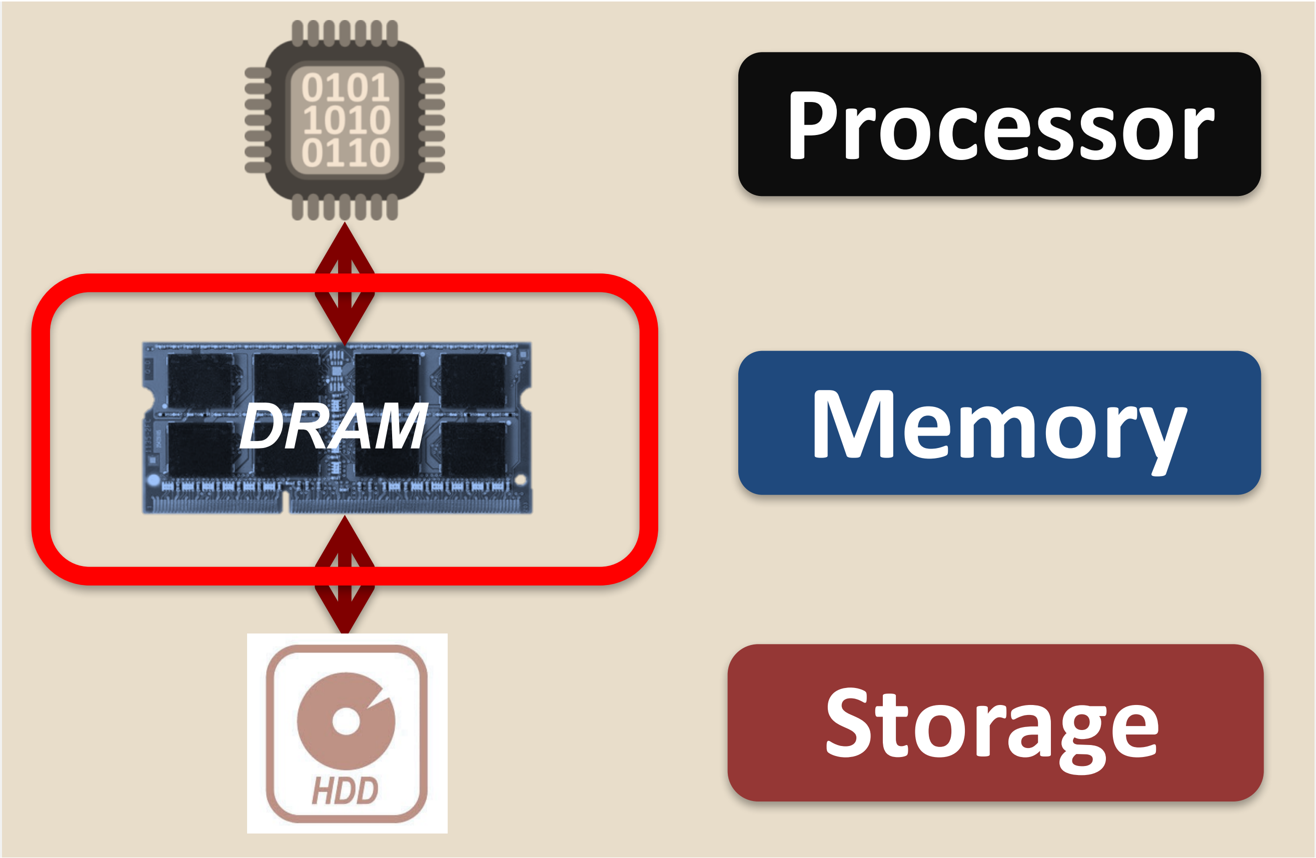
PARBOR

AN EFFICIENT SYSTEM-LEVEL TECHNIQUE TO DETECT DATA-DEPENDENT FAILURES IN DRAM

Samira Khan
Donghyuk Lee
Onur Mutlu

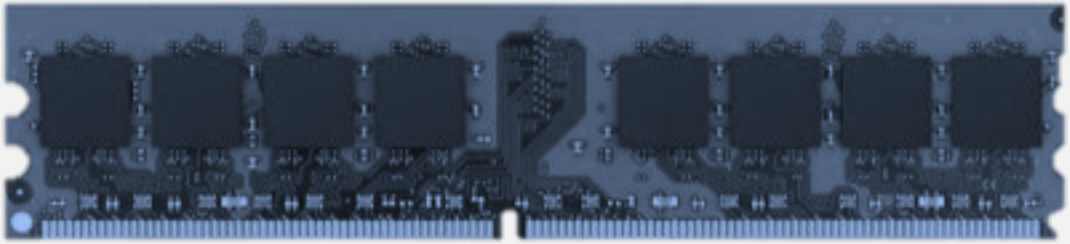
University of Virginia
Carnegie Mellon
ETH Zürich

MEMORY IN TODAY'S SYSTEM



DRAM is a critical for performance

MAIN MEMORY CAPACITY



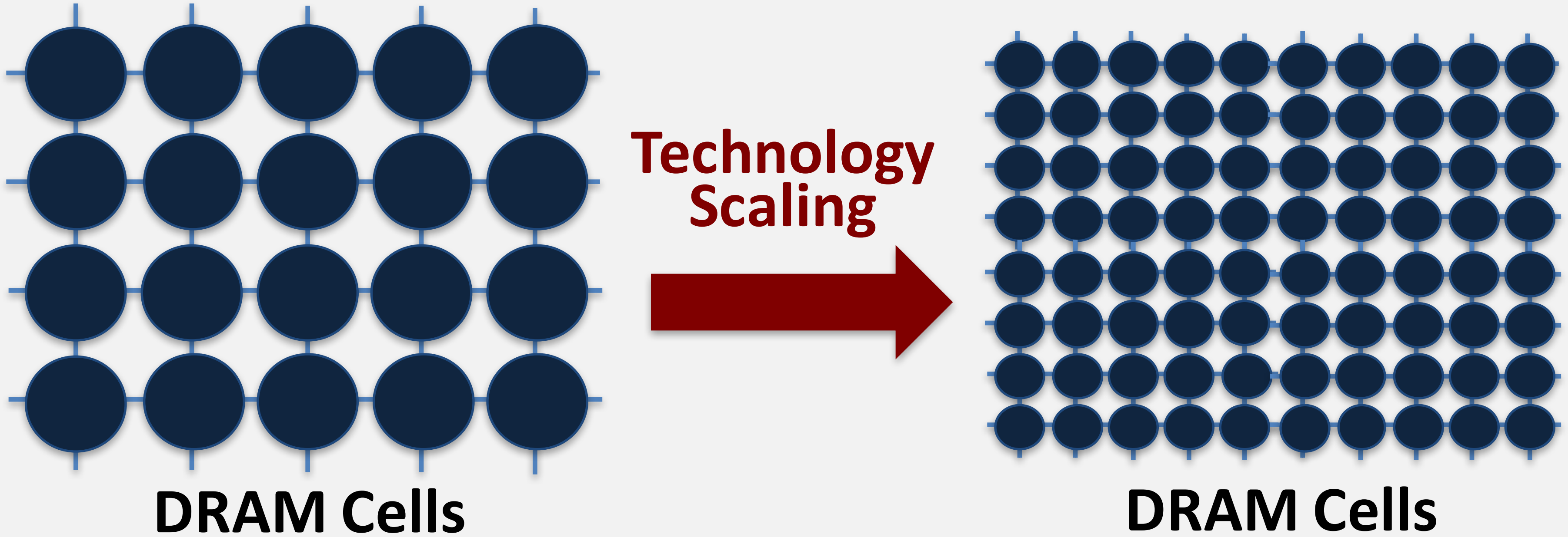
Gigabytes of DRAM

Increasing demand ***for high capacity***

1. More cores
2. Data-intensive applications

How did we get more capacity?

