shows that for single-level cell (SLC) PCM, WDE rate along bit-line is $\sim 11.5$ percent.

### 2.2 Related Work

Previous works address WDE along wordline [6], [10], [11] or bitline [8], [11]. Jiang et al. [6] propose a data-compression based technique. For blocks compressed below a certain length, data encoding is performed to avoid WDE-vulnerable patterns. While writing the encoded block, a write is completed if the number of WDEs is within correction capability of ECC, otherwise, VnC is performed. For uncompressed blocks, write completes only when no WDE is present. The effectiveness of this technique, however, depends on compressibility of the data. For mitigating VnC overhead, Wang et al. [10] exploit imbalanced writes to different cells within a page, whereas Decongest exploits imbalanced writes to different pages.

Wang et al. [8] propose lazy-correction scheme to consolidate multiple VnC operations. They also propose issuing verification requests early when a write operation is waiting in the write queue. For lines marked as 'no-use' by OS, VnC is not performed. Eslami

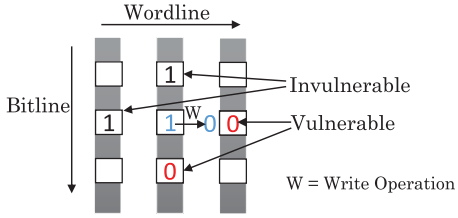


Fig. 1. An example of WDE and its data pattern dependence.

et al. [11] propose a chckerboard layout of PCM cells where only half the cells are used at a time to mitigate WDE. This, however, reduces memory capacity. Decongest is orthogonal to these techniques [6], [8], [11] and can be easily integrated with them to provide even larger improvement.

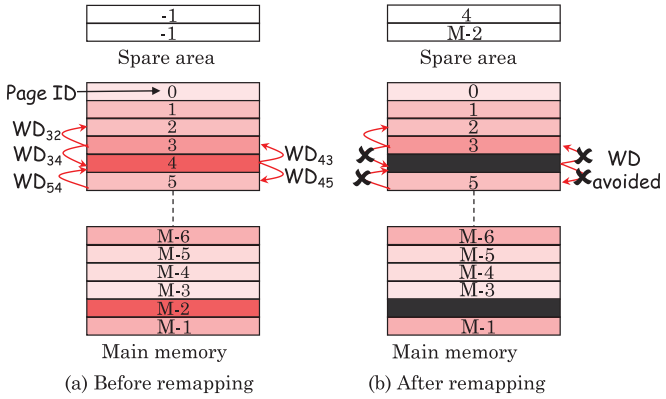


Fig. 2. An illustration of remapping approach of Decongest. $WD_{xy}$ is disturbance to page $y$ due to a write to page $x$.

## 3 Decongest: Overall Architecture

### 3.1 Key Idea and Working

*Key Idea.* Under WDE, PCM write latency increases even further due to iterative VnC operations since its neighboring lines need to be checked and corrected (if required). This leads to large performance loss especially for memory-intensive applications. It is well known that due to temporal locality, only a small number of pages, termed *hot pages*, are expected to be repeatedly accessed. Decongest exploits the non-uniform memory access pattern to reduce VnC overhead. Specifically, by remapping these hot pages to a WD-free spare area, need of VnC operations in neighboring pages can be avoided. Further, once a hot page is remapped, on write to a page, VnC operation need not be performed for these remapped pages. Thus, Decongest brings twofold reduction in VnC operations. Further, by reducing the writes to PCM main memory, it also improves the PCM lifetime. Let technique refer to baseline or Decongest. The absolute lifetime $\Pi_{\text{technique}}$ of PCM is defined as $\Phi/B_{\text{technique}}$, where $\Phi$ is PCM write endurance ($10^9$), and $B_{\text{technique}}$ is number of writes to the most write-intensive page in a unit time interval. The improvement in lifetime due to Decongest is defined as $\Pi_{\text{Decongest}}/\Pi_{\text{baseline}}$. The main memory and spare area are designed with $4F^2$ and $8F^2$ cells [6], respectively. Since memory systems generally use extra pages for reliability purposes, they can be easily used as spare area.

