Counter Advance for Reliable Encryption in Phase Change Memory

Donald Kline Jr., Rami Melhem, and Alex K. Jones

Abstract—The use of hardware encryption and new memory technologies such as phase change memory (PCM) are gaining popularity in a variety of server applications such as cloud systems. While PCM provides energy and density advantages over conventional DRAM memory, it faces endurance challenges. Such challenges are exacerbated when employing memory encryption as the stored data is essentially randomized, losing data locality and reducing or eliminating the effectiveness of energy and endurance aware encoding techniques. This results in increasing dynamic energy consumption and accelerated wear out. In this paper we propose *counter advance*, a technique to leverage the process of encryption to improve reliability and lifetime while maintaining low-energy and low-latency operation. Counter advance is compatible with standard error-correction codes (ECC) and error correction pointers (ECP), the standard for mitigating endurance faults in PCM. Counter advance achieves the same fault tolerance using three ECP pointers for a 10^{-4} cell failure rate compared to the leading approach to consider energy savings and reliability for encrypted PCM (SECRET) using five ECP pointers. At a failure rate of 10^{-2}, counter advance can achieve an uncorrectable bit error rate (UBER) of 10^{-10}, compared to $< 10^{-4}$ for SECRET, using six ECP pointers. This leads to a lifetime improvement of 3.8× while maintaining comparable energy consumption and access latency.

Index Terms—Emerging memories, reliability, stuck-at faults, and error correction

1 Introduction

FUTURE high performance servers are expected to utilize tiered memory that employs new solid state technology like phase-change memory (PCM). PCM is gaining popularity in these systems as evidenced by products like the Micron/Intel 3D XPoint memory, due to their near DRAM read latency with improved density, energy, and non-volatility [1], [2]. A challenge of PCM is its high dynamic write power. Many proposed solutions to this challenge leverage data locality where a newly written value typically has high similarity to the currently stored data. Thus, techniques like differential write [3] and flip-N-write [4] leverage this similarity to reduce dynamic power. Additionally, PCM has a challenge of limited write endurance (circa 10^8 writes) [5] before failing as a "stuck" cell. Cells "stuck-at" a particular value can be read but their state is immutable. The techniques to reduce bit changes to save energy also improve the effective memory endurance.

The non-volatility of PCM can be a further challenge if physical security of the hardware is compromised. Dedicated encryption in the memory is one proposed solution to help mitigate these risks. Unfortunately, when storing encrypted data, locality is lost as the probability of any particular cell being a '1' or '0' becomes 50 percent. Thus, for each write, approximately 50 percent of the cells must be written, even if actual value changes minimally. This leads to increased energy consumption and faster wear out.

- *D. Kline and A. K. Jones are with the Department of Electrical and Computer Engineering, University of Pittsburgh, PA 15260. E-mail: {dek61, akjones}@pitt.edu.*
- *R. Melhem is with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260. E-mail: melhem@cs.pitt.edu.*

Manuscript received 23 Mar. 2018; revised 29 June 2018; accepted 19 July 2018. Date of publication 30 July 2018; date of current version 9 Nov. 2018.
(Corresponding author: Donald Kline.)
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.2018.2861012

We propose *counter advance*, a technique to leverage the existing encryption hardware to improve reliability in PCM with dedicated counter-based encryption. Counter advance utilizes several observations: First, for stuck-at PCM cells, depending on the value to be written and the state of each faulty cell, the data can either be stuck-at the right value (stuck-at right, SA-R) or the wrong value (stuck-at wrong, SA-W). Second, a new encryption of the same data item can be generated by running the encryption with the next counter value. Third, for each new encrypted candidate for storage, there is a 50 percent chance for each faulty cell that the data will be SA-R. Thus, by exploring multiple counter values, a value that eliminates (or minimizes) SA-W values may be obtained. In this case, even for a row with some faulty cells due to endurance, the system may continue to operate, extending the useful lifetime of the memory. Assuming a 27- to 32-bit counter, 10^8 to 10^9 writes per row, which meets or exceeds the projected PCM cell endurance, can be achieved for a single key. If multiple encryptions are used only to tolerate faults, then the average counter advances per write will be $<< 2$.

