Carnegie Mellon University SEAGATE ETH zürich SAFARI

Vulnerabilities in MLC NAND Flash Memory Programming:

Experimental Analysis, Exploits, and Mitigation Techniques

Saugata Ghose

Carnegie Mellon University

Flash Memory Summit

August 9, 2017

Santa Clara, CA

Executive Summary

SAFARI

- MLC (multi-level cell) NAND flash uses two-step programming
- We find ***new* reliability and security vulnerabilities**
  - In between two steps, cells are in a **partially-programmed** state
  - Program interference, read disturb much worse for partially-programmed cells than for fully-programmed cells
- We **experimentally characterize** vulnerabilities using real state-of-the-art MLC NAND flash memory chips
- We show that **malicious programs can exploit vulnerabilities** to corrupt data of other programs and reduce flash memory lifetime
- We propose three solutions that target vulnerabilities
  - One solution **completely eliminates vulnerabilities**, at the expense of 4.9% program latency increase
  - Two solutions **mitigate vulnerabilities**, increasing flash lifetime by 16%

Presentation Outline

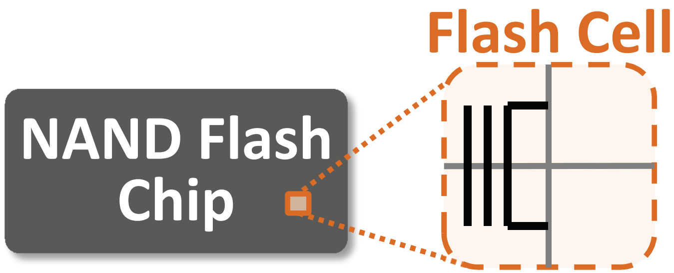
SAFARI

- Executive Summary
- **NAND Flash Background**
- Characterizing New Vulnerabilities in Two-Step Programming
- Example Sketches of Security Exploits
- Protection and Mitigation Mechanisms
- Conclusion

Storing Data in NAND Flash Memory

SAFARI

- Flash cell uses the **threshold voltage** of a floating-gate transistor to represent the data stored in the cell

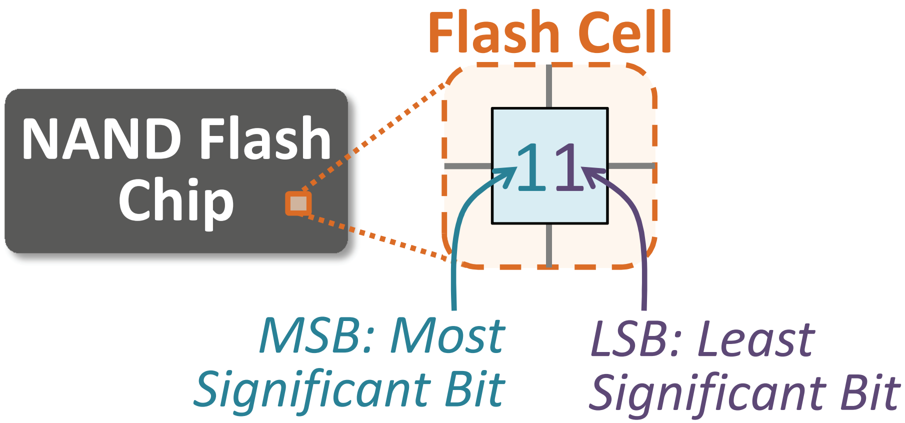


- Per-bit cost of NAND flash memory has greatly decreased
  - Aggressive process technology scaling

Storing Data in NAND Flash Memory

SAFARI

- Flash cell uses the **threshold voltage** of a floating-gate transistor to represent the data stored in the cell

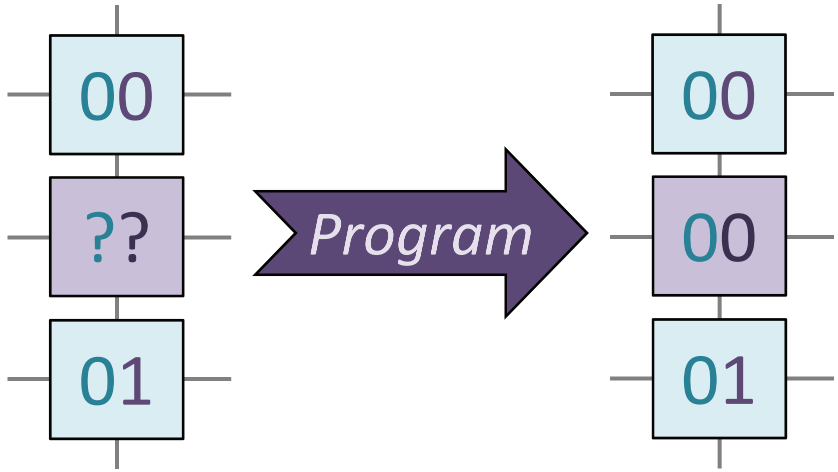


- Per-bit cost of NAND flash memory has greatly decreased
  - Aggressive process technology scaling
  - Multi-level cell (MLC) technology

Programming Data to a Multi-Level Cell

SAFARI

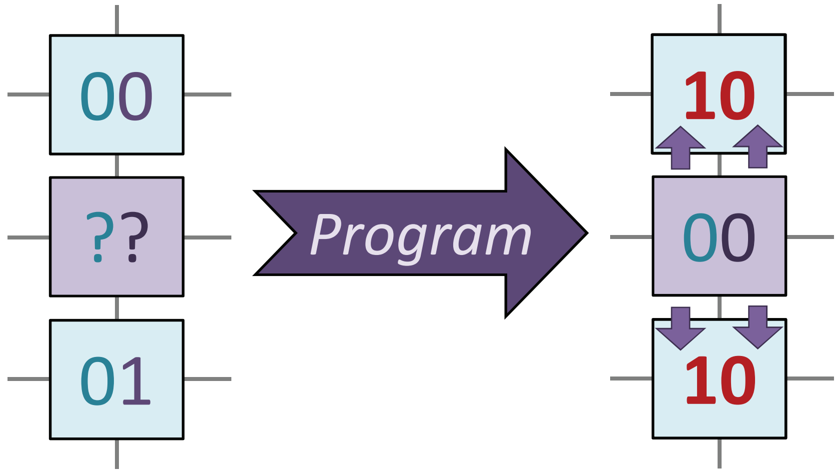
- Cell programmed by pulsing a large voltage on the transistor gate



Programming Data to a Multi-Level Cell

SAFARI

- Cell programmed by pulsing a large voltage on the transistor gate

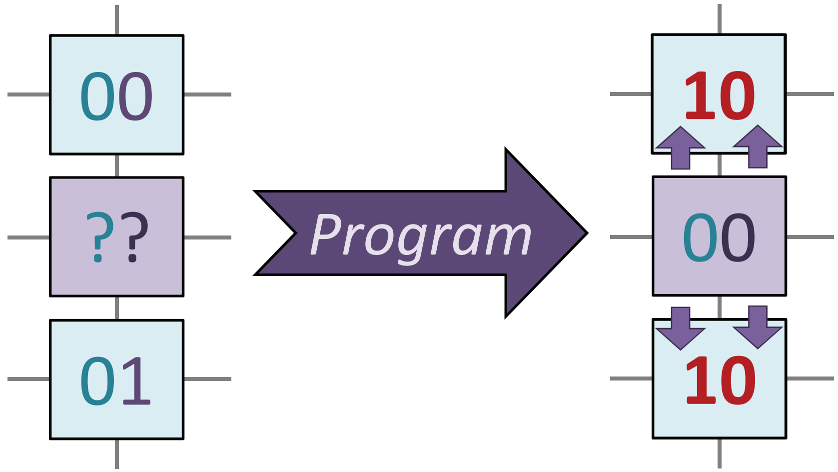


- ***Cell-to-cell program interference***
  - Threshold voltage of a neighboring cell **inadvertently increases**
  - Worsens as flash memory scales

Programming Data to a Multi-Level Cell

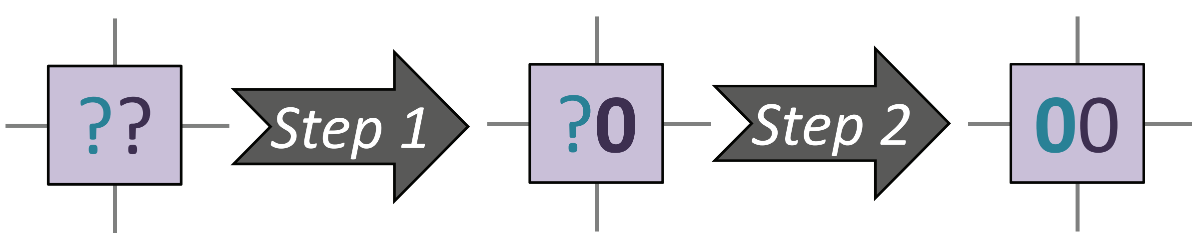
SAFARI

- Cell programmed by pulsing a large voltage on the transistor gate



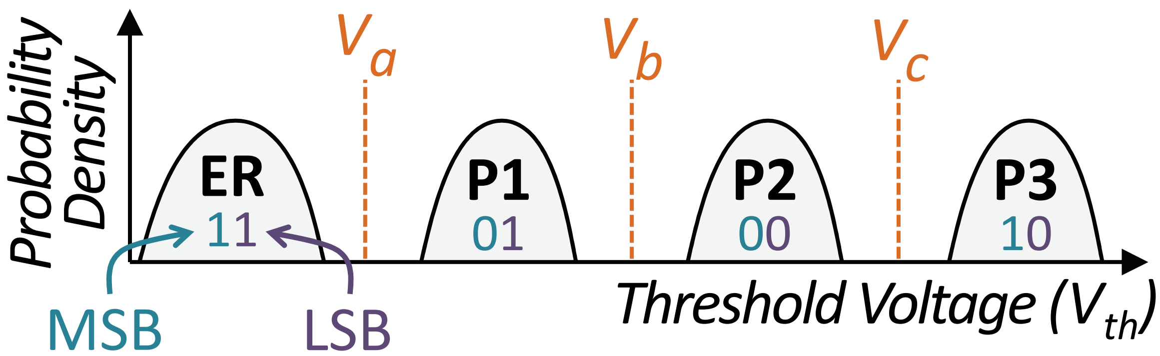
- ***Cell-to-cell program interference***
  - Threshold voltage of a neighboring cell **inadvertently increases**
  - Worsens as flash memory scales

- Mitigation: **two-step programming**



Reading Data from a Multi-Level Cell

- **Threshold voltages represented as a probability distribution**
  - Due to process variation
  - Each two-bit value corresponds to a ***state*** (a range of threshold voltages)

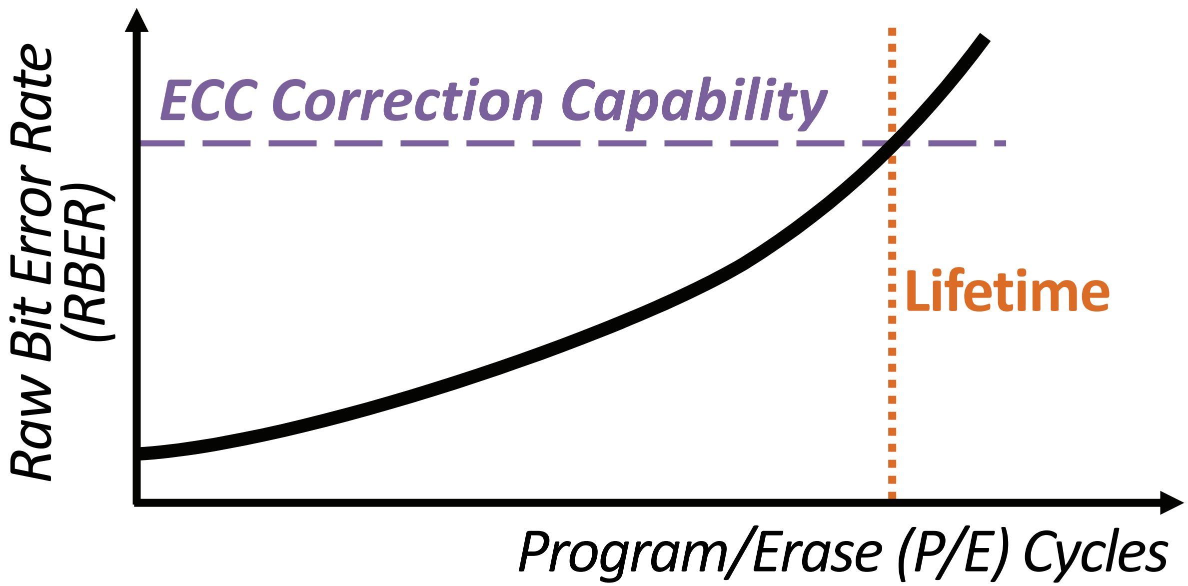


- **Read reference voltages** (V_a, V_b, V_c)
  - Identify the state a cell belongs to
  - Applied to the transistor gate to see if a cell turns on

NAND Flash Memory Errors and Lifetime

SAFARI

- During a read, ***raw bit errors*** occur when the cell threshold voltage **incorrectly shifts** to a different state



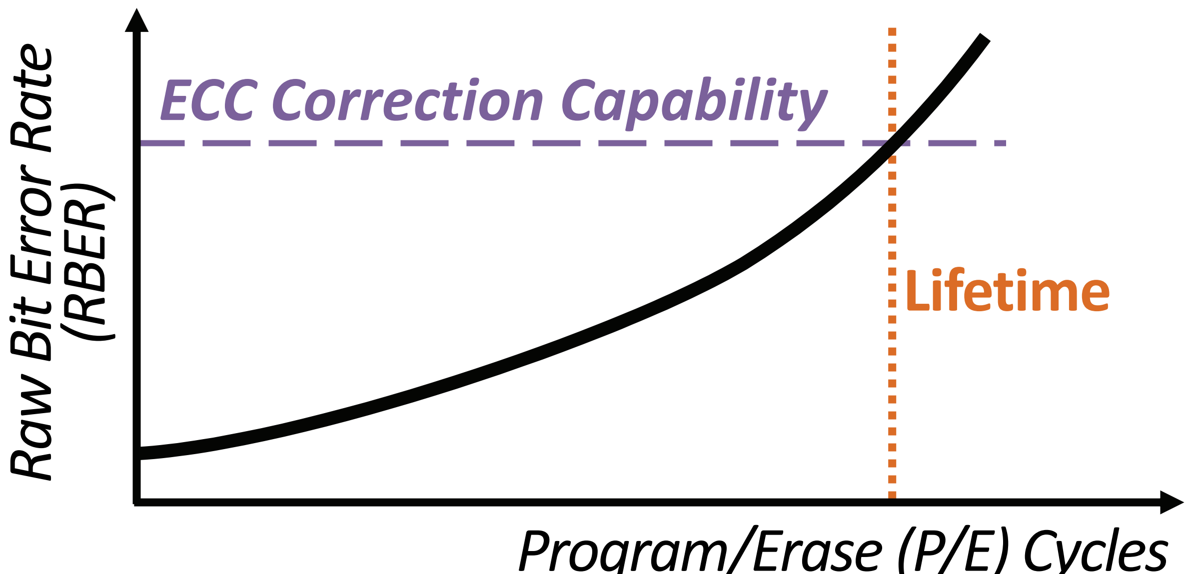
- Controller employs **sophisticated ECC** to correct errors

- If errors exceed ECC limit, flash memory has exhausted its ***lifetime***

NAND Flash Memory Errors and Lifetime

SAFARI

- During a read, ***raw bit errors*** occur when the cell threshold voltage **incorrectly shifts** to a different state



OUR GOAL

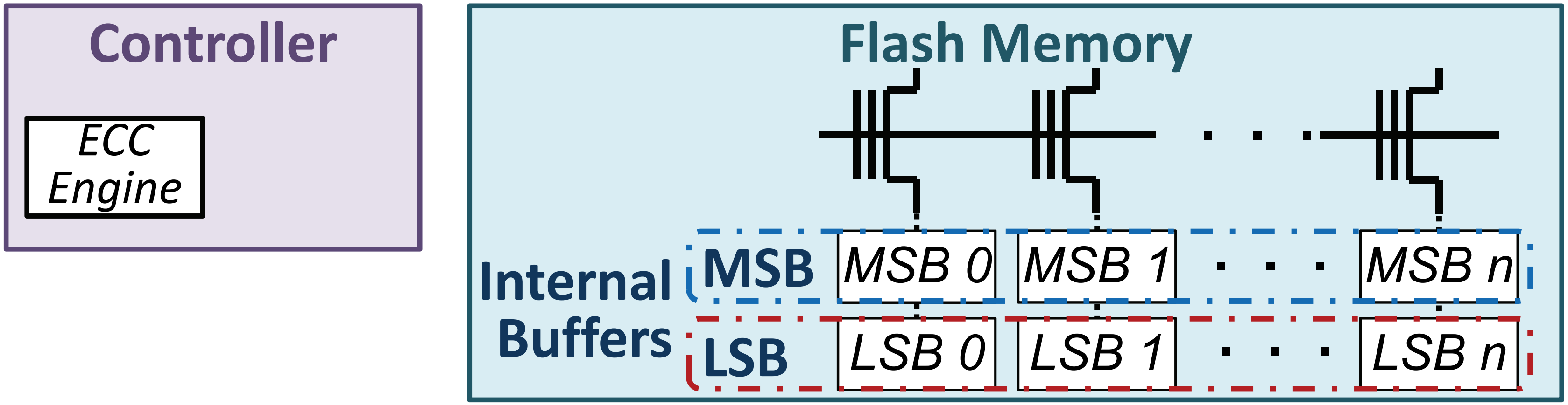
Understand how two-step programming affects flash memory errors and lifetime (and what potential vulnerabilities it causes)

Presentation Outline

SAFARI

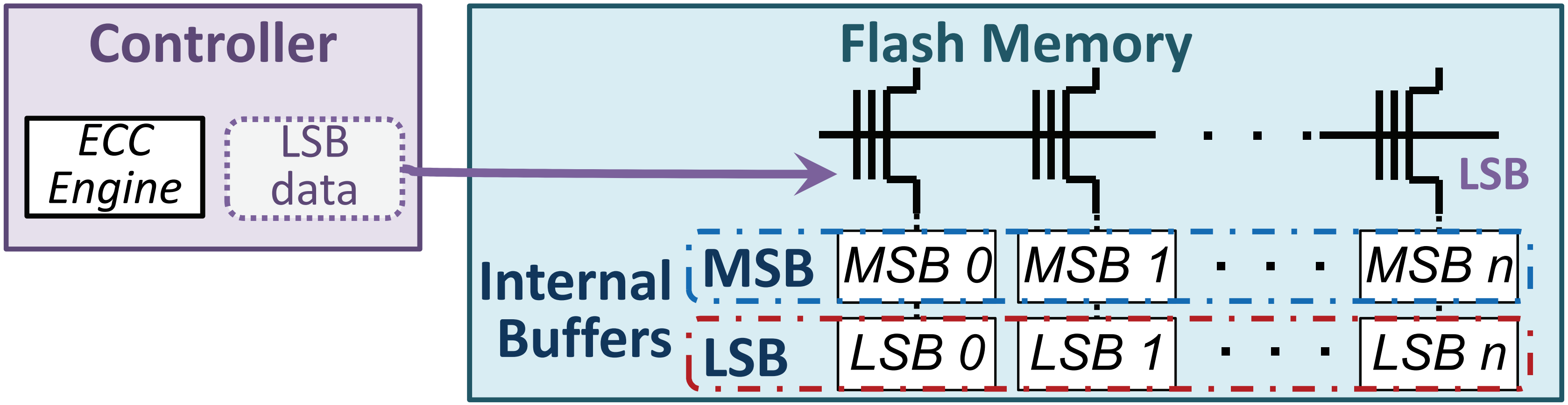
- Executive Summary
- NAND Flash Background
- **Characterizing New Vulnerabilities in Two-Step Programming**
  - How Can Two-Step Programming Introduce Errors?
  - Program Interference
  - Read Disturb
- Example Sketches of Security Exploits
- Protection and Mitigation Mechanisms
- Conclusion

