반도체디스플레이기술학회지 제21권 제4호(2022년 12월)
Journal of the Semiconductor & Display Technology, Vol. 21, No. 4. December 2022.

동작 분석을 통한 비휘발성 메모리에 대한 Wear-out 공격 방지 기법

최주희[*†]
[*†]상명대학교 스마트정보통신공학과

Exploiting Memory Sequence Analysis to Defense Wear-out Attack for Non-Volatile Memory

Juhee Choi[*†]
[*†]Dept. of Smart Information Communication Engineering, Sangmyung University

ABSTRACT

Cache bypassing is a scheme to prevent unnecessary cache blocks from occupying the capacity of the cache for avoiding cache contamination. This method is introduced to alleviate the problems of non-volatile memories (NVMs)-based memory system. However, the prior works have been studied without considering wear-out attack. Malicious writing to a small area in NVMs leads to the failure of the system due to the limited write endurance of NVMs. This paper proposes a novel scheme to prolong the lifetime with higher resistance for the wear-out attack. First, the memory reference pattern is found by modified reuse distance calculation for each cache block. If a cache block is determined as the target of the attack, it is forwarded to higher level cache or main memory without updating the NVM-based cache. The experimental results show that the write endurance is improved by 14% on average and 36% on maximum.

Key Words : Non-Volatile Memories, Wear-out Attack, Cache Replacement Policy, Cache Bypassing

1. 서 론

캐시 바이패싱(cache bypassing)은 자주 사용되지 않을 만한 캐시 블록을 캐시에 저장하지 않게 하여, 캐시의 오염(pollution)을 줄이는 기법이다[1-3]. 캐시는 기본적으로 캐시 적중 실패(cache miss)가 발생할 경우, 다른 캐시 블록을 방출한 후(evict)에 그 자리에 해당 캐시 블록을 할당하게(linefill) 된다. 그런데, 일부 캐시 블록은 자주 사용되지 않음에도 불구하고, 자주 사용되는 캐시 블록 대신 캐시의 용량을 차지하여, 전체적으로 적중률(cache hit ratio)을 떨어뜨린다. 이 문제를 해결하기 위해서, 캐시 블록을 캐시에 저장하지 않고, 상위 캐시로 보내는 캐시 바이패싱 기법이 제안되었다.

그리고, 이 캐시 바이패싱 기법은 쓰기 횟수를 줄이는 것이 중요한 비휘발성 메모리(non-volatile memory, NVM) 연구 분야에서도 도입되고 있다[4-6]. 비휘발성 메모리는 용어에서 나타나듯이, 전력을 공급하지 않아도 저장된 정보를 잃어버리지 않는 메모리를 뜻한다[7,8]. 기존의 휘발성 메모리가 전하(electronic charge)를 통해 0과 1를 구별하여 정보를 나타내는 반면, 비휘발성 메모리는 주로 고체의 상태나 자기장의 상태를 통해, 정보를 저장한다. Fig. 1은 비휘발성 메모리 중 가장 널리 연구되고 있는 종류 중에 하나인 Phase change memory(PCM)의 메모리 셀(cell)을 나타내고 있다. PCM의 셀은 Fig. 1과 같이 합금 형태의 소자와

†E-mail: jhplus@smu.ac.kr

이를 조작하는 로직들로 구성되어 있다. 이 셀은 2가지 상태를 나타낼 수 있어, 각 상태에 따라 '0'값과 '1'값을 저장할 수 있다. 비휘발성 메모리는 정적 전력 소모량(static energy consumption)이 극히 적고, 저장 용량의 밀도(density)가 커서, 기존의 캐시에서 사용되었던 SRAM을 대신해서 도입이 고려되고 있다.

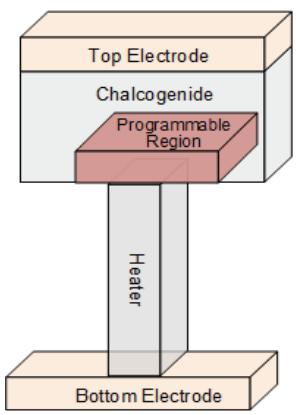

Fig. 1. Phase change memory (PCM).

