DRAM 로우 해머링 방지를 위한 확률 기반 추가 로우 리프레시의 최적 확률을 찾는 방법

우정현, 정기석*

한양대학교

jhwoo1007@hanyang.ac.kr, *kchung@hanyang.ac.kr

A Method to Find the Optimal Probability for Probability-driven Additional Row Refresh to Prevent DRAM Row Hammering

Woo, Jeong-Hyun, Chung, Ki-Seok*

Hanyang University, Seoul, Korea

요 약

DRAM 공정 기술의 스케일링은 더 큰 용량의 DRAM 생산을 가능하게 하였지만, DRAM의 신뢰성 또한 저하시켰다. 이로 인한 대표적인 문제로 같은 로우를 여러 번 엑티베이트 (Activate) 함에 따라 이웃한 로우의 셀에 저장된 값이 원치 않게 변하는 현상인 로우 해머링 (Row Hammering)이 있다. 로우 해머링은 DRAM이 기본적으로 수행하는 오토 리프레시 (Auto Refresh)만으로는 해결되지 않기 때문에 추가의 해결책이 필수적이다. 본 논문에서는 이 로우 해머링 문제를 극복하기 위한 방법 중의 하나인 확률 기반 추가 로우 리프레시 기법에서, 로우 해머링이 발생하지 않도록 하면서 추가의 리프레시에 의한 성능 저하와 에너지 소모를 최소로 하는 추가 로우 리프레시를 해야 하는 최적의 확률을 찾는 방법을 제시한다. 실험 결과, 로우 해머 문턱이 4K인 경우 추가 로우 리프레시 확률은 0.006이 최적임을 알 수 있었다.

Ⅰ. 서론

Dynamic Random Access Memory (DRAM)는 주로 현대 컴퓨팅 시스템의 주기억장치로 사용된다. DRAM은 하나의 캐패시터와 트랜지스터로 구성된 셀에 데이터를 저장한다. 계속된 DRAM 공정 기술의 스케일링 덕분에 DRAM의 용량은 계속해서 증가할 수 있었지만, 이는 DRAM의 신뢰성을 심각하게 저하시켰다. 이러한 이유로 발생하는 대표적인 예로 DRAM의 로우 해머링 (Row Hammering)이 있다[1].

로우 해머링은 DRAM이 같은 로우를 여러 번 엑티베이트 (Activate) 함에 따라 원치 않게 이웃한 로우의 셀에 저장된 값이 변하는 문제이다. DRAM이 기본적으로 수행하는 오토 리프레시 (Auto Refresh)만으로는 로우 해머링이 해결되지 않기 때문에 추가의 해결책이 필수적이다. 이를 해결하기 위한 간단한 방법으로는 DRAM의 모든 로우를 더 자주 리프레시 해주는 방법이 있다. 하지만 이러한 방법은 성능과 에너지 면에서 심각한 오버헤드를 야기한다는 문제점이 있다. 더 효율적인 해결을 위해서 이전의 연구들은 두 가지 하드웨어 기반 해결책을 제시하였다. 첫 번째는 계수기 (Counter)를 활용해 엑티베이트 횟수가 로우 해머링을 발생시키는 로우 해머링 문턱 (Row Hammering Threshold)에 다다르면 인접한 로우를 추가적으로 리프레시 하는 방법이다. 이러한 결정론적 해결책은 어떠한 경우에도 로우 해머링을 방지할 수 있다는 장점이 있지만, 계수기를 추가함에 따라 발생하는 에너지와 비용이 상당하다는 문제점이 있다. 또 다른 해결책으로는 확률 기반 해결책이 있다 [1, 4]. 확률 기반 해결책은 정해진 확률에 따라 매 엑테베이트마다 엑티베이트된 로우의 인접한 로우 중 하나를 추가적으로 리프레시 하는 방법이다. 확률 기반 해결책은 구현이 단순하고 추가적인 비용도 적다는 장점이 있다. 다만, 추가 로우 리프레시 확률이 높아질수록 더 큰 로우 해머링 방지 효과를 거둘 수 있지만, 이에 따라 성능과 에너지 오버헤드가 증가한다는 문제점이 있다. 따라서 확률 기반 기법을 최적으로 적용하기 위해서는 로우 해머링이 발생하지 않도록 하면서 추가의 리프레시에 의한 성능 저하와 에너지 소모를 최소로 하는 것이 중요하다. 결국은, 이런 요구 조건을 만족하는 추가 로우 리프레시를 해야 하는 최적의 확률을 찾는 것이 핵심이라고 할 수 있다..

