RowHammer: A Retrospective

Jongchan Woo

11/09/2020

Massachusetts Institute of Technology

Outline

- Motivation
- Background
- Discussion (Strengths/Weaknesses)
- RowHammer-based Attack
- Mitigation Techniques
- Circuit-level Studies
- Other Works
- RowHammer in a Broader Context
- Future Work

Motivation

- Scaling of DRAM process technology

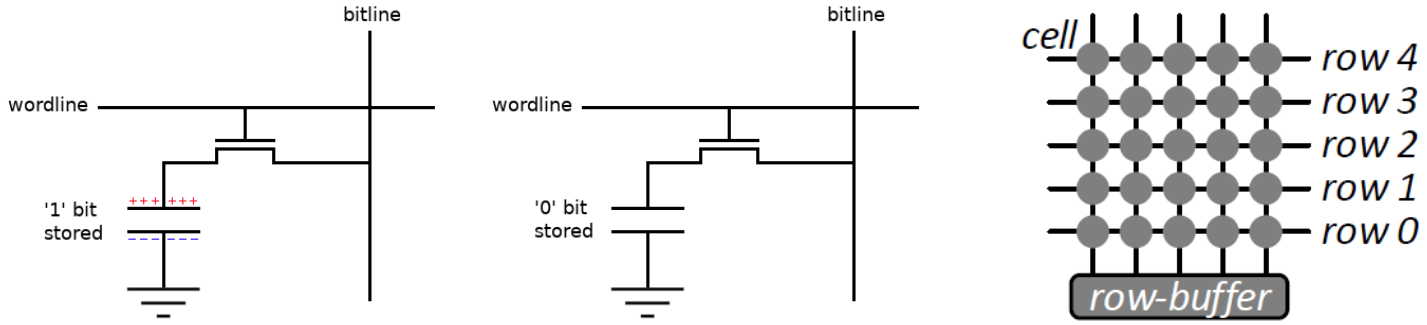
  🙂 Reduce cost-per-bit of memory

  ☹ Memory reliability issue

    - Small cell has limited amount of charge → reduced noise margin
    - Close proximity of cells → electromagnetic coupling effects cause undesirable interaction

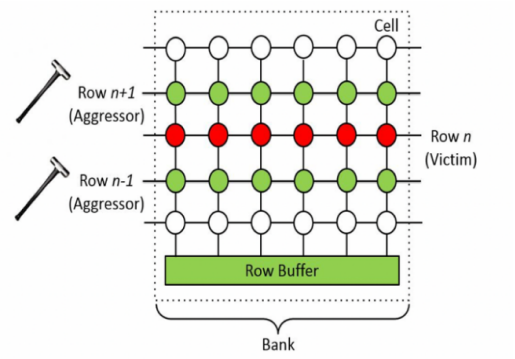
  – Disturbance errors exist in ***commodity*** DRAM chips in the form of RowHammer

Background



- DRAM
  - 1T 1C Cell composition
  - Wordline (0 → 1) connects the capacitor to bitline.
  - Write to bitline (0 or 1) / Read by sense amplifier and stored in row-buffer
  - All access to the row are served by the row-buffer

Background




```
1 code1a:
2   mov (X), %eax
3   mov (Y), %ebx
4   clflush (X)
5   clflush (Y)
6   mfence
7   jmp code1a
```

a. Induces errors

```
1 code1b:
2   mov (X), %eax
3   clflush (X)
4
5
6   mfence
7   jmp code1b
```

b. Does not induce errors

- RowHammer
  - Activating a row too often causes “disturbance errors”
  - Code1a : Loop (Read → Flush) for two address (X, Y : same bank with different row)

    → Memory controller **opens** and **closes** the two rows repeatedly
  - Cpde1b : Loop (Read → Flush) for single address

    → Memory controller minimize **opens** and **closes** (only once): No disturbance

Background

- Proposed causes of RowHammer
  - Electromagnetic coupling
    - Changing the voltage of a wordline could inject noise into an adjacent wordline
  - Bridge (class of DRAM faults)
    - Accelerate the flow of charge between two bridged cells
  - Hot-carrier injection
    - Permanently damage the wordline by hot-carrier injection
      → Modify the amount of charge in cells or alter the characteristic

Overview of This Paper

- RowHammer Summary of their prior work
  - RowHammer mechanisms and characteristics
  - User-level RowHammer and security threat
  - RowHammer solutions
- Works that build on RowHammer
  - Exploits using RowHammer
  - Defenses against RowHammer
  - Circuit-level studies
  - Other works exploiting RowHammer
- Ongoing and Future work