How Can Two-Step Programming Introduce Errors? SAFARI



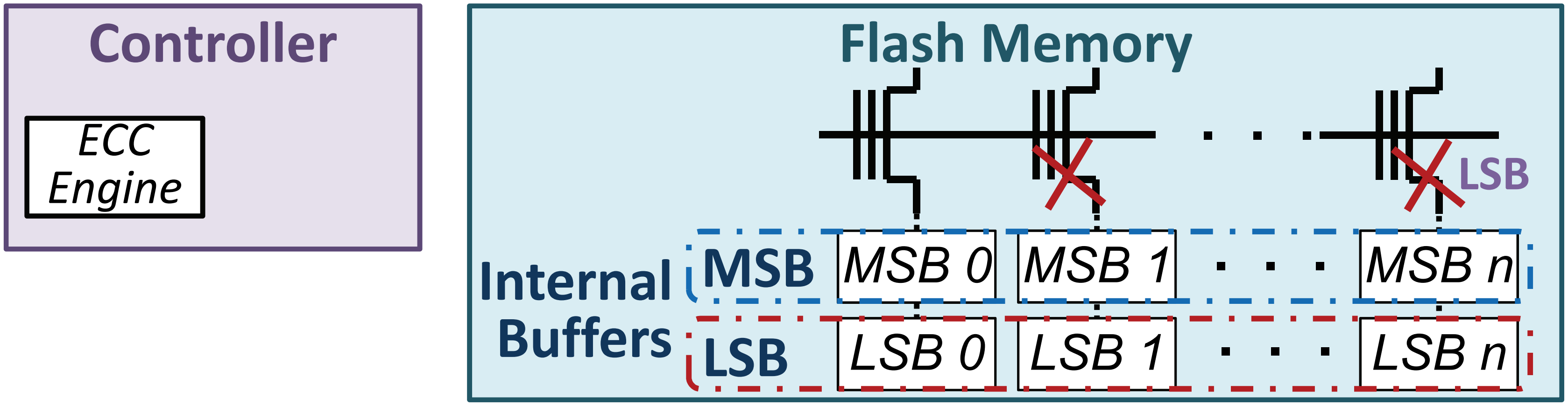
- Cell starts in the erased state

How Can Two-Step Programming Introduce Errors? SAFARI



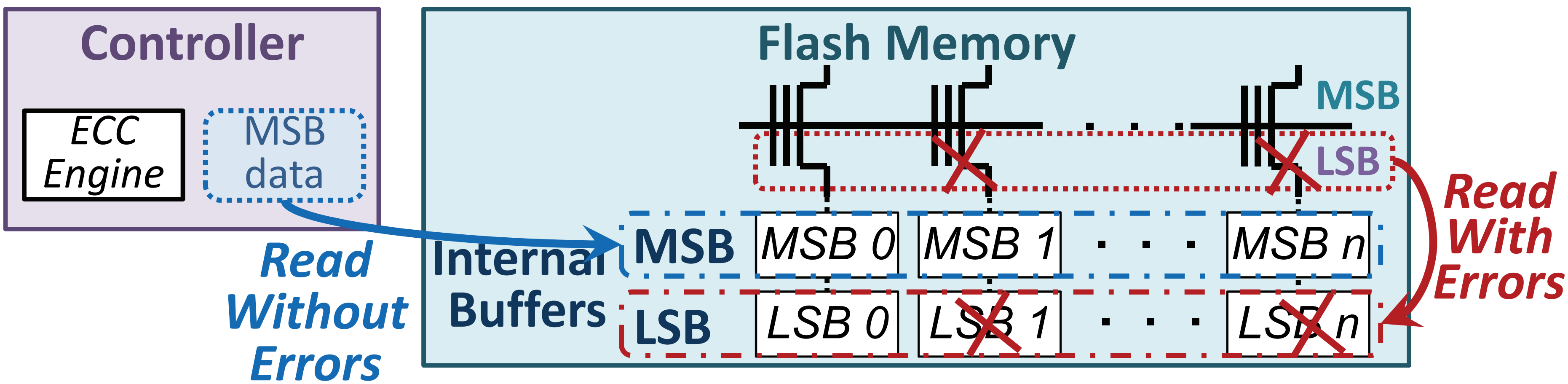
- Cell starts in the erased state
- Step 1 – LSB: Partially program the cell to a temporary state

How Can Two-Step Programming Introduce Errors? SAFARI



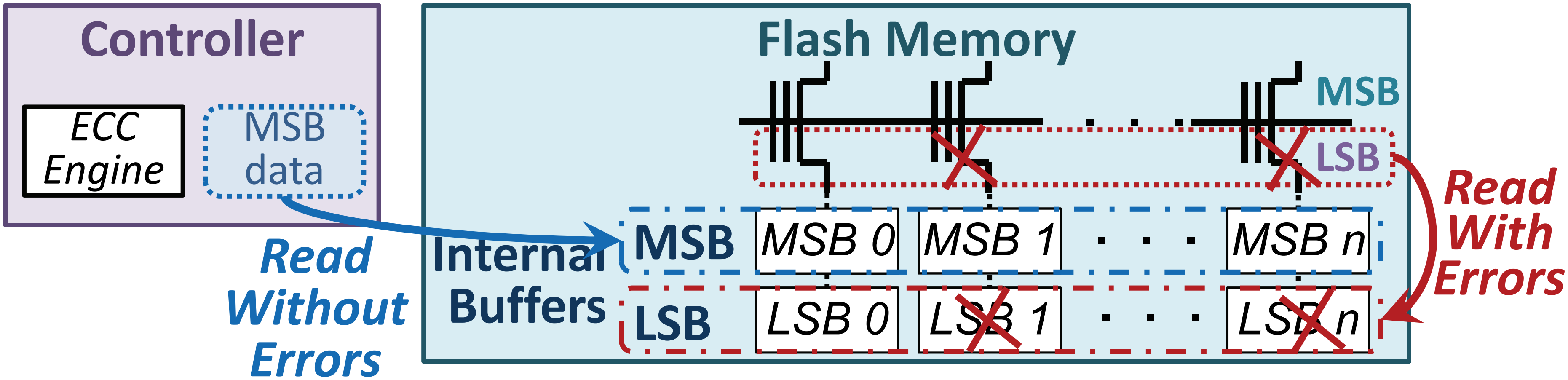
- Cell starts in the erased state
- Step 1 – LSB: Partially program the cell to a temporary state
- Errors are introduced into the **partially-programmed LSB data**

How Can Two-Step Programming Introduce Errors? SAFARI



- Cell starts in the erased state
- Step 1 – LSB: Partially program the cell to a temporary state
- Errors are introduced into the **partially-programmed LSB data**
- Step 2 – MSB: Program the cell to its final state
  - LSB data is read with errors into internal LSB buffer, **not corrected** by ECC
  - MSB data comes from controller to internal MSB buffer

How Can Two-Step Programming Introduce Errors? SAFARI



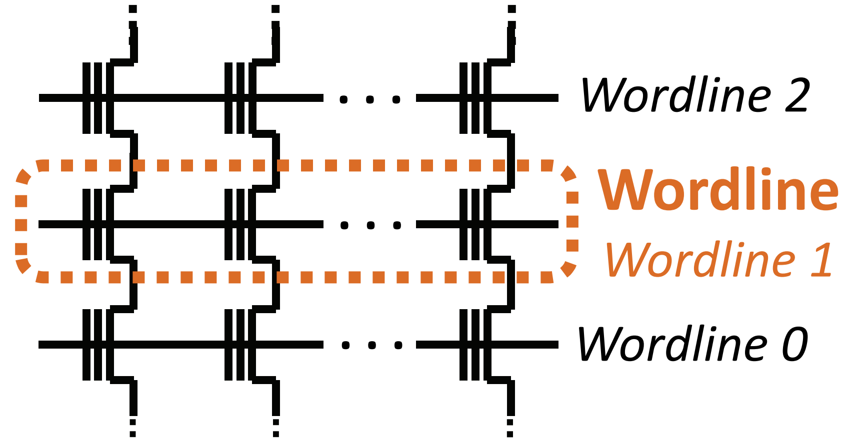
- Cell starts in the erased state
- Step 1 – LSB: Partially program the cell to a temporary state

Errors in internal LSB buffer data cause the cell to be programmed to an incorrect state

Cell-to-Cell Program Interference

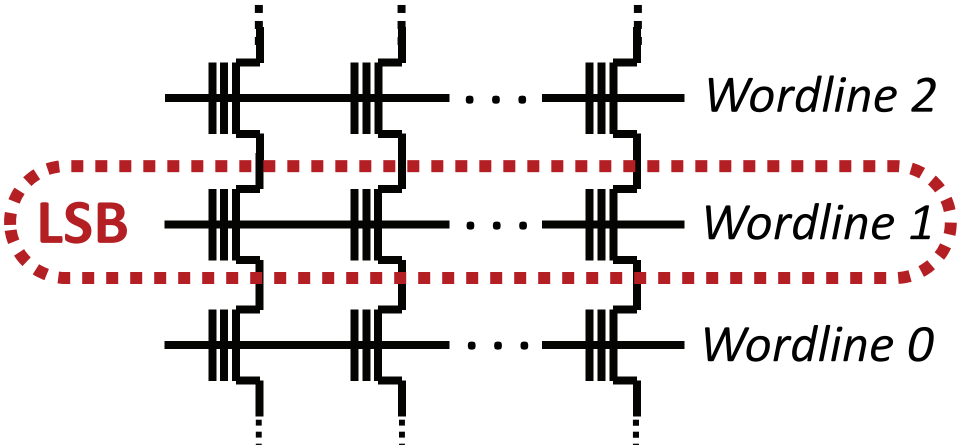
SAFARI

- Flash cells are grouped into multiple **wordlines** (rows)

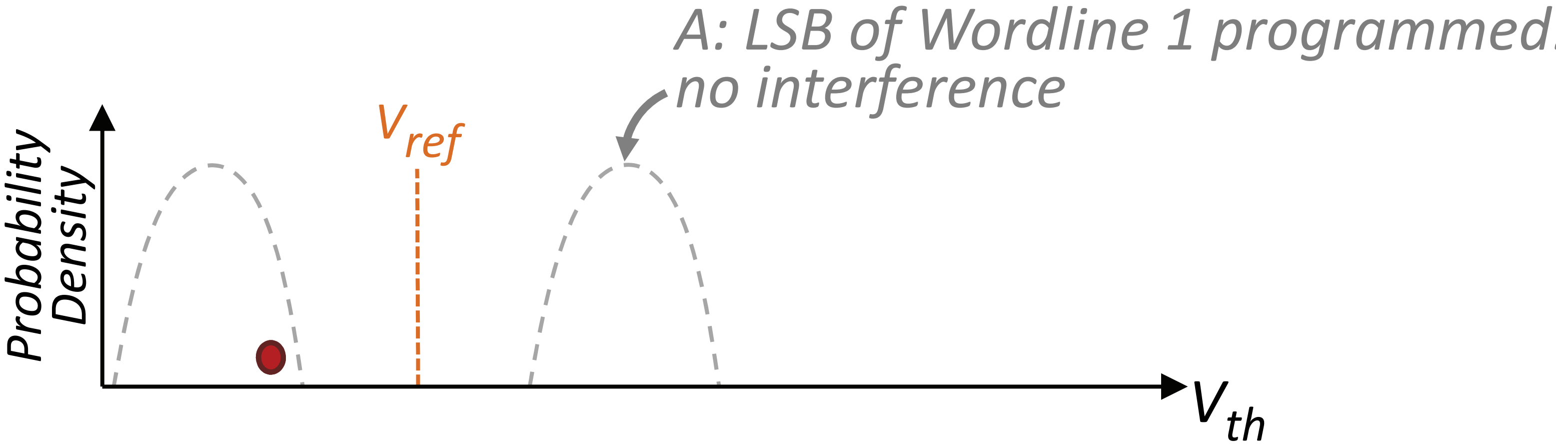


Cell-to-Cell Program Interference

- Flash cells are grouped into multiple **wordlines** (rows)
- Two-step programming **interleaves LSB, MSB steps** of neighboring wordlines

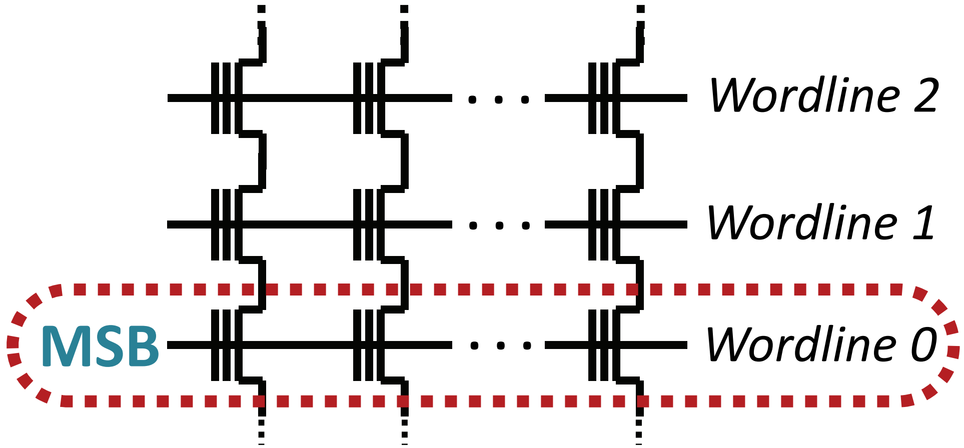


- Steps interleaved using shadow program sequencing

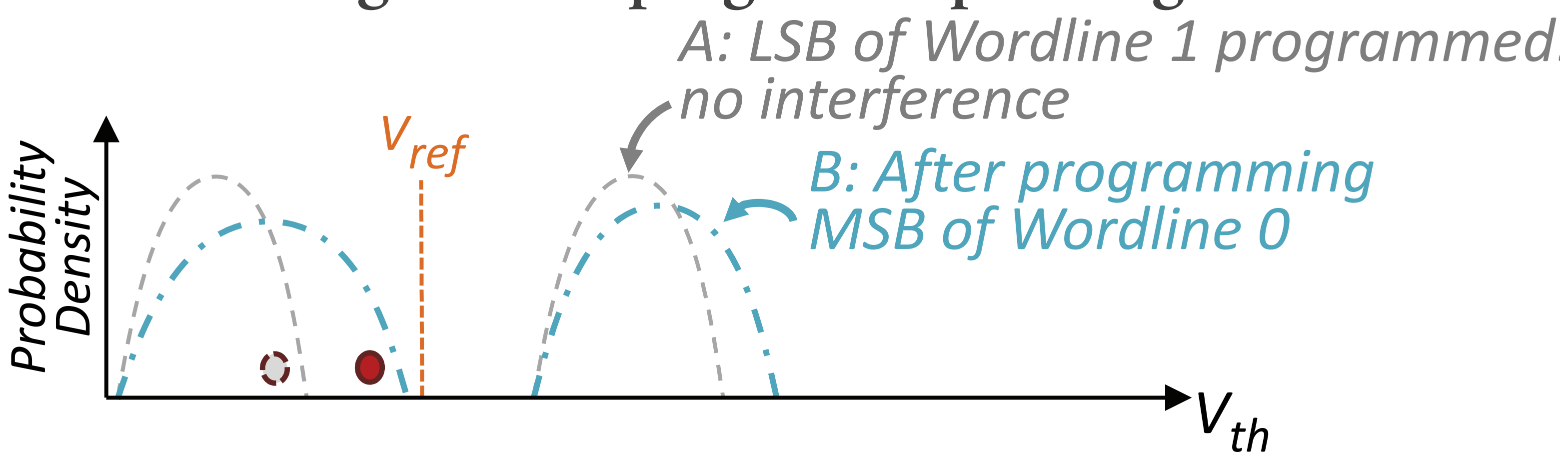


Cell-to-Cell Program Interference

- Flash cells are grouped into multiple **wordlines** (rows)
- Two-step programming **interleaves LSB, MSB steps** of neighboring wordlines

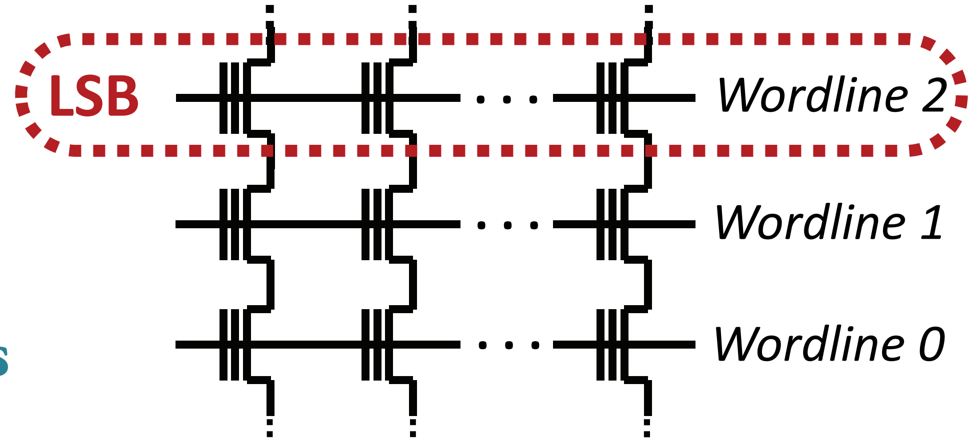


- Steps interleaved using shadow program sequencing



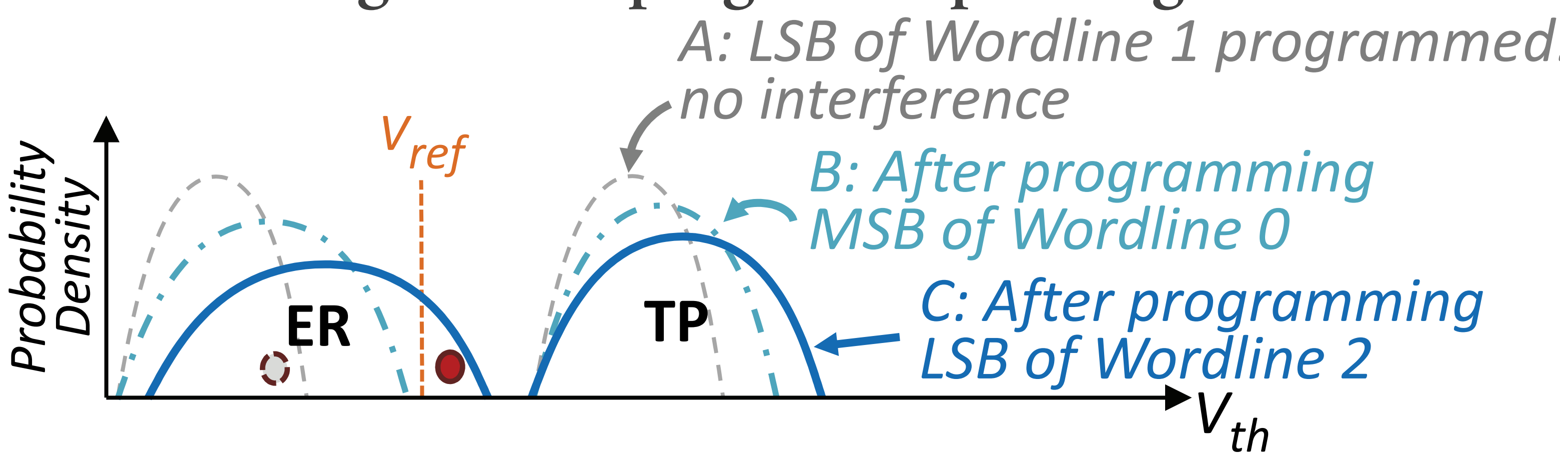
Cell-to-Cell Program Interference

- Flash cells are grouped into multiple **wordlines** (rows)



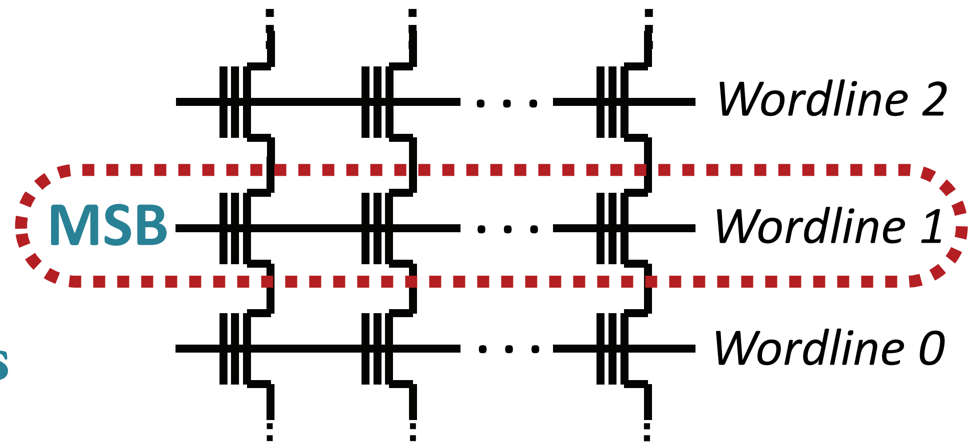
- Two-step programming **interleaves LSB, MSB steps** of neighboring wordlines