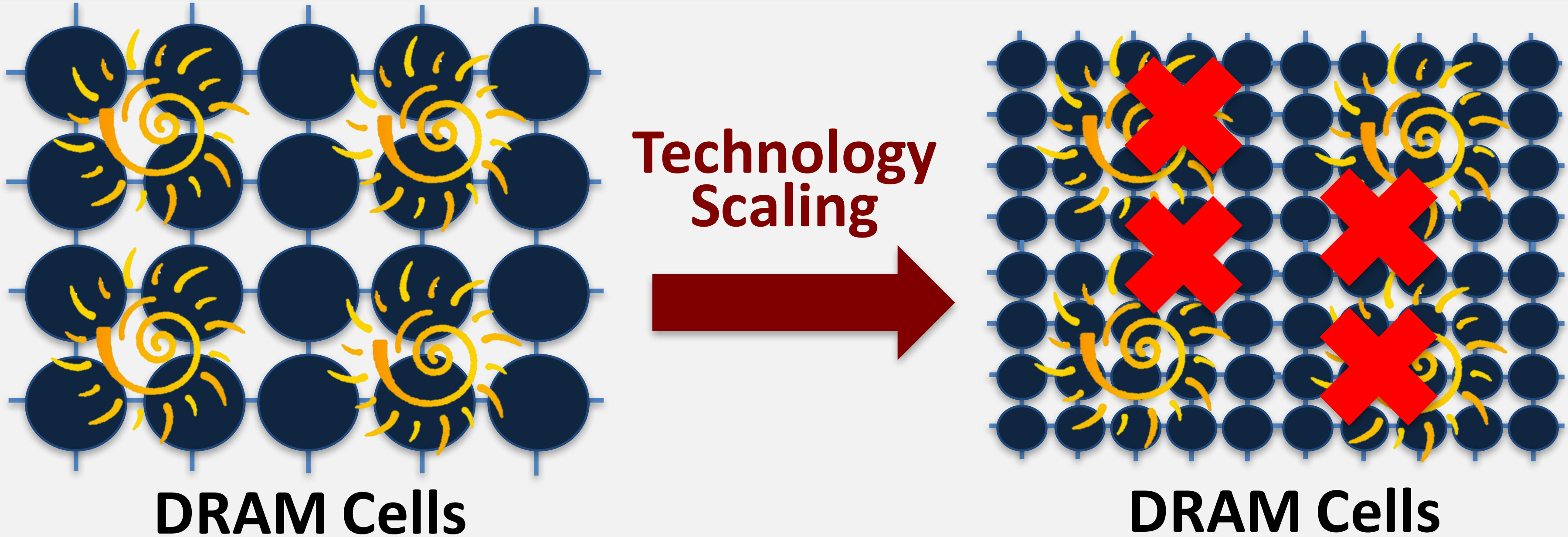
DRAM SCALING



DRAM scaling enabled high capacity

DRAM SCALING TREND

Scaling places cells in close proximity, increasing cell-to-cell interference

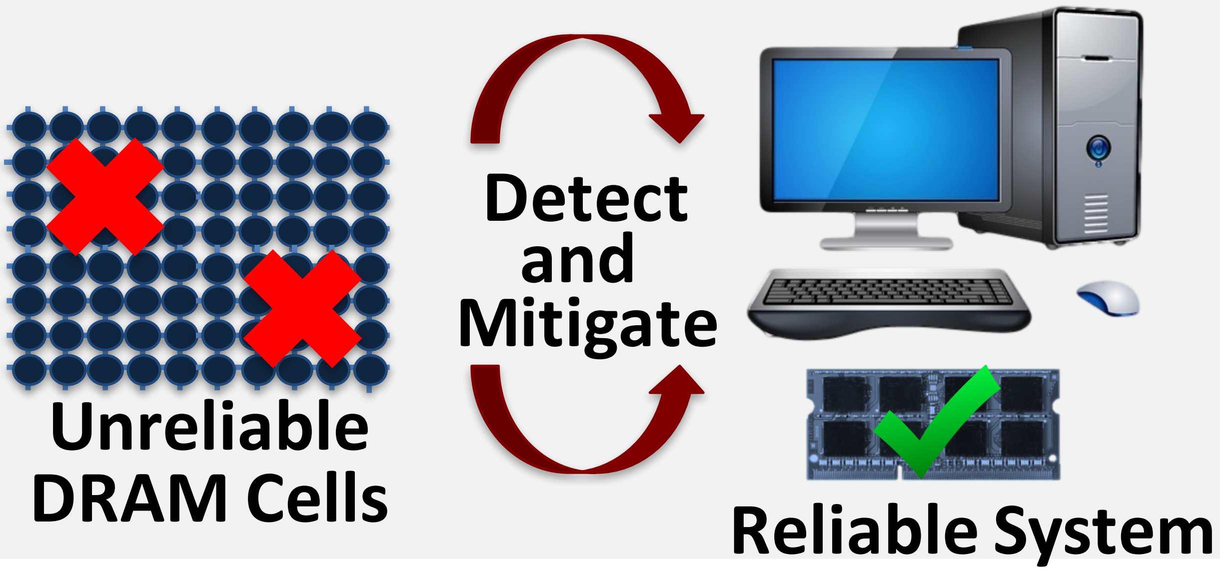


More interference results in more failures

How can we enable DRAM scaling without sacrificing reliability?

SYSTEM-LEVEL DETECTION AND MITIGATION

Manufacturers can make ***cells smaller***
without mitigating all failures



Detect and mitigate failures after the system has become operational

SYSTEM-LEVEL DETECTION AND MITIGATION

✓ Enables *scalability* *[SIGMETRICS'14, DSN'14, DSN'15]*
- *Lets vendors manufacture smaller, unreliable cells*

✓ Improves *reliability* *[ISCA'13, ISCA'14, DSN'14, DSN'15]*
- *Can detect failures that escape the manufacturing tests*

✓ Improves *latency* *[HPCA'15, HPCA'16, SIGMETRICS'16]*
- *Reduces latency for cells that do not fail at lower latency*

✓ Enables *refresh optimizations* *[ASPLOS'11, ISCA'12, DSN'15]*
- *Reduces refresh operations by using low refresh rate for robust cells*

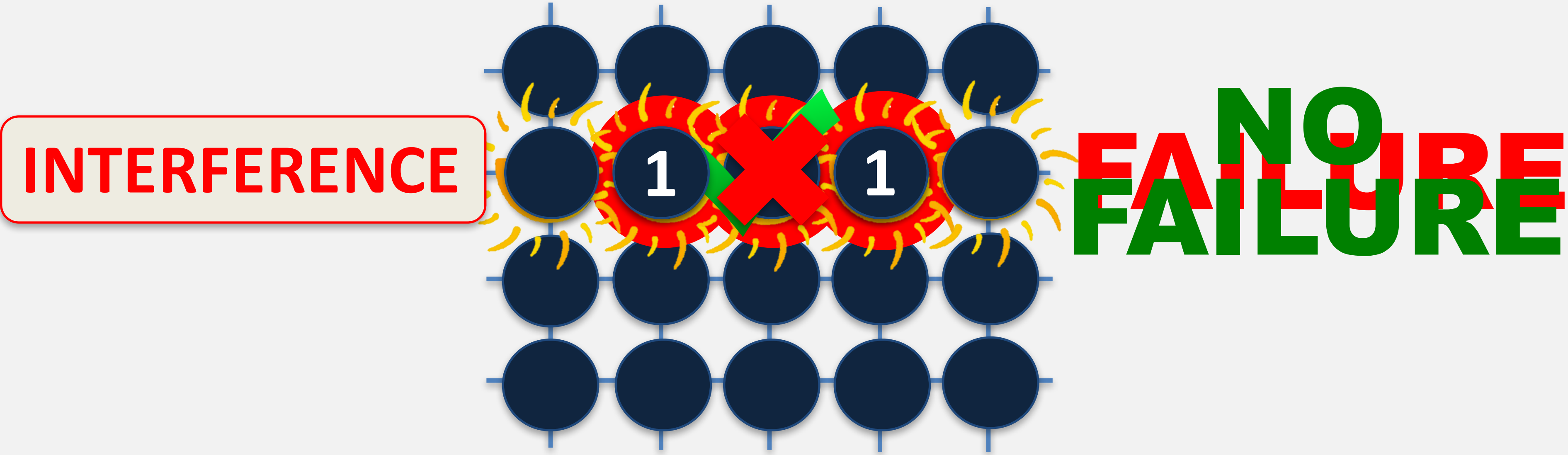
CHALLENGE

System-level detection and mitigation faces ***a major challenge*** *due to a specific type of* ***failure:***

DATA-DEPENDENT FAILURES

DATA-DEPENDENT FAILURES

Data-dependent failure is a major type of cell-to-cell interference failure



Some cells can fail depending on the data stored in neighboring cells

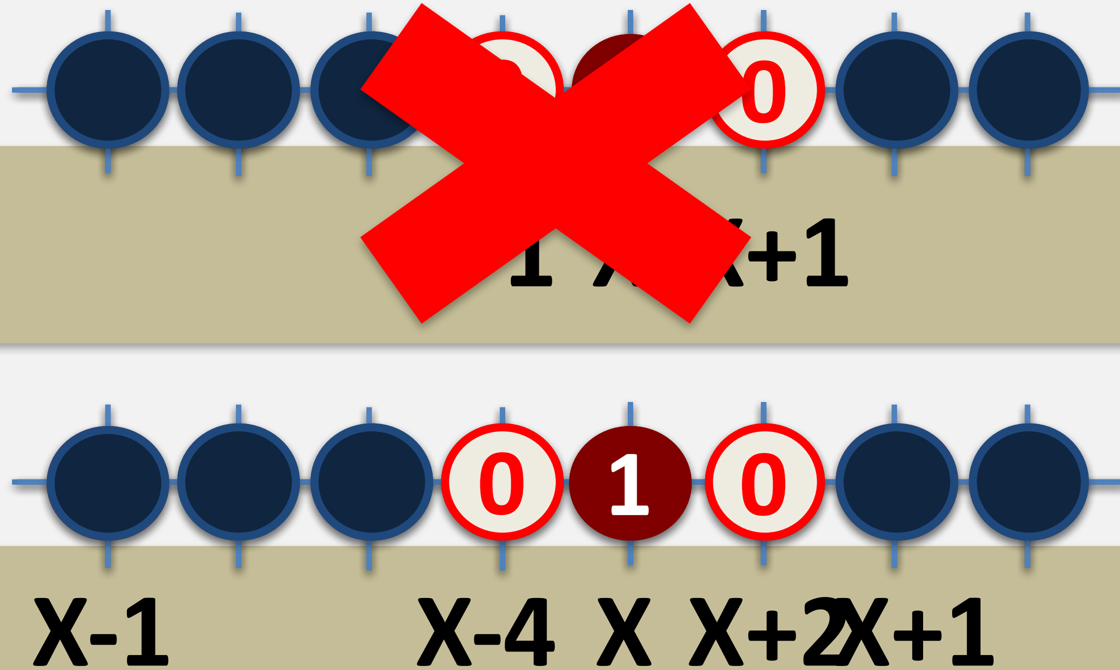
JSSC'88, MDTD'02

CHALLENGE IN DETECTING DATA-DEPENDENT FAILURES

Detect failures by writing specific patterns
in the neighboring cell addresses

LINEAR ADDRESS

SCRAMBLED ADDRESS



PROBLEM: Scrambled address is not visible to system (e.g. memory controller)

CAN WE DETERMINE THE LOCATION OF PHYSICALLY ADJACENT CELLS?

SCRAMBLED ADDRESS

X-? X X+?

NAÏVE SOLUTION

For a given failure X,

test every combination of two bit addresses in the row

$O(n^2)$

8192*8192 tests, 49 days for a row with 8K cells

Not feasible in a real system

OUR APPROACH: PARBOR

Goal:
A fast and efficient way to determine the locations of neighboring cells

PARBOR: Summary

A new technique to determine the locations of neighboring DRAM cells

- Reduces test time using *two key ideas:*

- **Exploits** *heterogeneity in cell interference* to reduce test time by detecting *only one neighbor*

- **Exploits** *DRAM regularity and parallelism* to detect *all neighbor locations by running parallel tests in multiple rows*

Detects neighboring locations within 60-99 tests in 144 real DRAM chips, a 745,654X reduction compared to naïve tests

