DRAM 의 Activation 횟수를 이용한 Row Hammering 개선 방법

김나연, 김광래, 정기석
한양대학교

wingedna@hanyang.ac.kr, kng212@hanmail.net, kchung@hanyang.ac.kr

Improved Row Hammering Prevention Capability using Activation Count

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

요 약

DRAM 공정 기술의 스케일링이 10nm 대로 진입함에 따라 인접한 DRAM 셀들 간의 전자기적 영향이 증가하며, 이로 인해 DRAM 의 신뢰성이 저하되고 있다. 이 신뢰성 저하의 대표적인 문제 중의 하나로, DRAM 의 뱅크 내의 같은 로우를 여러 번 집중적으로 액티베이트함에 따라 인접한 로우의 셀에 저장된 값이 의도치 않게 플립되는 현상인 로우 해머링(Row Hammering)이 있다. 본 논문에서는 액티베이트의 횟수가 정해진 시간 간격 내에 접근 한계 값을 초과하면 내부의 카운터 주소가 아닌 마지막 액티베이트 주소를 저장하여 인접한 로우를 추가 리프레시하는 방안을 제안하고, 이를 위한 적합한 접근 한계 값을 제시한다. 실험 결과, 우리가 제안한 모델은 MRLoc 모델에 비해 37.9% 적은 추가 리프레시로 로우 해머링을 완벽하게 막을 수 있었다.

Ⅰ. 서 론

Dynamic Random Access Memory (DRAM)의 저장 공간인 셀은 메모리의 용량을 높이기 위해서 더욱 밀접하게 배치할 수 있도록 공정 기술이 발전하고 있다. DRAM 의 행의 밀도가 증가함에 따라 메모리 성능이 향상되지만 인접한 행의 신뢰성 문제를 유발하는 로우 해머링(Row Hammering)이 발생한다. 로우 해머링은 DRAM 의 특정 행이 집중적으로 자주 액세스될 때 인접한 행의 셀 값이 의도치 않게 비트 플립되는 현상으로 최근에는 악의적으로 로우 해머링을 발생시키는 패턴 발생 등의 공격 등에 의한 심각한 보안 문제의 원인이 되었다[1]-[3].

로우 해머링을 해결하기 위해서는 정해진 시간 동안, 특정 로우가 로우 해머링 문턱 값 (Row Hammering Threshold) 보다 더 많이 액티베이트 (Activate) 되기 전에 비트 플립이 발생할지 모르는 로우를 추가적으로 리프레시 해줘야 한다. 리프레시할 로우를 안전하게 많이 선택하기 보다는, 리프레시할 때 전력 소모 및 성능 저하가 발생하므로 추가 리프레시를 최소화해야 리프레시로 인한 과도한 오버헤드를 방지할 수 있다. 따라서 본 논문에서는 DRAM 의 Bank 별로 로우 액티베이트 접근 횟수를 카운트하여 접근 한계 값 (Limited Access) 이상이 되면 DRAM 으로 액티베이트를 초과하였다는 신호를 보내 로우 해머링을 대응할 수 있는 방법을 제안한다. 접근 한계 값은 DRAM 에 비트 플립을 일으키지 않는 최대의 액티베이트 횟수로 미리 메모리 컨트롤러에 저장하여 접근 횟수 모니터링에 활용한다. 제안한 모델은 최근 발표된 MRLoc[4] 모델에 비해 37.9% 적은 추가 리프레시로 로우 해머링을 완벽하게 막을 수 있었다.

Ⅱ. 본 론

로우 해머링을 대응하기 위한 추가적인 리프레시는 일반적으로 확률적 리프레시 방법에 기반을 둔다. 그림 1 과 같이 시스템의 호스트가 보내는 뱅크 액티베이트 명령어의 횟수를 카운트할 수 있는 카운터 모듈을 설계하여 메모리 컨트롤러에 배치한다. 뱅크 별로 액티베이트 명령어를 카운트하여 레지스터에 미리 저장되어 있는 접근 한계 값을 초과하면 DRAM 에 접근 회수 초과 신호를 전송한다. DRAM 은 접근 횟수 초과 신호를 받으면 로우 액티베이트 주소를 DRAM 내의 래치 회로에 저장하도록 한다.