- Steps interleaved using shadow program sequencing

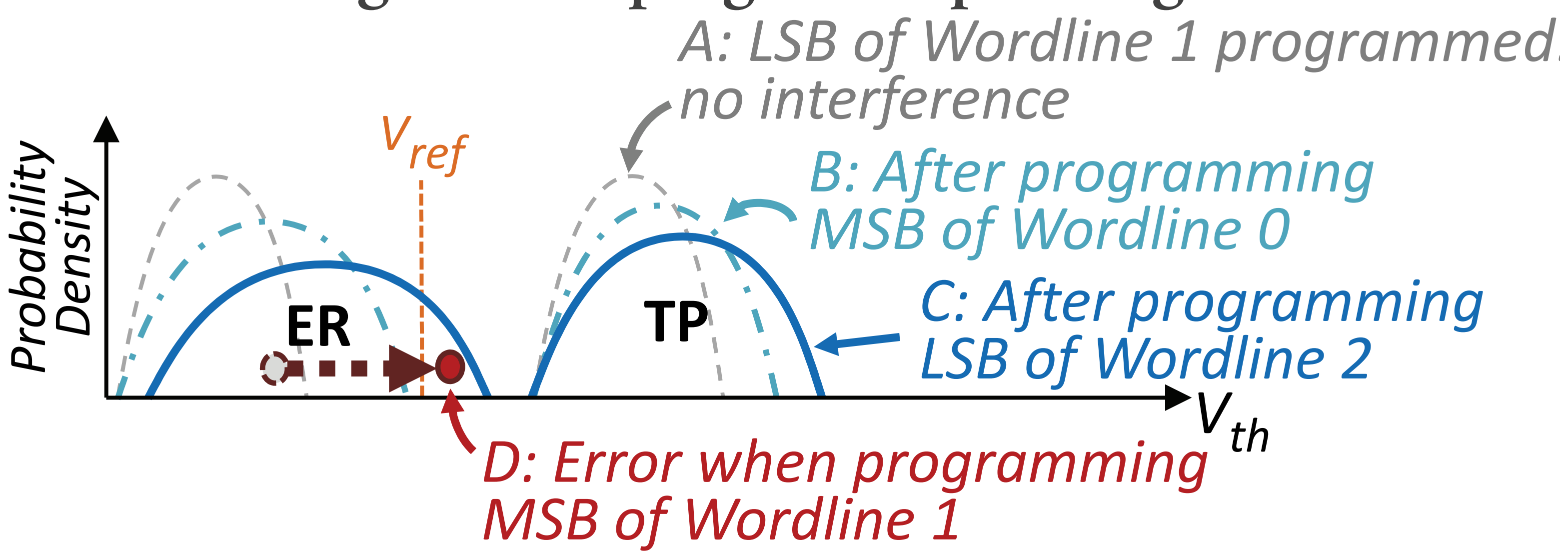


Cell-to-Cell Program Interference

- Flash cells are grouped into multiple **wordlines** (rows)
- Two-step programming **interleaves LSB, MSB steps** of neighboring wordlines

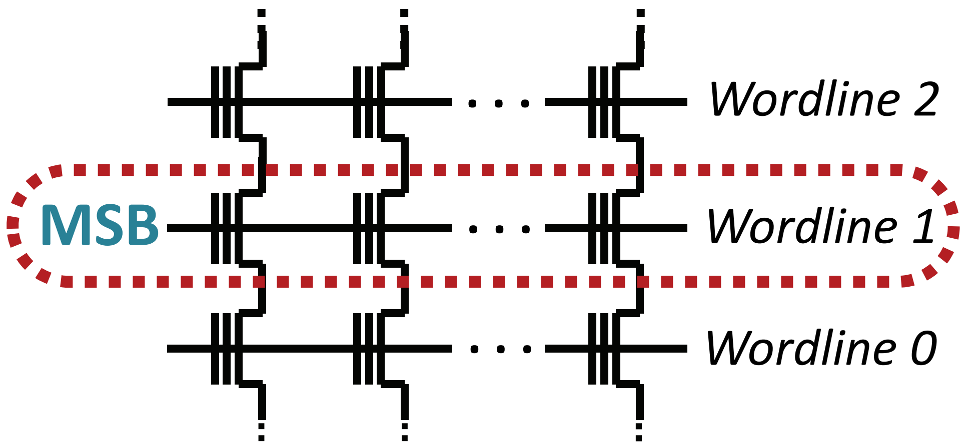


- Steps interleaved using shadow program sequencing



Cell-to-Cell Program Interference

- Flash cells are grouped into multiple **wordlines** (rows)
- Two-step programming **interleaves LSB, MSB steps** of neighboring wordlines



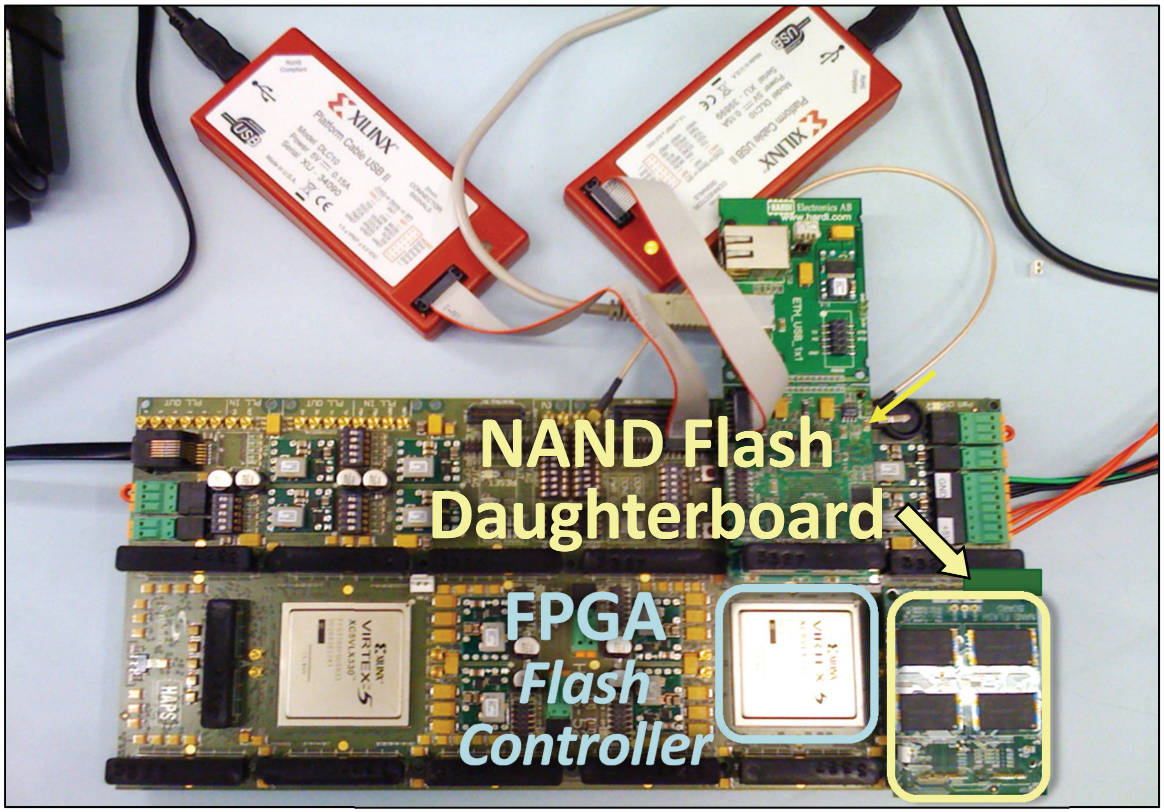
- Steps interleaved using shadow program sequencing

A: LSB of Wordline 1 programmed: no interference

Steps for neighboring wordlines cause interference on *partially-programmed cells*

How bad is this interference?

- We perform experiments on **real** state-of-the-art 1x-nm (i.e., 15-19nm) MLC NAND flash memory chips

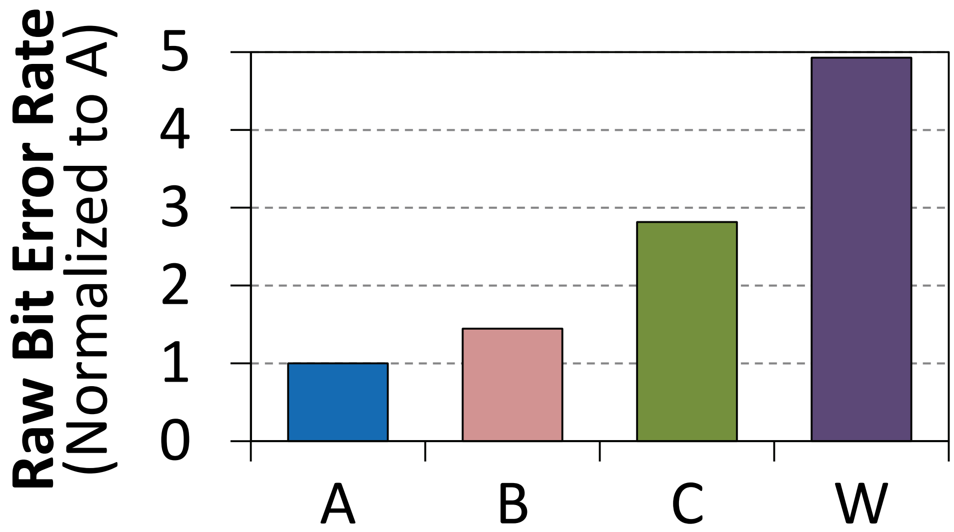


- More info: Cai et al., *FPGA-Based Solid-State Drive Prototyping Platform*, FCCM 2011

Measuring Errors Induced by Program Interference

SAFARI

- Error rate **increases with each programming step**
  - A: **Before interference** (LSBs in Wordline *n* just programmed)
  - B: After programming pseudo-random data to MSBs in **Wordline *n*-1**
  - C: After programming pseudo-random data to MSBs in **Wordline *n*-1 *and*** LSBs in **Wordline *n*+1**

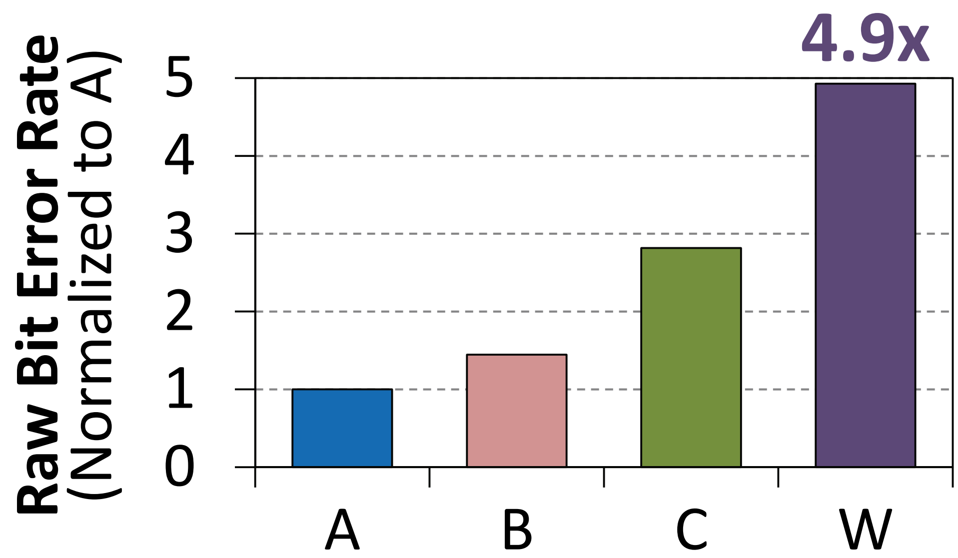


- Interference **depends on the data value** being programmed
  - Higher voltage → more programming pulses → more interference
  - W: After programming **worst-case data pattern** to Wordlines *n*-1 and *n*+1

Measuring Errors Induced by Program Interference

SAFARI

- Error rate **increases with each programming step**
  - A: Before interference (LSBs in Wordline *n* just programmed)
  - B: After programming pseudo-random data to MSBs in Wordline *n*-1
  - C: After programming pseudo-random data to MSBs in Wordline *n*-1 *and* LSBs in Wordline *n*+1

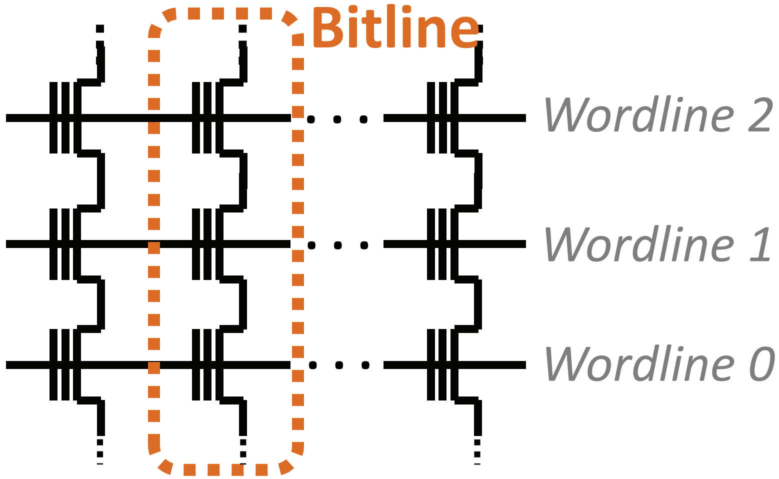


Program interference with worst-case data pattern increases the error rate of partially-programmed cells by 4.9x

Read Disturb

SAFARI

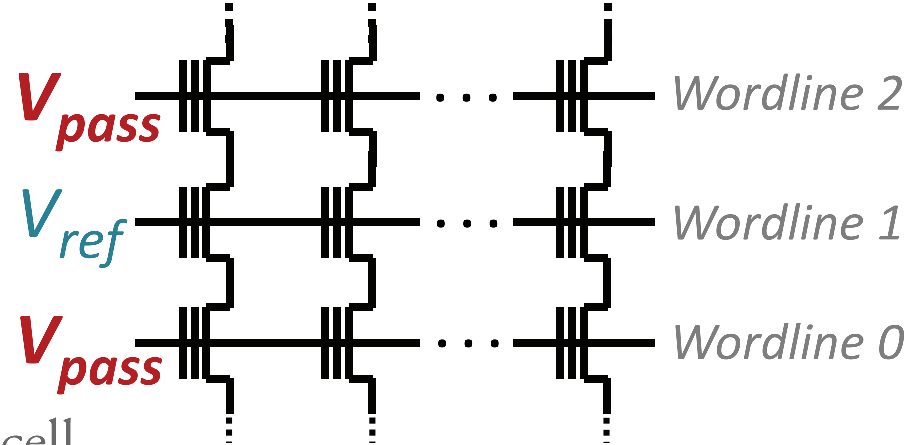
- **Flash block**: cells from multiple wordlines connected together on **bitlines** (columns)



Read Disturb

SAFARI

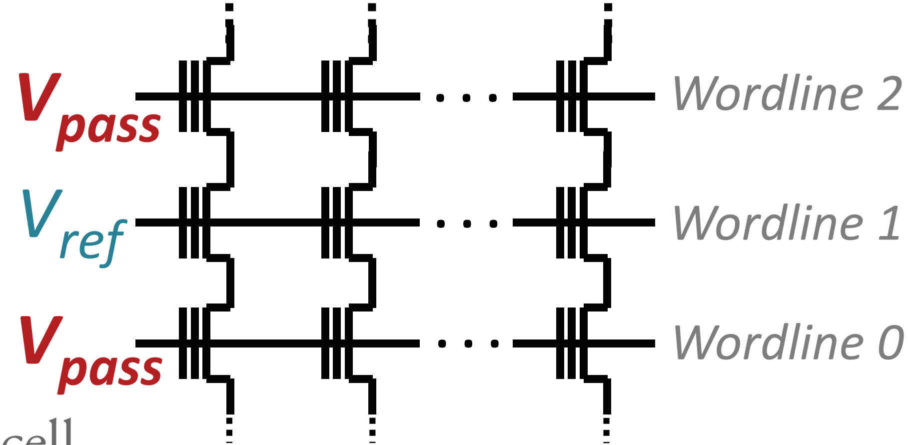
- **Flash block**: cells from multiple wordlines connected together on **bitlines** (columns)



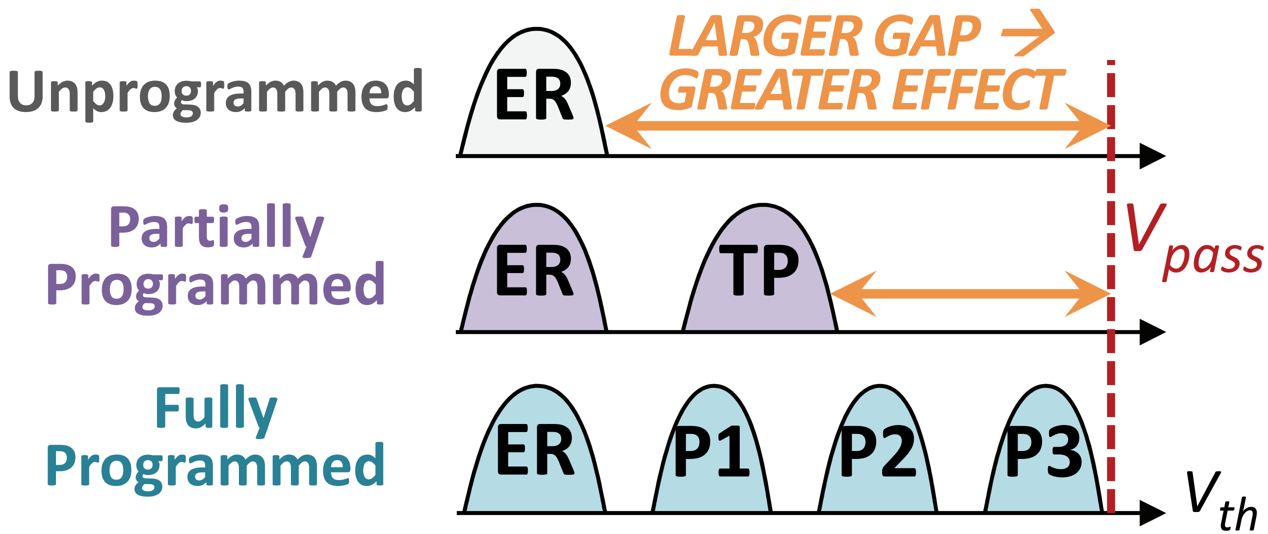
- Reading a cell from a bitline
  - Apply read reference voltage (V_{ref}) to cell
  - Apply a **pass-through voltage** (V_{pass}) to turn on all **unread** cells

Read Disturb

- **Flash block**: cells from multiple wordlines connected together on **bitlines** (columns)



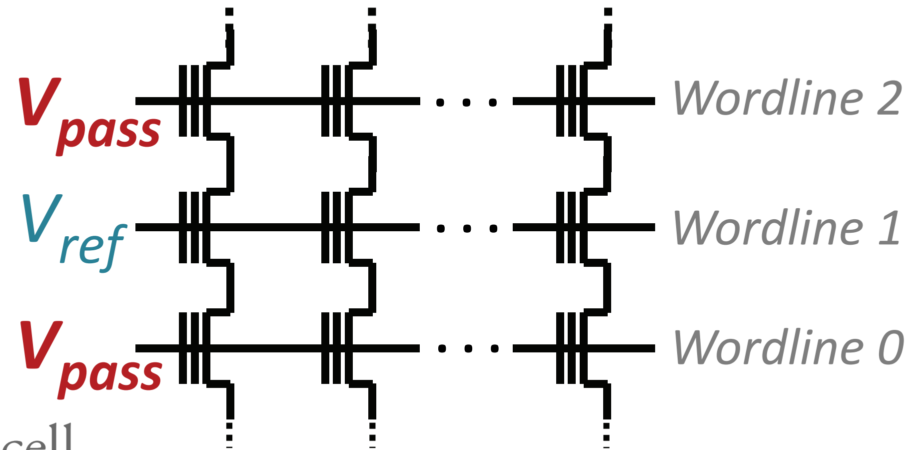
- Reading a cell from a bitline
  - Apply read reference voltage (V_{ref}) to cell
  - Apply a **pass-through voltage** (V_{pass}) to turn on all **unread** cells
- Pass-through voltage has a **weak programming effect**



Read Disturb

SAFARI

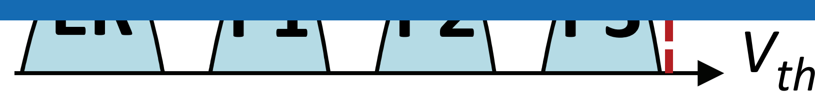
- **Flash block**: cells from multiple wordlines connected together on **bitlines** (columns)



- Reading a cell from a bitline
  - Apply read reference voltage (V_{ref}) to cell
  - Apply a **pass-through voltage** (V_{pass}) to turn on all **unread** cells