In particular, we make the following contributions:

1) We present counter advance to improve the reliability and lifetime of PCM storage subject to endurance faults.
2) We demonstrate an architecture to apply counter advance in the context of the leading method to reduce energy for counter encrypted PCM.
3) We demonstrate the combined capability of counter-advance with ECC and ECP [6] protection to improve reliability and minimize storage overheads.
4) We provide detailed analyses of counter advance for the SPEC benchmark suite.

2 Background

Counter-mode encryption [7] to secure main memory, depicted in Fig. 1, was originally proposed for DRAM [8] by adding a cipher into the memory controller and adding counter storage for each memory row. Using a private key, a unique counter value, and the row address, the cipher generates a one-time pad (OTP). The OTP is XORed with the data to create an encrypted ciphertext. Decryption functions in the reverse, reading the data ciphertext and the counter in plaintext from the memory row to recreate the OTP and reverse the encryption process.

Unfortunately, encryption has the side effect of disrupting data locality, as for each plaintext value written, a unique OTP is generated containing a random set of 0's and 1's. Because the OTP changes for each write and the OTP is XOR'd with the plaintext, small or large changes in the plaintext results in similarly random ciphertext. This unpredictable output for each OTP is what gives the encryption its strength. When applied to a memory technology like PCM, standard energy saving techniques such as encoding and differential write are defeated and the increase in bit changes can lead to early cell wear-out.

SECRET or Smartly EnCRypted Energy EfficienT non-volatile memories [9] addresses these challenges by allocating a dedicated "epoch sub-counter" for blocks within the row. Thus, row writes must only encrypt dirty blocks. The sub-counter maintains independent count values for each block within an epoch. When a sub-counter saturates, the epoch ends, the main counter is advanced, all sub-counters are reset, and the entire row is encrypted and written with the new count value. SECRET addresses reliability by allocating ECP pointers [6] per row to tolerate faults.

3 Counter Advance

Counter advance leverages the property that encryption of plaintext with a new counter generates a new random ciphertext. Recall that

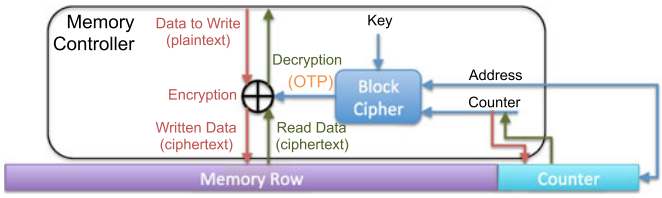

Fig. 1. Counter mode encryption in the memory controller.

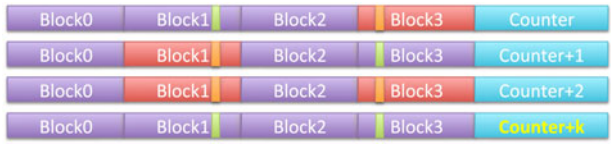

Fig. 2. Counter advance example. Green indicates a SA-R fault and orange a SA-W fault. Purple blocks are error free and red blocks contain an error.

data that matches the stuck-at value is SA-R and data opposed to the stuck-at value is SA-W. When writing a ciphertext candidate in the presence of stuck-at faults, each faulty cell has a 50 percent probability of being SA-R, and similar for SA-W. Thus, by incrementing the counter, it is possible to improve fault tolerance by finding a ciphertext candidate that maximizes SA-Rs.

Consider the example in Fig. 2 for a row with two stuck-at faults such that, given the ciphertext, the first is SA-R and the second is SA-W. Advancing the counter (`Counter+1`) resulted in the first fault becoming SA-W and the second becoming SA-R. This is due to the property that each fault in each ciphertext candidate has a 50 percent chance to be SA-R, but is equally likely to be SA-W. Advancing the counter again (`Counter+2`) was unlucky, resulting in two SA-Ws. The probability of finding an error free candidate with f faults is 2^{-f}, or 25 percent for f=2, which required multiple advancements in the example. Both word-level encryption and error correction can dramatically reduce the number of advancements to find an error free candidate. For example with single bit error correction (e.g., ECP-1), the example of Fig. 2 would have been successful without counter advancement. If SECRET is used with independent sub-counters per block and assuming all blocks were dirty, blocks zero to two would have been written with `Counter` and block three would have used `Counter+1`.