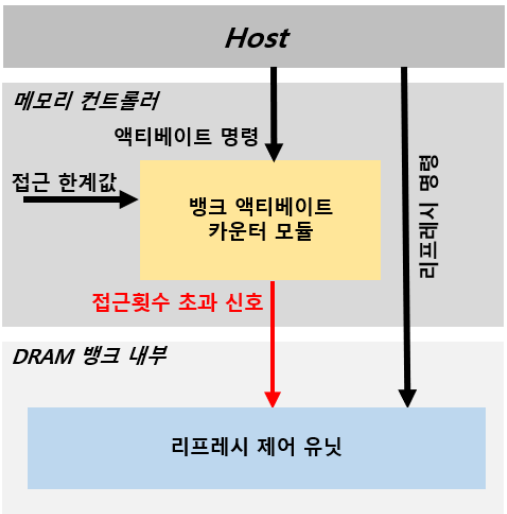

그림 1. 제안된 모델

그림 2 는 제안된 모델의 동작의 한 예시를 보여준다. DRAM 은 tREFI 마다 진입하는 리프레시 동작 시 내부의 카운터를 이용하여 뱅크의 로우를 순차적으로 리프레시시키지만 접근회수 초과 신호를 받으면 카운터 주소가 아닌 액티베이트시 래치에 저장된 주소의 인접한 행이 비트 플립이 발생하지 않도록 추가적으로 리프레시되도록 한다. 뱅크에서 액티베이트 되는 횟수가 증가할수록 래치에 확률이 높기 때문에 로우 해머링을 막을 수 있는 확률이 증가한다. 해당 행에 대해 추가 리프레시가 끝나면, 접근 횟수 초과 신호를 초기화하여 다음 접근 횟수

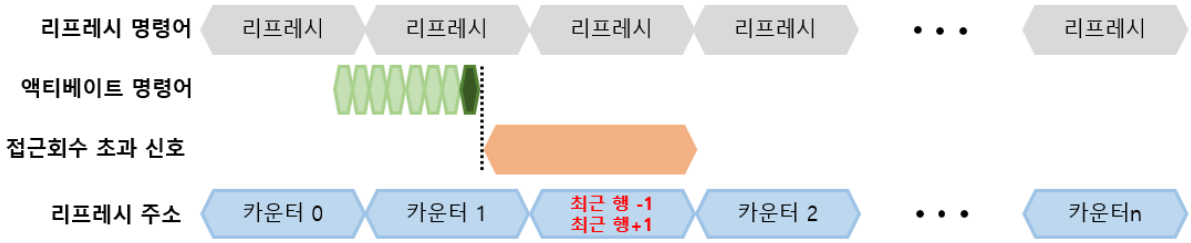

그림 2. 제안된 모델의 동작

초과 신호를 메모리 컨트롤러로부터 받을 수 있도록 한다. 이 때, 비트 플립이 발생할 가능성이 있는 로우를 흔히 피해 행이라고 부르며, 이 때 몇 개의 피해 행에 대해 추가 리프레시를 실행할지를 결정하는 것은 중요한 문제이다. 결국, 로우 해머링에 의한 비트 플립을 방지하는 방법의 성능 평가는 비트 플립이 일어나지 않도록 하는 것이 가장 중요하지만, 이를 보장하면서도 최소 횟수의 추가 리프레시가 발생하는 지를 판단하는 것이 중요하다.

로우 해머링 문턱 값 설정에 따른 비트 플립을 방지하기 위한 추가 리프레시 횟수를 분석하기 위해 DRAM 시뮬레이터로 주로 사용되는 DRAMSim2[5] 시뮬레이터를 사용하였으며 DRAM 의 주요 파라미터들은 JEDEC 에서 지정한 DDR4 파라미터[6]을 사용하였다. 한 로우에 몇 번 접근해야 비트 플립이 일어나는지 결정하는 지표인 로우 해머링 문턱은 4K 로 설정하였다. 본 논문에서는 액티베이트시 집중적으로 액티베이트되는 로우 수를 다양하게 변경해가면서 피해 행의 비트 플립 발생을 줄일 수 있는 최적의 로우 해머링 문턱 값을 찾는 실험을 진행하였다.

