블룸 필터를 이용한 로우 해머링 방지 방법

김광래, 정기석*
한양대학교

kksilver91@hanyang.ac.kr, *kchung@hanyang.ac.kr

Avoiding Row-hammering Problem by Using Bloom Filters

Kim, Kwang-Rae and Chung, Ki-Seok*
Hanyang University, Seoul, Korea

요 약

최신 DRAM 공정 스케일링이 20nm 이하의 기술로 축소됨에 따라 인접한 DRAM 셀들이 서로 전자기적인 영향을 피하기가 더 어려워졌다. 그래서 만약 DRAM 셀들이 각각 적절하게 고립되지 않으면, 몇몇 셀로의 빈번한 접근 (Access)이 인접 셀에서의 의도하지 않은 비트 플립을 유발할 수 있다. 이 현상을 일반적으로 로우 해머링 (Row Hammering) 문제라고 한다. 특히, 악의적으로 생성된 많은 액세스 패턴으로 인해 보안 문제가 발생할 수 있다. 로우 해머링 문제를 해결하기 위해 널리 사용되는 두가지 방법이 있는데 하나는 카운터 기반 리프레시 (Refresh) 방법이고, 다른 하나는 확률적 리프레시 방법이 있다. 우리는 이 논문에서 블룸 필터를 이용한 확률적 솔루션을 제안하고자 한다. 실험 결과, 우리가 제안한 모델이 파라 (PARA) 모델이 로우 해머링을 막기 위해 하는 추가 리프레시 (Additional Refresh) 개수의 약 0.39배의 추가 리프레시만으로 로우 해머링을 완벽하게 막을 수 있었다.

Ⅰ. 서 론

Dynamic Random Access Memory (DRAM) 공정 스케일링 기술이 20nm 이하 공정 기술까지 축소되었다. 이렇게 증가된 DRAM 밀도로 인해 메모리 비용이 절감되고 메모리 성능이 향상되었으나 셀의 고밀도화에 따라 두 가지 우려가 생긴다. 첫째로 작은 셀은 제한된 양의 전하만을 갖기 때문에 노이즈 마진이 줄어든다. 둘째로 더 인접해진 셀들 간의 거리로 인해 전자기적인 커플링 효과로부터 보호하기가 매우 어려워진다. 결과적으로, 메모리 셀은 다양한 신뢰성 문제를 겪게 된다. 특히, 일부 셀에 대한 빈번한 접근 (access)은 인접한 셀에서 의도하지 않은 비트 플립으로 이어질 수 있다. 이러한 현상을 흔히 로우 해머링 (Row Hammering) 문제라고 한다. 특히, 악의적으로 생성된 과도한 접근 패턴 때문에 심각한 보안 문제가 발생할 수 있다. 최근 몇 가지 연구에서 로우 해머링 공격에 대한 심각한 보안 문제가 나타났었다[1]-[5]. IT업계에서 중요한 보안 문제로써 로우 헤머링 공격으로부터 방어를 하는 것을 시도해왔다. 이 문제를 해결하기 위한 접근 방식으로는 확률적 리프레시 (refresh) 방법과 카운터 기반 리프레시 방법이 있다. 확률적 리프레시 방법은 메모리 컨트롤러의 난수 발생기에 의해 생성된 난수를 이용해서 확률적 리프레시를 수행함으로써 로우 해머링 공격을 하는 공격자 로우 (aggressor row)로부터 희생자 셀들 (victim cell)을 보호한다. 이 문제를 확률적으로 해결하기 때문에, 모든 희생자 셀들을 로우 해머링 공격으로부터 보호하는 것을 보장할 수는 없다. 한편, 카운터 기반 리프레시 방법은 카운터를 사용하여 각 행의 로우 해머링 공격을 추적한다. 특정 로우의 접근 카운터 값이 특정 값에 도달하면 주변의 셀 값을 유지하기 위해 리프레시를 수행한다. 따라서, 카운터 기반 리프레시 방법은 로우 해머링 공격으로부터 보호를 보장할 수 있다. 그러나 카운터를 관리하기 위해서 상당한 오버헤드가 발생할 수 있다는 문제가 있다[3]. 그래서 낮은 오버헤드를 가진 확률적 리프레시 방법이 더 일반적으로 채택되었다. 대표적인 확률적 리프레시 방법으로 접근할 때마다 인접 로우들 중 하나를 일정 확률로 추가 리프레시를 해주는 파라 (PARA) 모델이 있다[1]. 최적의 확률적 리프레시의 조건은 로우 해머링이 최대한 발생하지 않도록 하는 것과 에너지 소모를 최소화하기 위해 추가적인 로우 리프레시을 줄이는 것이다.