OUTLINE

Data-Dependent Failures

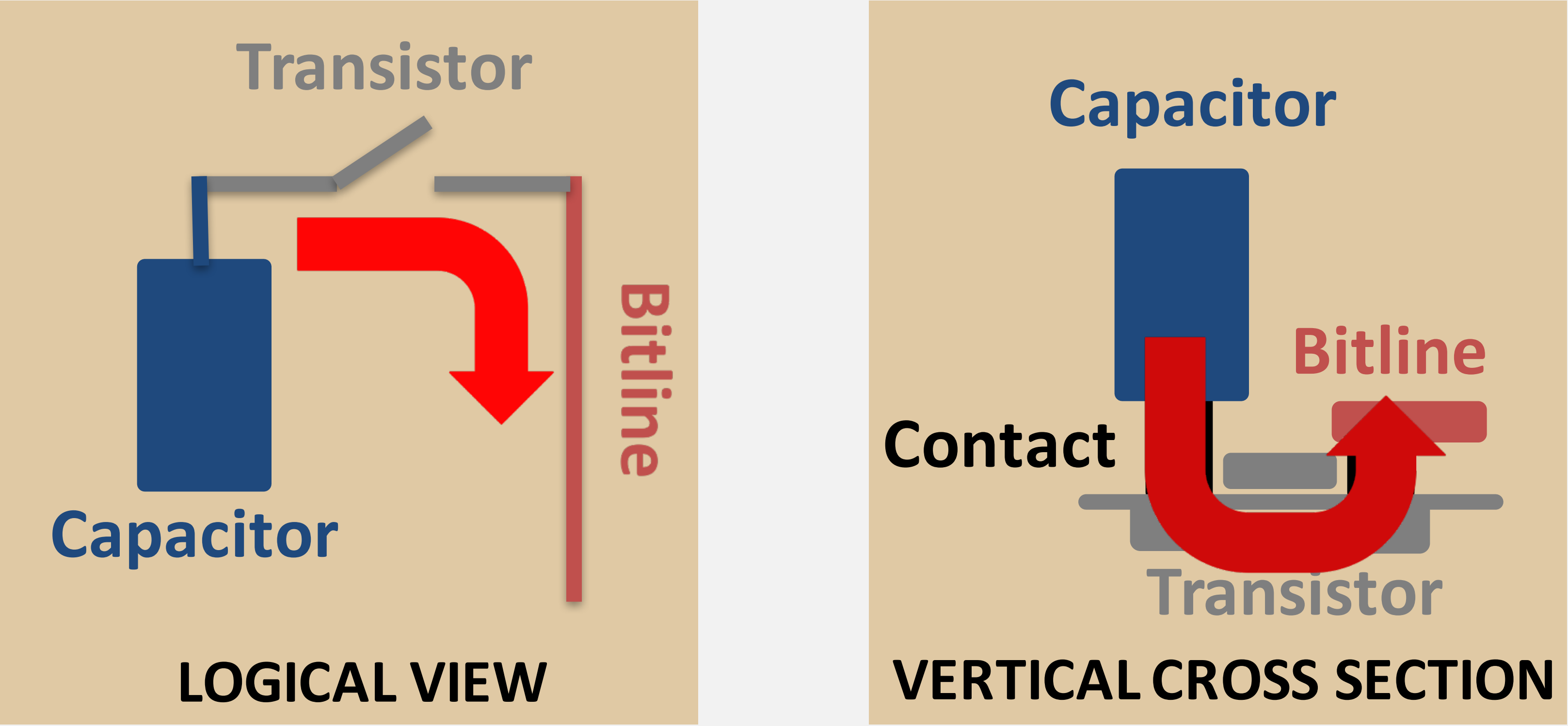
Challenges in System-Level Detection

Our Mechanism: PARBOR

Experimental Results from Real Chips

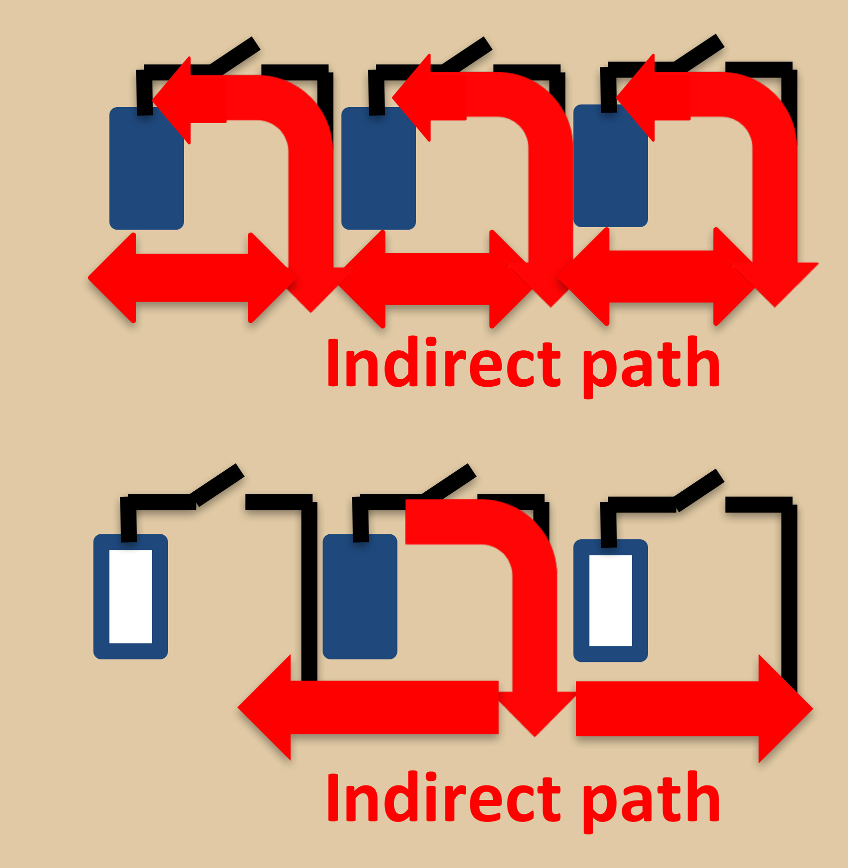
Use Cases

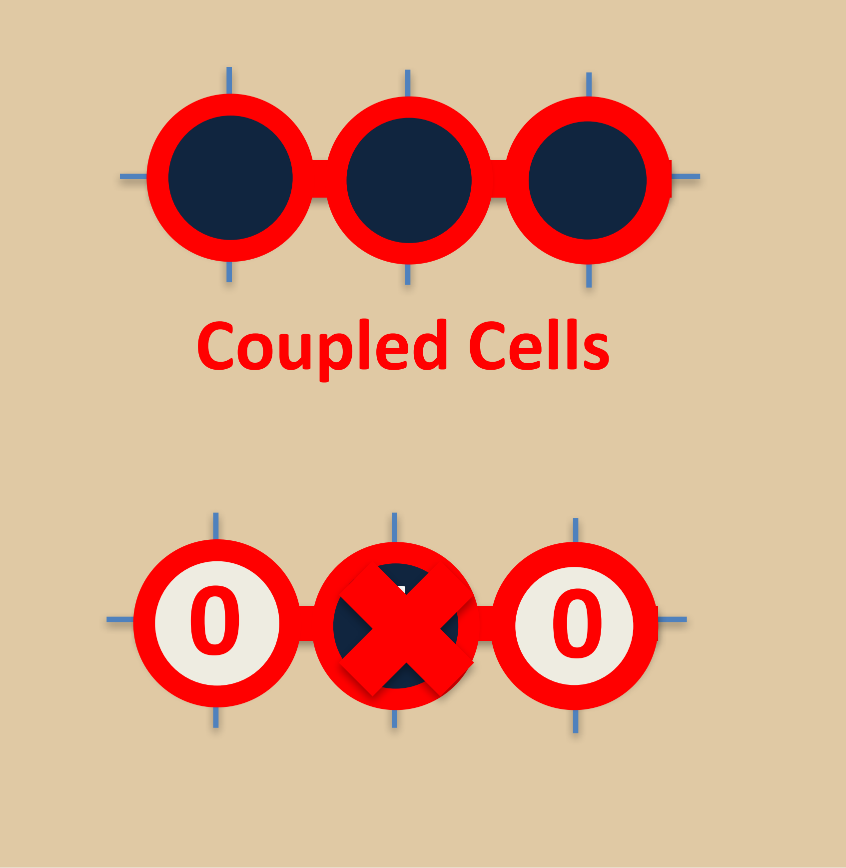
A DRAM CELL



A DRAM cell

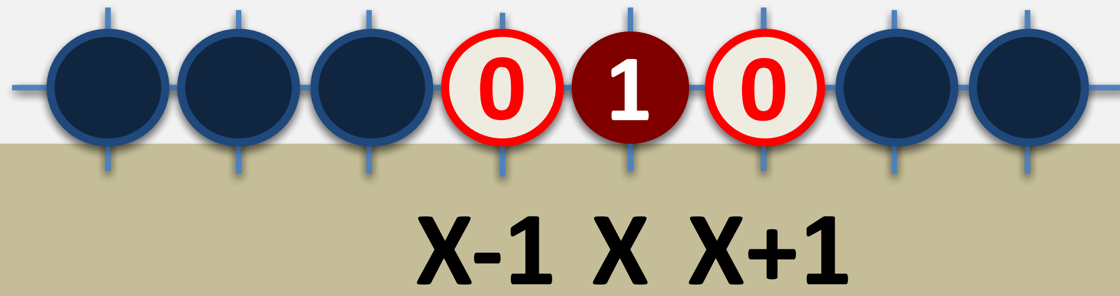
DATA-DEPENDENT FAILURES





Failures depend on the data content in neighboring cells

DETECTING DATA-DEPENDENT FAILURES



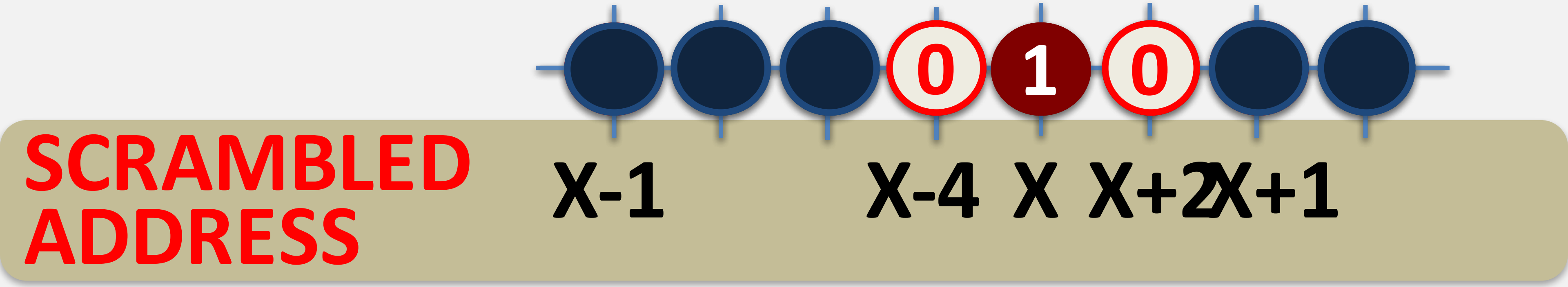
To test cell at ***address X,*** write ***1 at address X*** and ***0s at address X+1 and X-1***

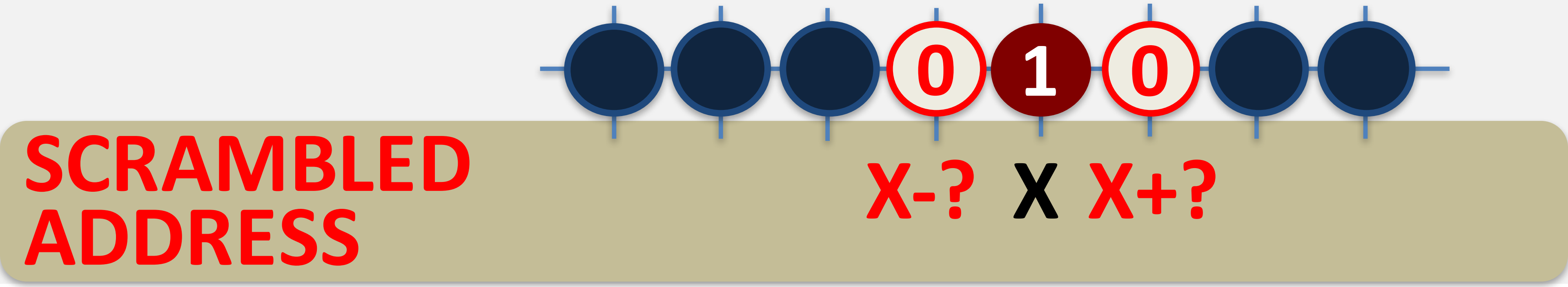
Need to write specific data patterns in neighboring addresses