그러나, 비휘발성 메모리는 캐시로 채택되기 위해서는 해결해야 할 문제점이 있다. 비휘발성 메모리의 셀의 상태를 변화시키는 것은 휘발성 메모리와는 달리 많은 전력과 긴 시간을 요구한다. 또한, 쓰기 횟수에 제약이 있어서, 일정 횟수의 쓰기가 일어나면, 그 셀은 마모되어(worn-out) 이후에는 사용할 수 없다[9,10]. 이것은 전체 메모리 시스템의 수명이 줄어드는 문제를 야기한다. 따라서, 캐시 바이패싱 기법은 필요한 캐시 블록만을 캐시에 저장하게 되므로, 자연스럽게 불필요한 쓰기 횟수가 감소하여, 비휘발성 메모리의 수명 향상에 도움이 된다.

기존에 제안된 바이패싱 기법들은 프로그램이 정상적으로 실행되는 상황을 가정하여 연구되어 왔으며, 이에 대해서는 좋은 성능을 보이고 있다. 그러나, 악의적으로 비휘발성 메모리의 특정한 셀에 쓰기 동작을 발생시켜 시스템의 수명을 단축시키는 Wear-out 공격에는 상대적으로 취약한 면모를 보인다[11,12]. 본 논문에서는 이러한 공격에도 저항성이 있는 비휘발성 메모리 시스템 구축을 위해서 새로운 캐시 바이패싱 기법을 제안한다.

본 논문에서 제안된 기법은 쓰기 동작의 패턴을 분석하여, 악의적인 공격 여부를 확인하고 이에 해당하는 경우, 비휘발성 메모리로 이루어진 캐시에 저장하지 않고, SRAM으로 이루어진 하위 레벨 캐시 또는 DRAM으로 이루어진 메인 메모리에 저장하게 한다.

2. 관련 연구

캐시 바이패싱 기법은 저장할 가치가 적은 데이터를 캐시에 저장하지 않고 상위 레벨의 캐시로 전달하여, 캐시의 적중률을 높이는 기법이다[1-3]. 특히, 프로그램의 크기가 커지고, 멀티 코어 환경의 보편화 되면서, 최하위 레벨 캐시(last-level cache, LLC)의 적중률의 중요성이 커지고 있어서, 캐시 바이패싱 기법에 대한 연구가 더 활발히 진행되고 있다[1-3]. 캐시 바이패싱 기법의 효율성에 가장 크게 영향을 미치는 것은 저장하지 않은 캐시 블록을 예측하는 부분이며, 이 예측율을 높이기 위해 다양한 접근 방식이 제안되어 왔다.

이러한 예측 방식 중에 하나가 해당 캐시 블록의 재활용성(reuse distance)을 측정하여 이를 활용하는 방식이다[13]. 캐시 블록이 캐시에 저장되었을 때부터 지워질 때까지 접근 횟수를 저장하고, 자주 사용되지 않는 캐시 블록으로 판정되는 경우, 다음에 같은 캐시 블록이 접근되는 경우, LLC에 저장하지 않는다.

3. Wear-out 공격 탐지 기법

3.1 캐시 블록의 메모리 접근 패턴 분석

Wear-out 공격은 캐시의 수명을 단축시키기 위해 소수의 캐시 블록들에 극단적으로 많은 쓰기 동작을 집중적으로 수행시킨다. 따라서, 기존의 정상적인 프로그램 들에서 보이는 쓰기 패턴과는 다르게 쓰기 동작 외에 유의미한 동작이 적으며, 쓰기 동작은 짧은 기간에 좁은 영역에서만 일어난다(Fig. 2). 캐시 블록의 재활용성을 측정을 하는 기존의 기법들 에서는 주로 접근의 횟수만을 저장하여, 이를 바탕으로 바이패싱 여부를 결정하였다. 그러나, 이런 방식으로는 정상적인 쓰기 동작과 Wear-out 공격을 위한 쓰기 동작을 분리하기 어려우므로, 좀 더 정교한 모델이 필요하다. 이를 위해서 Fig. 3에서 설명하듯이, 쓰기 연속적으로 일어나는 횟수가 일정 횟수를 넘어서는 경우에 대해서만 쓰기_횟수를 더하고 이를 읽기 횟수로 나누어서 쓰기 집중도(write intensity)를 계산한다.