**Algorithm 1.** Decongest Workflow for Each Memory Access (`RT`: Remapping Table; `addr`: Current Request Address; `WrCounter`: Write Frequency Counter)

**Input** Remapping Threshold $T_p$

```
1: if addr in RT then
2:    Complete read/write from RT[addr];
3: else
4:    if Access is a read access then
5:       read content from addr ;
6:    else
7:       WrCounter[addr]++;
8:       if WrCounter[addr] > Tp then
9:          addr' = RT.findRemappingAddrInSpareArea;
10:         Write at addr';
11:         RT[addr] = addr';
12:      else
13:         write at addr;
```

Fig. 2 provides an overview of our technique (only few WD effects are shown for brevity). Without remapping, a write to page 4 leads to WD in neighboring pages 3 and 5 ($WD_{43}$ and $WD_{45}$) and hence, VnC operations need to be performed for those pages. If an error is found in page 3, correcting it using a RESET operation in VnC may disturb page 2 and 4 ($WD_{32}$ and $WD_{34}$) which will require more VnC operations. These cascading VnC operations lead to large performance and energy penalty.

Assuming that page 4 and page M-2 are two most frequently accessed pages, remapping them to a WD-free spare area avoids WD *from these pages* ($WD_{34}$ and $WD_{54}$) and *to these pages* ($WD_{43}$ and $WD_{45}$). Due to this, future write accesses to these hot pages can be accelerated since no VnC operations need to be performed while writing them. Algorithm 1 shows the workflow of Decongest for each memory access.

*Intuitive Explanation.* The following analogy is helpful to get insights into WDE phenomenon and Decongest technique. In a congested locality, a person with contagious disease disturbs its neighbors by spreading the disease and some persons are more prone to such sickeness than others. The disease can be greatly mitigated by relocating most sick persons to an uncongested disease-free area. Similarly, in the super-dense PCM, a write operation to a cell disturbs its neighbors and some (hot) pages are more prone to WD than others. Hence, WD can be greatly mitigated by relocating hot pages to a non-dense WD-free area.

*Promotion Policy.* The effectiveness of Decongest depends on accurate identification of hot pages. An ideal approach for finding the hot pages will require future knowledge. Since future knowledge is not available, we use past access frequency as the indicator of future access frequency. We associate a counter with each page which is incremented on each write access to record write access frequency. A page whose counter reaches a threshold ($T_p$) is promoted to spare area.

*Demotion Policy.* When the spare area is full, a page needs to be replaced and demoted to main memory. For simplicity, we use first in first out (FIFO) scheme which replaces the oldest page in the spare area. This scheme maintains a pointer to the next usable location in spare area. If this location is empty, new page is allocated here, otherwise, currently stored page is demoted to main memory and new page is allocated here. After this, the pointer is incremented to the next usable location.

*Page Remapping.* Whenever a page is promoted into the spare area, its address needs to be remapped. We use a remapping table (RT) to record the remapping information. Each entry in RT contains the original page address and remapped page address. In order to remap a page, the spare area pointer is checked to get the new address for current page. The page replaced from spare area (if any) is migrated back to the main memory, and its remapping information in RT is cleared. Then, the hot page is written directly to the spare area, and the corresponding remapping information is stored in RT.

*Salient Features of Decongest.* Compared to the fixed allocation approach proposed in [8], Decongest does not require OS involvement, which simplifies the overall design. The counter-based hot

TABLE 1
System Configuration

| | |
|---|---|
| Core | 4-core single issue in-order CMP, 4 GHz |
| L1 cache | separate I/D, 32 KB/core, 64 B block |
| L2 cache | 256 KB/core, 4-way LRU, 64 B block, writeback |
| L3 cache | 2 MB/core, 8-way LRU, 64 B block, writeback |
| Main memory | 4 GB, 1 channel, 2 ranks, 8 banks per rank, 32-entry write queue per bank |
| PCM parameters | Read: 100 ns, Write (SET): 200 ns, Write (RESET): 100 ns, 128-bit parallel write [13] |

page identification is easy to implement. Decongest directly focuses on write-intensive pages which are primarily responsible for WD. By comparison, previous techniques direct same effort for all the pages (e.g., [8], [11]) and fail to account for write-variation present in applications.