본 논문에서는 효율적인 확률 기반 해결책의 활용을 위한 최적의 추가 로우 리프레시 확률을 찾는 방법을 제시한다. 이를 위해서, 로우 해머링을 발생시키는 악의적인 패턴을 만들고, 다양한 추가 로우 리프레시 확률이 적용된 경우의 로우 해머링 감소 효과를 비교하였다.

표 1. 악의적인 패턴

패턴	설명	예
패턴 1	반복되는 임의로 선택된 N개의 로우	$(X_1, X_2, \cdots, X_N)^*$
패턴 2	반복되는 임의로 선택된 N개의 로우 + 임의로 선택된 로우	$X_1, 4, 14, X_2, \cdots, X_N, \cdots$
패턴 3	이웃한 로우	$(Y_1-1, Y_1+1, \cdots, Y_N-1, Y_N+1)^*$
패턴 4	이웃한 로우 + 임의로 선택된 로우	$Y_1-1, 8, 10, Y_1+1, \cdots, 20, \cdots, Y_N+1, \cdots$
패턴 5	반복되는 임의로 선택된 N개의 로우 + 이웃한 로우	$X_1, X_2, Y_1-1, X_3, \cdots, Y_N-1, Y_N+1, \cdots, X_N, \cdots$

Ⅱ. 본론

최적의 추가 로우 리프레시 확률을 찾기 위해서, 먼저 이전에 제시된 로우 해머링 기반의 공격 시나리오들[5-7]을 기반으로 로우 해머링을 발생시키는 악의적인 패턴을 만들었다. 표 1은 실험에 사용된 5가지의 악의적인 패턴을 보여준다. 패턴 1은 임의로 선택된 N개의 로우를 순차적으로 반복해서 접근하는 경우이다. 패턴 2는 패턴 1에 임의로 선택된 로우들을 혼합해 만들었다. 패턴 3은 임의로 선택된 N개 로우의 이웃한 로우들을 순차적으로 반복해서 접근하는 경우이다. 패턴 4는 패턴 3에 임의로 선택된 로우들을 혼합하여 만들었다. 패턴 5는 패턴 1과 패턴 3을 혼합하여 만들었으며 본 논문에서는 N 값으로 8을 사용하였다.

추가 로우 리프레시 확률에 따른 로우 해머링 감소 효과를 분석하기 위해 DRAM 시뮬레이터로 주로 사용되는 DRAMSim2[8]를 사용하였으며 DRAM AC 파라미터는 [9]에서 주어진 값을 사용하였다. 로우 해머링 문턱 값은 4K로 설정하였다.