- Pass-through voltage has a **weak programming effect**

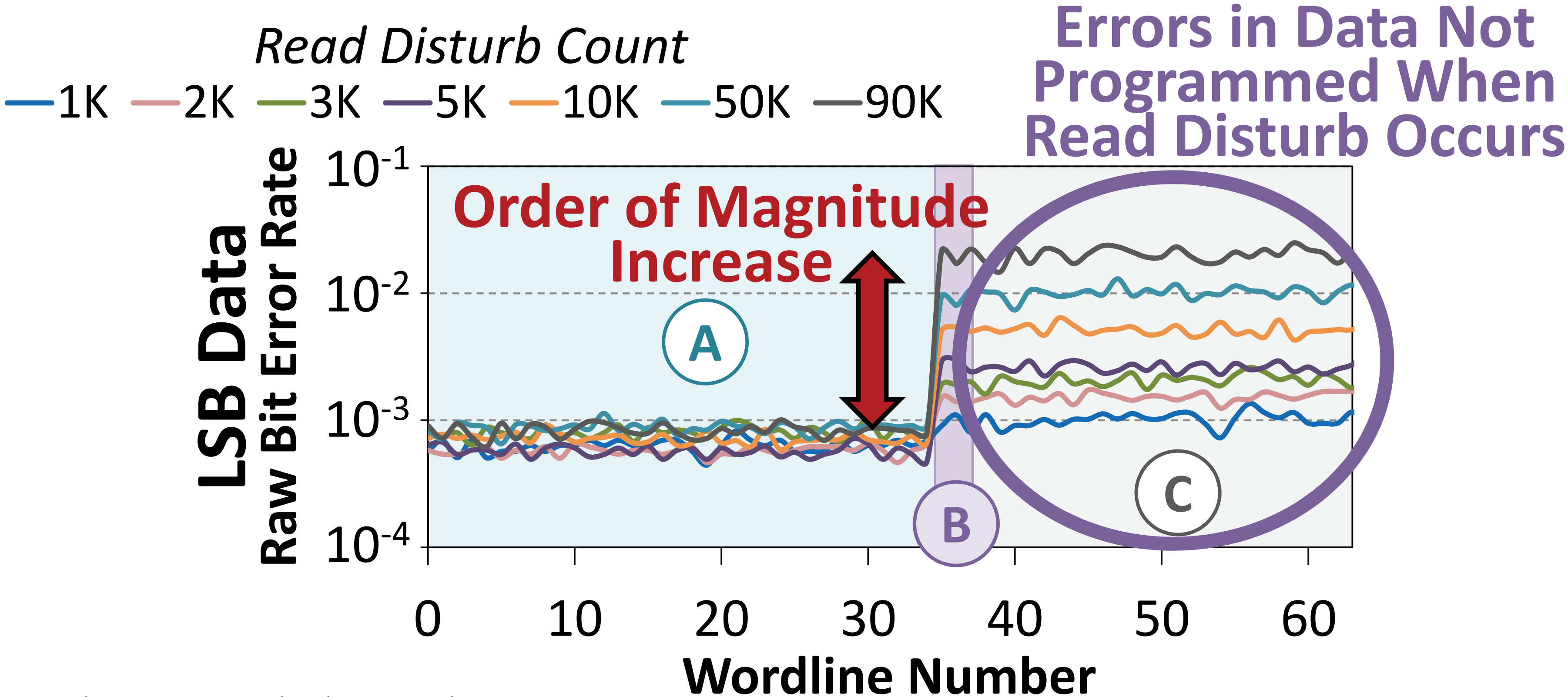
LARGER GAP →



Partially-programmed and unprogrammed cells more susceptible to read disturb errors

Measuring Errors Induced by Read Disturb

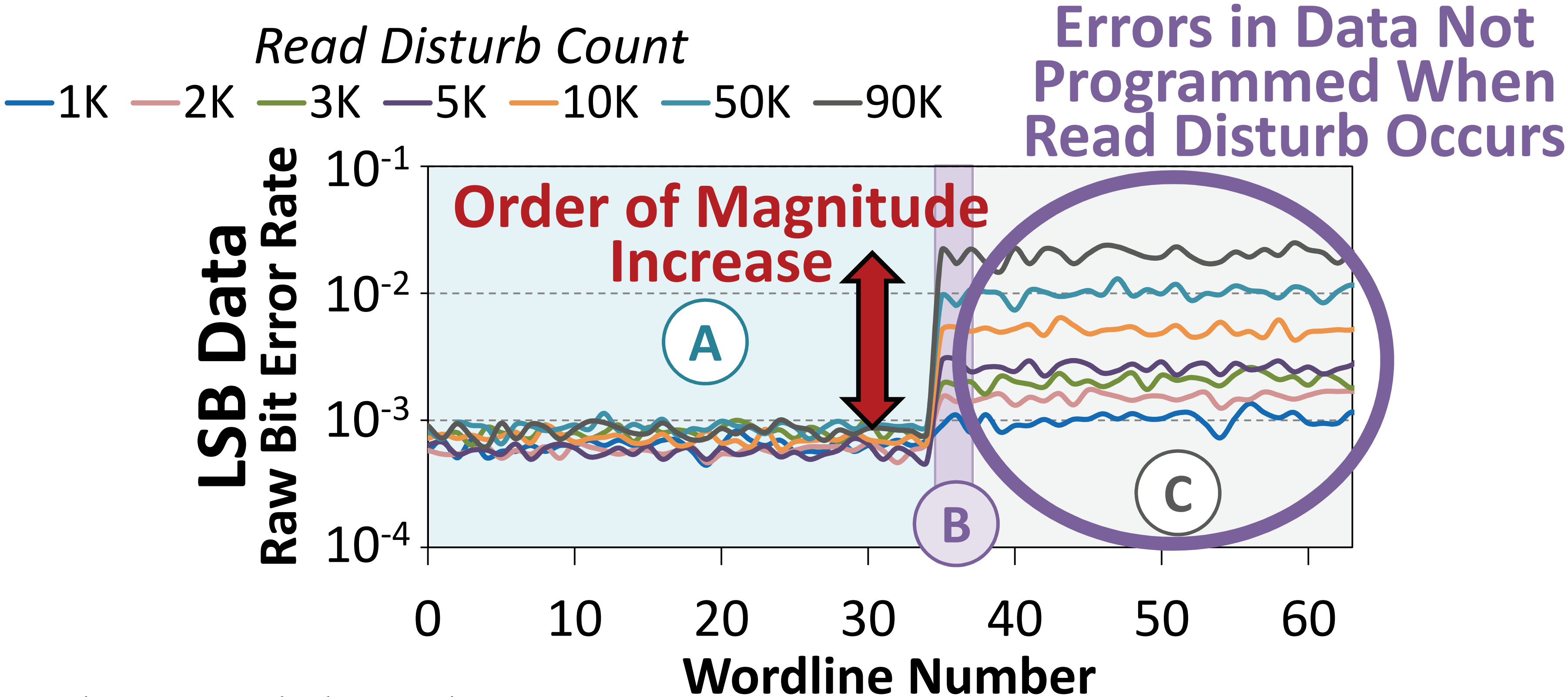
SAFARI



- **Induce read disturbs on:**
  - A: Fully-programmed cells
  - B: Partially-programmed cells
  - C: Unprogrammed cells
- **After read disturb, program remaining data and check error rate**

Measuring Errors Induced by Read Disturb

SAFARI



LSB data in partially-programmed and unprogrammed cells most susceptible to read disturb

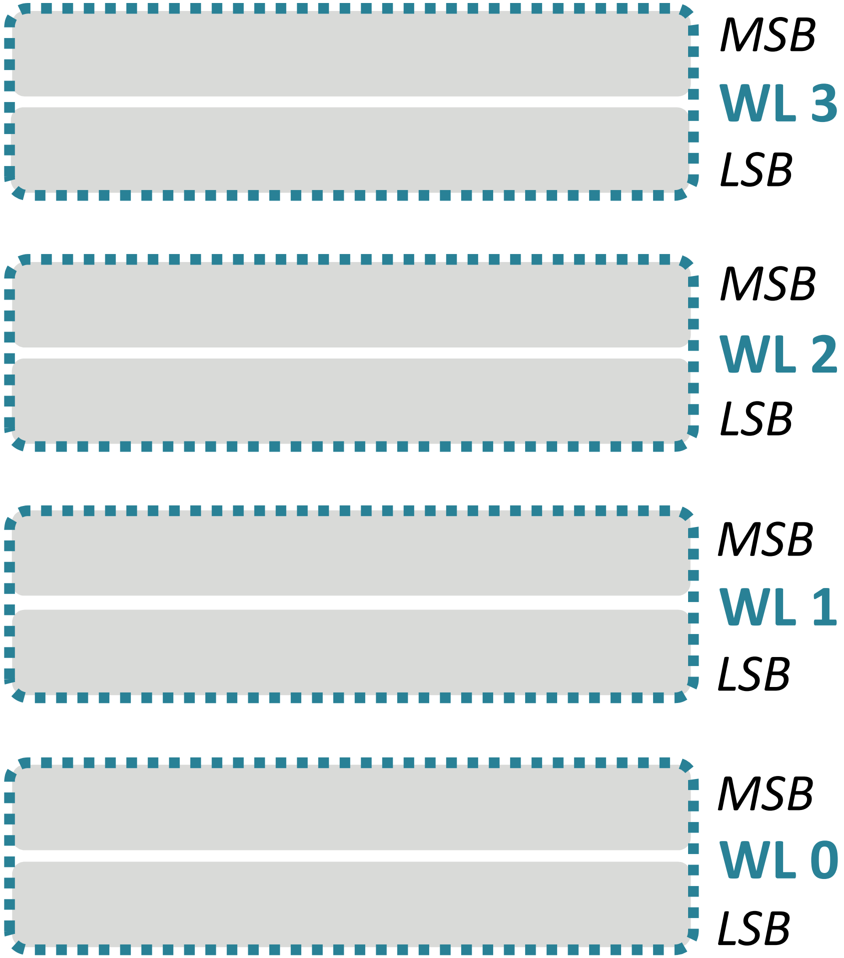
Presentation Outline

- Executive Summary
- NAND Flash Background
- Characterizing New Vulnerabilities in Two-Step Programming
- **Example Sketches of Security Exploits**
  - **Program Interference Based Exploit**
  - Read Disturb Based Exploit
- Protection and Mitigation Mechanisms
- Conclusion

Sketch of Program Interference Based Exploit

- Malicious program targets a piece of data that belongs to a victim program
- Goal: Maximize program interference induced on victim program's data

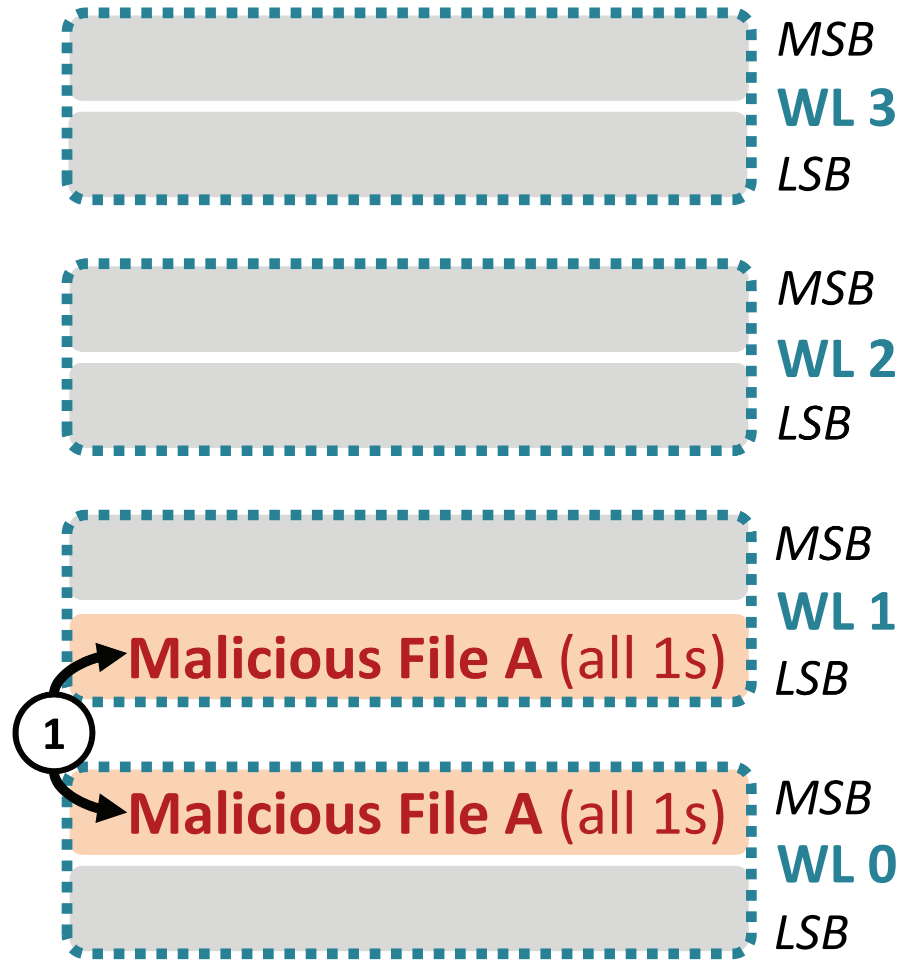
- **Write worst-case data pattern** to neighboring wordlines (*WL*)



Sketch of Program Interference Based Exploit

SAFARI

- Malicious program targets a piece of data that belongs to a victim program
- Goal: Maximize program interference induced on victim program's data
- **Write worst-case data pattern** to neighboring wordlines (*WL*)
  1. Wordlines 0/1: all 1s to keep at *lowest* possible threshold voltage

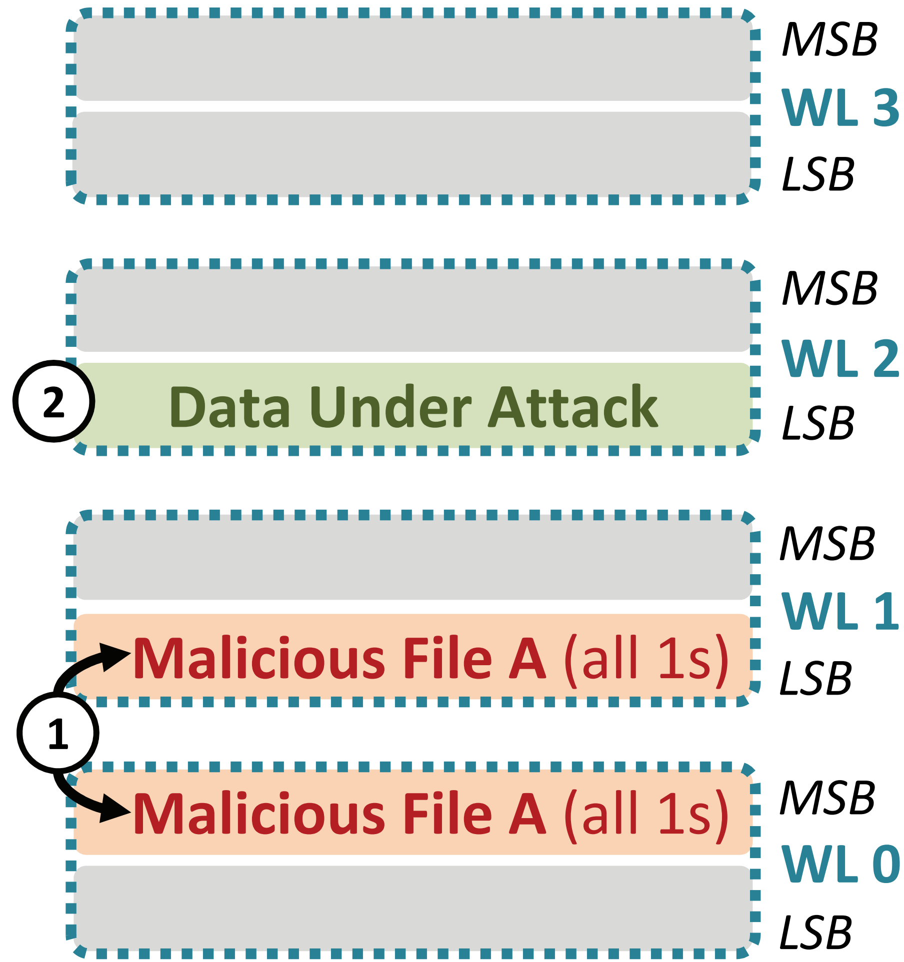


Sketch of Program Interference Based Exploit

SAFARI

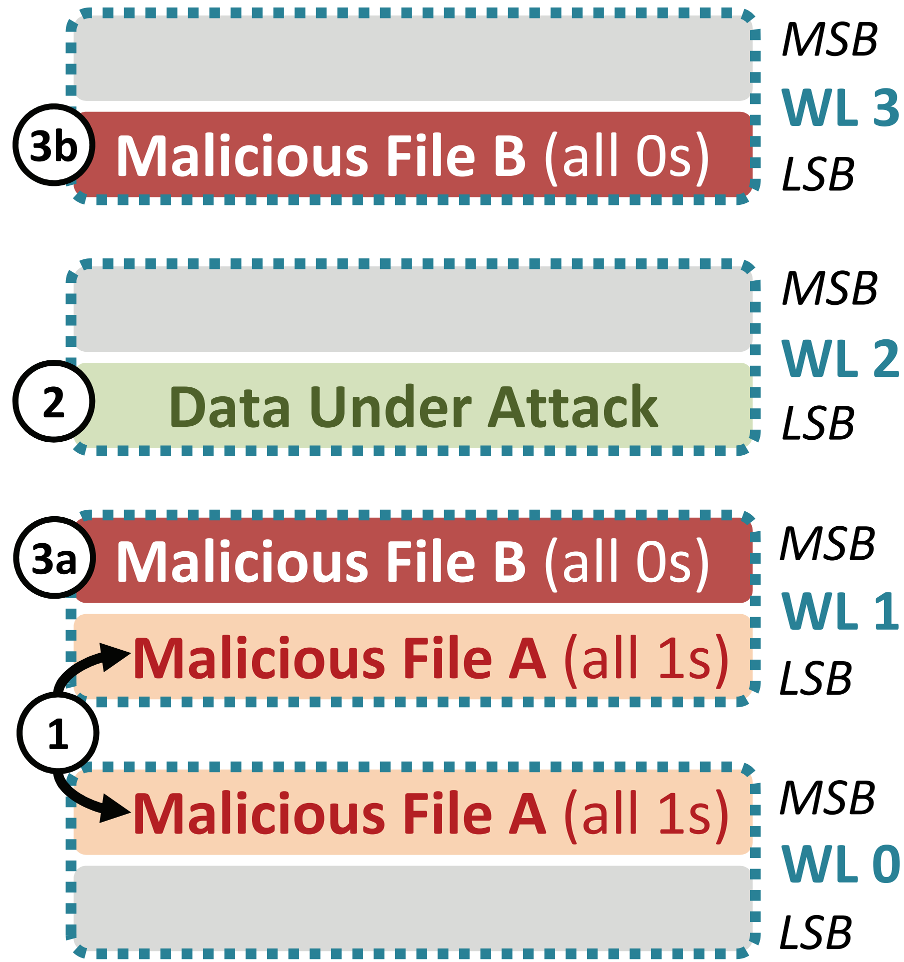
- Malicious program targets a piece of data that belongs to a victim program
- Goal: Maximize program interference induced on victim program's data

- **Write worst-case data pattern** to neighboring wordlines (*WL*)
  1. Wordlines 0/1: all 1s to keep at *lowest* possible threshold voltage
  2. Wordline 2: victim program writes data



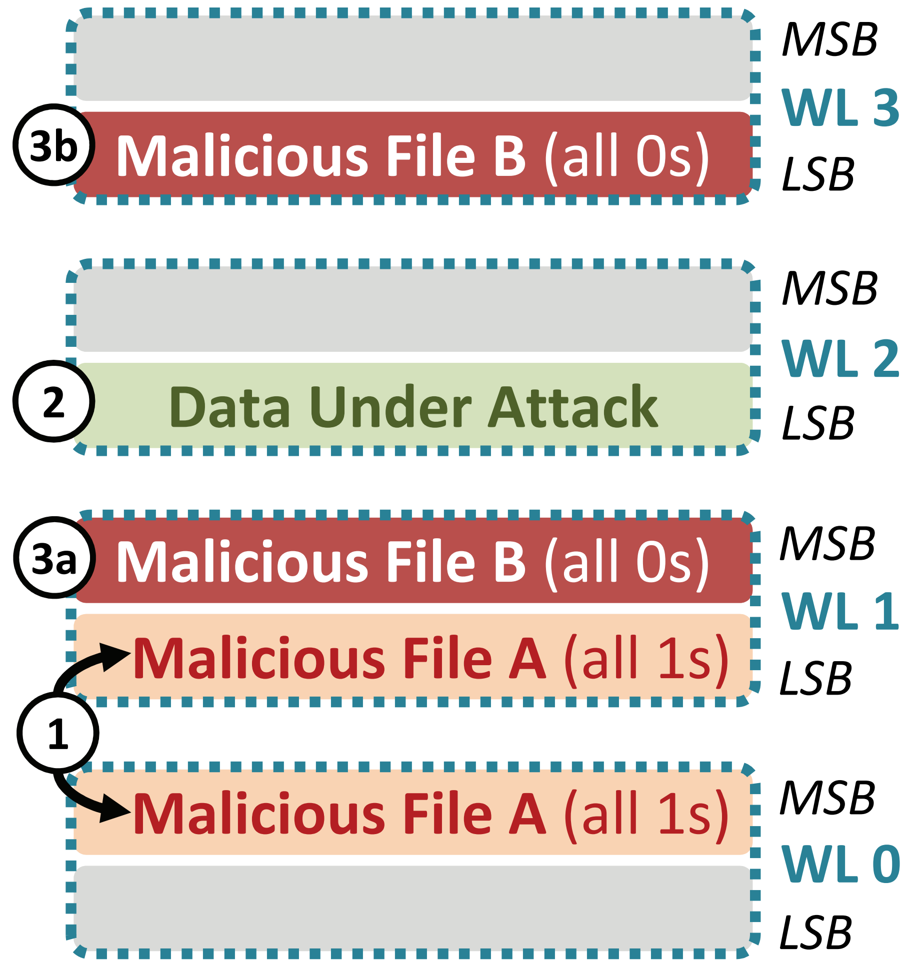
Sketch of Program Interference Based Exploit