We explore counter advance with block level encryption and error correction in Fig. 3. Fig. 3a expands on Fig. 1 with sub-counters per 128-bit block in the style of SECRET [9]. Counter advance applied in this context examines the stuck-at bits independently between blocks and only advances the counter when SA-W bits appear. Stuck-at faults can be determined by storing and reading patterns of all '1's and '0's or using a fault cache [10]. It is straightforward to extend block-level encryption with ECC as the parity bits (e.g., SECDED (64,72) ECC) would not cross blocks. In this case, counter advance could protect fewer SA-W errors and allow ECC protection to correct others at the cost of reduced transient error protection.

ECP, in contrast, uses pointers that are shared by the blocks of a given row. Block-level counter advance faces the trade off of using a pointer or advancing the counter to mitigate a fault. Using a pointer will reduce the availability of pointers to tolerate faults in other blocks. The algorithm for selecting the appropriate write candidate for ECP with counter advance is shown in Fig. 3b.

Assuming a counter advancement epoch window w where $w = 2^b$ and b is the number of bits for each sub-counter, w serves as a threshold of how many counter values will be explored to accomplish a particular write successfully. If the encrypted data experiences E errors (i.e., SA-Ws) but E is less than a threshold T, the write proceeds with the current c value. Otherwise, if this is the "best" candidate (i.e., fewest SA-Ws) so far it is retained. If c is still within the epoch w, c is incremented and the next candidate is evaluated. If c reaches the limit of the epoch without finding a candidate within the error threshold, the best candidate is written if sufficient ECP pointers are available, otherwise the write fails. In our evaluation we consider two schemes: the *counter minimization* (CM) approach sets T to the number of available ECP pointers, allowing a write to proceed with the minimum counter value that discovers a possible solution, while the *pointer minimization* (PM) approach sets T=1 requiring a fault free solution to write immediately.

4 Evaluation

To evaluate the effectiveness of counter advance we studied a 4 GB main memory, with 64-bit words, and 512-bit rows organized in 4 KB pages using eleven SPEC CPU 2006 benchmarks [11]: bzip2, cactus, gamess, gcc, gobmk, gromacs, leslie3d, mcf, namd, pearl, and zeusmp. Each workload was executed for at least 1 billion write accesses. Three bits were allocated as a sub-counter for each block, setting $w = 8$. High cell failure rates (10^{-3}, and 10^{-2}) representing different points during the memory lifetime, as shown in Fig. 4, were used as stimuli for counter advance. To model the stuck-at faults, we created fault maps, including fault bits stuck at '0' or '1,' at these cell failure rates using Bayesian distribution to mimic the impact of process variation with spatial correlation of faults [12]. As process variation increases with scaling, the memory will incur cell faults more quickly and better error correction will be necessary to maintain effective lifetimes. For example, at a coefficient of variation (CoV) of 0.2, increasing fault tolerance to handle cell failures of 10^{-2} instead of 10^{-4} will extend the lifetime by 1.1×. For CoV of 0.25 and 0.3 effective lifetime can be extended by 3.8× and 16.4×, respectively, making operation in this failure range critical to reasonable memory lifetimes as scaling increases.

Fig. 5 shows a summary of the uncorrectable bit error rate (UBER), defined as the number of bit errors that occur per bit

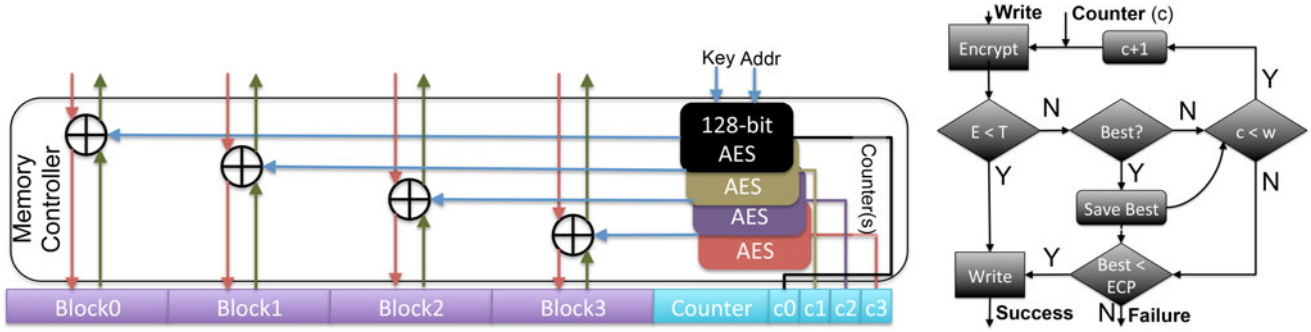

Fig. 3. Block level counter advance architecture.