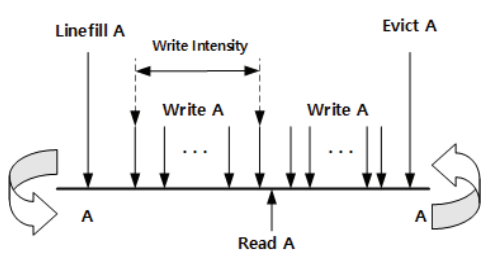

Fig. 2. Write intensity for detecting wear-out attack.

```
* Parameters:
S[i]: ith data access of the sequence of the target cache block
N: Number of accesses of the target cache block
W: Total number of write accesses
R: Total number of read accesses
Wp: Total number of write accesses of the current period
T: Threshold for wear-out attack

W←0, R←0, Wp←0
for i←1 to N:
    if S[i] == Write_access then
        Wp←Wp+1
    else // Read_access
        R←R+1
        if Wp > T then
            Wp←W+Wp
        end if
        Wp←0
    end if
end for
Write Intensity←W/R
```

Fig. 3. Memory access pattern analysis.

3.2 Wear-out 공격 방어를 위한 캐시 바이패싱 정책

Wear-out 공격의 대상이 되는 캐시 블록을 탐지한 후에는 캐시 바이패싱 정책을 적용한다. Fig. 4에서는 기존의 방식과 바이패싱 정책을 사용하는 경우의 캐시 블록의 쓰기 횟수를 측정하여 비교한 것이다. 캐시는 2개의 캐시 블록을 저장할 수 있으며, 캐시 블록 A,B,C에 접근한다. 처음에는 B와 C가 저장되어 있으며, A,A,B,A,A,B,C의 순서로 캐시 블록의 접근이 일어나는데, A는 공격 대상이 되는 캐시 블록이고, B와 C는 정상적인 쓰기 동작을 수행하는 캐시 블록이다. 기존의 방식으로는 A에 대한 접근이 자주 일어나므로, A에 캐시 접근 실패가 발생할 경우, 이를 다시 캐시에 저장하면서 다른 캐시 블록을 쫓아낸다.

Fig. 4(a)에서는 처음 A가 접근되었을 때, 캐시 적중 실패가 발생하며, 해당 캐시 블록이 다른 캐시 블록 대신 저장된다(①). 그리고, 다음 A에 대한 접근이 일어나면 캐시 적중이 일어난다(②). 그리고, B에 접근을 시도하게 되는데 이 때 다시 캐시 적중 실패가 발생하여 C의 자리에 저장된다(③). 이후 A,B에 대한 접근에 대해서는 모두 캐시 적중이 일어나며(④⑤⑥), C가 들어오면 다시 캐시 블록이 쫓겨난다(⑦). 결과적으로 캐시 적중 실패는 3번만 일어난다. 새로운 캐시 블록이 캐시에 들어오는 것은 쓰기 과정으로 취급되므로, 실제 쓰기 횟수는 A가 4번, B가 2번, C가 1번으로 총 7번의 쓰기가 일어난다. 그러나, A를 쓰는 동작은 공격을 위한 것으로 불필요한 쓰기 동작이다. 따라서, 3번만 유의미한 쓰기이며, 4번의 공격이 성공한 것이다.

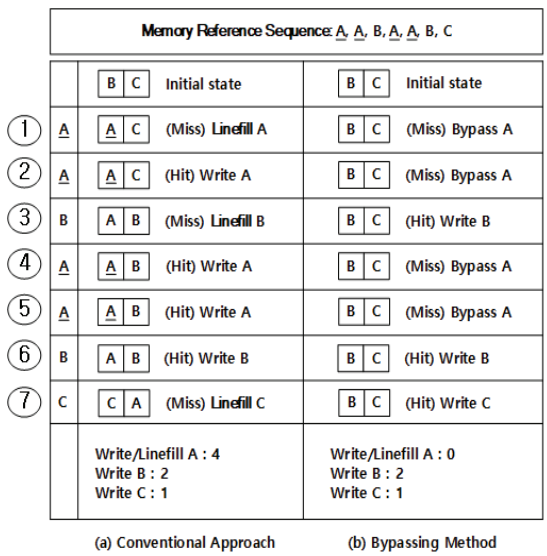

Fig. 4. Cache bypassing method. Block A is a target of wear-out attack and Block B/C are normal blocks.

반면, Fig. 4(b)에서는 처음 A가 접근되었을 때, A가 저장되지 않는다(①). 그래서, 다음 A에 대해서 캐시 적중 실패가 일어난다(②). 그러나, 그 다음 B에 접근을 시도할 때는 캐시 적중이 발생한다(③). 이후 A에 대한 접근에 대해서는 모두 캐시 적중 실패가 일어나지만, 바이패싱 되며(④⑤), B,C에 대한 접근은 성공한다(⑥⑦). 결과적으로 캐시 적중 실패는 4번이 일어나서, 실패의 횟수는 증가한다. 그러나, A에 대한 불필요한 쓰기를 회피할 수 있으므로, 전체적인 쓰기 횟수는 3번으로 줄어들어, 공격을 회피할 수 있다.