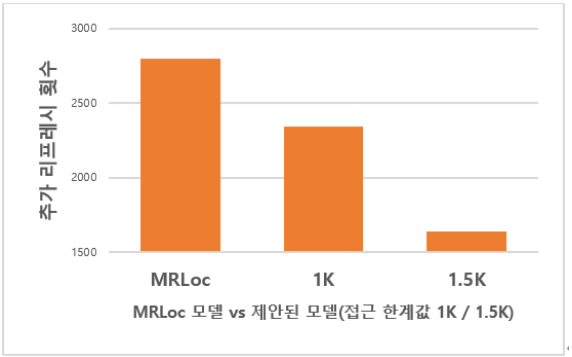

그림 3. 로우 해머링 문턱에 따른 추가 리프레시 비교

표 1 은 다섯 가지의 피해 행 수에 따른 로우 해머링에 의한 비트 플립 발생 빈도의 비교 결과를 보여준다. 최신 로우 해머링 방지 모델 중의 하나인 MRLoc 모델과 제안된 모델의 접근 한계 값을 1K, 1.5K, 2K 로 변경하면서 실험하였다. 제안하는 방법이 비교적 적은 수의 피해 행에 대한 추가 리프레시를 통해서도 비트 플립이 발생하지 않는 것을 알 수 있다. 접근 한계 값을 작게 설정할수록 로우 해머링이 발생하지 않는 경향을 보이며 접근 한계 값을 2K 로 했을 때 로우 해머링이 발생하였다. 따라서 접근 한계 값은 1.5K 적합하다는 것을 보여준다.

표 1. 피해행 수에 따른 비트 플립 발생 빈도 비교

(단위 : 행 개수)

	MRLoc	제안된 구조		
		접근 한계값1K	접근 한계값1.5K	접근 한계값2K
case1(피해행4개)	4	0	0	4
case2(피해행8개)	8	0	0	8
case3(피해행12개)	12	0	0	12
case4(피해행16개)	16	0	0	16
case5(피해행20개)	20	0	0	20

그림 3 은 MRLoc 모델과 제안된 모델의 접근 한계 값에 따른 로우 해머링 대응을 위한 추가 리프레시 횟수를 나타낸다. 제안된 구조는 MRLoc 에 비해 적은 추가 리프레쉬로 로우 해머링에 대한 대응이 가능하며 로우 해머링 문턱 값을 크게 설정할수록 추가 리프레시 횟수가 감소함을 확인할 수 있다. 실험 결과에 따르면 접근 한계 값은 로우 해머링에 의한 비트 플립을 발생시키지 않는 1.5K 로 설정하였을 때 MRLoc 모델에 비해 37.9% 적은 추가 리프레시로 로우 해머링을 완벽하게 대응함을 보여준다.

Ⅲ. 결론

로우 해머링은 DRAM 의 신뢰성을 훼손하는 대표적인 원인이며 이에 대한 다양한 연구가 진행되고 있다. 로우 해머링을 막기 위한 여러 가지 연구 중 하나가 확률적 리프레시 방안이고 본 논문에서는 더 효율적인 방법으로 로우 해머링에 의한 의도치 않은 비트 플립을 막을 수 있는 방법을 제시하였다. DRAM 시뮬레이션을 통해 우리가 제안한 모델이 기존의 연구와 비교했을 때 37.9% 더 적은 리프레시로 로우 해머링에 의한 비트 플립을 막을 수 있다는 것을 보여주었다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.2021-0-00131, 제조검사장비 경량화를 위한 지능형 엣지컴퓨팅 반도체 개발)을 받아 수행된 연구임

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] 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.

[4] 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.

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

[6] JEDEC, “DDR4 SDRAM,” JESD79-4C,2017.