OUTLINE

- Data-Dependent Failures
- Challenges in System-Level Detection
- Our Mechanism: PARBOR
- Experimental Results from Real Chips
- Use Cases

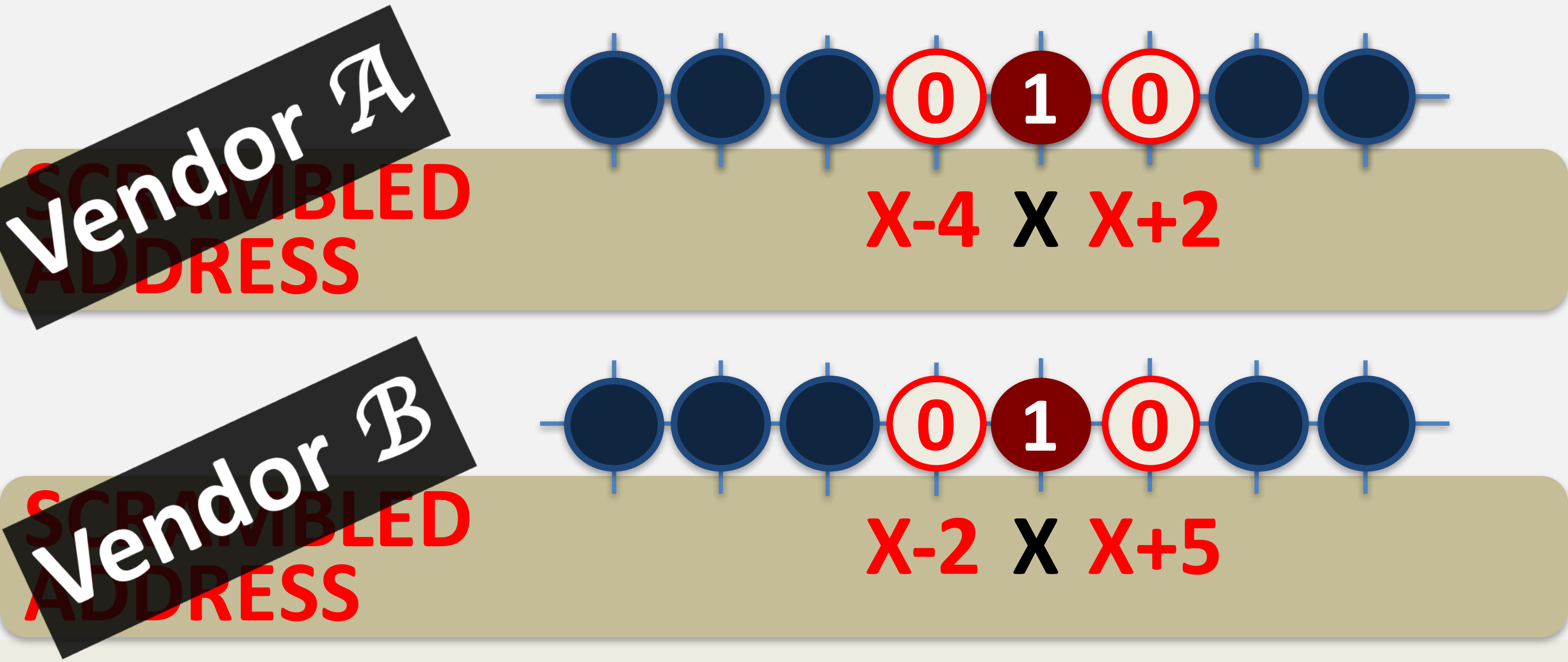
CHALLENGE: SCRAMBLED ADDRESS SPACE





- Scrambled address *not visible to system*
- Cannot detect failures without the *address mapping information*

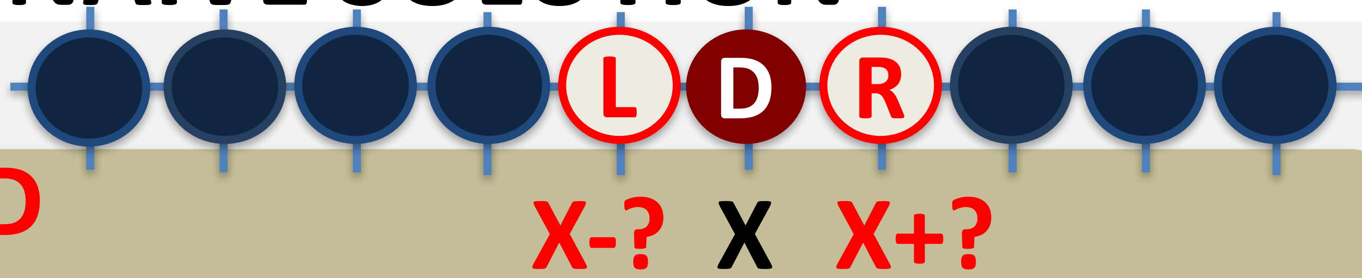
CHALLENGE: SCRAMBLED ADDRESS SPACE



- Different for *each generation and vendor*
- Need a *dynamic way* to detect address mapping information *in the system*

NAIVE SOLUTION

SCRAMBLED ADDRESS



Determine the location of neighboring cells

NAÏVE SOLUTION: O(n^2)

- **For a given failure X, test *every combination of two bit addresses* in the row**
  - **Address bits: (0, 0), (0, 1), … (X-1, X), (X, X+1) … (n-1, n)**
- For vendor A
  - X will ***fail only*** when ***X-4, X+2*** tested

8192*8192 tests, 49 days for a row with 8K cells
Not feasible in a real system

GOAL

A fast and efficient way to determine the locations of neighboring cells

OUTLINE

Data-Dependent Failures

Challenges in System-Level Detection

Our Mechanism: PARBOR

Experimental Results from Real Chips

Use Cases

PARBOR: KEY OBSERVATIONS

Reduces test time based on two key observations:

<u>Key observation 1:</u>

- *Data-dependent failures* depend on *the heterogeneity in coupled cells*
  - Some cells are *strongly coupled* and fail based on the *data content in just one neighbor*
  - Reduce test time by detecting *only one neighbor*

- ***CHALLENGE: Detecting failures with only one neighbor information cannot find all failures***

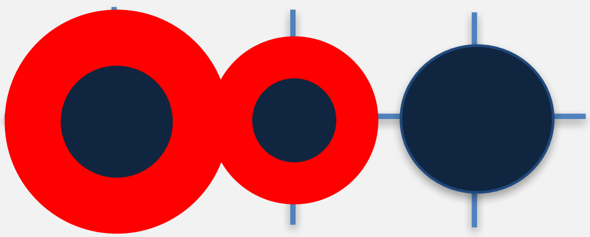
PARBOR: KEY OBSERVATIONS

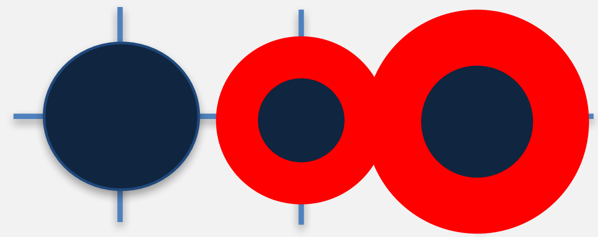
Reduces test time based on two key observations:

Key observation 2:

- *DRAM exhibits regularity and parallelism*
  - Neighbors are located *at the same distance in different rows of DRAM*
  - Detect *all neighbor locations by running parallel tests in multiple rows*

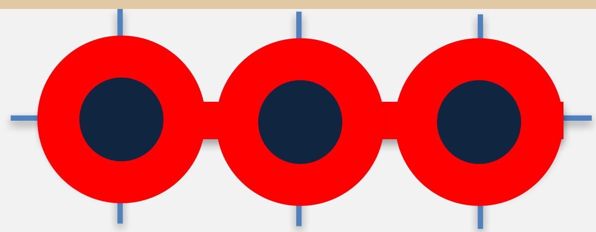
KEY OBSERVATION 1: STRONGLY VS. WEAKLY COUPLED CELLS





STRONGLY COUPLED CELL

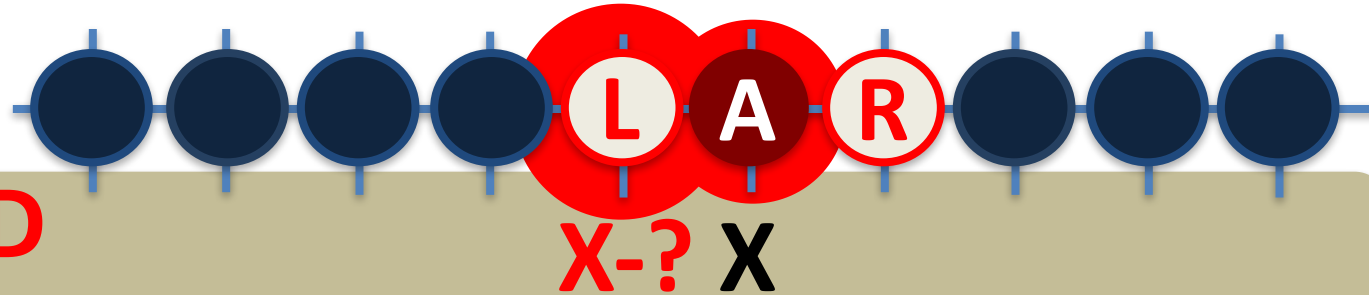
Fails even if only one neighbor's data changes