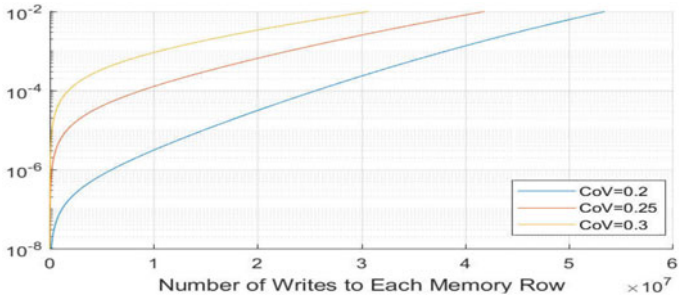

Fig. 4. Cell fault rate for different coefficients of variation.

written, for both row and block-level encryption with different strengths of error correction used to protect the data and counter bits. A "word-level" 64-bit block size was selected to match the word size in modern architectures. With no error correction, word-level counter advance (word CA) provides two orders of magnitude improvement in UBER compared to word-level encryption alone, such as SECRET [9], for as high as a 10^{-3} cell failure rate. As error correction is employed the improvement is amplified, providing 3–5 orders of magnitude improvement by introducing one ECP pointer (ECP1). At a cell failure rate of 10^{-3}, an UBER of $\leq 10^{-10}$ required only ECP4 for word and row CA. Employing PM reduced the requirement to ECP3. In contrast, SECRET with ECP6 can only achieve a 10^{-7} UBER.

At a cell failure rate of 10^{-2}, unsurprisingly, UBER is drastically reduced. For ECP6, the protection proposed by SECRET [9], counter advance achieves a 10^{-7} UBER versus 10^{-4} for ECP6 alone. Relaxing counter advance to explore eight epochs allowed word CA with PM to function at a $< 10^{-10}$ UBER with ECP6. This indicates that while a device might operate using the CM approach initially to minimize counter advancements, it could switch into PM and expand the searching window for *gracefully degraded* operation mode when the cell failure rate became sufficiently high.

Counter advance is sensitive to block size. Small block sizes will increase the flexibility to eliminate faults. Our block-size sensitivity study indicates counter advance is nearly as effective for 64-bit blocks as 32-bit blocks. 128-bit blocks have a noticeable degradation (0.5-1 orders of magnitude UBER), particularly with ECC and ECP, however, the counter advance improvements are still dramatic.

A logical concern about counter advance is the impact to performance from evaluating multiple ciphertext candidates and the potential to saturate the encryption counter more quickly. Fig. 6 shows the number of counter increments per write operation. Word-level encryption naturally reduces the average counter advancements per write (A) to A=0.98 compared to the row-level baseline of A=1, as each write only advances the dirty words' sub-counters. This provides sufficient "room" for word-level fault-induced counter advancements for lower fault rates (e.g., $\leq 10^{-4}$) without exceeding the row-level counter lifetime. At 10^{-3}, for ECP $\geq$3, $A < 1$. At 10^{-2} there are significant fault-induced counter advancements, owing to gracefully degraded operation. However, increasing the number of epochs searched for larger numbers of ECP pointers, especially ECP6, provides significant improvements in protection, with only slight increases to A. To achieve an UBER of 10^{-10} only requires A=1.2 with ECP6 after a cell failure rate of 10^{-2}.

As this gracefully degraded mode would only occur very late in the memory lifetime, these counter advancements would only saturate the counter nominally sooner while extending the usable life dramatically. If the system is reset with a new encryption key or the data is moved for another reason (e.g., wear-leveling [13]), the counter can also be reset. Moreover, given that writing is not typically on the performance critical path, our experiments indicate that these additional encryptions (A=1.2) do not significantly degrade performance.

5 Alternative Implementations

Counter-mode encryption has a downside that it requires the storage of a counter for each row. Unfortunately, this overhead cannot be eliminated for counter-mode encryption. However, the storage dedicated to the per-word sub-counters, initially proposed by SECRET [9] to reduce energy and improve endurance, could be retargeted to improve fault tolerance. This storage would be insufficient to add additional ECC, but could add two additional ECP pointers. This comparison is shown in Figs. 5 and 6 by comparing word CA with ECP-N to row CA with ECP-N+2. The results indicate that for lower fault rates, row CA would provide an advantage in fault tolerance at the cost of increased energy and reduced endurance. As the fault rate increases, word CA in PM mode is more fault tolerant while maintaining energy and endurance benefits over row CA.

