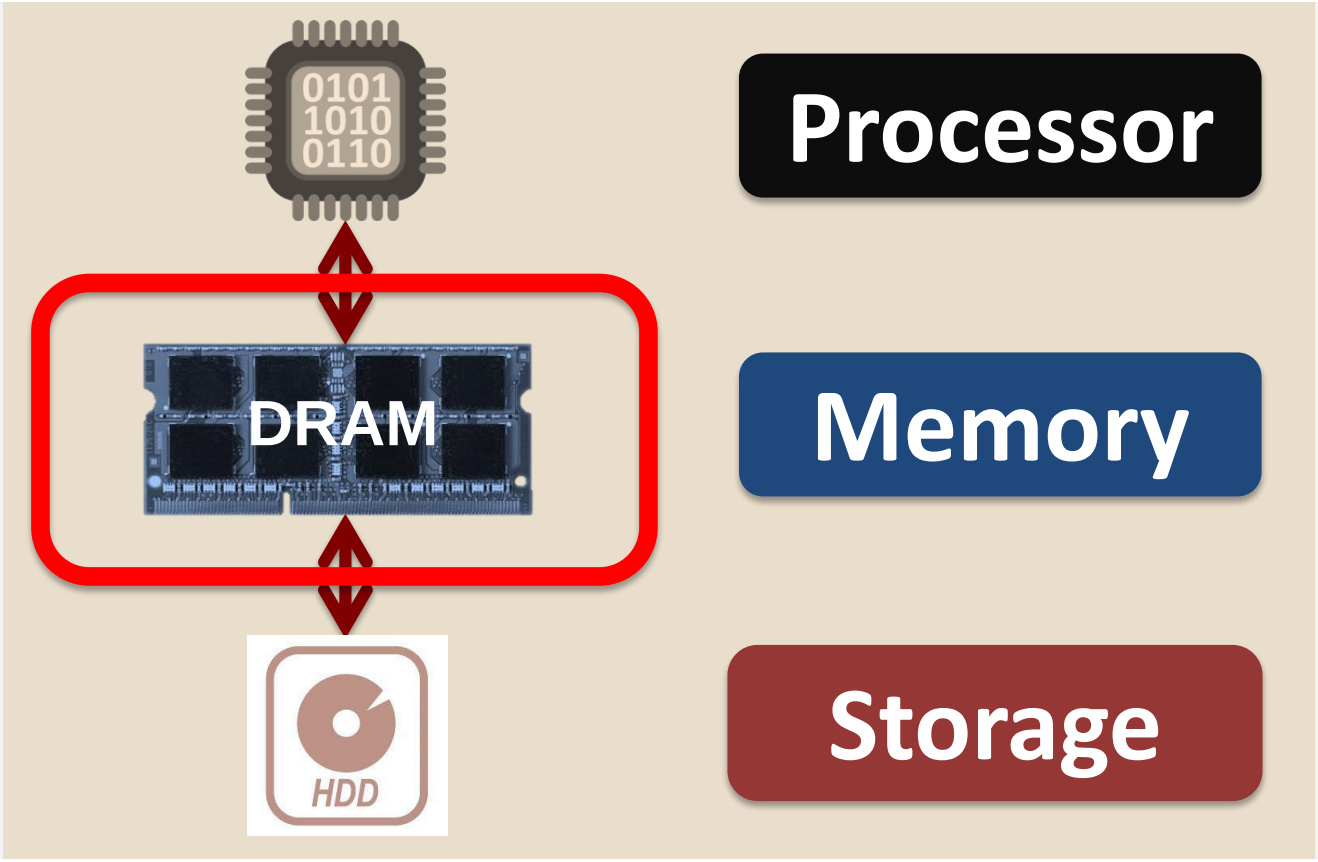
SOLVING THE DRAM SCALING CHALLENGE:

RETHINKING THE INTERFACE BETWEEN CIRCUITS, ARCHITECTURE, AND SYSTEMS

Samira Khan

Carnegie Mellon

MEMORY IN TODAY'S SYSTEM



DRAM is critical for performance

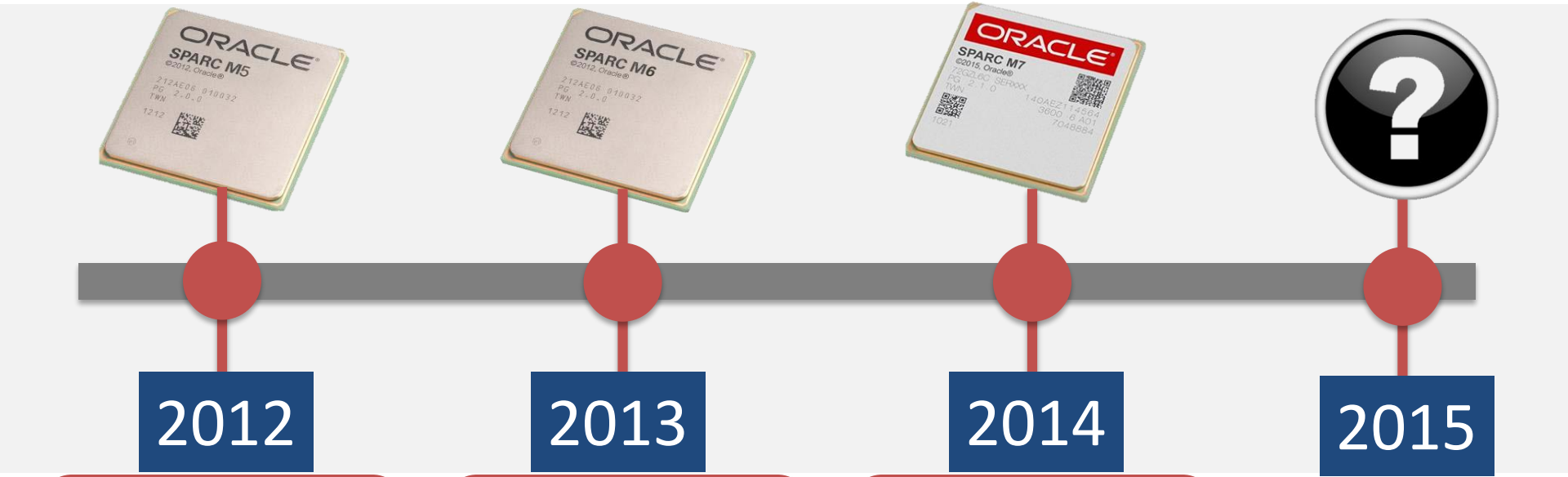
MAIN MEMORY CAPACITY

Gigabytes of DRAM

Increasing demand for high capacity

1. More cores
2. Data-intensive applications

DEMAND 1:
INCREASING NUMBER OF CORES



SPARC M5
6 Cores

SPARC M6
12 Cores

SPARC M7
32 Cores

More cores need more memory

DEMAND 2:
DATA-INTENSIVE APPLICATIONS

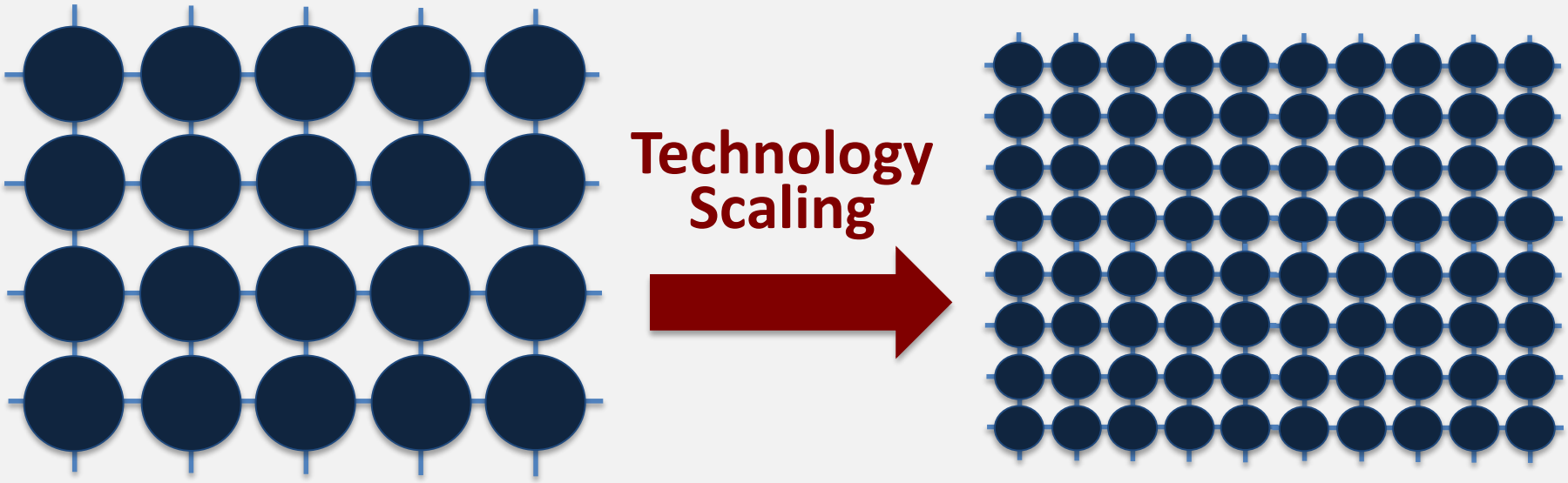
redis

MEMORY CACHING

IN-MEMORY DATABASE

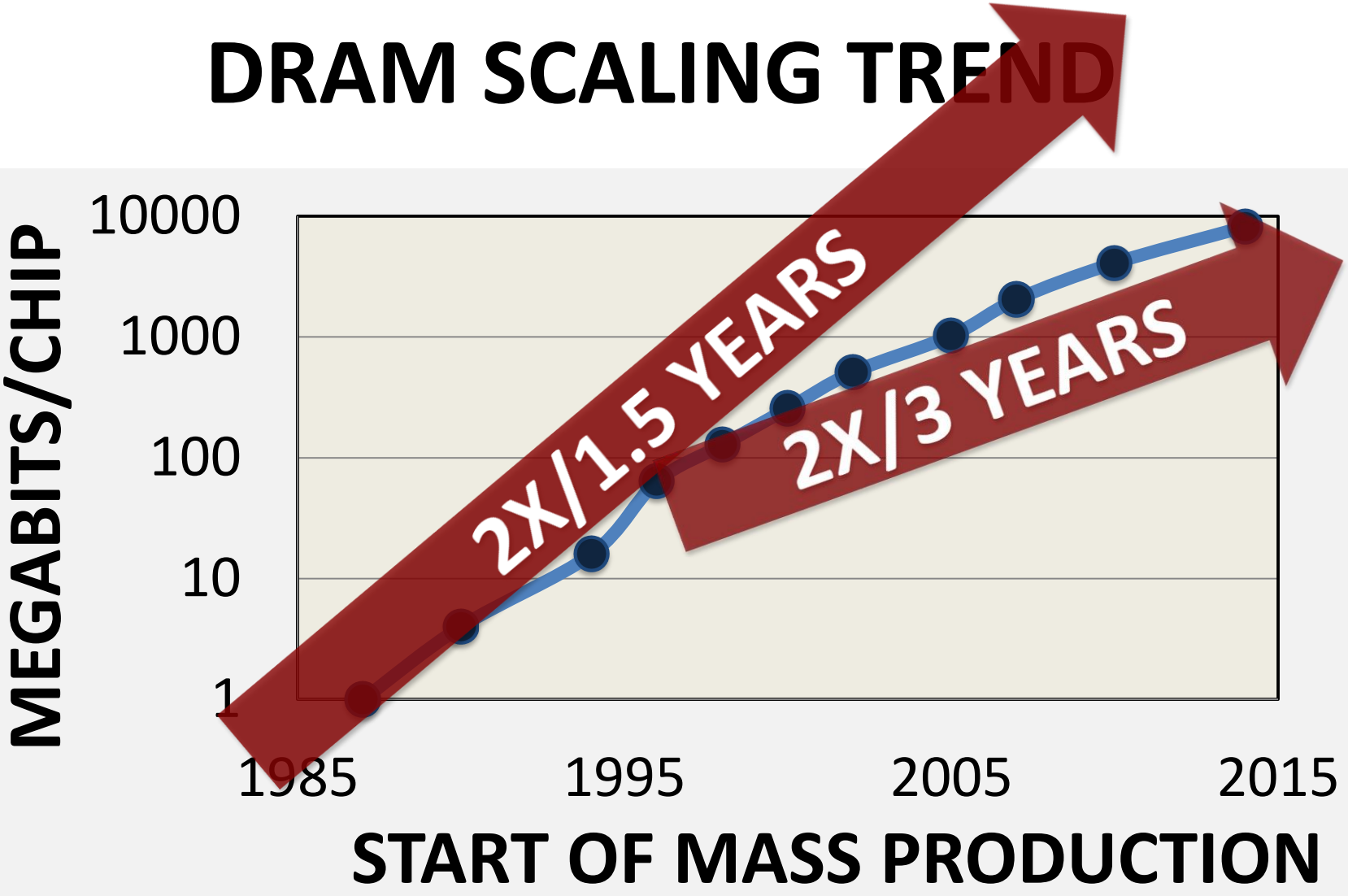
More demand for memory

HOW DID WE GET MORE CAPACITY?