WEAKLY COUPLED CELL

Fails if both neighbors' data change

KEY IDEA 1: EXPLOITING STRONGLY COUPLED CELLS



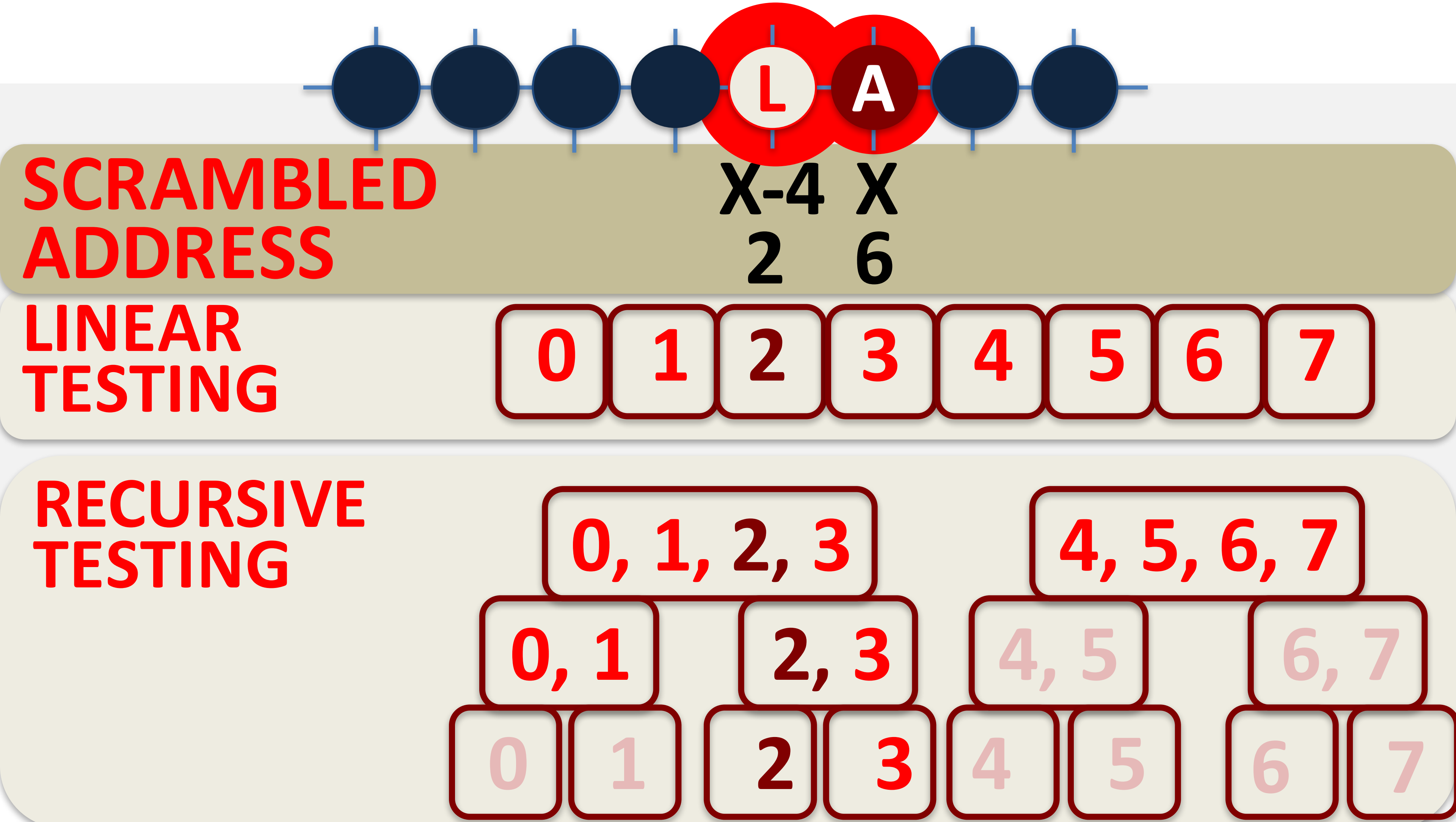
SCRAMBLED ADDRESS

- Instead of detecting both neighbors, ***reduce test time*** by detecting ***only one neighbor location in strongly coupled cells***
  - Does not need to detect ***every two bit addresses***
  - ***Linearly tests every bit address***
  - 0, 1, … , X, X+1, X+2, … n

ADVANTAGES

- **Reduces test time to linear O(n)**
- **Can reduce test time further by applying recursive tests to linear tests**

RECURSIVE TEST



Recursive test reduces test time compared to linear testing

CHALLENGE:

Detecting failures with only one neighbor information cannot find *all* data-dependent failures

PARBOR: KEY OBSERVATIONS

Reduces test time based on two key observations:

Key observation 1:

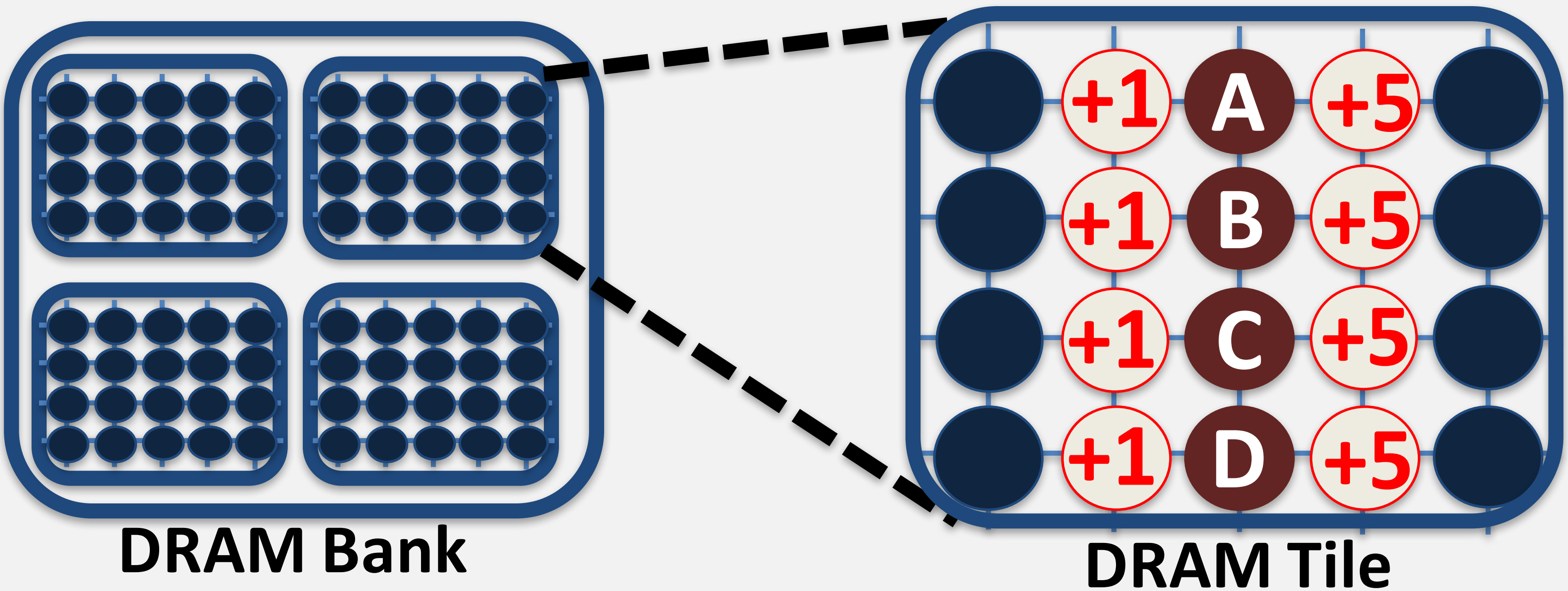
- *Data-dependent failures depend on the heterogeneity in coupled cells*
  - Some cells are *strongly coupled* and fail based on the *data content in just one neighbor*
  - Reduce test time by detecting *only one neighbor*

Key observation 2:

- *DRAM exhibits regularity and parallelism*
  - Neighbors are located *at the same distance in different rows of DRAM*
  - Detect *all neighbor locations by running parallel tests in multiple rows*