- Malicious program targets a piece of data that belongs to a victim program
- Goal: Maximize program interference induced on victim program's data
- **Write worst-case data pattern to neighboring wordlines** (*WL*)
  1. Wordlines 0/1: **all 1s** to keep at *lowest* possible threshold voltage
  2. Wordline 2: victim program writes data
  3. Wordlines 1 and 3: **all 0s** to program to *highest* possible threshold voltage



Sketch of Program Interference Based Exploit

- Malicious program targets a piece of data that belongs to a victim program
- Goal: Maximize program interference induced on victim program's data
- **Write worst-case data pattern** to neighboring wordlines (*WL*)
  1. Wordlines 0/1: **all 1s** to keep at *lowest* possible threshold voltage
  2. Wordline 2: victim program writes data
  3. Wordlines 1 and 3: **all 0s** to program to *highest* possible threshold voltage



- In the paper
  - More details on why this works
  - Procedure to work around data scrambling

Presentation Outline

SAFARI

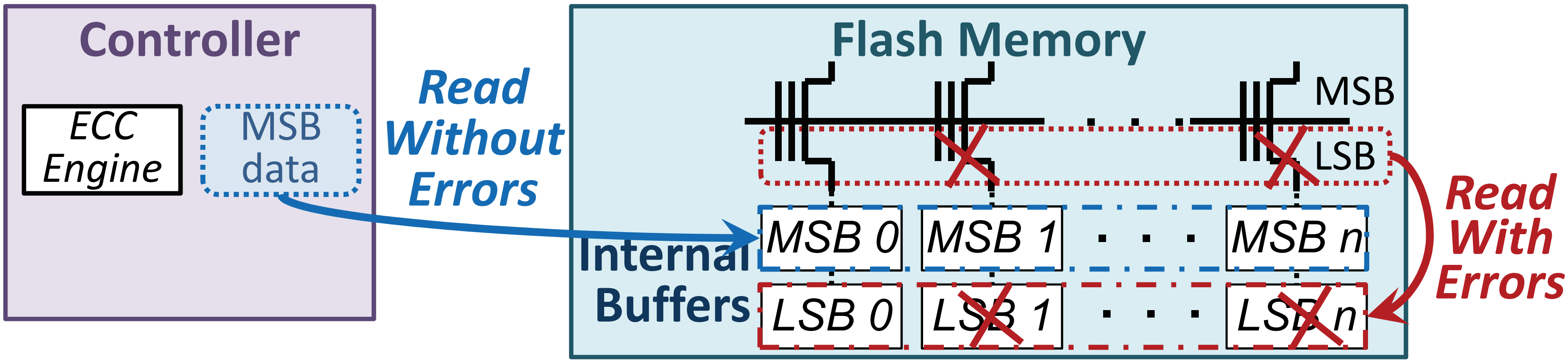
- Executive Summary
- NAND Flash Background
- Characterizing New Vulnerabilities in Two-Step Programming
- **Example Sketches of Security Exploits**
  - Program Interference Based Exploit
  - **Read Disturb Based Exploit: in the paper**
- Protection and Mitigation Mechanisms
- Conclusion

Presentation Outline

SAFARI

- Executive Summary
- NAND Flash Background
- Characterizing New Vulnerabilities in Two-Step Programming
- Example Sketches of Security Exploits
- **Protection and Mitigation Mechanisms**
  - **Buffering LSB Data in the Controller**
  - **Multiple Pass-Through Voltages**
  - Adaptive LSB Read Reference Voltage
- Conclusion

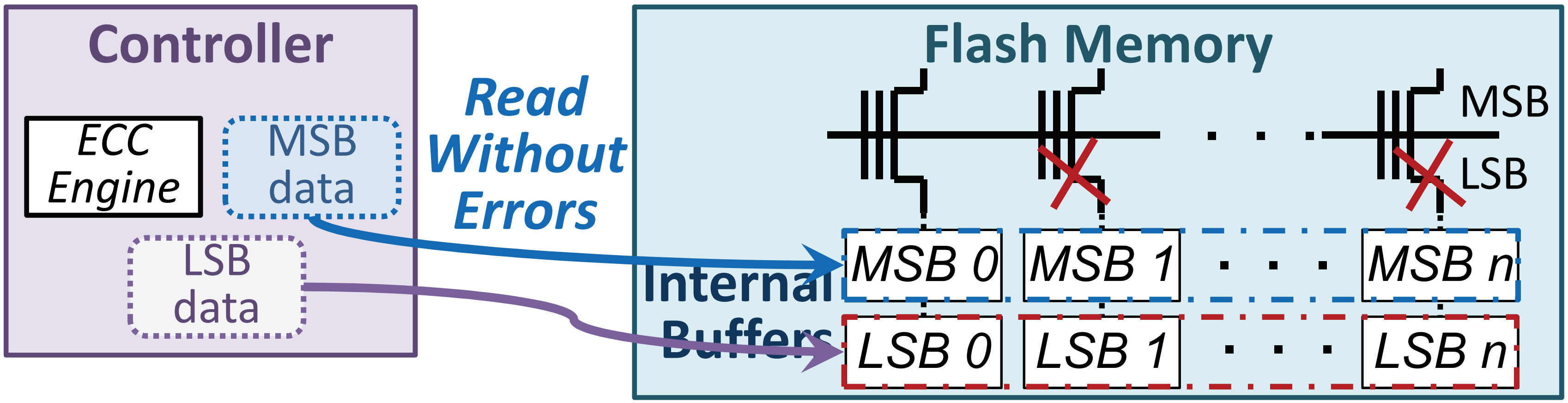
- Key Observation: During MSB programming, LSB data is read from flash cells with **uncorrected interference and read disturb errors**



1. Buffering LSB Data in the Controller

SAFARI

- Key Observation: During MSB programming, LSB data is read from flash cells with **uncorrected interference and read disturb errors**



- Key Idea: **Keep a copy of the LSB data in the controller**

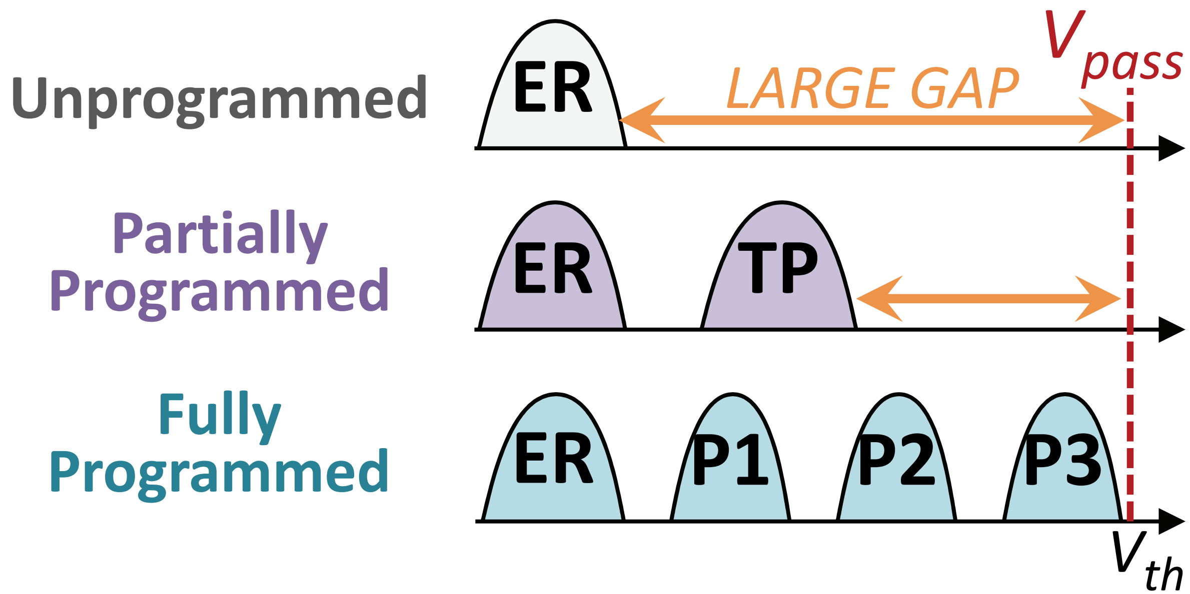
Completely eliminates vulnerabilities to program interference and read disturb

Typical case: 4.9% increase in programming latency

2. Multiple Pass-Through Voltages

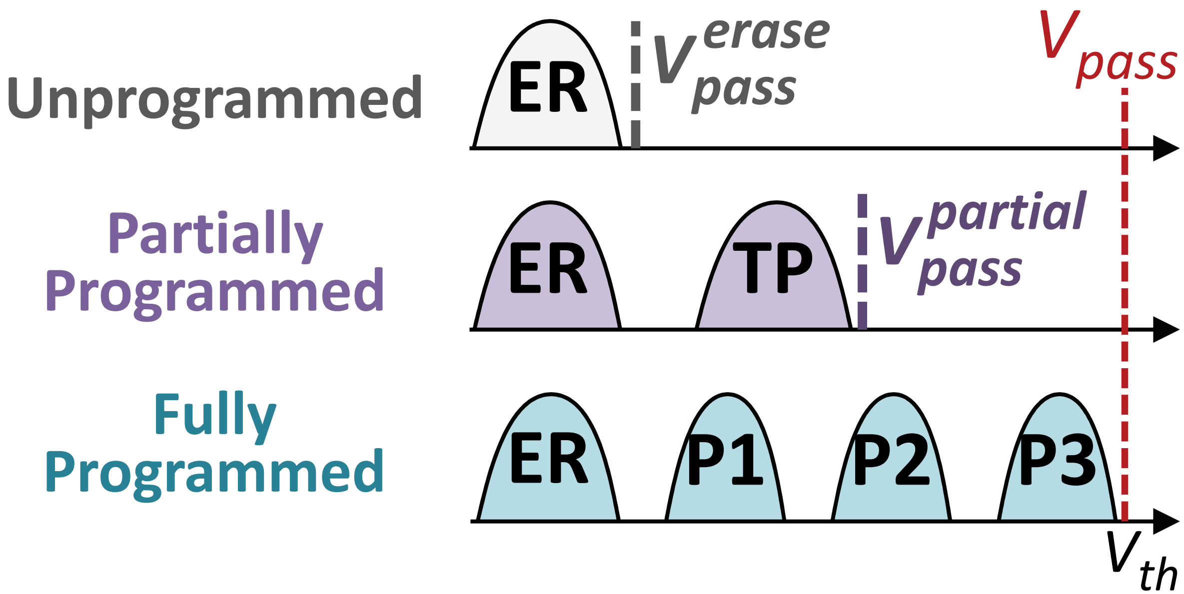
SAFARI

- Key Observation: **Large gap** between threshold voltage and pass-through voltage (V_{pass}) increases errors due to read disturb



2. Multiple Pass-Through Voltages

- Key Observation: **Large gap** between threshold voltage and pass-through voltage (V_{pass}) increases errors due to read disturb



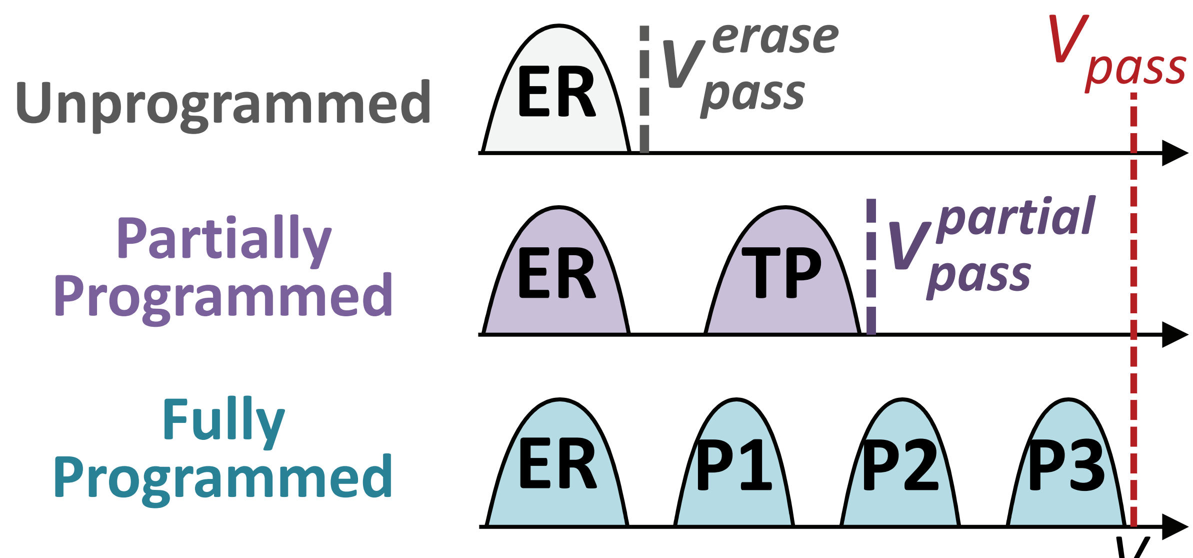
- Key Idea: **Minimize gap** by using **three pass-through voltages**

- Reduces raw bit error rate by 72%
- Increases flash lifetime by 16%

2. Multiple Pass-Through Voltages

SAFARI

- Key Observation: **Large gap** between threshold voltage and pass-through voltage (V_{pass}) increases errors due to read disturb



Mitigates vulnerabilities to read disturb

No increase in programming latency

Presentation Outline

- Executive Summary
- NAND Flash Background
- Characterizing New Vulnerabilities in Two-Step Programming
- Example Sketches of Security Exploits
- **Protection and Mitigation Mechanisms**
  - Buffering LSB Data in the Controller
  - Multiple Pass-Through Voltages
  - Adaptive LSB Read Reference Voltage: in the paper
- Conclusion

Presentation Outline

SAFARI

- Executive Summary
- NAND Flash Background
- Characterizing New Vulnerabilities in Two-Step Programming
- Example Sketches of Security Exploits
- Protection and Mitigation Mechanisms
- **Conclusion**

Executive Summary

SAFARI

- We find *new* **reliability and security vulnerabilities** in MLC NAND flash memory
  - In between two steps, cells are in a **partially-programmed** state
  - **Program interference, read disturb much worse for partially-programmed cells** than for fully-programmed cells
- We **experimentally characterize** vulnerabilities using real state-of-the-art MLC NAND flash memory chips
- We show that **malicious programs can exploit vulnerabilities** to corrupt data of other programs and reduce flash memory lifetime
- We propose **three solutions** that target vulnerabilities
  - One solution **completely eliminates vulnerabilities**, at the expense of 4.9% program latency increase
  - Two solutions **mitigate vulnerabilities**, increasing flash lifetime by 16%

Carnegie Mellon University SEAGATE ETH zürich SAFARI

Vulnerabilities in MLC NAND Flash Memory Programming

Saugata Ghose, Carnegie Mellon University

Our HPCA 2017 paper

Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques

Yu Cai[†] Saugata Ghose[†] Yixin Luo[‡†] Ken Mai[†] Onur Mutlu[§†] Erich F. Haratsch[‡]
[†]*Carnegie Mellon University* [‡]*Seagate Technology* [§]*ETH Zürich*

Modern NAND flash memory chips provide high density by storing two bits of data in each flash cell, called a multi-level cell *(MLC). An MLC partitions the threshold voltage range of a flash cell into four* voltage states. *When a flash cell is programmed, a high voltage is applied to the cell. Due to parasitic capacitance coupling between flash cells that are physically close to each other, flash cell programming can lead to* cell-to-cell program interference, *which introduces errors into* neighboring *flash cells. In order to reduce the impact of cell-to-cell interference on the reliability of MLC NAND flash memory, flash manufacturers adopt a* two-step programming method, *which programs the MLC in two separate steps. First, the flash memory partially programs the least significant bit of the MLC to some intermediate threshold voltage. Second, it programs the most significant bit to bring the MLC up to its full voltage state.*

In this paper, we demonstrate that two-step programming exposes new reliability and security vulnerabilities. We expe-

belongs to a different flash memory *page* (the unit of data programmed and read at the same time), which we refer to, respectively, as the least significant bit (LSB) page and the most significant bit (MSB) page [5].

A flash cell is programmed by applying a large voltage on the control gate of the transistor, which triggers charge transfer into the floating gate, thereby increasing the threshold voltage. To precisely control the threshold voltage of the cell, the flash memory uses *incremental step pulse programming* (ISPP) [12, 21, 25, 41]. ISPP applies multiple short pulses of the programming voltage to the control gate, in order to increase the cell threshold voltage by some small voltage amount (V_{step}) after each step. Initial MLC designs programmed the threshold voltage in *one shot*, issuing all of the pulses back-to-back to program *both* bits of data at the same time. However, as flash memory scales down, the distance between neighboring flash cells decreases, which

https://users.ece.cmu.edu/~saugatag/papers/17hpca_tsp.pdf

Our Proceedings of the IEEE paper

INVITED PAPER

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

ABSTRACT | NAND flash memory is ubiquitous in everyday life today because its capacity has continuously increased and

KEYWORDS | Data storage systems; error recovery; fault tolerance; flash memory; reliability; solid-state drives

https://arxiv.org/pdf/1706.08642

https://www.ece.cmu.edu/~safari/

Come to our posters! **Booth 833**

References to Papers and Talks

Our FMS Talks and Posters

SAFARI

- Aya Fukami, **Improving Chip-Off Forensic Analysis for NAND Flash**
- Saugata Ghose, **Vulnerabilities in MLC NAND Flash Memory Programming**
- Posters (**Booth 833**)
  - Accurate and Practical Online Flash Channel Modeling for MLC NAND Flash
  - Vulnerabilities in MLC NAND Flash Programming: Analysis, Exploits, and Mitigation
  - Error Characterization, Mitigation, and Recovery in Flash Memory Based SSDs

- **FMS 2016**
  - Onur Mutlu, **ThyNVM: Software-Transparent Crash Consistency for Persistent Memory**
  - Onur Mutlu, **Large-Scale Study of In-the-Field Flash Failures**
  - Yixin Luo, **Practical Threshold Voltage Distribution Modeling**
  - Saugata Ghose, **Write-hotness Aware Retention Management**
- **FMS 2015**
  - Onur Mutlu, **Read Disturb Errors in MLC NAND Flash Memory**
  - Yixin Luo, **Data Retention in MLC NAND Flash Memory**
- **FMS 2014**
  - Onur Mutlu, **Error Analysis and Management for MLC NAND Flash Memory**

Our Flash Memory Works (I)

SAFARI

- Summary of our work in NAND flash memory
  - Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu, Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid-State Drives, *Proceedings of the IEEE*, Sept. 2017

- Overall flash error analysis
  - Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis, DATE 2012.
  - Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai, Error Analysis and Retention-Aware Error Management for NAND Flash Memory, ITJ 2013.
  - Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu, Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory, *IEEE JSAC*, Sept. 2016

- **Flash-based SSD prototyping and testing platform**
  - Yu Cai, Erich F. Haratsh, Mark McCartney, Ken Mai, FPGA-based solid-state drive prototyping platform, FCCM 2011.

- **Retention noise study and management**
  - Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai, Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime, ICCD 2012.
  - Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu, Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery, HPCA 2015.
  - Yixin Luo, Yu Cai, Saugata Ghose, Jongmoo Choi, and Onur Mutlu, WARM: Improving NAND Flash Memory Lifetime with Write-hotness Aware Retention Management, MSST 2015.
  - Aya Fukami, Saugata Ghose, Yixin Luo, Yu Cai, and Onur Mutlu, Improving the Reliability of Chip-Off Forensic Analysis of NAND Flash Memory Devices, *Digital Investigation*, Mar. 2017.