4. 실험 환경 및 결과

4.1 실험 환경

본 논문에서 제안한Wear-out 공격을 대비한 바이패싱 기법을 검증하기 위해 기존의 논문들에서 사용되었던 Gem5시뮬레이터를 수정하여 실험을 진행하였다[14]. 시뮬레이션에서는 전체 시스템의 캐시 구조를 다음과 같이 가정하였다. L1 캐시는 명령어 캐시와 데이터 캐시를 각각 32KB로 구성하였으며, L2 캐시는 512KB이고, LLC는

2MB이다. L1 캐시들과 L2 캐시는 SRAM이고, LLC는 PCM(Phase change memory)을 채택하였다. 구체적인 시스템 구성은 Table 1에 표시하였다. 실험을 위한 워크로드는 SPEC2006에서 일부 프로그램들을 선택하였다[15].

Table 1. Processor configurations

Architecture	x86, out-of-order, 2GHz
INT / MEM / FP	4/4/4
Branch Predictor	gshare predictor, 16 history length
ROB Size	256
I/D Cache	32KB, 4-way, 64B blocks, Read/Write: 1-cycle latency
L2 Cache	512KB, 8-way, 64B blocks, Read/Write: 5-cycle latency
LLC with PCM	2MB, 16-way, 64B blocks Read: 19-cycle latency Write: 93-cycle latency
Memory Latency	250 cycles

각 워크로드 자체에는 Wear-out를 위한 공격 코드가 없으므로, 공격 코드를 삽입하였으며, 공격 코드의 실행 비율을 여러가지로 변경시켜 가면서 실험을 진행하였다. 그리고, 캐시 수명은 하나의 캐시 블록에 마모도가 기준치를 넘은 셀이 4개 이상 발생할 때를 기준으로 한다.

본 논문에서 제안한 기법의 효율성을 비교하기 위해서, 비휘발성 메모리에서 사용하는 기본적인 쓰기 방지 기법들인 Read-before-write(RBW)과 Data inverting(DI)기법이 적용된 Baseline [16]과 다른 바이패싱 기법인 Hybrid bypassing block selector(HBS) [7]의 실험 결과도 함께 표시하였다. 단, HBS는 하이브리드 캐시 구조를 가정하고 있으므로, 정확한 비교를 위해서 SRAM은 사용하지 않도록 한다.

4.2 실험 방법

캐시의 수명을 정확하게 측정하기 위해서는 실제로 전체 시스템을 칩으로 설계해서 확인해야 하지만, 비용과 시간의 문제로 많은 연구자들은 시스템을 추상화(abstract)하고 필요한 부분만 시뮬레이션하여 제안된 기법들의 정확성 및 효율성을 검증해왔다. 그 중에서 가장 많이 사용되는 시뮬레이터가 Gem5이다[14]. 기존의 컴퓨터 시스템을 구성하는 중앙처리장치, 캐시, 버스, 메모리 시스템 및 저장장치까지 모두 매 클럭 사이클(clock cycle)단위로 정확하게 시뮬레이션하도록 구성되어 있다. 프로그램에서 수행되는 명령어를 해석하고, 이를 수행하는 것을 모사하고, 그 과정에서 캐시접근 횟수나 접근 실패율 등의 같은 통계적인 데이터를 추출할 수 있다. 본 논문에서는 캐시의 교체정책을 담당하는 부분을 기존의 LRU(Least Recently Used)방식에서 본 논문에 맞도록 수정하고, 각 캐시 라인의 접근 횟수를 바탕으로 수명을 추정하였다.

4.3 실험 결과

Fig. 5는 Wear-out 공격이 없을 때를 기준으로 정규화된(normalized) 캐시 수명을 구한 것이다. X축은 사용된 프로그램이고, Y축은 캐시 수명이 얼마나 단축되었는지를 비율로 표시한 것이다. Y축의 값이 1에 가까울 수록 Wear-out 공격에 대한 저항성이 높다는 것을 의미한다. 그림에서 _10, _20, _30, _40은 각 기법에서 공격이 시작된 후에 수명이 다할 때까지 전체 실행 시간 중에 공격코드가 실행되는 비율이 10%, 20%, 30%, 40%라는 것을 나타낸 것이다. 예를 들어, HBS_20은 HBS 기법에서 실행 시간 중에 Wear-out 공격 코드가 20%실행되었다는 의미이다.