DRAM scaling enabled high capacity

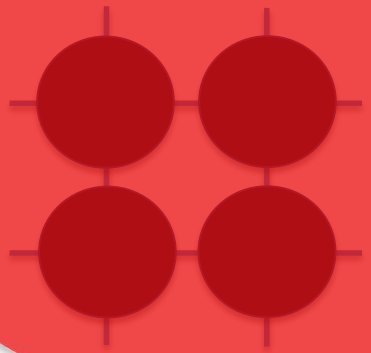
DRAM SCALING TREND



DRAM scaling is getting difficult

Source: Flash Memory Summit 2013, Memcon 2014

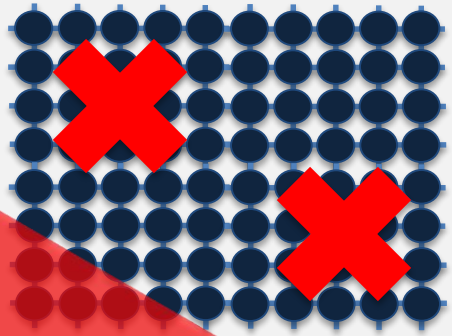
DRAM SCALING CHALLENGE



DRAM Cells

Technology Scaling

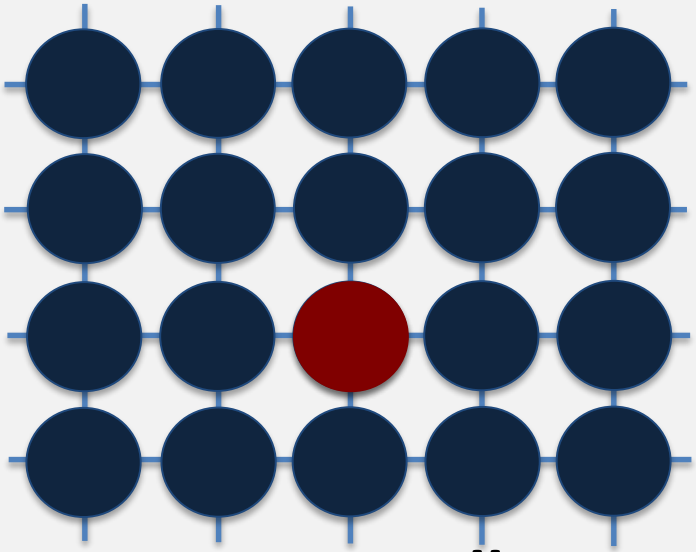
WHY?



DRAM Cells

Manufacturing reliable cells at low cost is getting difficult

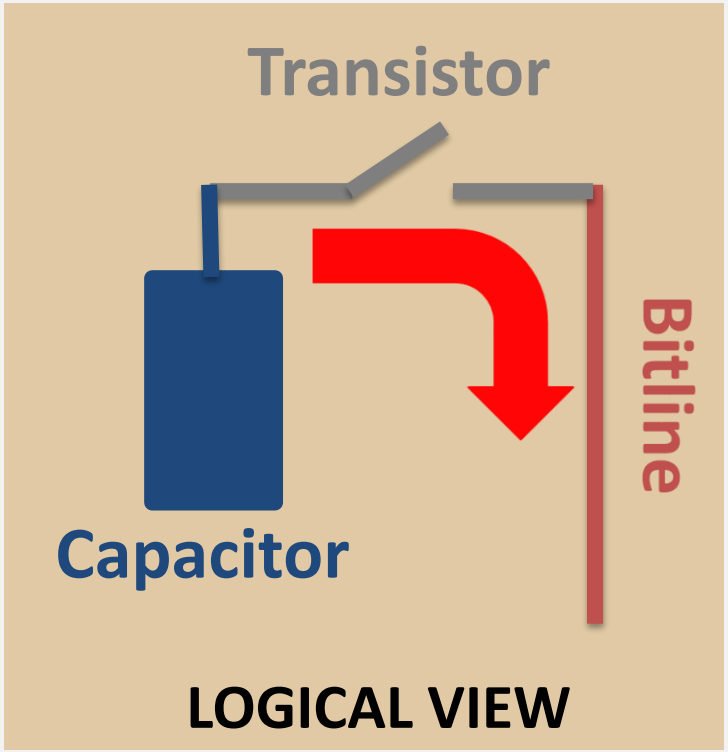
WHY IS IT DIFFICULT TO SCALE?

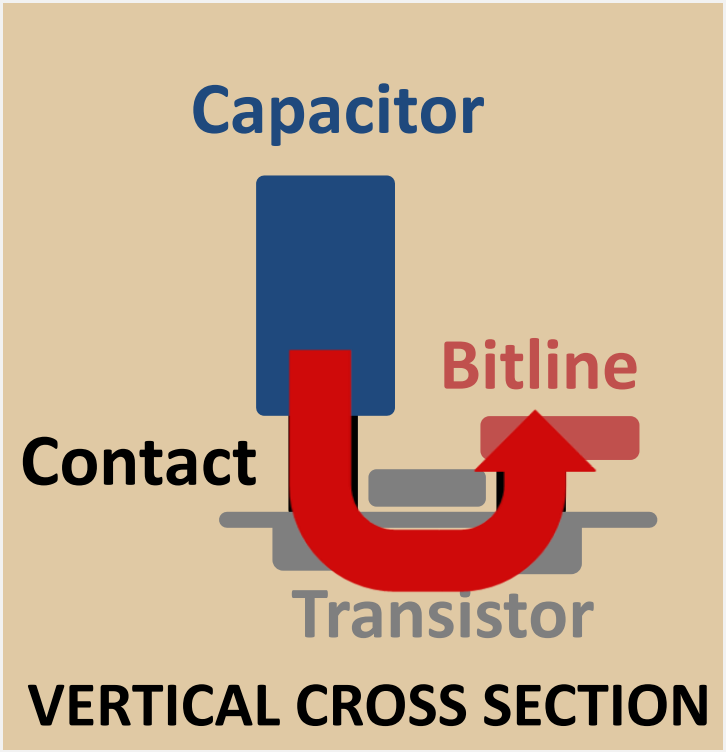


DRAM Cells

In order to answer this we need to take a closer look to a DRAM cell

WHY IS IT DIFFICULT TO SCALE?