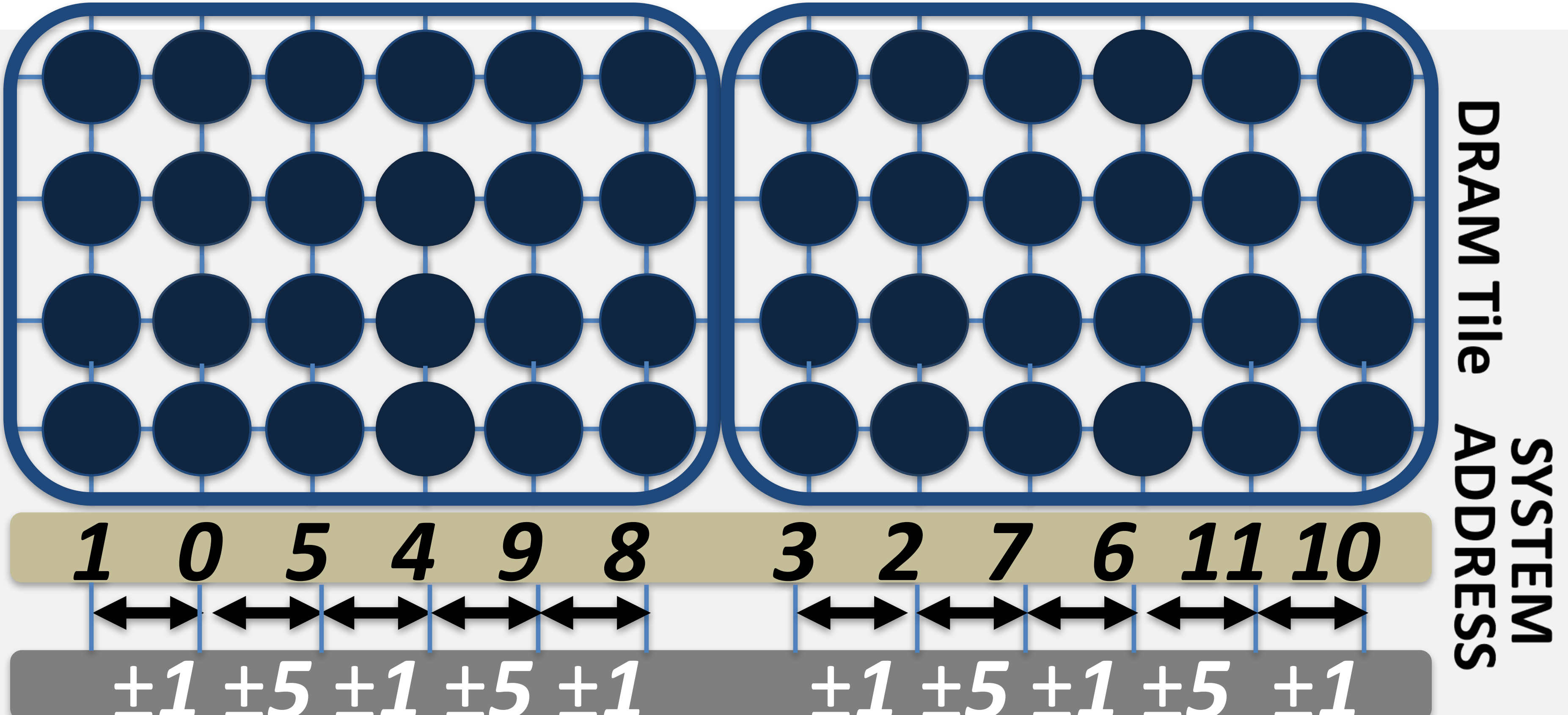
KEY OBSERVATION 2: REGULARITY AND PARALLELISM IN DRAM

- DRAM is internally organized as a *2D array of similar and repetitive tiles.*



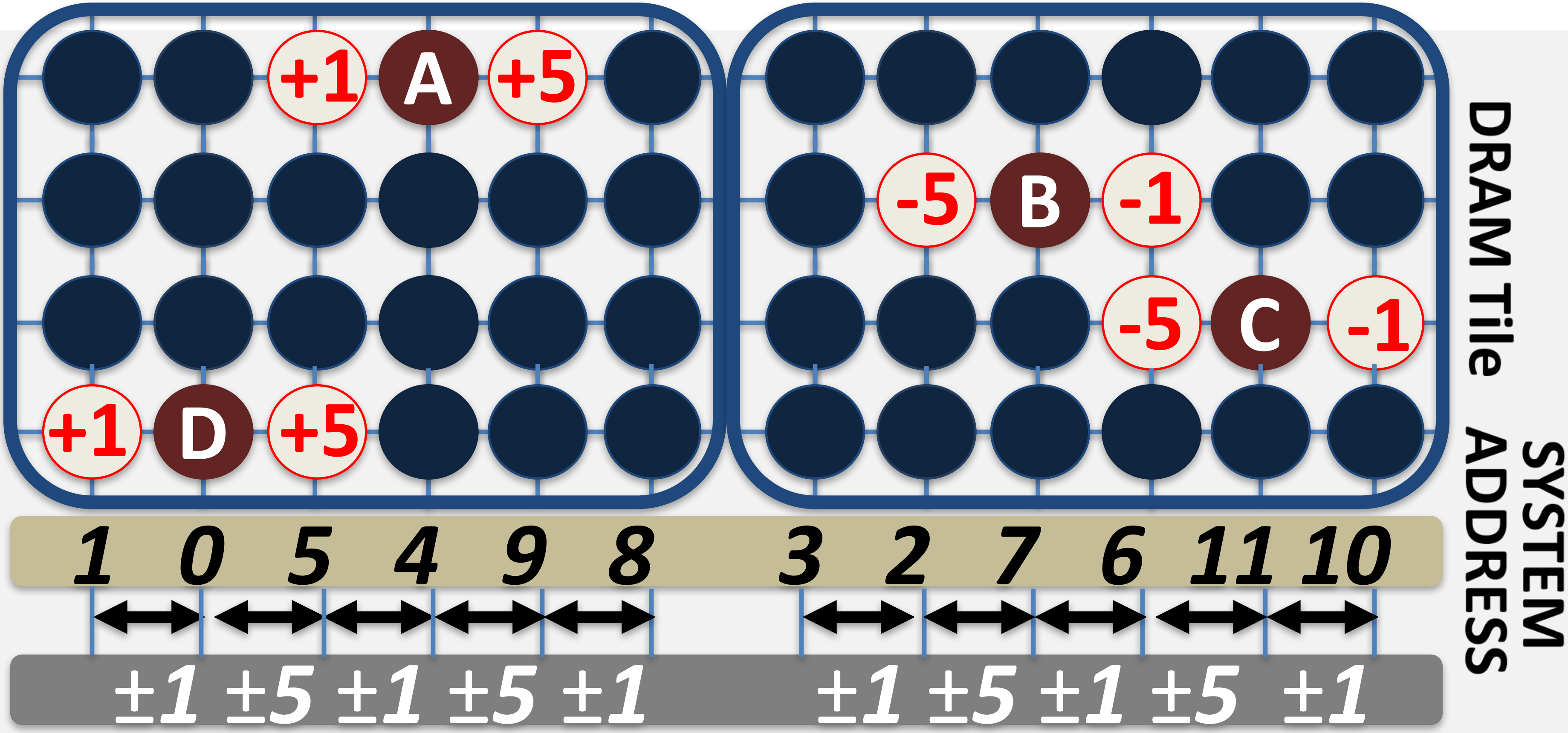
This regularity results in regularity in address mapping

KEY OBSERVATION 2: REGULARITY AND PARALLELISM IN DRAM



Due to regularity in tiles, neighbors can occur only in fixed distances

KEY OBSERVATION 2: REGULARITY AND PARALLELISM IN DRAM



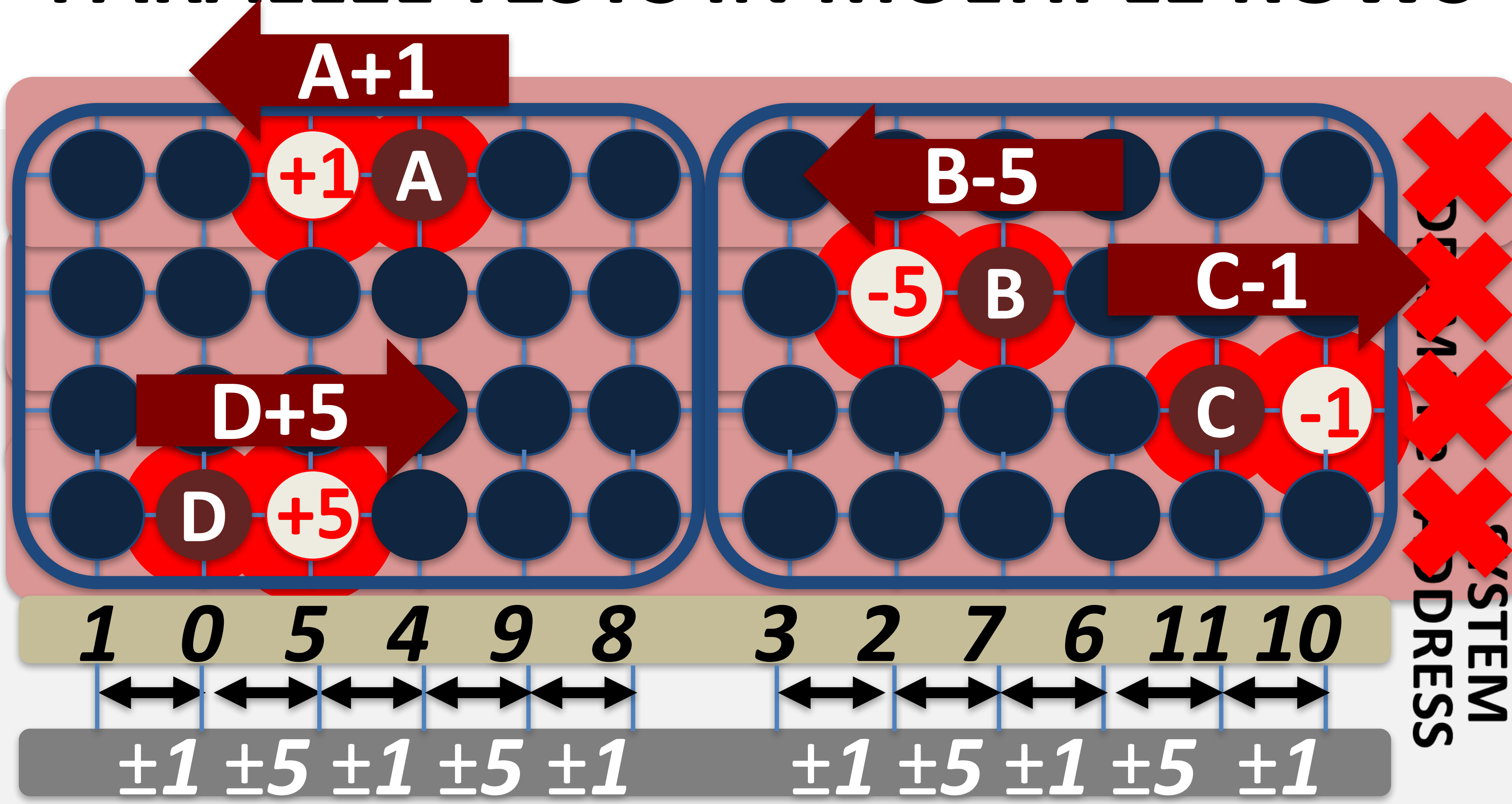
A, B, C, D provide all neighbor distances {+1, -5, +5, -1}

KEY IDEA 2: PARALLEL TESTS IN MULTIPLE ROWS

- ***Due to regularity in mapping,*** *it is possible to determine the neighbor locations* ***from different rows***
- *Run* ***parallel tests in multiple rows***
- *Detect the* ***neighbors' distances in these rows***
- ***Aggregate the locations from different rows***

Provides the neighbor distances for all cells

KEY IDEA 2: PARALLEL TESTS IN MULTIPLE ROWS



Aggregated neighbor locations {*+1, -5, +5, -1*}

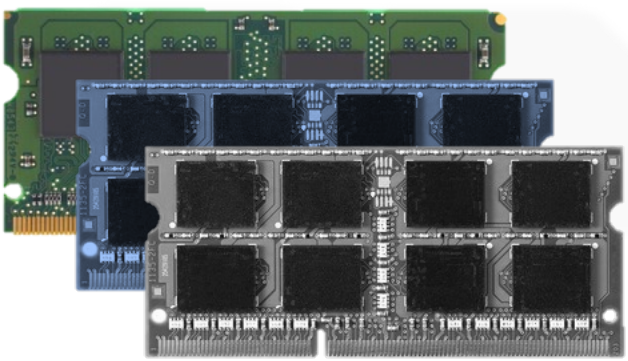
OUTLINE

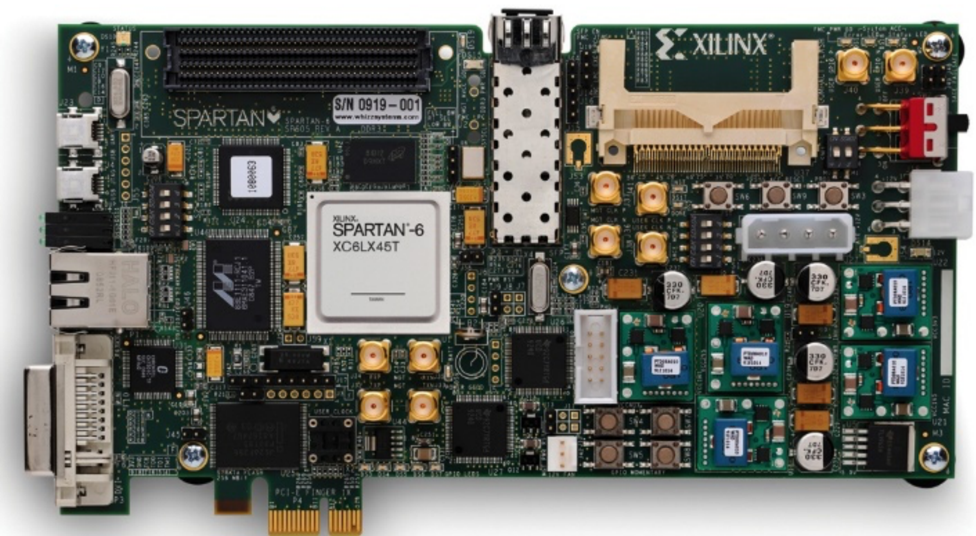
- Data-Dependent Failures
- Challenges in System-Level Detection
- Our Mechanism: PARBOR
- **Experimental Results from Real Chips**
- Use Cases