Baseline에서는 공격 코드가 10%만 실행되어도 수명이 27% 단축되고, 20%에서는 77% 가까이 수명이 줄어들다가, 30%, 40% 공격 구간에서는 수명이 1/10에서 1/20로 줄어든

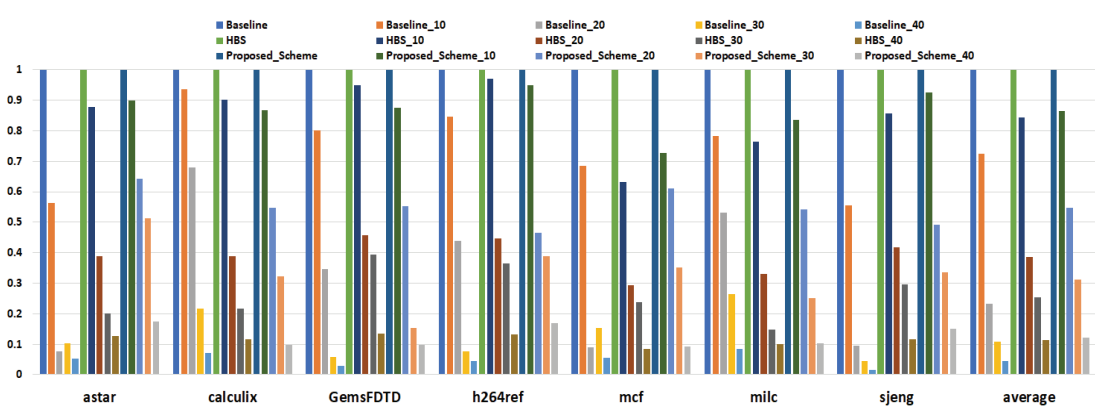

Fig. 5. Normalized lifetime.

다. 반면, HBS_10에서는 수명 저하가 15% 정도 수준이나, HBS_20에서는 수명이 62% 정도 줄어든다. HBS_30과 HBS_40에서는 각각 정규화된 수명이 0.25와 0.11 수준으로 떨어진다. 본 논문에서 제안한 기법은 10% 공격에는 13%의 수명 저하를 보이지만, 20% 공격에는 45%의 수명 저하를 보이고 있다. 30% 공격과 40% 공격에 대해서는 정규화된 수명이 0.31과 0.12가 된다.

Baseline과 HBS 및 본 논문의 기법과의 가장 큰 차이는 공격이 시작되면 Baseline은 급격하게 수명이 줄어든다는 것이다. 10% 공격에서도 이미1/4이상 수명이 단축되었으며, 20% 공격만 되어도 수명이 3/4이상 단축된다. HBS 및 본 논문의 기법은 기본적으로 쓰기가 집중된 캐시 블록을 찾아내는 기능이 있으므로, 어느 정도 공격에는 저항성을 가진다. 그러나, HBS도 이미 30% 공격에서 수명이 절반 이상 줄어들고 있으며, 40% 공격에서는 수명이 1/4 수준으로 떨어진다. 반면, 본 논문의 기법은 20% 공격에도 45%정도만 수명이 단축되고, 30% 공격에는 68%정도의 수명 저하를 보여, HBS에 비해서는 17%와 36%정도 높은 저항성을 보인다. 다만, 40% 공격에서는 비슷한 수명 저하를 보이는데, 이것은 이미 대부분의 쓰기가 공격에 의해 점유되고 있는 상황이라, 수명 저하를 방어하는데 한계를 보이기 때문이다.

실험 결과를 요약하면 다음과 같다. 기존이 기법들은 Wear-out 공격에 취약점을 노출하여 10% 또는 20%의 공격에도 특정한 캐시 라인에 집중적으로 쓰기 횟수가 집중되어 캐시의 마모도가 한계에 이르러서, 결국 시스템 전체의 고장으로 이어지는 시간이 급격하게 단축된다. 따라서, 전체적으로 시스템의 수명이 줄어든다. 그러나, 본 논문에서 제안한 기법은 30% 수준의 공격에서도 높은 저항성을 보이기 때문에, 시스템의 수명 자체가 유의미하게 늘어나게 된다.