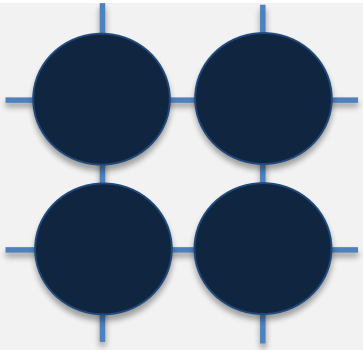
A DRAM cell

WHY IS IT DIFFICULT TO SCALE?

Challenges in Scaling

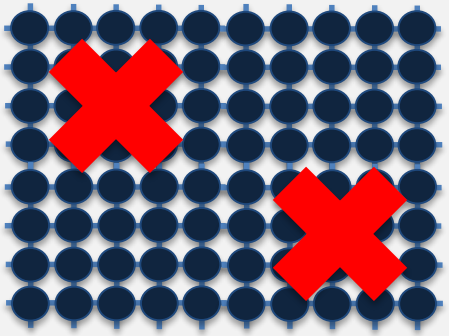
1. Capacitor reliability
2. Cell-to-cell interference

SCALING CHALLENGE 1:
CAPACITOR RELIABILITY

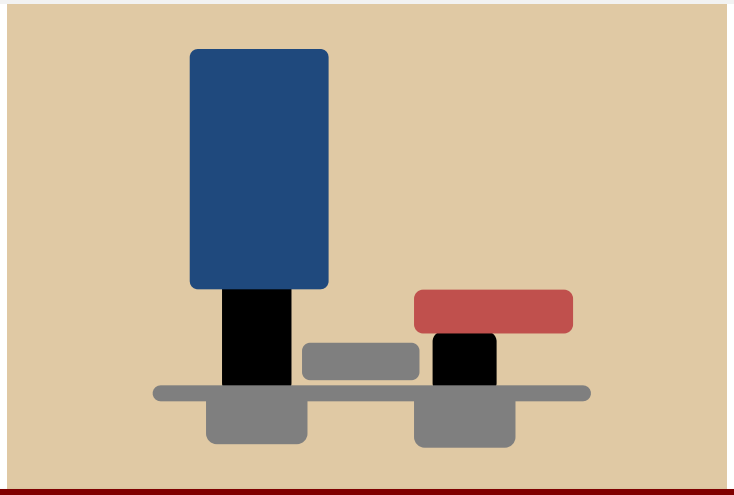


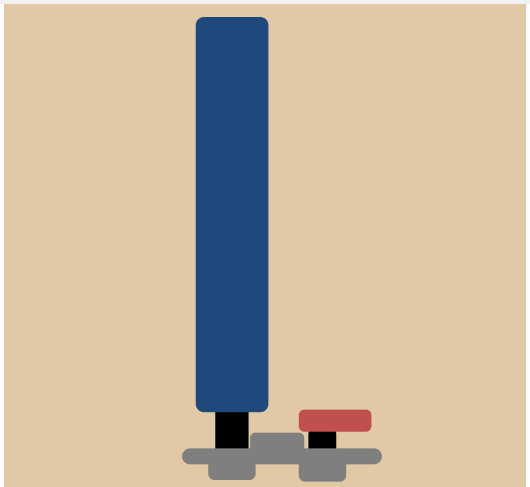
DRAM Cells

Technology Scaling



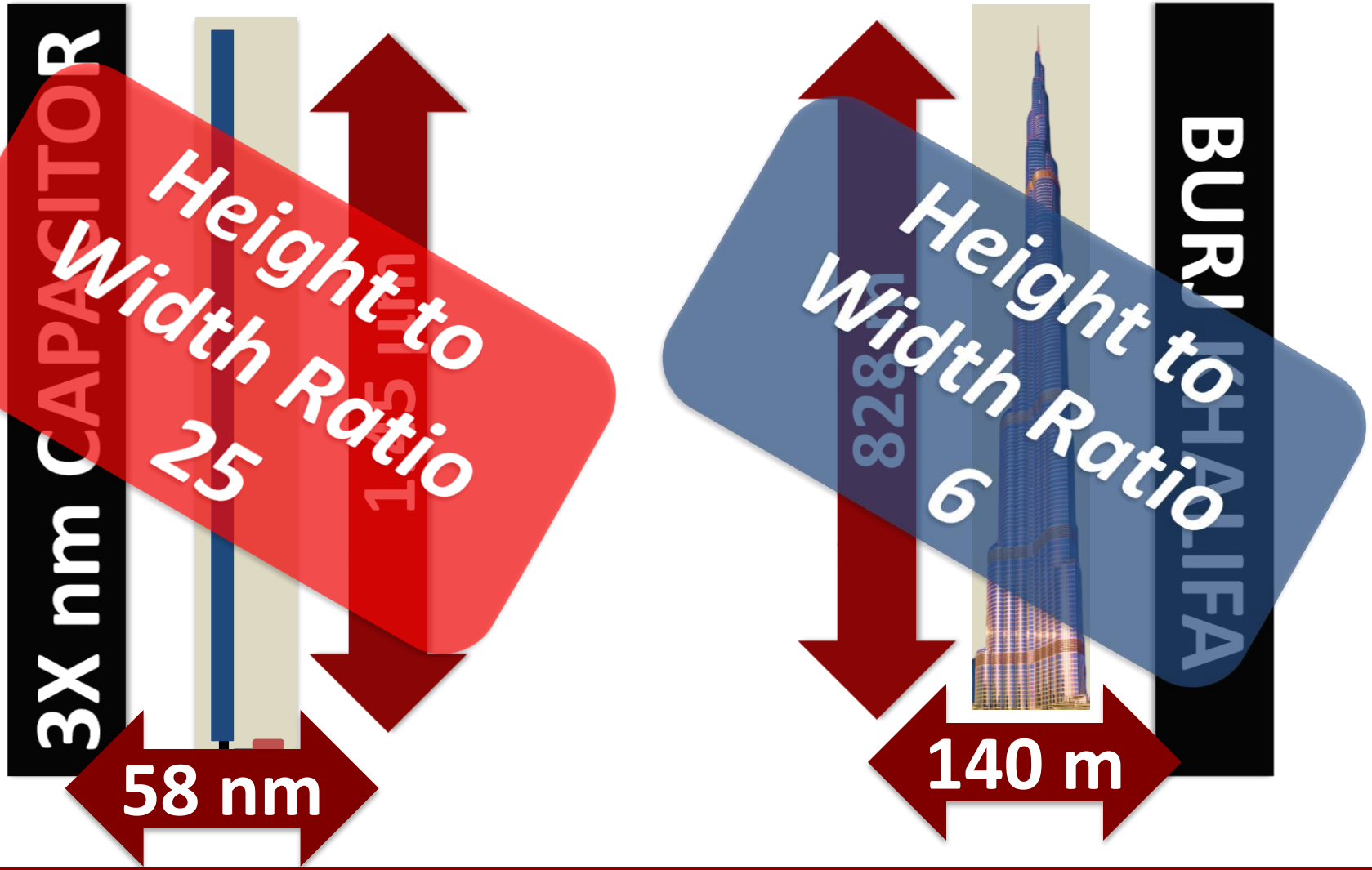
DRAM Cells





Capacitor is getting taller

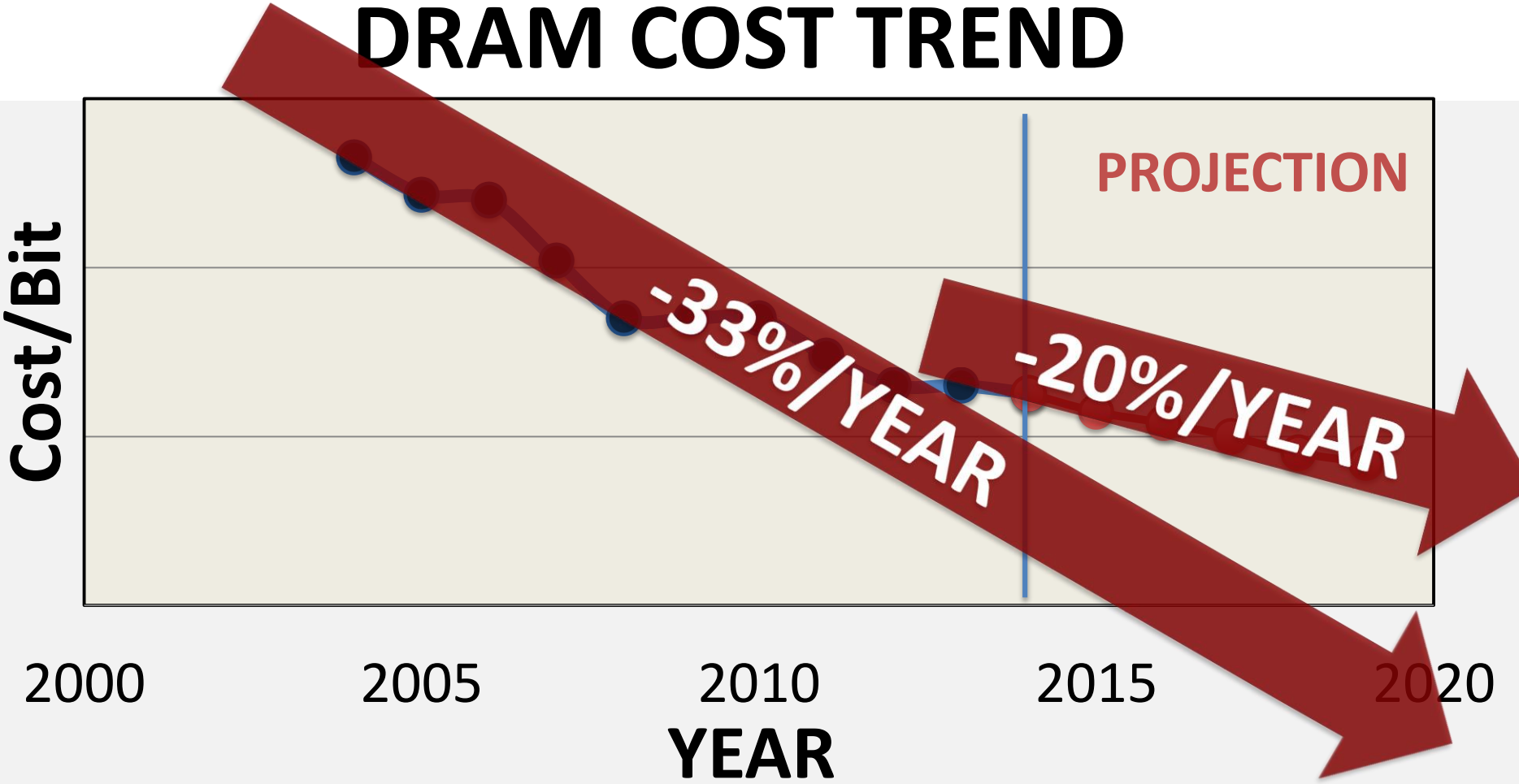
CAPACITOR RELIABILITY



Results in failures while manufacturing

Source: Flash Memory Summit, Hynix 2012

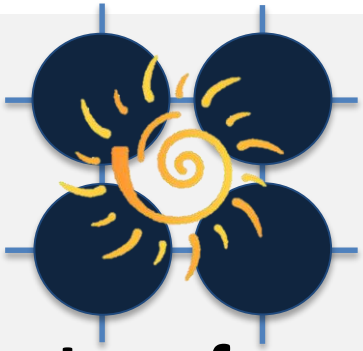
IMPLICATION:
DRAM COST TREND



Cost is expected to go higher