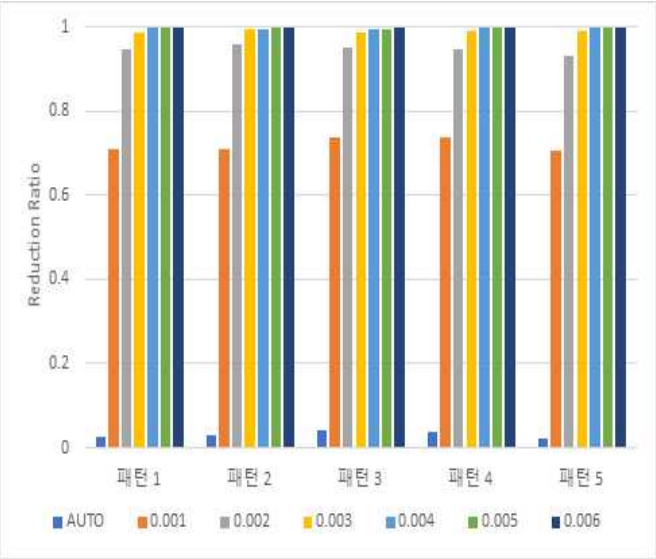

그림 1. 악의적인 패턴에 따른 로우 해머링 감소 효과 분석 결과

그림 1은 5가지의 제작된 악의적인 패턴에서 별도의 로우 해머링 해결책이 추가되지 않고 기존의 오토 리프레시만 사용하는 경우와 다양한 추가 로우 리프레시 확률이 적용된 경우의 로우 해머링 감소 효과를 비교한 결과이다. 에너지 소모와 성능 저하를 최소화하며 모든 로우 해머링을 해결하는 경우를 찾기 위해 추가 로우 리프레시 확률을 0.001부터 시작해 로우 해머링이 모두 해결될 때까지 0.001씩 증가시키며 실험을 진행하였다. 먼저 기존의 오토 리프레시만 사용한 경우에는 로우 해머링 감소 효과가 거의 없음을 알 수 있었다. 따라서 로우 해머링을 막기 위해서는 별도의 해결책이 필수적이다는 것을 확인할 수 있었다. 확률 기반 해결책이 적용된 경우 추가 로우 리프레시 확률이 0.001인 경우에는 모든 패턴에서 약 30%에 가까운 로우 해머링을 해결하지 못함을 알 수 있었다. 추가 로우 리프레시 확률이 높아짐에 따라 모든 패턴에서 로우 해머링 감소 효과가 커짐을 확인할 수 있었다. 추가 로우 리프레시 확률이 0.005인 경우에 대부분의 패턴에서 모든 로우 해머링을 해결하였지만, 패턴 3의 경우에 로우 해머링이 여전히 발생함을 알 수 있었다. 추가 로우 리프레시 확률이 0.006인 경우가 되어서야 모든 패턴에서 로우 해머링이 발생하지 않음을 알 수 있었고, 따라서 로우 해머링 문턱이 4K인 경우에는 추가 로우 리프레시 확률이 0.006이 최적의 값임을 알 수 있었다.

Ⅲ. 결론

로우 해머링 문제는 DRAM의 신뢰성을 심각하게 훼손하게 하는 대표적인 예이다. 확률 기반 추가 로우 리프레시 기법은 이 문제를 해결하기 위한 효과적인 기법 중의 하나이다. 이 방법에서는 추가 리프레시를 통하여 로우 해머링의 발생을 막으면서, 동시에 이로 인한 성능 저하와 에너지 소모를 최소화하기 위해서 추가로 리프레시를 해야 할 최적의 확률을 구하는 것이 가장 중요한데, 본 논문에서는 이 최적의 추가 로우 리프레시 확률을 찾는 방법을 제시했다. 본 논문에서는 로우 해머링을 발생시키는 악의적인 패턴을 만들고, 이를 기반으로 다양한 추가 로우 리프레시 확률에 따른 로우 해머링 감소 효과를 분석해 최적의 추가 로우 리프레시 확률을 찾는 방법을 연구하였다. 시뮬레이션을 통해 로우 해머링 문턱이 4K일 때 추가 로우 리프레시 확률은 0.006이 최적임을 알 수 있었다.

ACKNOWLEDGMENT

이 논문은 산업통산자원부 '산업전문인력역량강화사업'의 재원으로 한국산업기술진흥원(KIAT)의 지원을 받아 수행된 연구임. (2018년 지능형반도체 전문 인력 양성사업, 과제번호 :N0001883

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] Kim, Dae-Hyun, et al. "Architectural support for mitigating row hammering in DRAM memories." IEEE Computer Architecture Letters 14.1 (2015): 9-12

[3] Seyedzadeh, Seyed Mohammad, et al. "Counter-based tree structure for row hammering mitigation in DRAM." IEEE Computer Architecture Letters 16.1 (2017): 18-21

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

[5] Seaborn, Mark, and Thomas Dullienl. "Exploiting the DRAM rowhammer bug to gain kernel privileges." Black Hat 15 (2015)

[6] Gruss, Daniel, et al. "Rowhammer. js: A remote software-induced fault attack in javascript." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2016.

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

[8] Rosenfeld, Paul, et al. "DRAMSim2: A cycle accurate memory system simulator." IEEE Computer Architecture Letters 10.1 (2011): 16-19.

[9] DDR3L SDRAM MT41k1G8 [Online].Available:http://www.micron.com/~/media/documents/products/data-sheet/dram/ddr3/8gb_ddr3l.pdf