5. 결 론

본 논문에서는 Wear-out 공격에 대해서 비휘발성 메모리의 저항성을 높이는 기법을 제안하였다. Wear-out 공격의 대상이 되는 캐시 블록은 정상적인 쓰기 패턴을 가진 캐시 블록에 비해서 쓰기 집중도가 높다. 따라서, 캐시 블록의 쓰기와 읽기 패턴을 조사하여, 공격이 의심되는 캐시 블록을 찾아낸다. 그리고, 이런 캐시 블록은 캐시에 저장하지 않고, 다른 레벨의 캐시 또는 메인 메모리에만 저장하여, 비휘발성 메모리 셀의 마모도를 낭비하지 않도록 한다. 제안된 기법의 효율성을 검증하기 위해 실험을 수행한 결과, Wear-out 공격으로 인한 수명 단축 현상을 다양한 공격 비율에 대해서 평균적으로 14%, 최대로는 36% 정도 줄일 수 있었다.

참고문헌

1. Jang, Minwoo, et al., “Defending against flush+ reload attack with DRAM cache by bypassing shared SRAM cache.” IEEE Access, Vol. 8, pp. 179837-179844, 2020.
2. Egawa, Ryusuke, et al., “A layer-adaptable cache hierarchy by a multiple-layer bypass mechanism.” Proceedings of the 10th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies. 2019.
3. Emara, Moustafa, and Bo-Cheng Lai, “Selective bypassing and mapping for heterogeneous applications on GPGPUs,” Journal of Parallel and Distributed Computing, Vol. 142, pp. 106-118, 2020.
4. Xu, Yuanchao, et al., “Asymmetry & Locality-Aware Cache Bypass and Flush for NVM-Based Unified Persistent Memory,” 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), pp. 168-175, 2019.
5. Choi, Juhee, “Low Power Scheme Using Bypassing Technique for Hybrid Cache Architecture,” Journal of the Semiconductor & Display Technology, Vol. 20, No. 4, pp. 10-15, 2021.
6. Wang, Guan, et al., “Shared Last-Level Cache Management and Memory Scheduling for GPGPUs with Hybrid Main Memory,” ACM Transactions on Embedded Computing Systems (TECS), Vol. 17, No. 4, pp. 1-25, 2018.
7. Tae Hyun Kim, Yang On, and Sang Yeon Jun, “Design of Asynchronous Non-Volatile Memory Module Using NAND Flash Memory and PSRAM,” Journal of the Semiconductor & Display Technology Vol. 19, No.3, pp. 118-123, 2020.
8. H.-S. Philip Wong et al., “Phase Change Memory,” in Proceedings of the IEEE, Vol. 98, No. 12, pp. 2201-2227, Dec. 2010.
9. Agarwal, Sukarn, and Hemangee K. Kapoor, “Improving the lifetime of non-volatile cache by write restriction,” IEEE Transactions on Computers Vol. 68, No. 9. pp. 1297-1312, 2019.
10. Esfeden, Hodjat Asghari, et al., “BOW: Breathing operand windows to exploit bypassing in GPUs,” 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 996-1007, 2020.
11. Seong, Nak Hee, Dong Hyuk Woo, and Hsien-Hsin S. Lee, “Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping,” ACM SIGARCH computer architecture news, Vol. 38, No. 3, pp. 383-394, 2010.
12. Cronin, Patrick, Chengmo Yang, and Yongpan Liu, “A

collaborative defense against wear out attacks in non-volatile processors," ACM/ESDA/IEEE Design Automation Conference (DAC), p. 1-6, 2018.
13. Zhou, Fang, et al., "VAIL: A Victim-Aware Cache Policy to improve NVM Lifetime for hybrid memory system," Parallel Computing, Vol. 87, pp. 70-76, 2019.
14. J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood, "gem5-gpu: A heterogeneous cpu-gpu simulator," IEEE Computer Architecture Letters, Vol. 14, No. 1, pp. 34–36, 2015.
15. J. Henning, "Spec cpu2006 benchmark descriptions," SIGARCH Comput. Archit. News, Vol. 34, No. 4, pp. 1–17, 2006.
16. Yongsoo Joo, Dimin Niu, Xiangyu Dong, Guangyu Sun, Naehyuck Chang, and Yuan Xie, "Energy- and endurance-aware design of phase change memory caches," 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), pp. 136-141, 2010.

접수일: 2022년 11월 24일, 심사일: 2022년 12월 9일, 게재확정일: 2022년 12월 13일