NIST has proposed to use AES XTS as a standard for disk encryption [14]. While there are feasibility challenges to applying XTS in memory while guaranteeing protection, XTS would eliminate the need for counter storage in memory. XTS is based on XOR-encrypt-XOR (XEX) [15]. In disks, XEX/XTS encrypts twice, utilizing one key/encryption based on the sector and a second for each block within the sector. By extending the second encryption with an additional parameter supplied by the sub-counter, multiple candidates can be generated per block to improve fault tolerance. Because the ciphertext candidates for both XTS and counter-modes of AES have the same random properties, the results we obtained from XTS encryption are the same as those reported in Figs. 5 and 6.

6 Related Work

SECRET is the current state of the art in energy reduction and fault tolerance for encrypted PCM memory. However, SECRET improves on prior proposals such as DEUCE, or Dual Counter Encryption, which saves energy by distinguishing between dirty and clean words and encrypting on the dirty words within the epoch. DEUCE does not consider reliability and SECRET provides

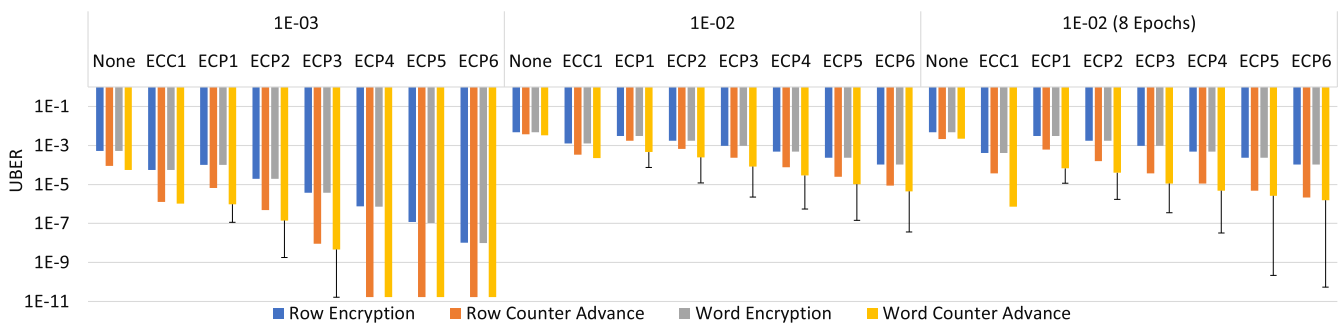

Fig. 5. UBER for various error rates. Counter advance explores one epoch (w=8), except where noted. Word CA+ECP is reported for CM with an error bar indicating PM.

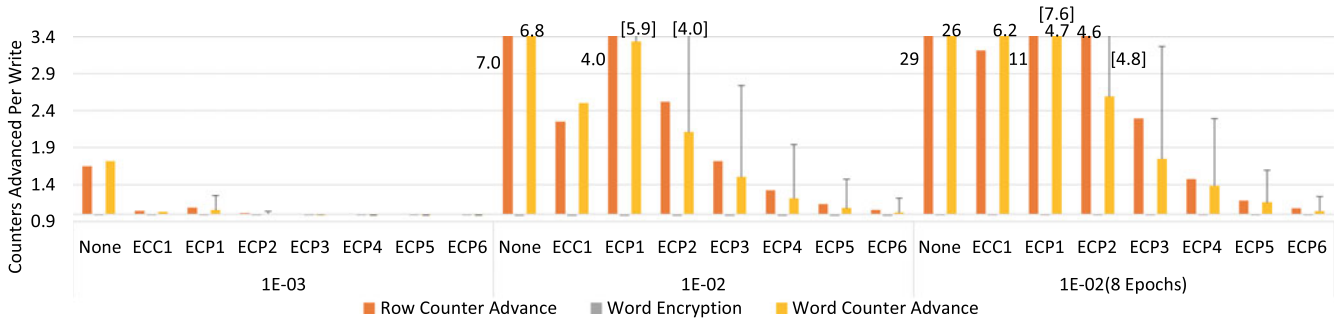

Fig. 6. Counters advanced per write at various error rates. Counter advance explores one epoch (w=8) except where noted. Word CA+ECP is reported for CM with an error bar indicating PM. Row Encryption is always unity.