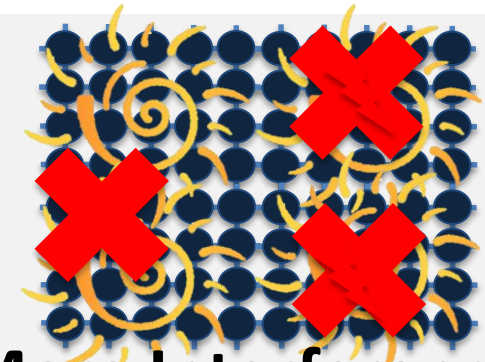
Source: Flash Memory Summit, Hynix 2012

SCALING CHALLENGE 2:
CELL-TO-CELL INTERFERENCE

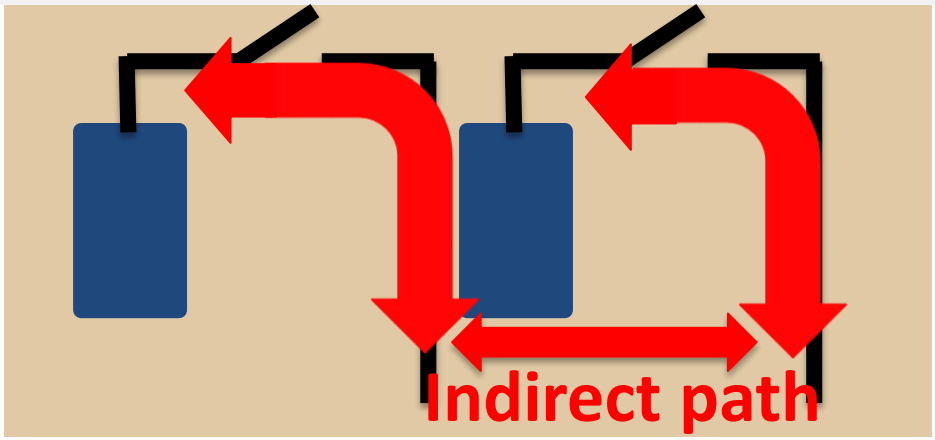


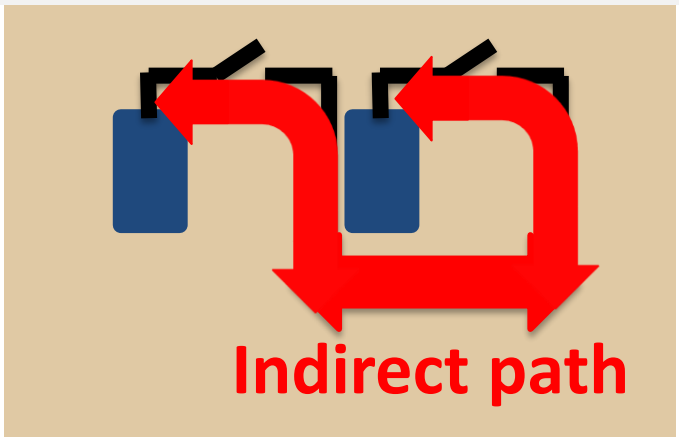
Less Interference

Technology Scaling



More Interference





More interference results in more failures

IMPLICATION:
DRAM ERRORS IN THE FIELD

DRAM Errors in the Wild: A Large-Scale Field Study

1.52% of DRAM modules failed in Google Servers

A Study of DRAM Failures in the Field

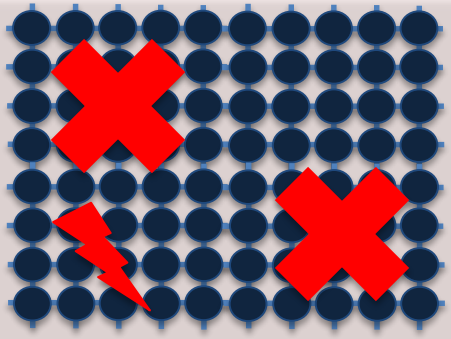
1.6% of DRAM modules failed in LANL

SIGMETRICS'09, SC'12

GOAL

ENABLE
HIGH CAPACITY MEMORY
WITHOUT
SACRIFICING RELIABILITY

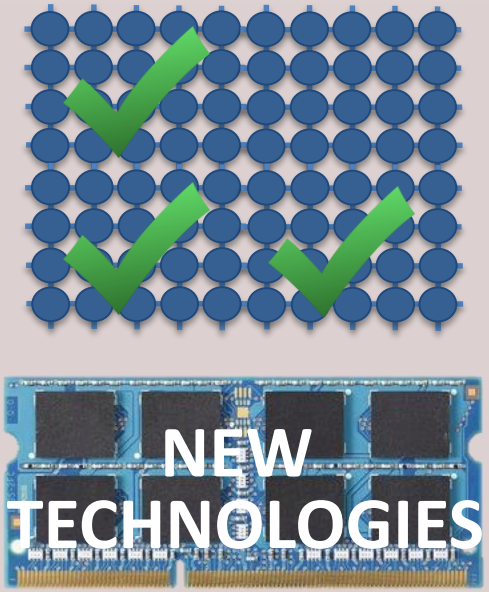
TWO DIRECTIONS