### 3.2 Overhead Estimation

*Storage Overhead*. We assume that there are $B$ banks, each with $P$ pages of $S$ bits. The spare area for each bank has $K$ pages. In our experiments, $B = 16$, $P = 32{,}768$, $S = 65{,}536$ (Table 1) and $K = 512$. Spare pages require $B \times K \times S$ bits, access counters require $B \times P \times \lceil \log_2 T_p \rceil$ bits and FIFO policy requires $B \times \lceil \log_2 K \rceil$ bits. These are stored in main memory and for our experiments, their total overhead is 64.5 MB, which is only 1.57 percent of total memory capacity (4 GB). Since spare area is designed with $8F^2$ cells, the area overhead is $\approx 3.14$ percent. Since Decongest allows lowering the memory area for a given memory capacity by virtue of using the super-dense PCM cells, a small storage overhead is easily acceptable.

RT is stored in memory controller and it requires $K(\lceil \log_2 B \rceil + \lceil \log_2 P \rceil + \lceil \log_2 K \rceil)$ bits for each bank. In our experiments, this totals to 14 Kb per bank, i.e., 28 KB for entire memory, which is only 0.34 percent of LLC capacity (8 MB). Also, this is in the range of overhead of previous techniques [12].

*Latency and Energy Overheads*. We assume 20 cycles penalty of accessing RT for both reads and writes. Note that this latency is easily hidden since PCM access latency is high and memory requests generally wait for long time in scheduler. Each remapping event triggers two writes to memory which leads to latency overhead. We account for this latency overhead in our experiments. As our results show, the performance improvement provided by Decongest outweighs the latency overhead of page remapping.

## 4 Results and Analysis

*Experimentation Details*. We use an in-house simulator which models a 4-core in-order processor with entire memory hierarchy. The simulator faithfully models memory timing constraints, bank conflicts and DDR scheduling constraints. Table 1 summarizes the system configuration. We use the scaling model from Jiang et al. [6] and assume the WDE rate at 20 nm node.

The simulated workloads are from SPEC2006 benchmark suite. We used the PIN tool to capture and filter 50 million memory references to L1 cache after skipping the application initialization phase. In the experiments, each core runs a copy of one application, forming four-core multi-programmed workloads running in different virtual address spaces.

We adopt PCM energy model from Zhou et al. [1] and also compute energy consumption of VnC operations and page migrations. The differential write is adopted except when performing replacement from spare area. We assume that a wear-leveling scheme is used in spare area to avoid its early wear-out.

*Comparison with Other Approaches*. Our baseline is a WD-affected $4F^2$ PCM which uses VnC operations to correct WDEs. We also show results with '2Xarea' PCM, which is an iso-capacity WD-free $8F^2$ PCM (i.e., having nearly double the area than baseline) but does not require VnC operations. This allows us to compute the maximum achievable improvement from any WDE-mitigation technique. All settings adopt DIN [6] to mitigate WDE along wordline.

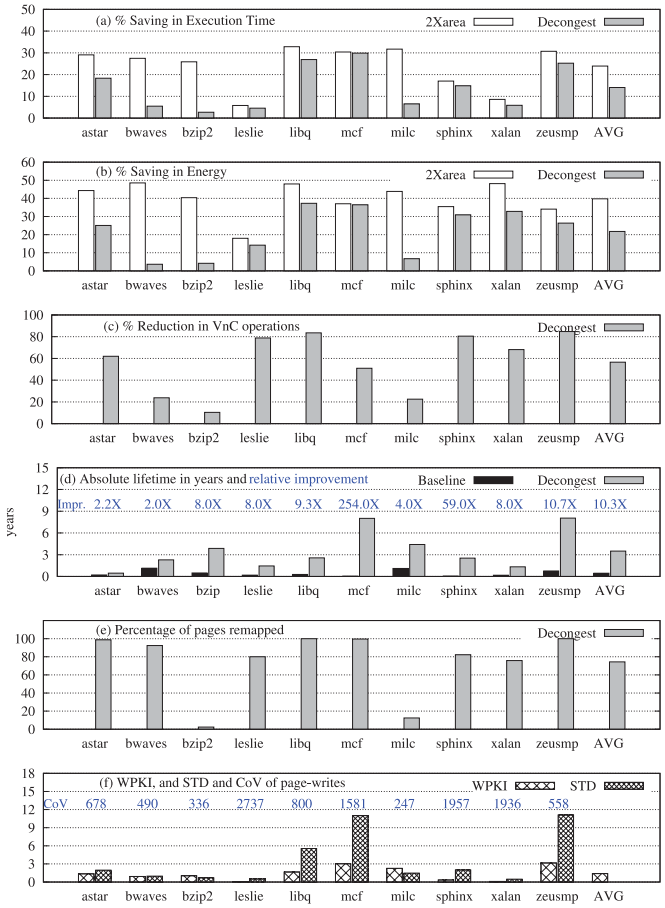


Fig. 3. Results with '2Xarea' and Decongest schemes (leslie = leslie3d, libq = libquantum). STD and CoV are standard deviation and coefficient of variation (in %) of per-page writes).