METHODOLOGY

An FPGA-based testing infrastructure

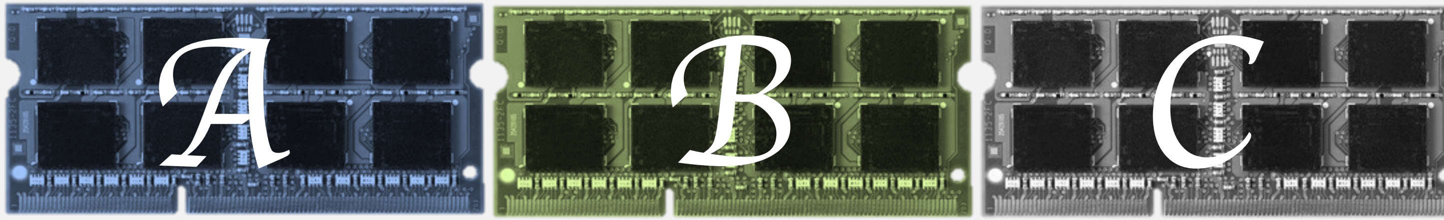
[ISCA'13, SIGMETRICS'14, ISCA'14, HPCA'15, DSN'15, SIGMETRICS'16]





Evaluated 144 chips from three major vendors

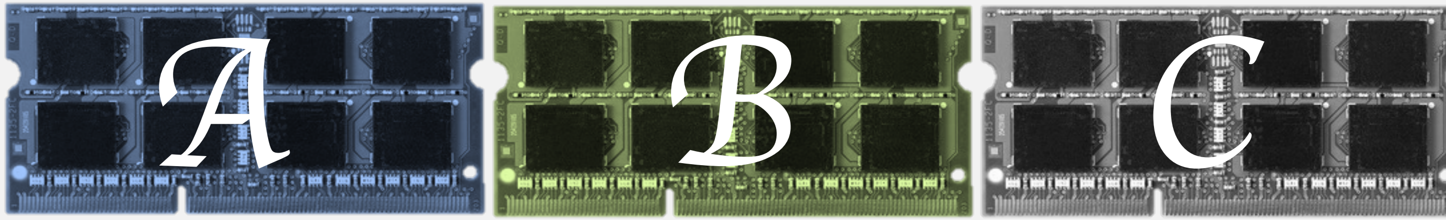
PARBOR: TEST CHARACTERISTICS



NUM TEST REDUCED	745654X	1016800X	745654X

Can detect neighbor locations in 66-90 tests

PARBOR: TEST CHARACTERISTICS



	A	B	C
NUM TEST REDUCED	745654X	1016800X	745654X
NEIGHBOR LOCATIONS	±8, ±16, ±48	±1, ±64	±16, ±33, ±49

Can detect different address mapping in different chips

OUTLINE

- Data-Dependent Failures
- Challenges in System-Level Detection
- Our Mechanism: PARBOR
- Experimental Results from Real Chips
- **Use Cases**

USE CASES

USE CASE: PHYSICAL NEIGHBOR AWARE TEST

- *Use neighbor information to efficiently detect all data-dependent failures*

USE CASE: DATA-CONTENT BASED REFRESH

- *Use neighbor information and program content to reduce refresh count*

USE CASE: PHYSICAL NEIGHBOR-AWARE TEST

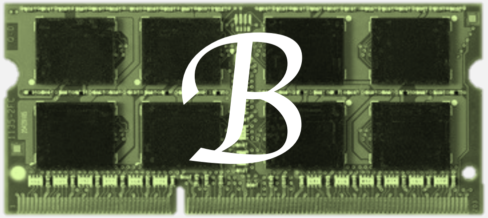
- *Use neighbor information to efficiently detect all data-dependent failures*

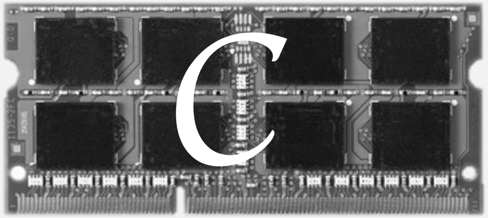
- *Use PARBOR to detect neighbor locations*
  - *Neighbor locations at* ***{±1 ±5}***

- *Can test every 11 bits in parallel*
  - *Reduces test time, needs only 11 tests*

- *At each test, write data pattern at the neighboring cells of each address*
  - *X-5, X+1, X, X-1, X+5 --> 0, 0, 1, 0, 0*

USE CASE:
PHYSICAL NEIGHBOR-AWARE TEST





NUM TESTS	32	32	16
EXTRA FAILURES DETECTED	42%	7%	18%

Detects more failures with small number of tests leveraging neighboring information

USE CASES

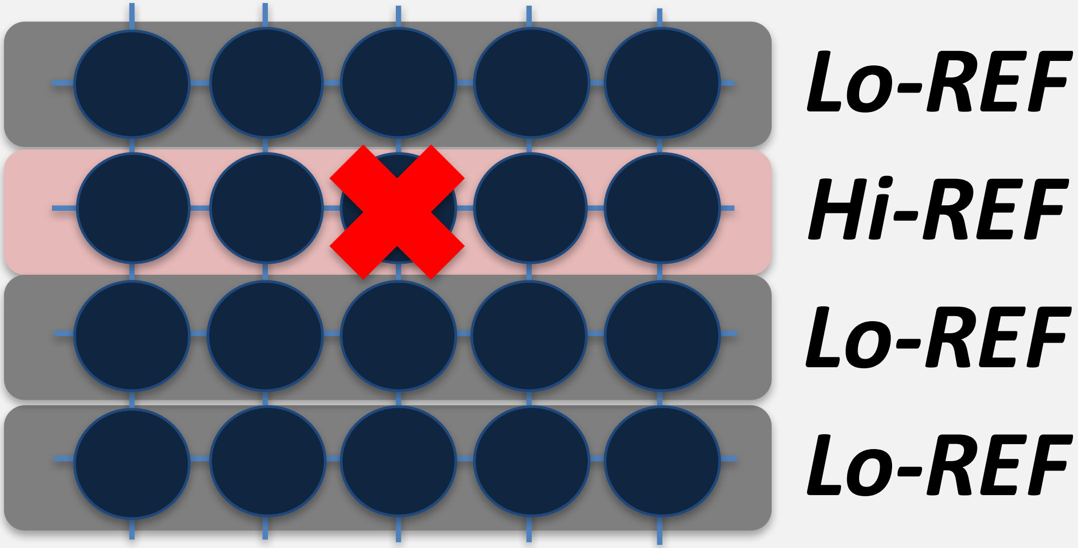
USE CASE: PHYSICAL NEIGHBOR AWARE TEST

- *Use neighbor information to efficiently detect all data-dependent failures*

USE CASE: DATA-CONTENT BASED REFRESH

- *Use neighbor information and program content to reduce refresh count*

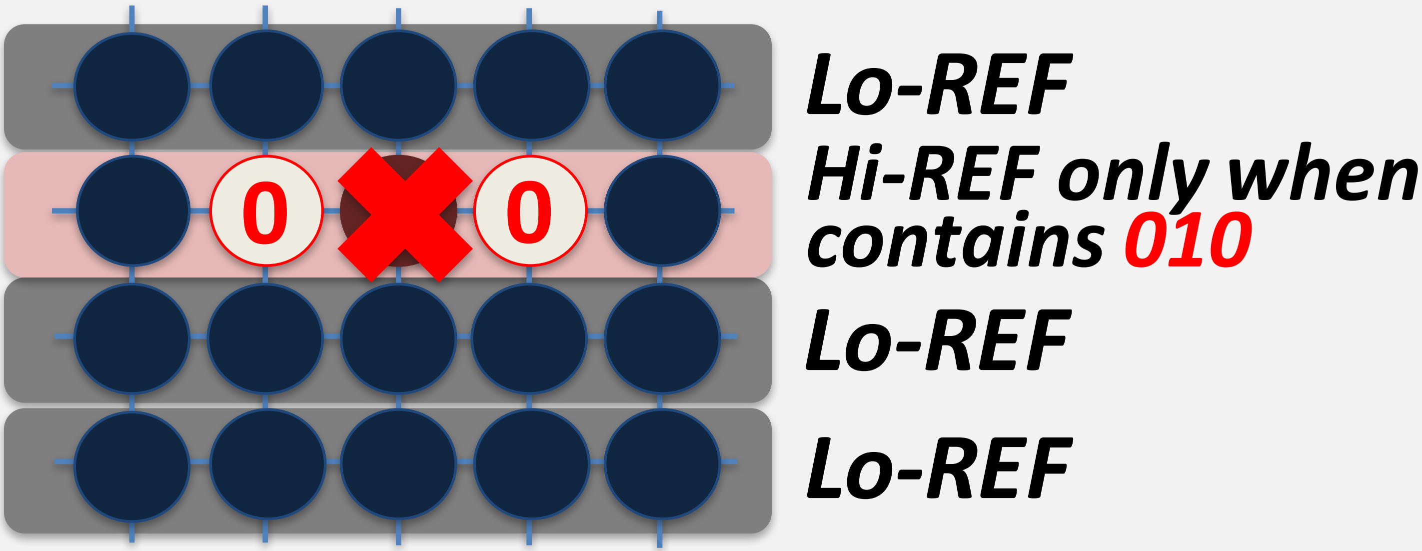
PROBLEM WITH TRADITIONAL REFRESH OPTIMIZATION



- ***Traditional refresh optimization:*** *[RAIDR ISCA'12]*
  - **High refresh rate** with rows with ***failures***
  - ***Low refresh rate*** for rows with ***no failure***