significant savings over DEUCE at the cost of additional sub-counter bits [9]. A specifically fault-tolerant proposal is to use the increasing counter value to serve as an indicator of PCM memory cell age and using this information to adopt increasingly capable ECC to combat memory faults [16]. Counter advance is complementary to this approach allowing use of lower overhead ECC for a longer duration, or achieving a particular UBER with a lower overall ECC storage. There has also been recent work to collaboratively design wearleveling with counter-mode encryption. The counter storage size is reduced by resetting the counter when the data is moved to a new location from the wear-leveling resulting in improved overhead and latency [13]. Again counter advance is complementary as the number of counter advancements to maintain reliability would require minimal impact to the total counter advancement, which can retain the storage savings of this wear-leveling approach.

7 Conclusion

Counter advance leverages the nature of block cipher encryption to improve reliability of systems that use in memory encryption for memory with endurance faults that manifest as stuck-at values. Counter advance in the presence of SA-Ws generates additional write candidates to maximize SA-Rs just by advancing the counter. Counter advance is compatible with row or word-level writes and provides multiplicative improvements in UBER compared to error correction alone. Counter advance can achieve the same protection as strong error correction (e.g., ECC or ECP5) with far fewer pointers (e.g., ECP1) at moderate error rates. It can also maintain an UBER of 10^{-10} with the same error correction as the leading related work [9] for extremely high fault rates of 10^{-2}. This can lead to lifetimes being extended by 2-10× or more depending on severity of process variation. We plan to explore the performance impact, lifetime improvement, and examine other modes of encryption in detail in our future work.

Acknowledgments

This work was supported by NSF Graduate Research Fellowship award number 1747452, and SHREC industry and agency members and by the IUCRC Program of the National Science Foundation (Grant No. CNS-1738783).

References

[1] O. Zilberberg, S. Weiss, and S. Toledo, "Phase-change memory: An architectural perspective," *ACM Comput. Surv.*, vol. 45, no. 3, 2013, Art. no. 29.
[2] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.
[3] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," *SIGARCH Comput. Archit. News*, vol. 37, pp. 14–23, Jun. 2009.
[4] S. Cho and H. Lee, "Flip-N-write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 347–357.
[5] C. J. Xue, G. Sun, Y. Zhang, J. J. Yang, Y. Chen, and H. Li, "Emerging non-volatile memories: Opportunities and challenges," in *Proc. 9th IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign Syst. Synthesis*, 2011, pp. 325–334.
[6] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," *ACM SIGARCH Comput. Archit. News*, vol. 38, pp. 141–152, 2010.
[7] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proc. IEEE*, vol. 67, no. 3, pp. 397–427, Mar. 1979.
[8] C. Yan, D. Englender, M. Prvulovic, B. Rogers, and Y. Solihin, "Improving cost, performance, and security of memory encryption and authentication," in *Proc. 33rd Annu. Int. Symp. Comput. Archit.*, 2006, pp. 179–190.
[9] S. Swami, J. Rakshit, and K. Mohanram, "SECRET: Smartly encrypted energy efficient non-volatile memories," in *Proc. 53nd ACM/EDAC/IEEE Des. Autom. Conf.*, 2016, pp. 1–6.
[10] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "SAFER: Stuck-at-fault error recovery for memories," in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2010, pp. 115–124.
[11] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Archit. News*, vol. 34, pp. 1–17, 2006.
[12] Z. Al-Ars, *DRAM Fault Analysis and Test Generation*. Delft, Netherlands, Delft Univ. Technol., 2005.
[13] F. Huang, D. Feng, Y. Hua, and W. Zhou, "A wear-leveling-aware counter mode for data encryption in non-volatile memories," in *Proc. Des. Autom. Test Eur. Conf. Exhibition*, 2017, pp. 910–913.
[14] M. Dworkin, "Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on storage devices," Nat. Inst. Standards Technol., Gaithersburg, Tech. Rep. SP 800–38E, 2010.
[15] P. Rogaway, "Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC," *Int. Conf. Theory Application Cryptology Inform. Security*, pp. 16–31, 2004.
[16] J. Kong and H. Zhou, "Improving privacy and lifetime of PCM-based main memory," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Jun. 2010, pp. 333–342.