Discussion

- Strengths
  - This paper investigates Rowhammer from the basic principle to the latest research. It's a really good starting point for studying Rowhammer attack
  - They set up a milestone on Rowhammer and present a direction for future work
- Weaknesses
  - This paper, called a retrospective, deals with details about Rowhammer but seems to lack novelty
  - I expected more in-depth research into the cause of Rowhammer on different DRAM devices and in circuit level so that future work could concentrate on solving the root cause of Rowhammer

RowHammer Based Attacks

- Google Project Zero
  - Exploited by user-level programs to gain kernel privileges on real systems
  - Runs Native Client program to escape from sandbox environment
  - Gain access to all of physical memory, take over the entire system
- Mobile Device
  - Use malicious user-level application without any permissions
  - Exploit Deterministic memory allocation patterns in the Android Linux Operating System
- WebGL
  - Takeover of a mobile system by triggering RowHammer using the WebGL interface on a mobile GPU
  - Takeover of a remote system by triggering RowHammer through the Remote Direct Memory Access (RDMA) protocol, and various other attacks

Mitigation Technique

- Requires **immediate** and **long-term** solutions
  - Immediate : Existing systems are patched → Vulnerable DRAM devices that are already in the field cannot be exploited.
    - Increase refresh rate (64ms → 8.2ms , energy/power consumption ↑)
    - Modify software : Monitor and detect RowHammer (intrusive to system operation / significant performance or memory overheads)
  - Long-term : Future DRAM devices do not suffer from the RowHammer problem when they are released into the field.
    - PARA(Probabilistic Adjacent Row Activation)

Mitigation Technique - PARA

- PARA(Probabilistic Adjacent Row Activation
  - Every time a row is opened and closed, one of its adjacent rows is also opened (refreshed) with some low probability
  - *Stateless* : No expensive hardware data structures required
  - Performance and power consumption overheads are very low due to the infrequent activation (probability of refresh : 0.001 to 0.005)

Pool Question

- Which of the following characteristics does PARA satisfy among the characteristics that Rowhammer's solution should have?
  - Immediate
  - Long-term
  - None of them
  - Both of them

Circuit-level Studies

- Gamma rays irradiation on DRAM
  - Data retention times ↓ Susceptibility to RowHammer failures ↑
  - Vulnerability to RowHammer & Data retention times : No correlation
  - Temperature annealing on the DRAM after gamma ray irradiation

  → Cells that experience a higher susceptibility to RowHammer maintain the higher susceptibility

Other Works

- PUF (Physical Unclonable Function)
  - Fingerprints of individual chips
  - Generate bit flips in the region whose locations are unique to the device and can be used to identify the device
- Attack on RowHammer-based PUF
  - Hammer on rows surrounding the region reserved by the RowHammer-based PUF

    → Rows at the edges of the reserved DRAM : # of bit flips ↑

RowHammer in a Broader Context

- Disturbance errors are a general class of reliability problems
  - All scaled memory technologies(SRAM, flash, and hard disk drives) exhibit such disturbance problems
  - Cell-to-cell interference is a fundamental issue, likely continue to appear in advanced technology
  - Such problems are expected to continue in future advanced technology (higher densities)

Future Work

- Focus on three perspectives
  - Security attack perspective
  - Defense/Mitigation perspective
  - Broader understanding, modeling, and prevention perspective
- Scalable solution is required

Discussion - Question

- How do you know where to specifically do the RowHammer attack to achieve what you want? Since you don't know a priori whether hammering a certain row flips other bits to 0 or 1 and even if the nearby cells will be susceptible.
- Can Rowhammer cause physical damage to a device, or does it solely cause transient effects?
- Are DRAM partitioning schemes (ex. placing distrusting parties far apart on the die, or adding dead 'spacer' rows) actually feasible in any real system?
- The mindset of "assume faulty hardware", while nice in theory, can only be taken so far. If hardware is completely faulty, then is it possible to safely run *any* code on it? So now we must figure out how to define this assumption of faulty hardware. Which/how many peripherals can be faulty? What can we trust? What is the minimum trust we *must* have to be able to safely operate?

Discussion - Question

- How could the probability p-value in PARA solution be determined in different DRAM technologies?

- If disturbance errors are a more general class of problems, is there a more general class of solutions?

Thanks!