Difficult to scale

MAKE DRAM SCALABLE

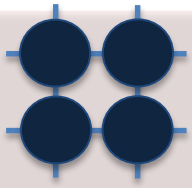
SIGMETRICS'14, DSN'15, ONGOING



Predicted to be highly scalable

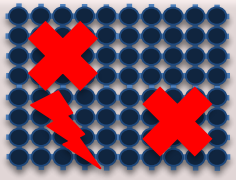
LEVERAGE NEW TECHNOLOGIES

WEED'13, ONGOING



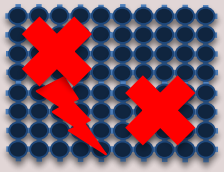
DRAM Cells

Technology Scaling



DRAM Cells

DRAM SCALING CHALLENGE



DRAM Cells

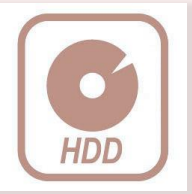
Detect and Mitigate

Reliable System

SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

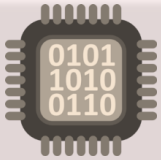
Non-Volatile Memory

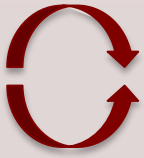
UNIFY

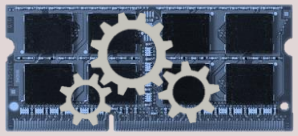


Storage

NON-VOLATILE MEMORIES: UNIFIED MEMORY & STORAGE