Our Flash Memory Works (III)

SAFARI

- **Program and erase noise study**
  - Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling, DATE 2013.
  - Y. Cai, S. Ghose, Y. Luo, K. Mai, O. Mutlu, and E. F. Haratsch, Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques, HPCA 2017.

- **Cell-to-cell interference characterization and tolerance**
  - Yu Cai, Onur Mutlu, Erich F. Haratsch, and Ken Mai, Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation, ICCD 2013.
  - Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Osman Unsal, Adrian Cristal, and Ken Mai, Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories, SIGMETRICS 2014.

Our Flash Memory Works (IV)

SAFARI

- **Read disturb noise study**
  - Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch, Ken Mai, and Onur Mutlu, Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation, DSN 2015.
- **Flash errors in the field**
  - Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, A Large-Scale Study of Flash Memory Errors in the Field, SIGMETRICS 2015.
- **Persistent memory**
  - Jinglei Ren, Jishen Zhao, Samira Khan, Jongmoo Choi, Yongwei Wu, and Onur Mutlu, ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems, MICRO 2015.

Referenced Papers and Talks

SAFARI

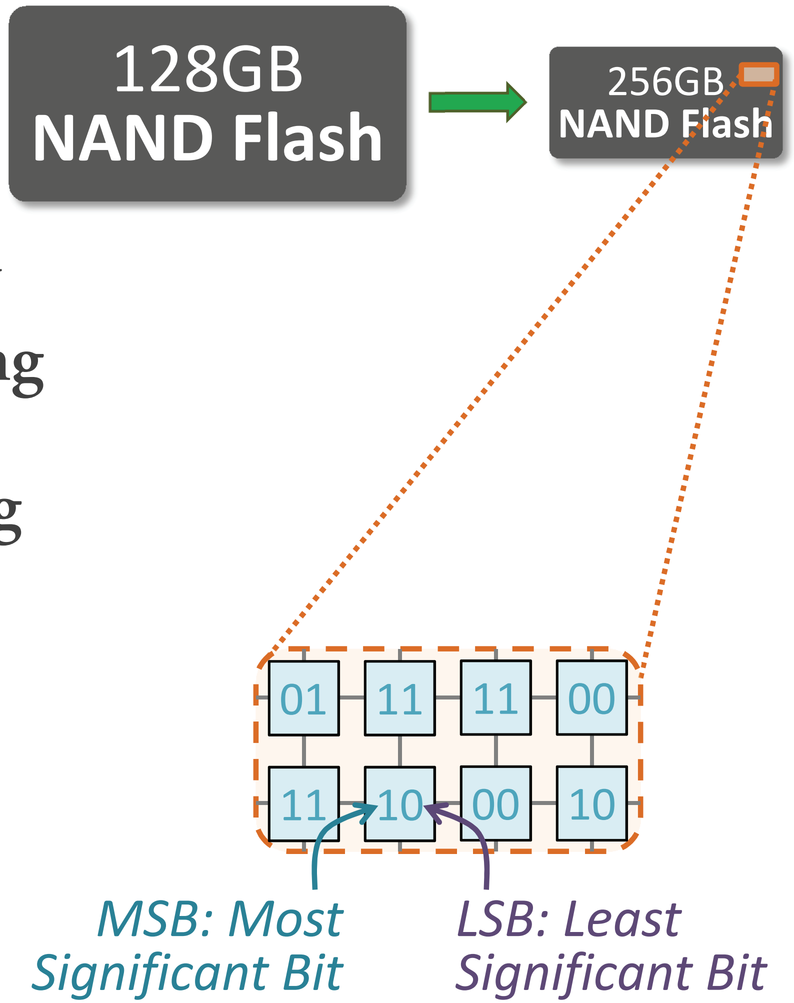
- All are available at
  - https://www.ece.cmu.edu/~safari/pubs.html
  - https://www.ece.cmu.edu/~safari/talks.html

- And, many other previous works on
  - Challenges and opportunities in memory
  - NAND flash memory errors and management
  - Phase change memory as DRAM replacement
  - STT-MRAM as DRAM replacement
  - Taking advantage of persistence in memory
  - Hybrid DRAM + NVM systems
  - NVM design and architecture

Backup Slides

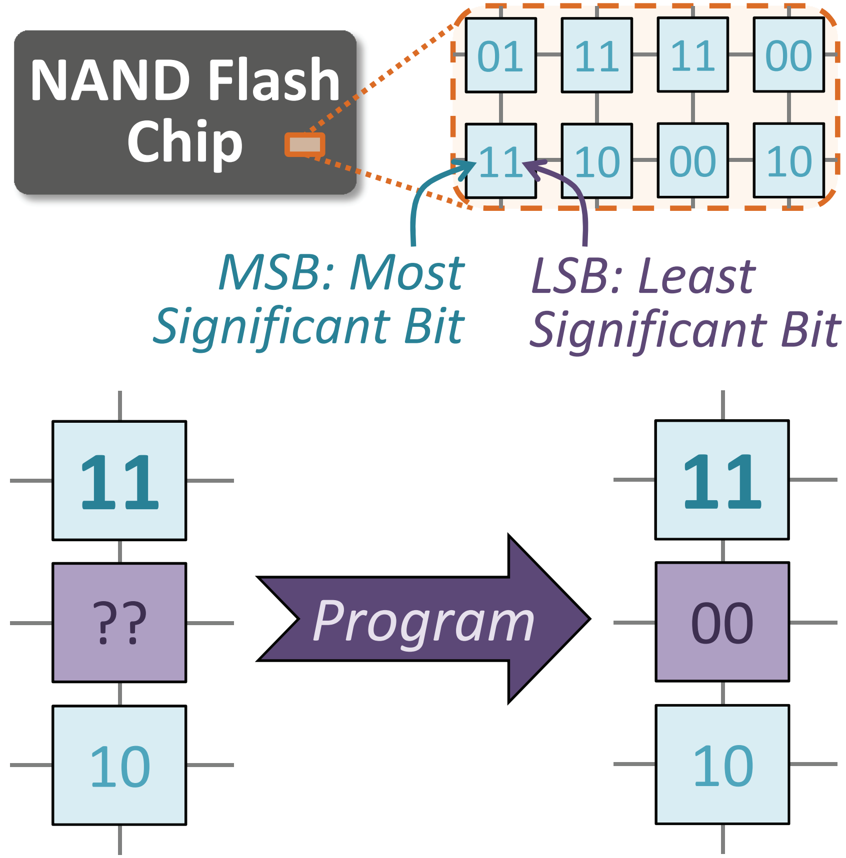
NAND Flash Memory Scaling

- SSDs use NAND flash memory chips, which contain billions of *flash cells*
- Per-bit cost of NAND flash memory has greatly decreased thanks to scaling
- Aggressive process technology scaling
  - Flash cell size decreases
  - Cells placed closer to each other
- **Multi-level cell (MLC)** technology
  - Each flash cell represents data using a *threshold voltage*
  - MLC stores **two bits of data in a single cell**



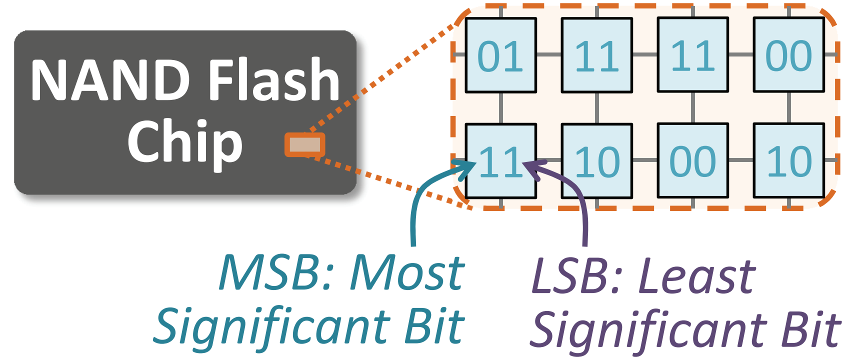
Two-Step Programming

- Per-bit cost of NAND flash memory has greatly decreased
  - Aggressive process technology scaling
  - Multi-level cell (MLC) technology
- Flash cell programmed by pulsing a large voltage to the cell transistor

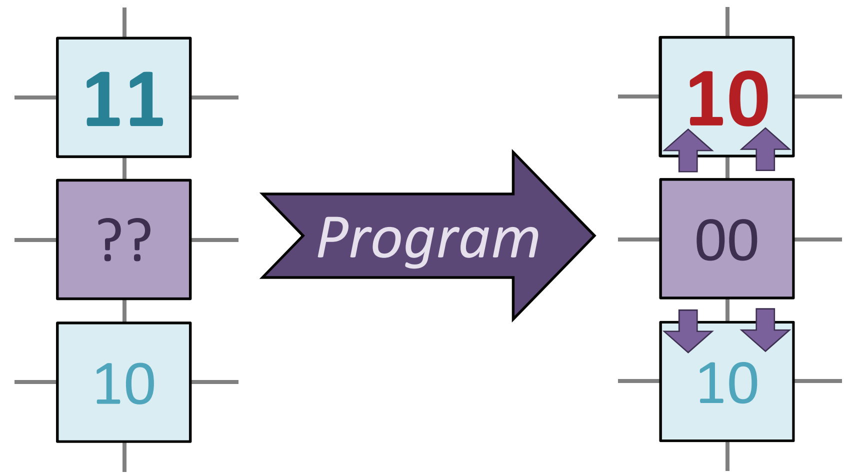


Two-Step Programming

- Per-bit cost of NAND flash memory has greatly decreased
  - Aggressive process technology scaling
  - Multi-level cell (MLC) technology

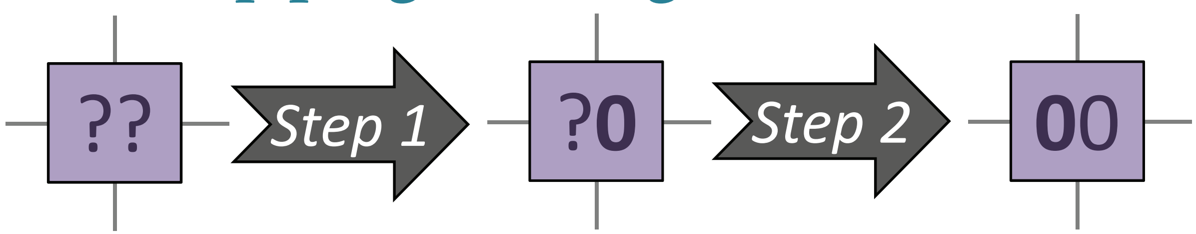


- Flash cell programmed by pulsing a large voltage to the cell transistor



- ***Cell-to-cell program interference***
  - Threshold voltage of a neighboring cell **inadvertently increases**
  - **Worsens as flash memory scales**

- Mitigation: **two-step programming**

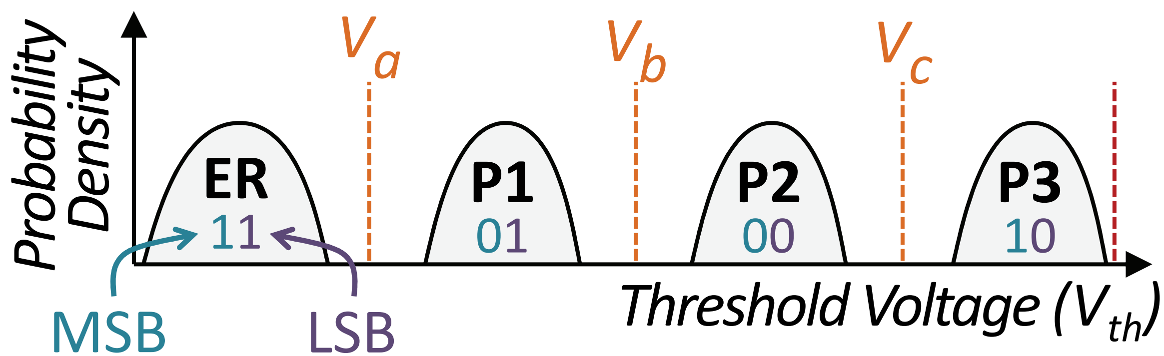


Representing Data in MLC NAND Flash Memory

SAFARI

- Flash cell uses floating-gate transistor **threshold voltage** to represent the data stored in the cell

- Threshold voltages represented as a probability distribution
  - Each two-bit value corresponds to a ***state*** (a range of threshold voltages)
  - **Read reference voltages** (V_a, V_b, V_c) identify the state a cell belongs to



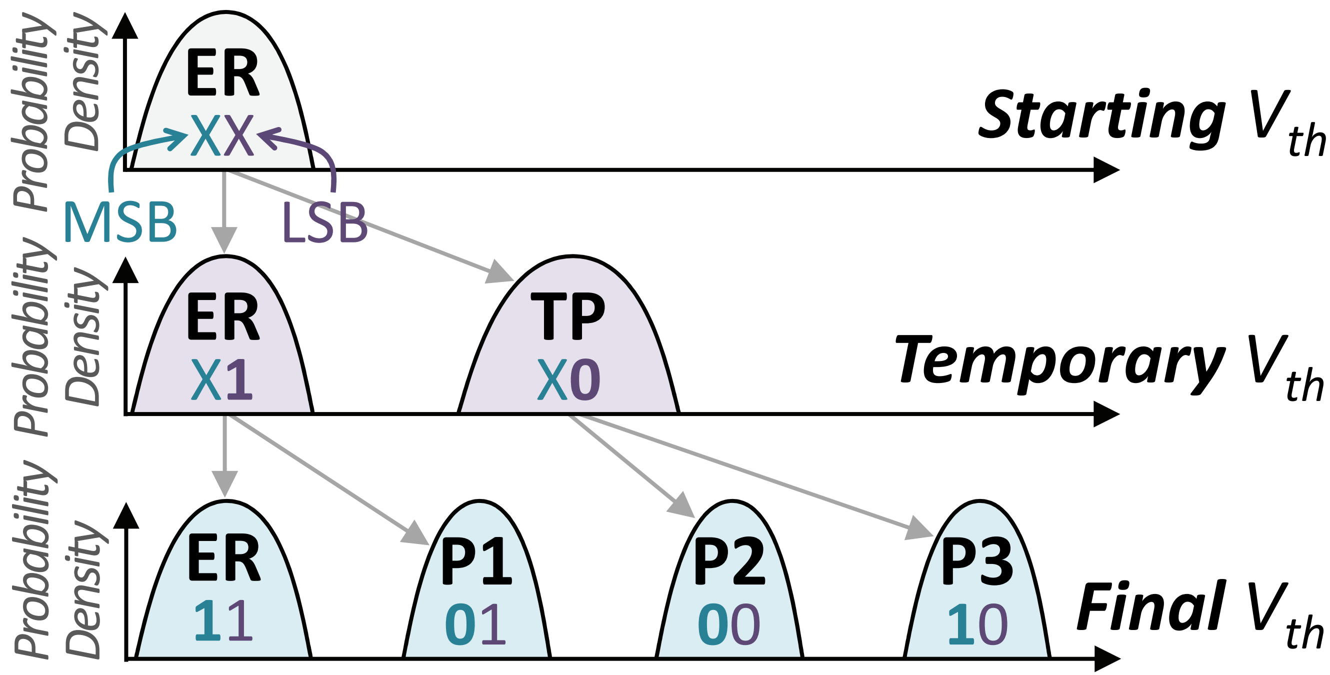
Threshold Voltage Distributions During Programming

SAFARI

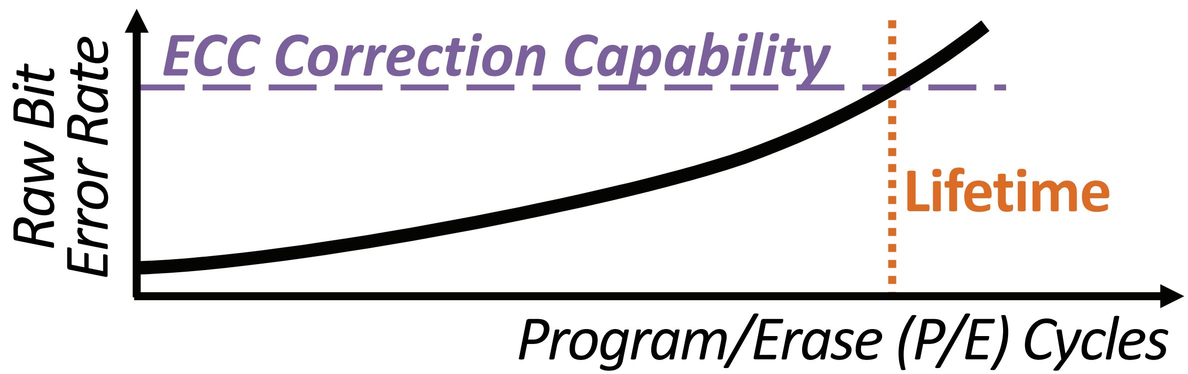
Unprogrammed

1. Program LSB

2. Program MSB

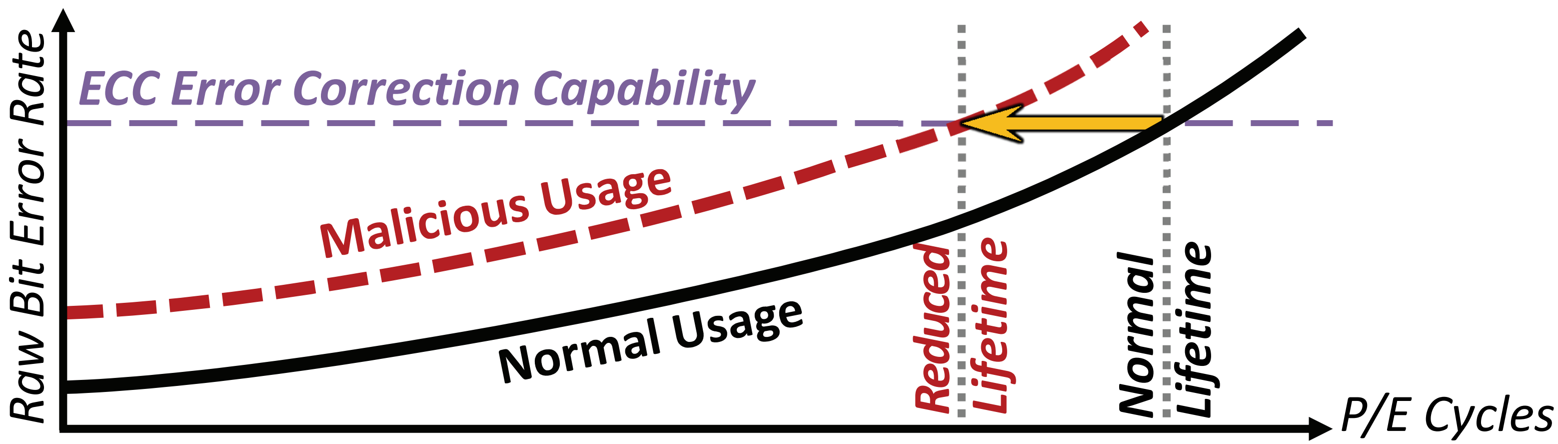


- ***Raw bit errors*** occur when the cell threshold voltage **incorrectly shifts** to a different state

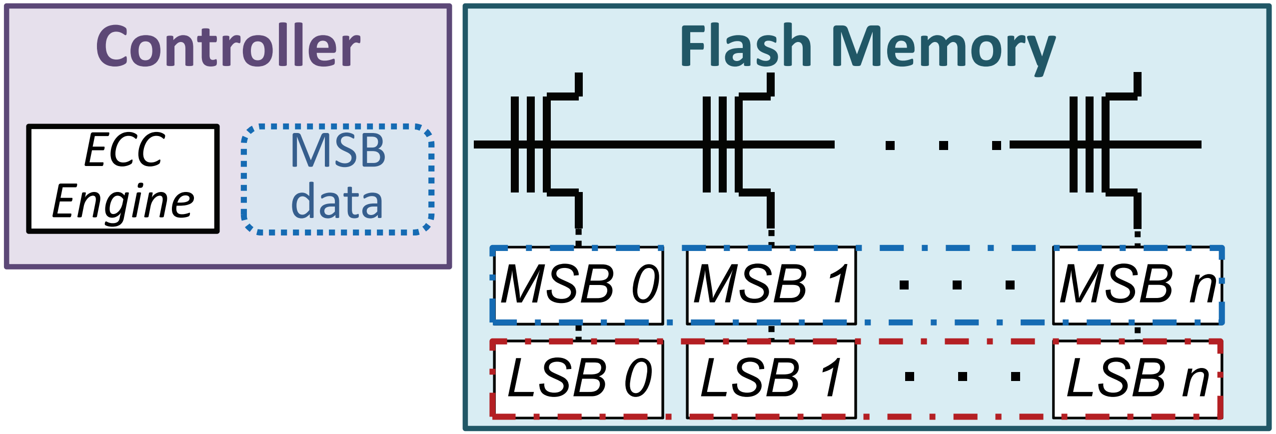


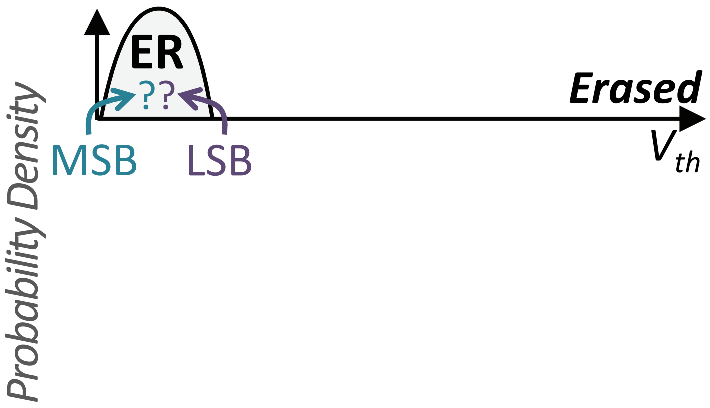
We experimentally characterize RBER, lifetime of state-of-the-art 1x-nm (i.e., 15-19nm) MLC NAND flash memory chips