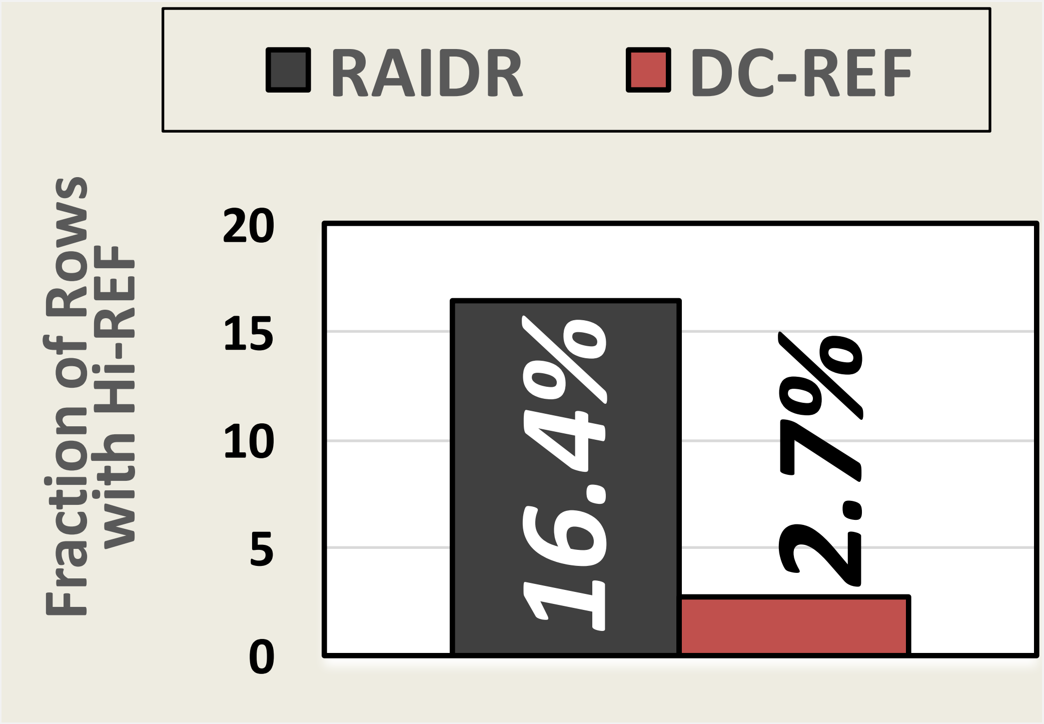
Does not take into account that failures occur only with specific content

A NEW USE CASE: DATA-CONTENT AWARE REFRESH



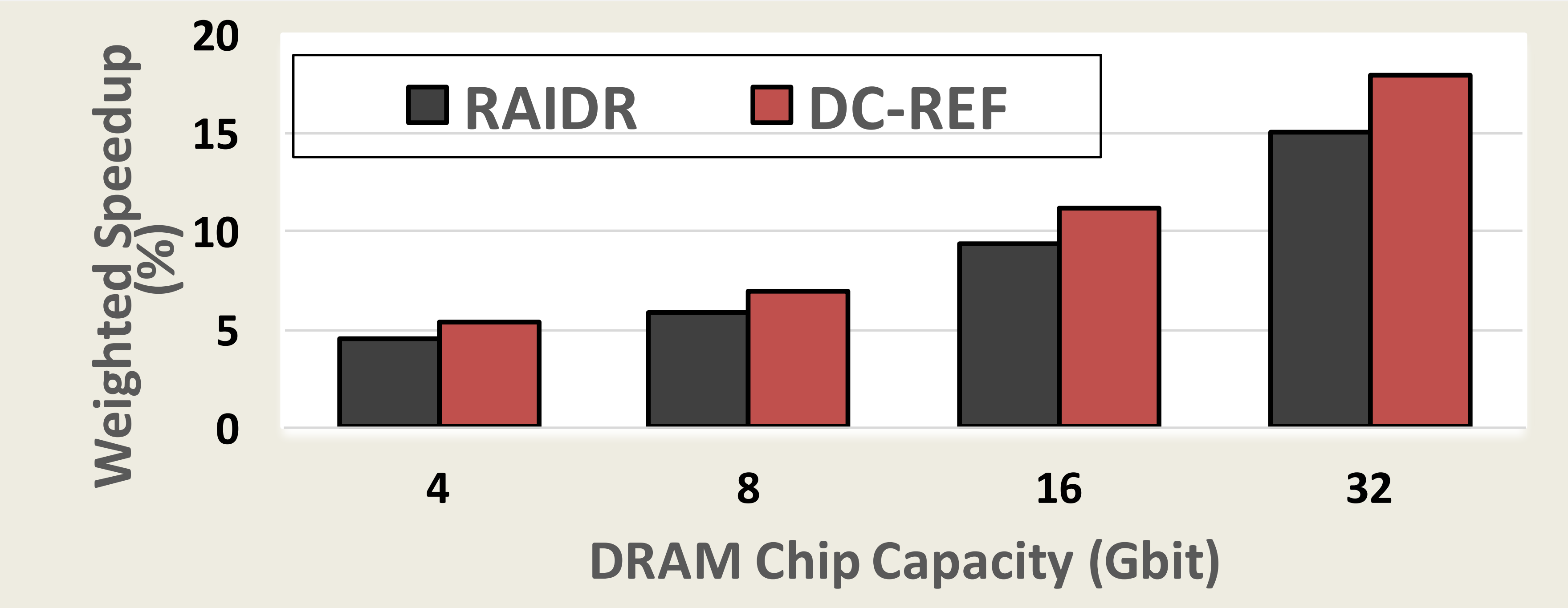
- ***DC-REF optimization:***
  - Builds on top of ***PARBOR*** *to track* ***locations of data-dependent failures*** *and* ***data patterns*** that cause the failures
  - **High refresh rate** for rows whose data content exhibits ***failures***
  - ***Low refresh rate*** for rows with ***no failure***

DATA-CONTENT AWARE REFRESH: Fraction of Rows with High Refresh Rate



DC-REF significantly reduces the number of high refresh operations

DATA-CONTENT AWARE REFRESH: PREFORMANCE IMPACT



DC-REF improves performance by reducing refresh operations

PARBOR: Summary

A new technique to determine the locations of neighboring DRAM cells

- **Exploits** *heterogeneity in data-dependent cells* to reduce test time by detecting only one neighbor

- **Exploits** *DRAM regularity and parallelism* to *aggregate neighbor locations from multiple rows to identify all neighbor locations*

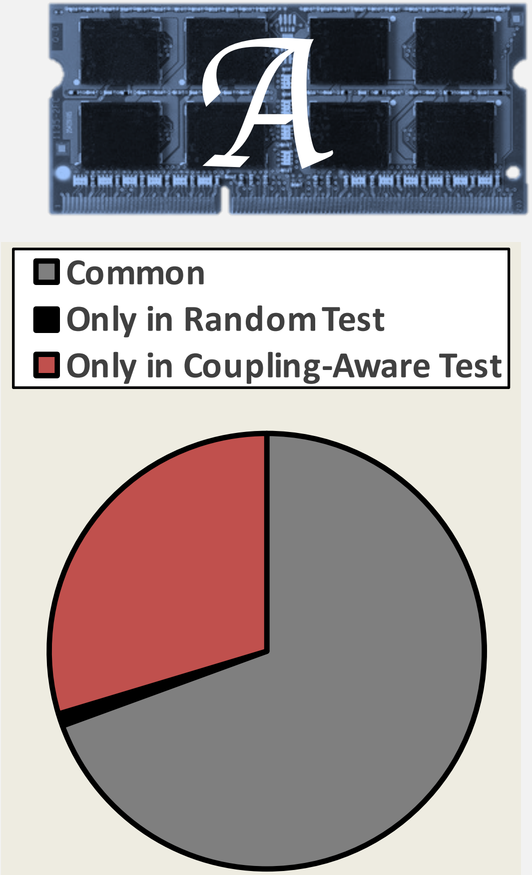
- **Enables** *new uses cases to* improve performance, reliability, and energy efficiency
  - *Physical neighbor-aware test*
  - *Data-content aware refresh*

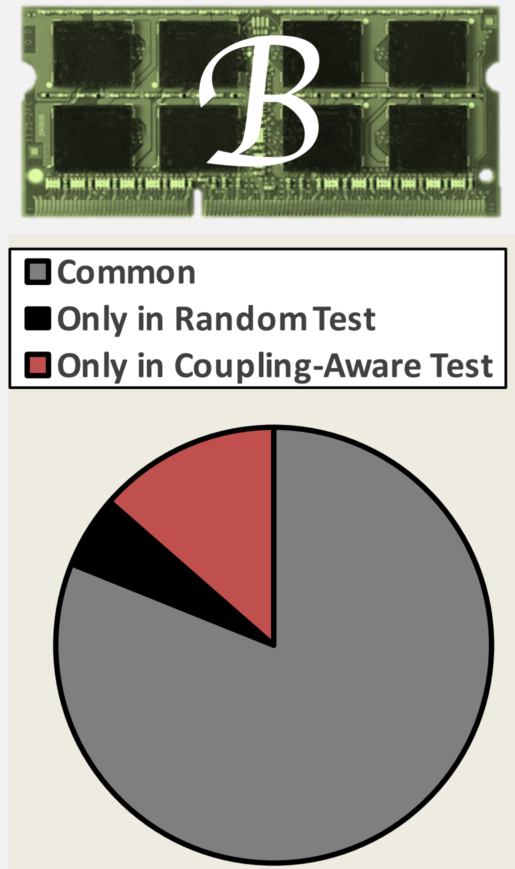
PARBOR

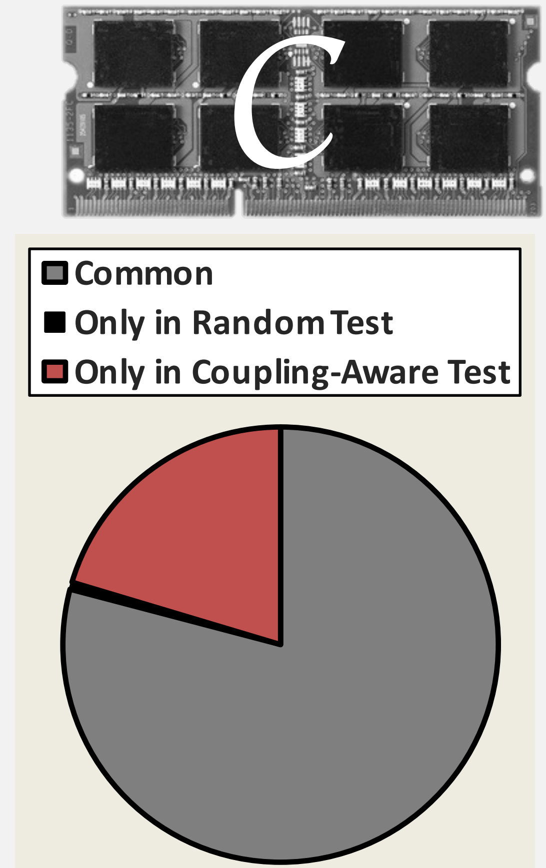
AN EFFICIENT SYSTEM-LEVEL TECHNIQUE TO DETECT DATA-DEPENDENT FAILURES IN DRAM

Samira Khan
Donghyuk Lee
Onur Mutlu

USE CASE: PHYSICAL NEIGHBOR-AWARE TEST







A significant fraction of failures can be detected only by PARBOR (20-30%)