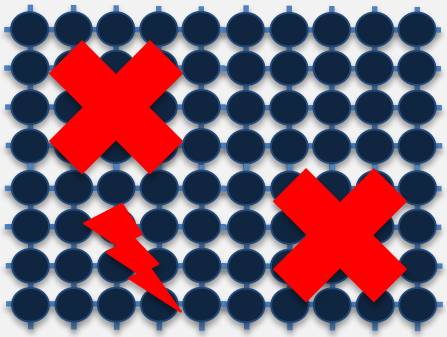






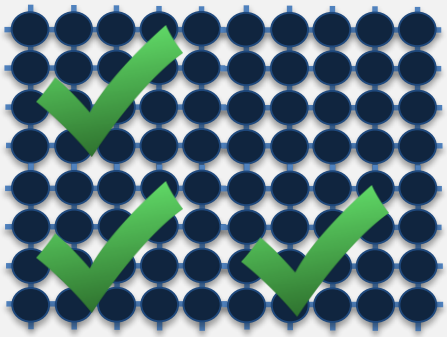
PAST AND FUTURE WORK

TRADITIONAL APPROACH TO ENABLE DRAM SCALING



Unreliable DRAM Cells

Make DRAM Reliable



Reliable DRAM Cells

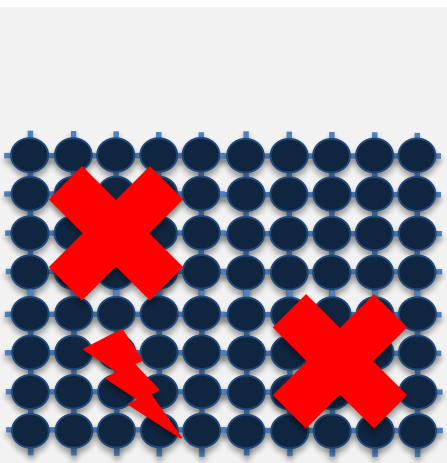
Reliable System

Manufacturing Time

System in the Field

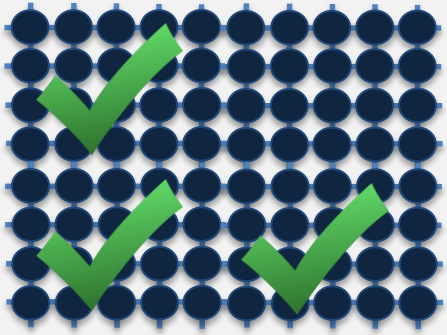
DRAM has strict reliability guarantee

MY APPROACH



Unreliable DRAM Cells

Make DRAM Reliable



Reliable DRAM Cells

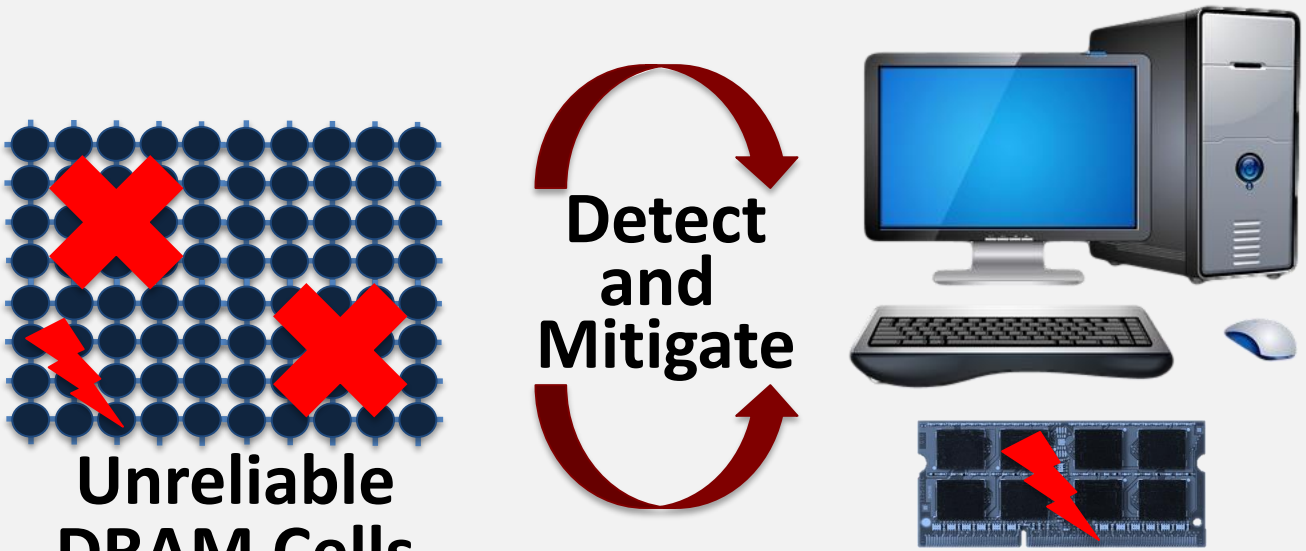
Reliable System

Manufacturing Time

System in the Field

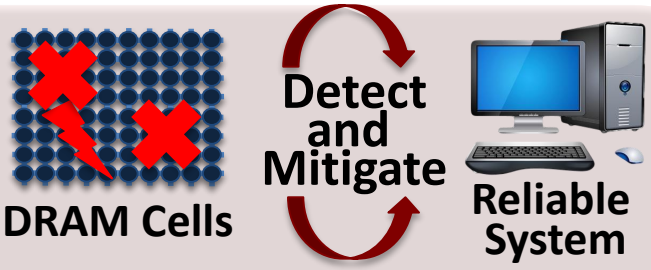
Shift the responsibility to systems

VISION: SYSTEM-LEVEL DETECTION AND MITIGATION