Malicious Program Behavior

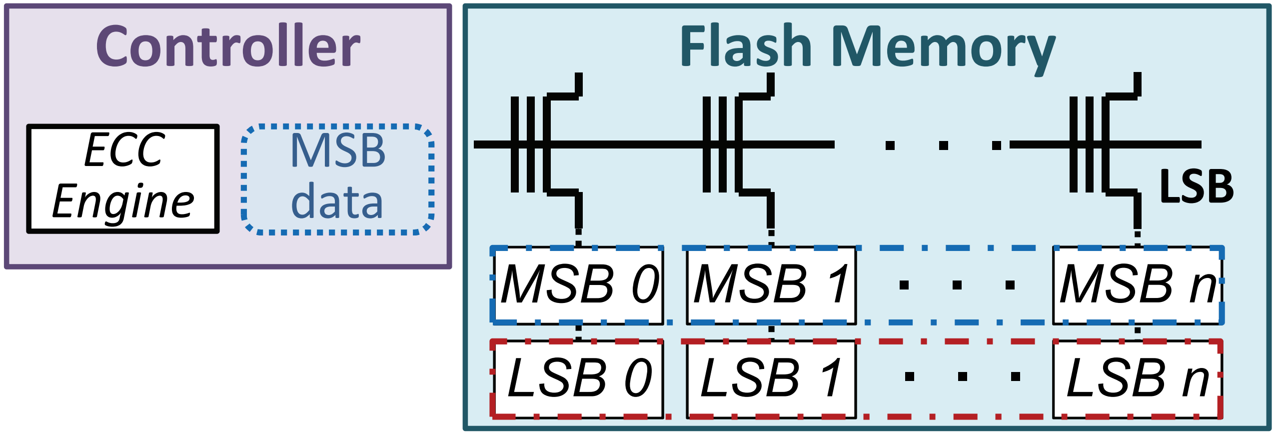


How Can Two-Step Programming Introduce Errors? SAFARI

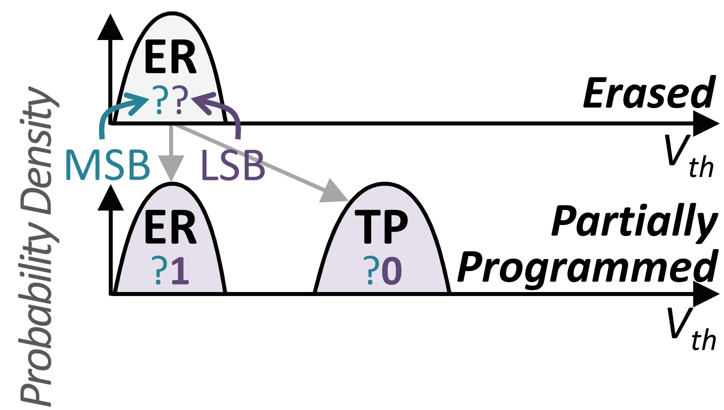




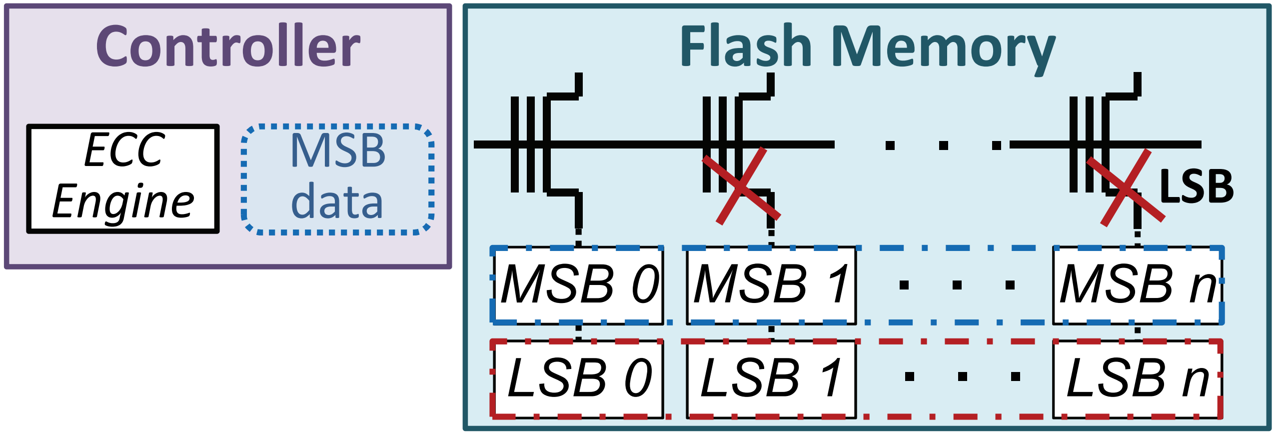
How Can Two-Step Programming Introduce Errors? SAFARI



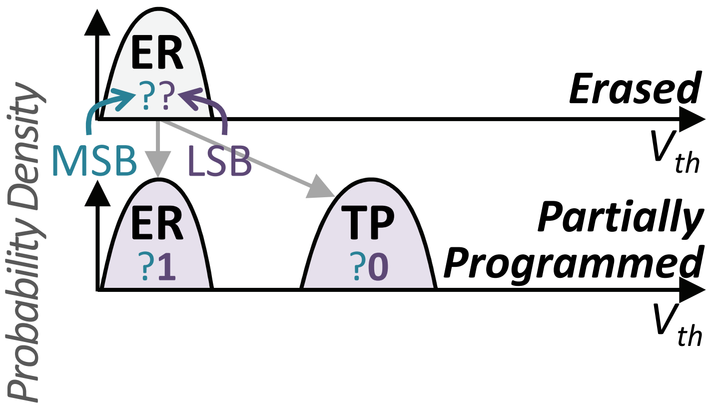
- Step 1: Program only the LSB data



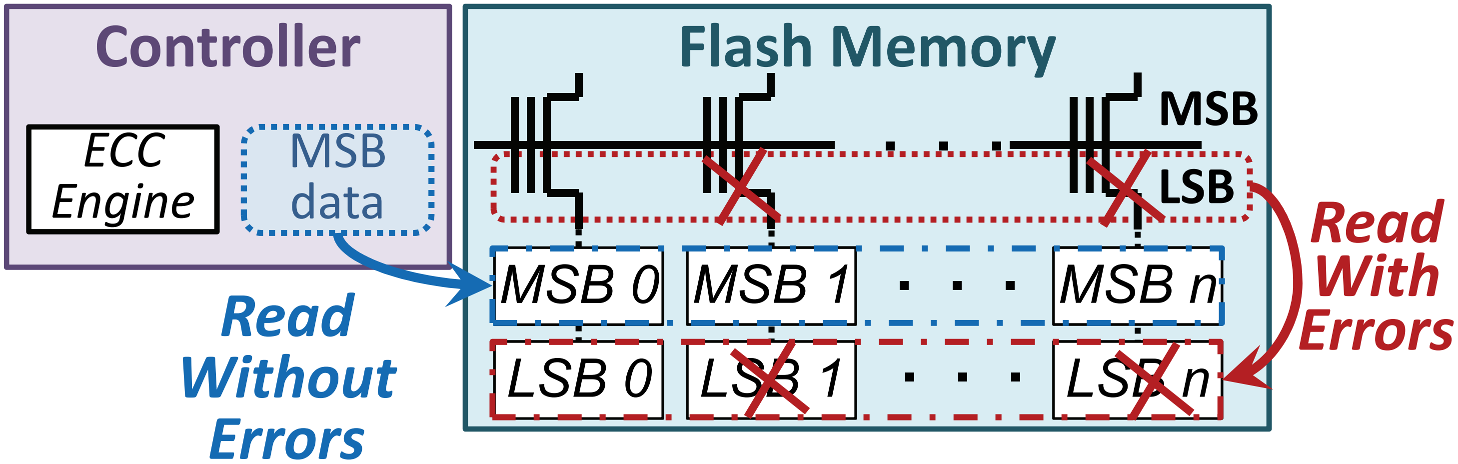
How Can Two-Step Programming Introduce Errors?



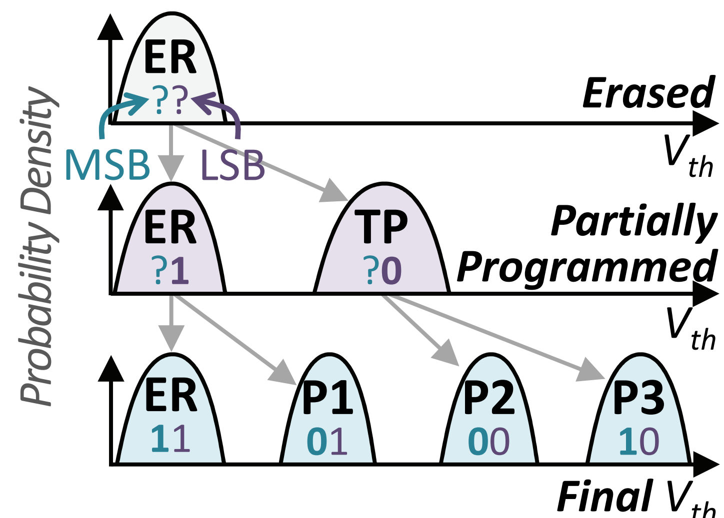
- Step 1: Program only the LSB data
- Errors are introduced into the **partially-programmed LSB data**



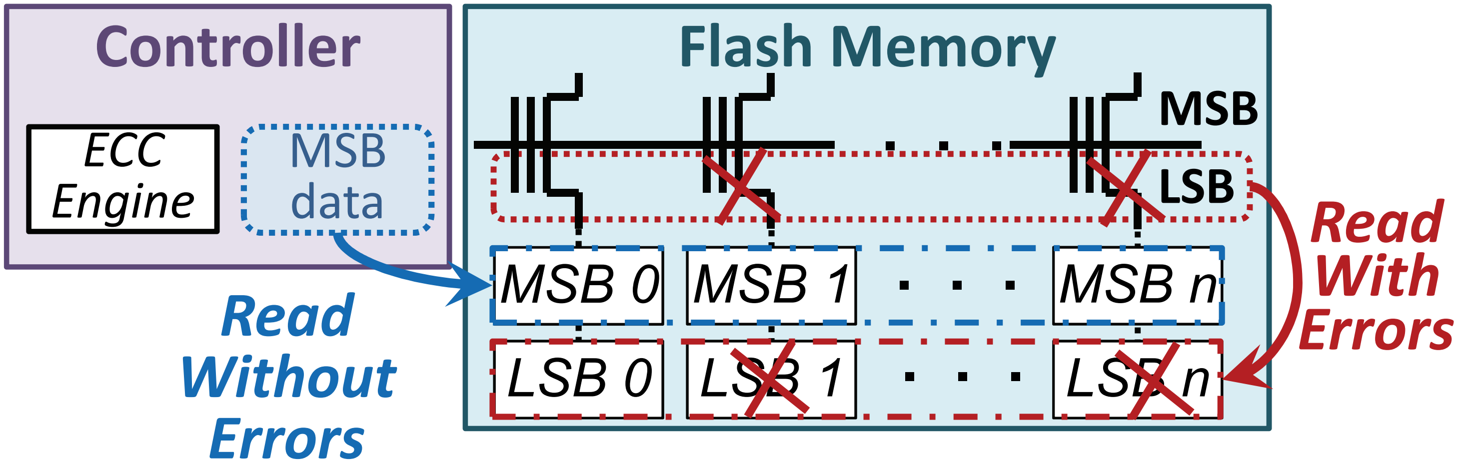
How Can Two-Step Programming Introduce Errors? SAFARI



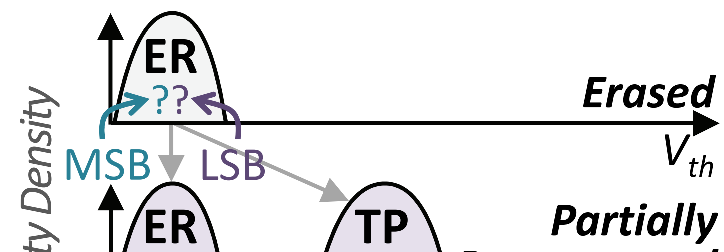
- Step 1: Program only the LSB data
- Errors are introduced into the **partially-programmed LSB data**
- Step 2: Program the MSB data
  - LSB data is read with errors directly into internal LSB buffer, **not corrected** by ECC
  - MSB data comes from controller to internal MSB buffer



How Can Two-Step Programming Introduce Errors? SAFARI



- Step 1: Program only the LSB data
- Errors are introduced into the **partially-programmed LSB data**

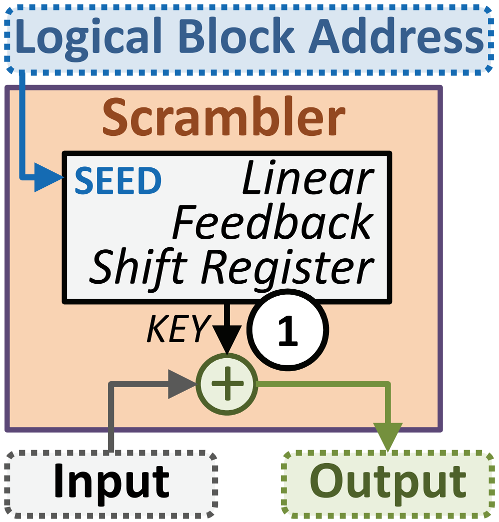


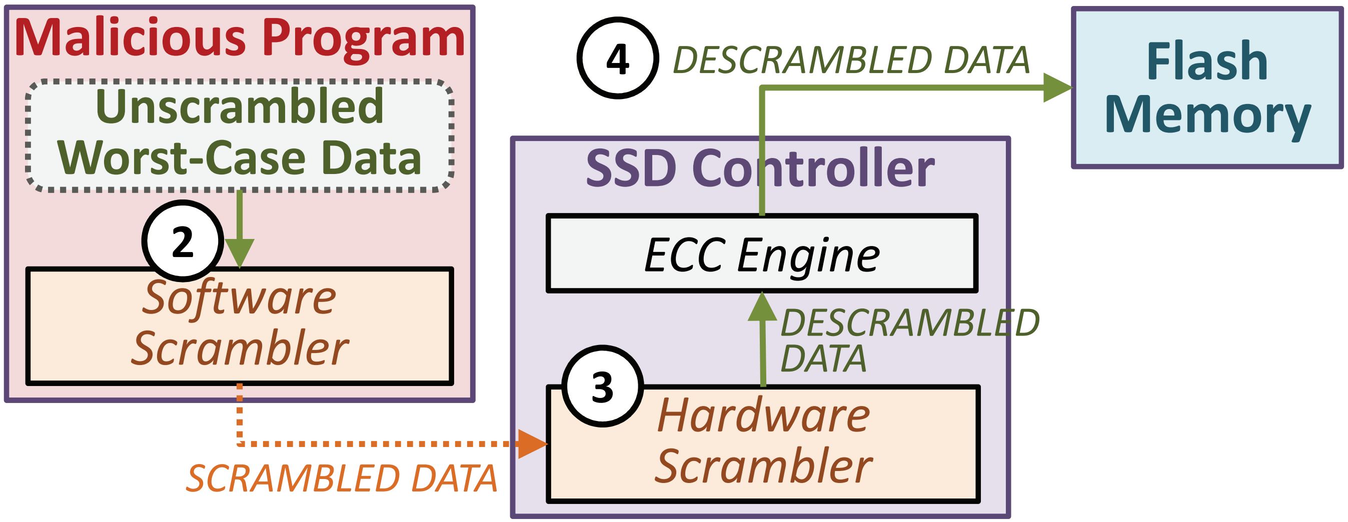
Errors in LSB data cause cell to be programmed to an incorrect state

Data Scrambler Workaround

SAFARI

- Some flash controllers employ XOR-based data scrambling





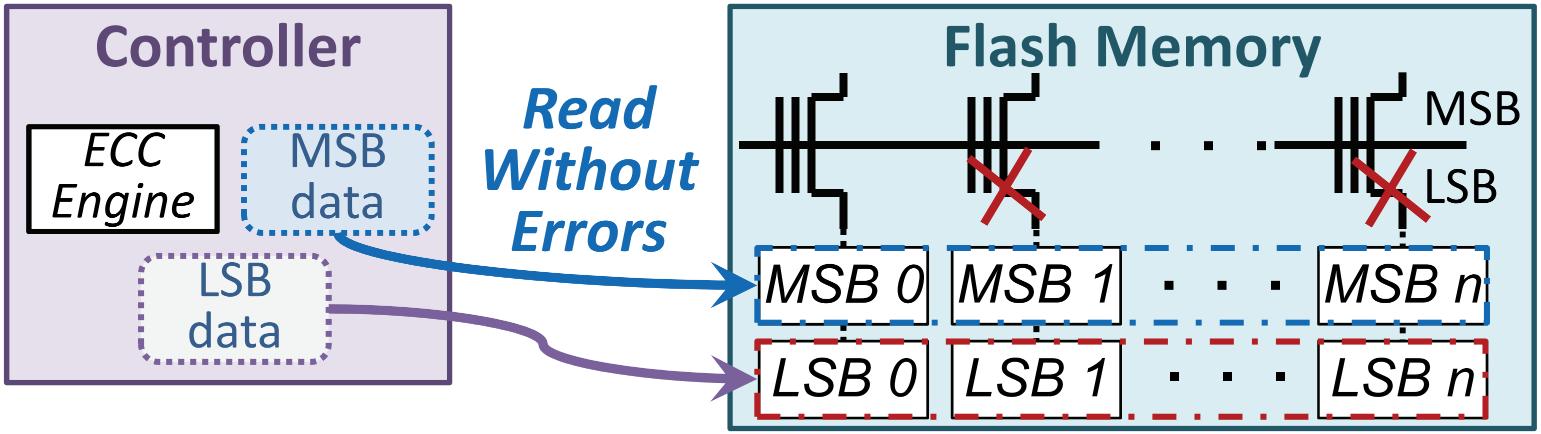
- Workaround to write worst-case data pattern
  - Recreate scrambler logic in software
  - Scramble data in software with the same seed
  - Hardware scrambler descrambles data using the same seed
  - Descrambled data written to flash memory

Sketch of Read Disturb Based Exploit

SAFARI

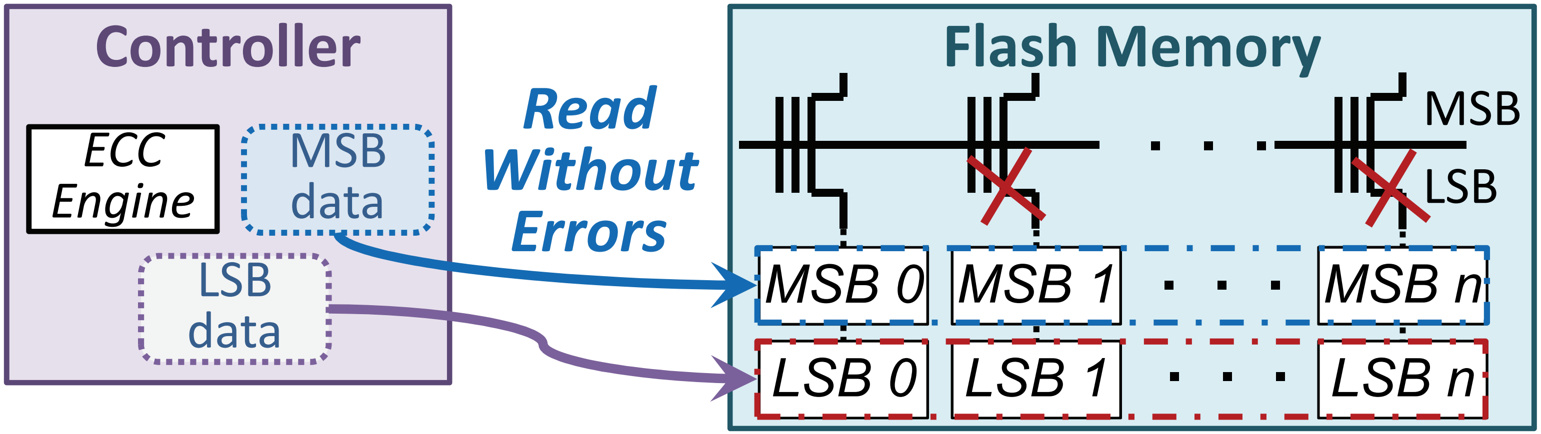
- Malicious program wants to induce errors into unprogrammed and partially-programmed wordlines in an *open block*
- Rapidly issues large number of reads to the open block
  - Write data to the open block
  - Issues ~10K reads per second **directly to the SSD** using syscalls
- Induces errors in partially-programmed data
- Induces errors in data *not yet programmed*
  - Programming can only **increase** threshold voltage
  - Exploit **increases threshold voltage before programming**, preventing cell from storing some data values
- In the paper: working around SSD caches