따라서 본 논문에서는 특수한 필터를 적용한 확률적 리프레시 방법을 사용하여 적은 추가 로우 리프레시로 로우 해머링 문제를 완화하는 새로운 방법을 제안하고자 한다. 이를 위해서, 제안된 모델과 파라 모델을 구현한 후, 로우 해머링 감소 효과와 추가 로우 리프레시의 수를 비교하였다.

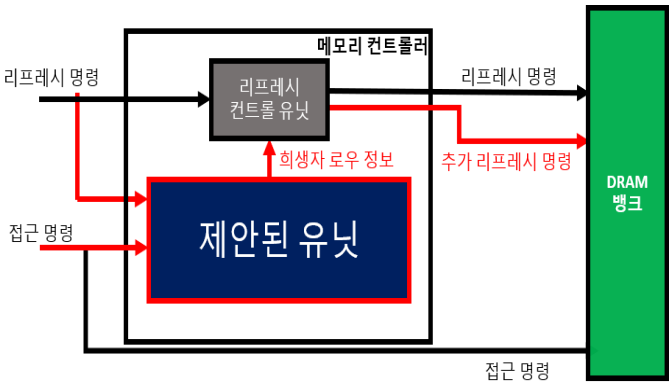

그림 1. 제안된 모델 및 동작

Ⅱ. 본론

로우 해머링 완화를 위해 제안된 방법은 이전 절에서 설명한 확률적 리프레시 방법에 기반을 둔다. 그림 1에서 볼

수 있듯이 제안된 유닛은 잠재적으로 위험할 수 있는 희생자 로우들의 정보를 리프레시 제어 장치에 제공하도록 설계되었다. 그런 다음, 리프레시 컨트롤 유닛이 해당 DRAM 뱅크로 추가 리프레시 명령을 전송한다. 이 방법은 메모리 컨트롤러를 약간 수정하는 것으로 구현이 가능하다. 그럼으로써 이 제안된 유닛은 낮은 오버헤드로 희생자 로우를 찾는다. 이 제안된 방법이 낮은 오버헤드로 로우 해머링을 완화할 수 있도록 하는 핵심 구성요소로 제안된 유닛 안의 블룸 필터 (bloom filter)[7]로 볼 수 있다. 블룸 필터는 큰 데이터를 해시 함수를 통해 작은 공간만으로 저장하는 것을 가능하게 하는 구조이기 때문에 효율적이다. 이 블룸 필터는 희생자 로우의 정보가 저장되어 있는 지를 확인하는 작업, 필터에 있는 희생자 로우의 정보를 업데이트하는 작업, 그리고 필터에서 희생자 로우의 정보를 삭제하는 세 가지 작업을 수행한다. 제안되는 유닛은 매 행이 접근할 때마다 이런 확인 작업을 통해 인접 로우의 정보가 저장되어 있는 지를 확인하고 저장 여부에 따라 그 인접 로우에 추가 리프레시 명령을 할 확률을 결정한다. 그 접근하는 로우의 인접한 희생자 로우들의 정보를 고정된 확률로 블룸 필터에 업데이트 해준다. 마지막으로 만약 어떤 희생자 로우가 추가적인 리프레시를 통해서 안전해지면, 삭제 작업을 통해서 필터 안의 희생자 로우의 정보를 삭제해준다.

제안된 모델이 얼마나 효율적으로 로우 해머링을 감소시켰는 지를 분석하기 위해 DRAMSim2[6] DRAM 시뮬레이터를 사용하였으며 한 로우에 몇 번 접근을 해야 로우 해머링이 일어나는 지를 결정하는 지표인 로우 해머링 문턱 (Row Hammering Threshold)는 8K로 설정하였다.