### 4.1 Main Results

Fig. 3 shows the results for our proposed Decongest technique. We set $K = 512$ and $T_p = 2$ as the default setting for Decongest. Overall, Decongest improves performance by 14.0 percent, and saves 21.8 percent energy, whereas the 2Xarea (i.e., $8F^2$) WD-free PCM improves performance by 23.9 percent and reduces energy by 39.8 percent. Compared to baseline, Decongest reduces 63.2 percent of VnC operations during write operations.

The performance and energy benefits provided by Decongest depend on the write intensity and temporal locality of applications. Write per kilo instructions (WPKI) shows the write intensity, whereas non-uniformity of writes can be inferred from STD and CoV values (Fig. 3f). For applications with large WPKI value, WD leads to large number of VnC operations which degrade performance and energy efficiency. Similarly, for applications with high locality, few pages attract all the writes and by remapping them, Decongest reduces WD significantly. Hence, Decongest provides large improvements for libquantum, mcf, sphinx and zeusmp benchmarks.

For some applications such as leslie3d and xalan, although write-intensity (WPKI) is relatively smaller, the locality of writes is high, as reflected in high CoV values. These applications have only few hot pages and hence, Decongest can effectively migrate them and hold them in spare area for avoiding WDEs. Conversely, for

TABLE 2
Parameter Sensitivity Results (Default: $K = 512$, $T_p = 2$ and FIFO Replacement Policy)

| | Percentage reduction in | | |
|---|---|---|---|
| | Execution time | Energy | VnC operations |
| Default | 14.04 | 21.77 | 63.15 |
| $K = 256$ | 11.66 | 17.58 | 56.55 |
| $K = 1{,}024$ | 15.20 | 24.14 | 66.25 |
| $T_p = 4$ | 11.38 | 16.69 | 50.81 |
| $T_p = 6$ | 10.27 | 15.09 | 43.94 |
| $T_p = 8$ | 9.76 | 14.13 | 39.45 |
| LRU policy | 14.04 | 21.77 | 63.15 |
| LWN policy | 9.63 | 17.10 | 47.04 |

write-intensive applications with poor locality, such as milc, there is high congestion for spare area. Hence, replacement happens frequently, which increases the migration overhead and reduces the improvement achieved. For reasons explained above, applications with low write-intensity and locality, such as bwaves and bzip2, also show small benefit with our technique.

Several benchmarks have very small lifetime with baseline (e.g., 0.03 years with mcf). By virtue of migrating hot pages to spare-area, Decongest improves lifetime significantly (e.g., 8.03 years for mcf) and average relative improvement is $10.28\times$ (Fig. 3d). Fig. 3e shows the percentage of pages remapped, computed as number of remapped pages/total accessed pages. The number of total accessed pages reflects the application working set size and remapped pages are those with write count higher than $T_p$. Percentage of remapped pages has strong correlation with percentage of VnC operations avoided (Fig. 3c), which is expected. Since the size of spare area is limited, in applications with large *absolute* number of remapped pages (figure omitted), there is contention for spare pages which lowers the energy saving from Decongest. This happens for bwaves and milc.

The 2Xarea approach fails to exploit temporal locality present in applications and hence, incurs nearly 100 percent area overhead. By comparison, Decongest enables use of super-dense PCM and provides large performance and energy gains.

### 4.2 Parameter Sensitivity Results

We now evaluate the sensitivity of Decongest to different algorithm parameters. Results are summarized in Table 2.

*Impact of Number of Pages in Spare Area (K).* On increasing $K$ to 1,024, higher number of hot pages can be migrated to spare area which leads to larger reduction in VnC operations. This translates into higher performance and energy improvement. Converse is true for reducing the value of $K$.