1. Buffering LSB Data in the Controller



- When LSB data is initially programmed, **keep a copy in the controller** DRAM
- During MSB programming, **send both LSB and MSB data from controller** to internal LSB/MSB buffers in flash memory
- Procedure to retrieve, correct data from flash memory if DRAM loses data (e.g., after power loss)

1. Buffering LSB Data in the Controller

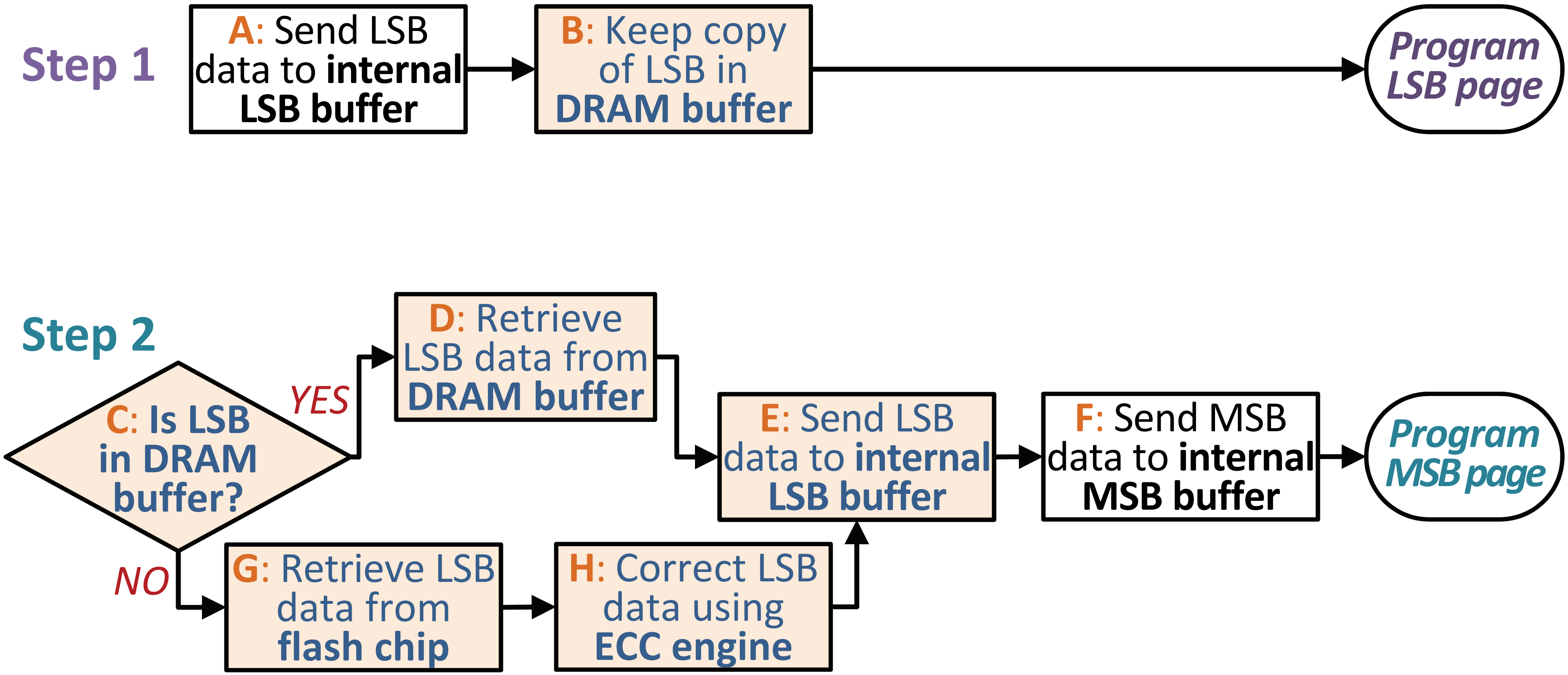


- When LSB data is initially programmed, **keep a copy in the controller** DRAM

Completely eliminates vulnerabilities to interference, read disturb

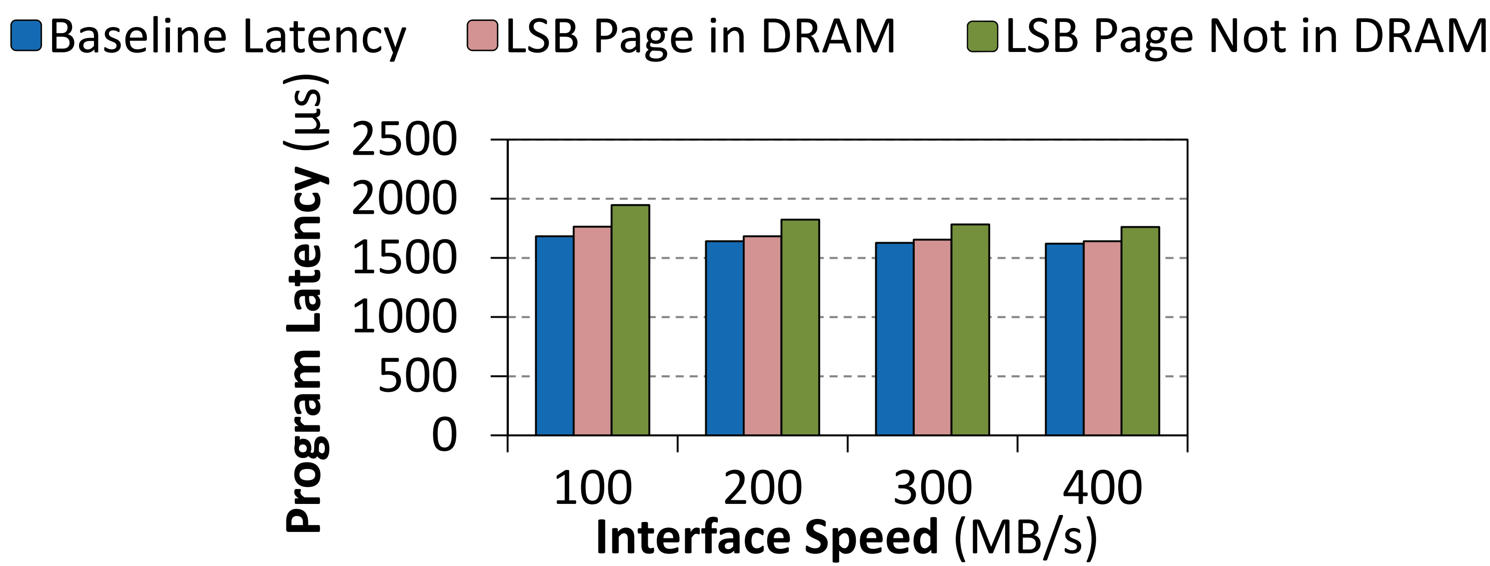
Typical case: 4.9% increase in programming latency

Algorithm for Buffering LSB Data

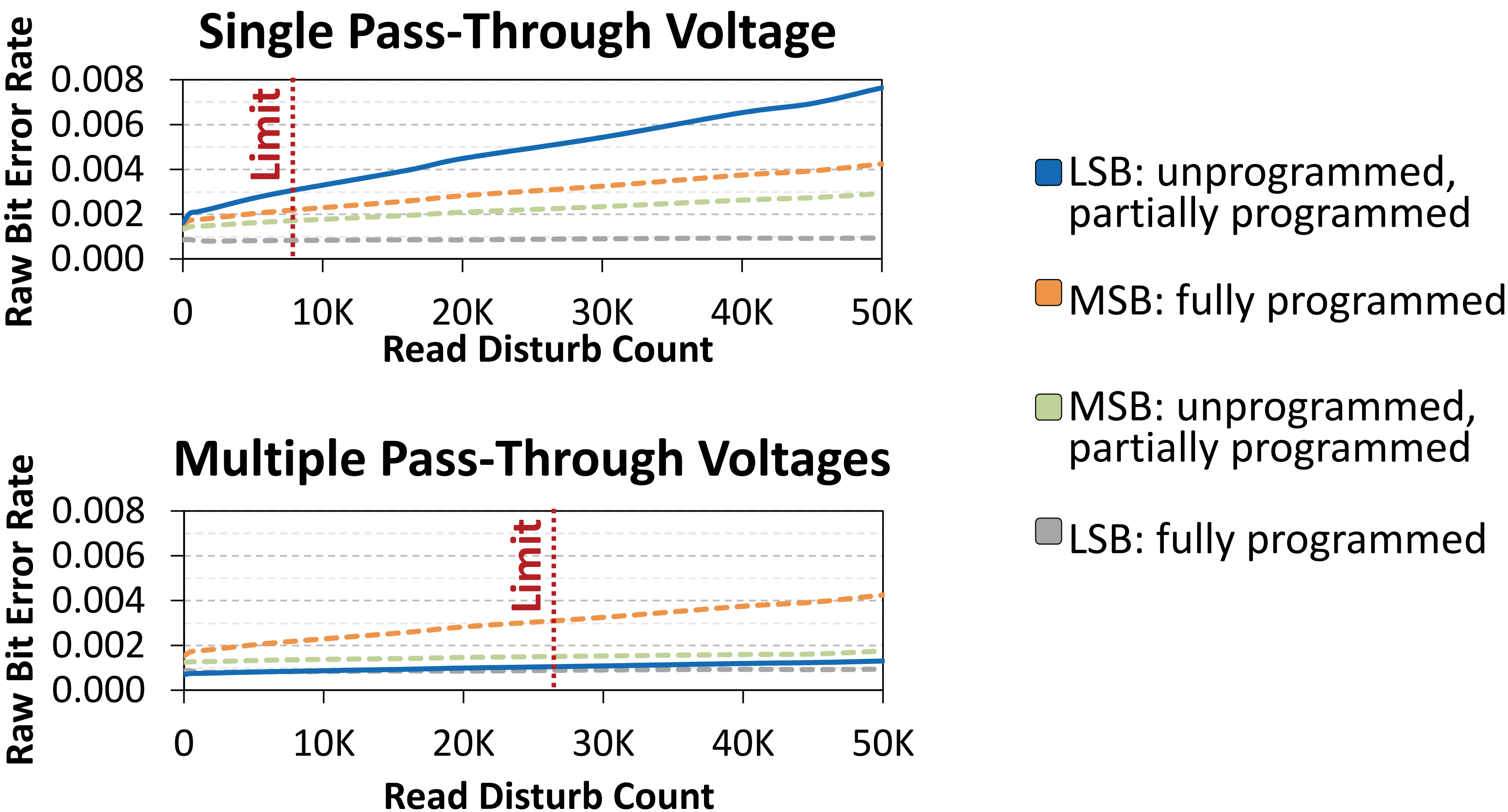


Latency Impact of Buffering

- Vary the speed of the interface between the controller and the flash memory
- Assumes 8KB page size

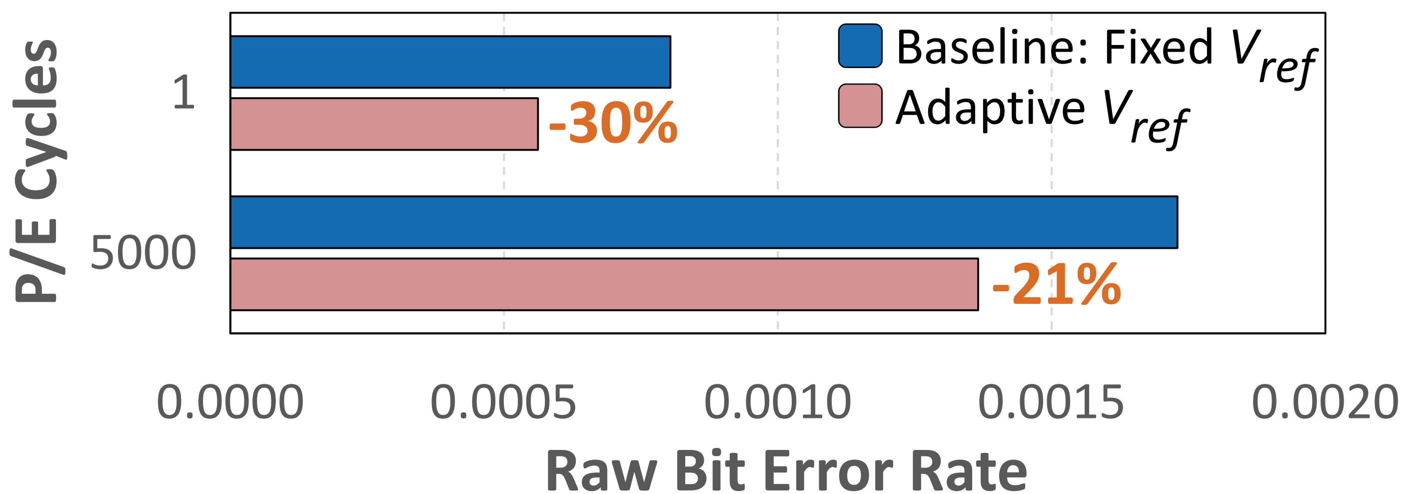


Error Rate with Multiple Pass-Through Voltages



3. Adaptive LSB Read Reference Voltage

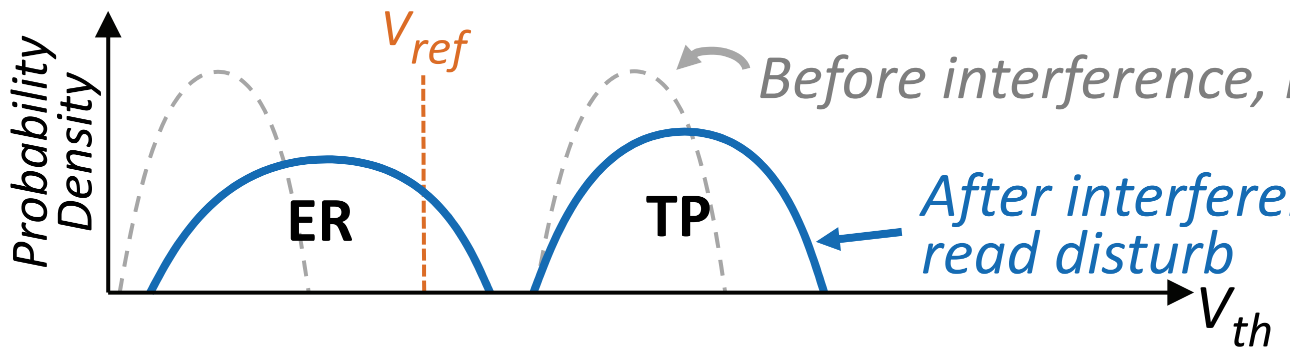
- **Adapt the read reference voltage** used to read partially-programmed LSB data
  - **Compensates for threshold voltage shifts** caused by program interference, read disturb
  - Maintain one read reference voltage per die
  - **Relearn voltage once a day** by checking error rate of test LSB data
- Reduces error count, but **does not completely eliminate errors**



3. Adaptive LSB Read Reference Voltage

SAFARI

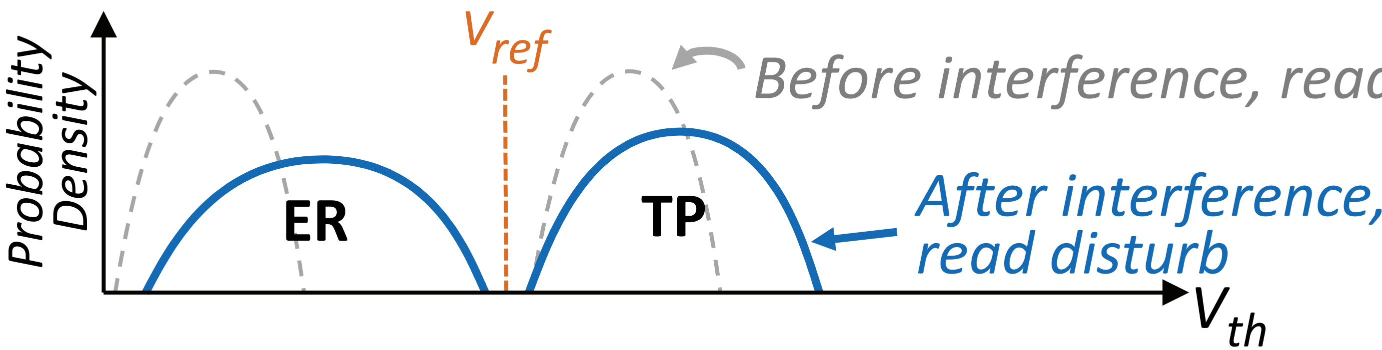
- **Adapt the read reference voltage** for partially-programmed LSB data to **compensate for voltage shifts**



3. Adaptive LSB Read Reference Voltage

SAFARI

- **Adapt the read reference voltage** for partially-programmed LSB data to **compensate for voltage shifts**

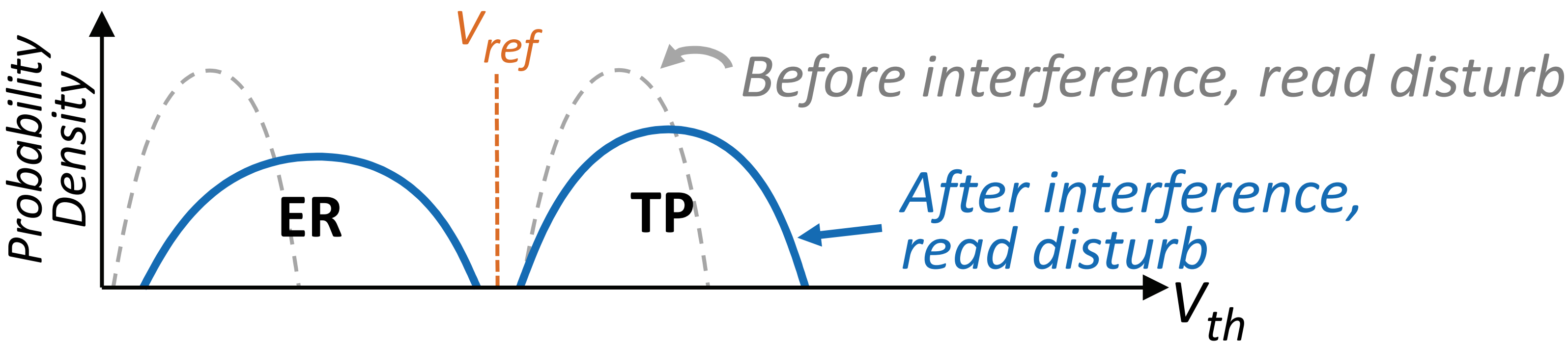


  - Program reference data value to LSBs of test wordlines
  - **Relearn voltage once a day** by checking error rate of test data

- Reduces error count by 21-30%, but **does not completely eliminate errors**

3. Adaptive LSB Read Reference Voltage

- **Adapt the read reference voltage** for partially-programmed LSB data to **compensate for voltage shifts**



- Program reference data value to LSBs of test wordlines
- **Relearn voltage once a day** by checking error rate of test data

Mitigates, but doesn't fully eliminate, vulnerabilities

No increase in programming latency

Conclusion

SAFARI

- Two-step programming used in MLC NAND flash memory
  - Introduces ***new* reliability and security vulnerabilities**
  - Partially-programmed cells susceptible to program interference and read disturb
- We **experimentally characterize** vulnerabilities using real NAND flash chips
- Malicious programs can exploit vulnerabilities to corrupt data belonging to other programs, and reduce flash memory lifetime

Solution	Protects Against	Latency Overhead	Error Rate Reduction
1. Buffering LSB in the Controller	program interference read disturb	4.9%	100%
2. Adaptive LSB Read Reference Voltage	program interference read disturb	0.0%	21-33%
3. Multiple Pass-Through Voltages	read disturb	0.0%	72% 16% lifetime increase