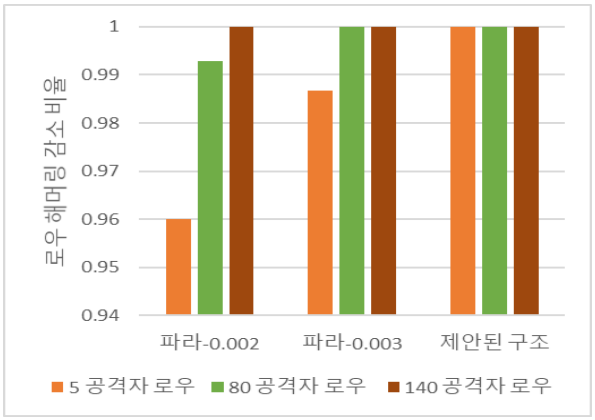

그림 2. 로우 해머링 감소 비율 비교 분석 결과

그림 2는 각 모델에서 공격자 로우의 개수에 따라서 로우 해머링 감소 비율을 비교한 것이다. 파라-0.002와 파라-0.003는 한 로우에 접근할 때마다 각각 0.002와 0.003의 고정된 확률로 인접한 로우 중 하나를 리프레시하는 파라 모델을 의미한다. 결과를 보면 우리가 제안한 모델이 공격자 로우의 개수와 관계없이 파라 모델보다 로우 해머링을 더 잘 방어하는 것을 관찰할 수 있다.

그림 3은 각 모델에서 공격자 로우의 개수에 따른 평균 추가 리프레시의 개수를 비교한 것이다. 제안된 모델에서의 추가 리프레시 개수는 파라-0.003에서의 추가 리프레시 개수의 0.39배정도로 다른 비교군들에 비해 현저히 적은 것을 볼 수 있다. 추가 리프레시는 DRAM의 성능의 저하와 에너지의 소모를 유발할 수 있기 때문에 추가 리프레시를 최소화하는 것은 중요한 문제이다. 결과적으로, 이 실험 결과는 우리의 제안된 모델이 비교군들 중에서 적은 추가 리프레시로 로우 해머링을 완벽하게 막을 수 있다는 것을 보여준다.

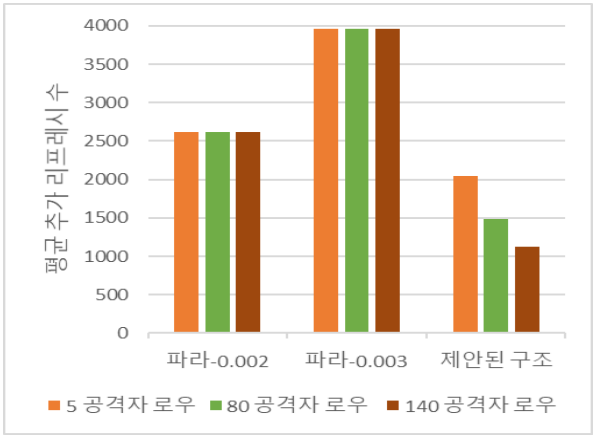

그림 3. 평균 추가 리프레시 수 비교 분석 결과

Ⅲ. 결론

DRAM의 보안 및 신뢰성 문제로써 로우 해머링 문제는 DRAM의 스케일링 기술이 더 축소되면서 더 심각해지고 있다. 이를 막기 위한 여러가지 접근 중 하나가 확률적 리프레시 방법이다. 확률적 리프레시 방법은 낮은 오버헤드로 로우 해머링을 방어하기에 효율적인 방법이다. 본 논문에서는 필터를 사용하여 더 효율적으로 로우 해머링을 방어하는 방법을 제시하였다. 그리고 시뮬레이션을 통해 우리가 제안된 모델이 이전의 연구와 비교했을 때 더 적은 추가 리프레시로 로우 해머링을 모두 막을 수 있는 것을 보여주었다.

ACKNOWLEDGMENT

이 논문은 2019년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (N0001883, 2019년 산업전문인력역량강화사업)

참 고 문 헌

[1] Kim, Yoongu, et al. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors." *ACM SIGARCH Computer Architecture News*. Vol. 42. No. 3. IEEE Press, 2014.

[2] Son, Mungyu, et al. "Making DRAM stronger against row hammering." *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017.

[3] You, Jung Min and Joon-Sung Yang. "MRLoc: Mitigating Row-hammering based on memory Locality." *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019.

[4] Kim, Dae-Hyun, Prashant J. Nair, and Moinuddin K. Qureshi. "Architectural support for mitigating row hammering in DRAM memories." *IEEE Computer Architecture Letters* 14.1 (2014): 9-12.

[5] Aweke, Zelalem Birhanu, et al. "ANVIL: Software-based protection against next-generation rowhammer attacks." *ACM SIGPLAN Notices* 51.4 (2016): 743-755.

[6] Rosenfeld, Paul, Elliott Cooper-Balis, and Bruce Jacob. "DRAMSim2: A cycle accurate memory system simulator." *IEEE computer architecture letters* 10.1 (2011): 16-19.

[7] Liu, Jamie, et al. "RAIDR: Retention-aware intelligent DRAM refresh." *ACM SIGARCH Computer Architecture News*. Vol. 40. No. 3. IEEE Computer Society, 2012.