*Impact of Threshold Value ($T_p$).* On increasing $T_p$, pages with higher number of writes are considered as hot and thus, it makes Decongest more conservative in migrating pages. Hence, with increasing $T_p$, Decongest reduces smaller number of VnC operations which reflects in smaller performance and energy advantages, although this also has the advantage of reducing page migration overhead. Thus, by changing the value of $T_p$, a designer can achieve a tradeoff between performance/energy improvement and the migration overhead. These results also justify our choice of $T_p$ value (as 2).

*Impact of Replacement Policy (RP) in Spare Area.* We change RP from FIFO to LRU and least write number (LWN). In LWN policy, the page with the fewest write number is selected as the victim. Clearly, LRU performs same as FIFO policy whereas LWN performs worse than both the policies. For each bank, the storage required for FIFO, LRU and LWN policies are $\log_2 K$, $K \times \log_2 K$ and $K \times 3$ (assuming a 3-bit counter), respectively. For large associativity ($K = 512$ in this paper) structures, the exact ordering maintained by LRU becomes unnecessary and hence, LRU provides no benefit over pseudo-LRU policies. Also, on any write, LRU may require changing up to $K$ LRU-age counters. Thus, LRU incurs high storage and operation overheads without providing additional benefit.

LWN policy requires comparing the write-counts of all the pages in spare area to find the one with minimum value. Since $K$ is large, pages remain in spare area for very long time before getting evicted. During this time, they exhaust all the reuse and hence, write-count becomes a poor indicator of future reuse which explains smaller improvements provided by LWN. These results justify our choice of FIFO.

## 5 Conclusion

We presented Decongest, a technique to reduce WDE overhead in PCM-based main memory. By reserving a small amount of WDE-free area and remapping hot pages into it, Decongest exploits application's locality and improves performance and energy efficiency by reducing VnC overhead and write latency.

## Acknowledgments

This work is partially supported by US NSF CCF#1617071 and a seed-grant from IIT, Hyderabad, India.

## References

[1] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. 36th Annu. Int. Symp. Comput. Archit.*, 2009, pp. 14–23.
[2] S. Mittal, "A survey of power management techniques for phase change memory," *Int. J. Comput. Aided Eng. Technol.*, vol. 8, pp. 424–444, 2014.
[3] T. Trader, "IBM Puts 3D XPoint on notice with 3 Bits/Cell PCM breakthrough," 2015. [Online]. Available: https://goo.gl/Z4sGZU
[4] K. Vatto, I. Cutress, and R. Smith, "Analyzing Intel-Micron 3D XPoint: The next generation non-volatile memory," 2015. [Online]. Available: https://goo.gl/r18xvc
[5] S. Lee, et al., "Programming disturbance and cell scaling in phase change memory: For up to 16nm based $4F^2$ cell," in *Proc. Symp. VLSI Technol.*, 2010, pp. 199–200.
[6] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2014, pp. 216–227.
[7] D.-H. Ahn, et al., "Reliability perspectives for high density PRAM manufacturing," in *Proc. Int. Electron Devices Meeting*, 2011, pp. 12.6.1–12.6.4.
[8] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "SD-PCM: Constructing reliable super dense phase change memory under write disturbance," in *Proc. 20th Int. Conf. Archit. Support Program. Languages Operating Syst.*, 2015, pp. 19–31.
[9] S. Mittal, "A survey of soft-error mitigation techniques for non-volatile memories," *Computers*, vol. 6, 2017, Art. no. 8.
[10] R. Wang, L. Jiang, Y. Zhang, L. Wang, and J. Yang, "Exploit imbalanced cell writes to mitigate write disturbance in dense phase change memory," in *Proc. 52nd ACM/EDAC/IEEE Des. Autom. Conf.*, 2015, Art. no. 88.
[11] A. Eslami, A. Velasco, A. Vahid, G. Mappouras, R. Calderbank, and D. J. Sorin, "Writing without disturb on phase change memories by integrating coding and layout design," in *Proc. Int. Symp. Memory Syst.*, 2015, pp. 71–77.
[12] L. E. Ramos, E. Gorbatov, and R. Bianchini, "Page placement in hybrid memory systems," in *Proc. Int. Conf. Supercomputing*, 2011.
[13] Y. Choi, et al., "A 20 nm 1.8 V 8Gb PRAM with 40 MB/s program bandwidth," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2012, pp. 46–48.