Detect and mitigate errors after the system has become operational

ONLINE PROFILING
Reduces cost, increases yield, and enables scaling



SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

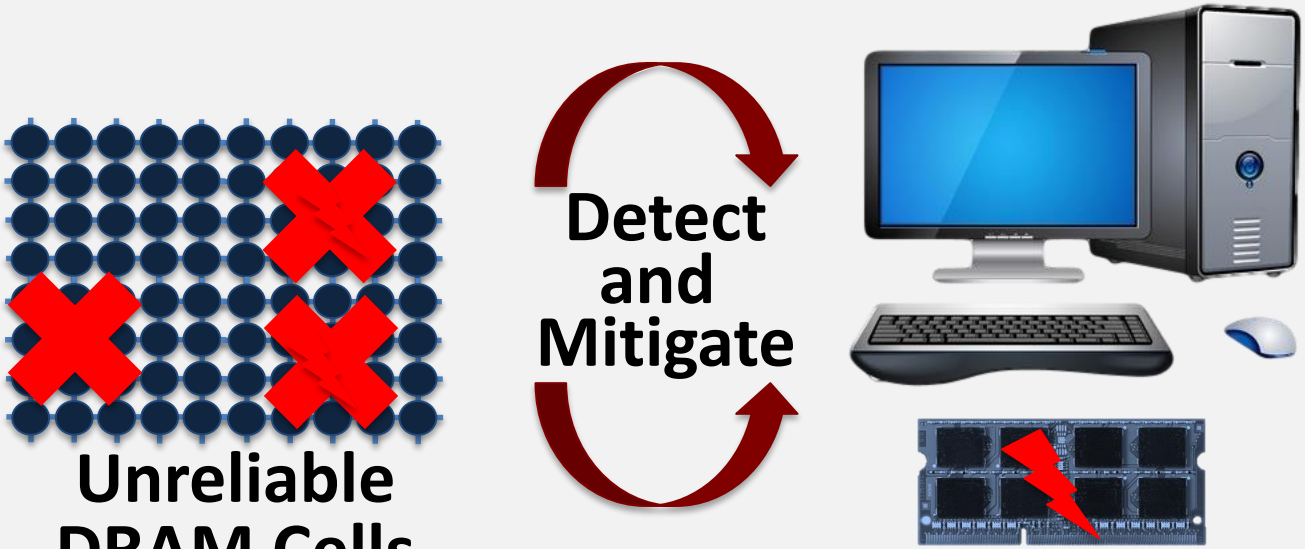
CHALLENGE: INTERMITTENT FAILURES

EFFICACY OF SYSTEM-LEVEL TECHNIQUES WITH REAL DRAM CHIPS

HIGH-LEVEL DESIGN

NEW SYSTEM-LEVEL TECHNIQUES

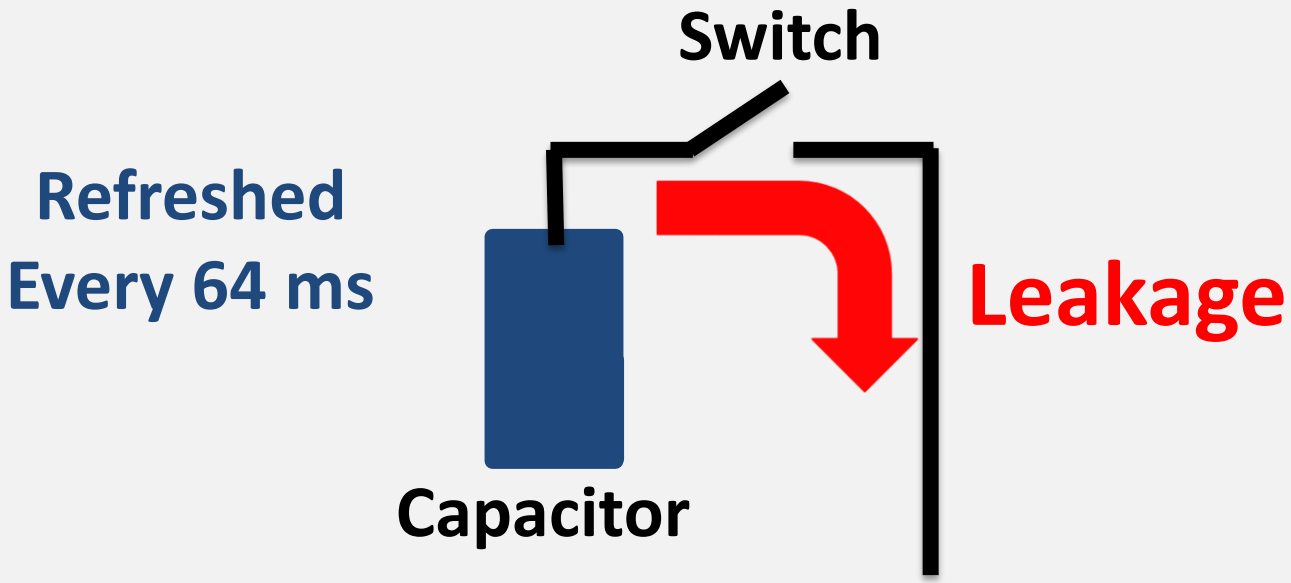
CHALLENGE: INTERMITTENT FAILURES

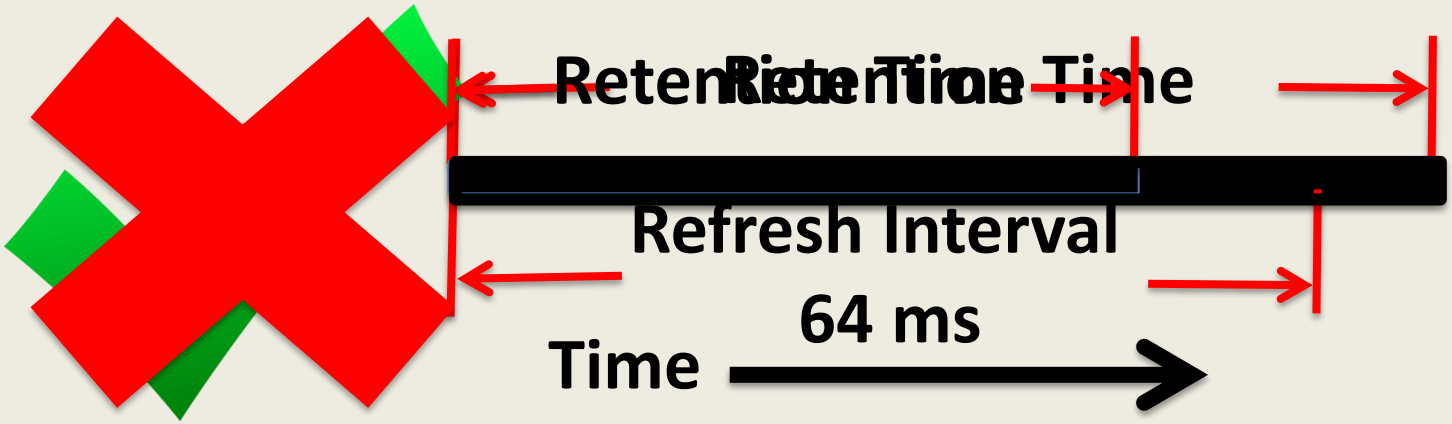


If failures were permanent, a simple boot up test would have worked

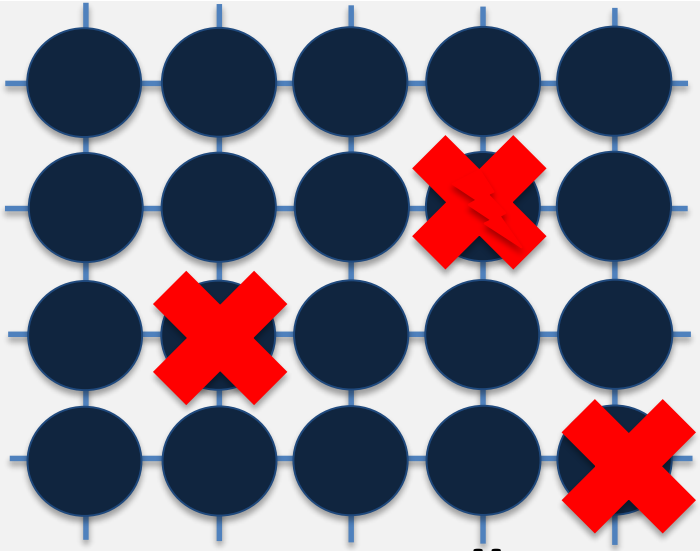
What are the characteristics of intermittent failures?

DRAM RETENTION FAILURE





INTERMITTENT RETENTION FAILURE



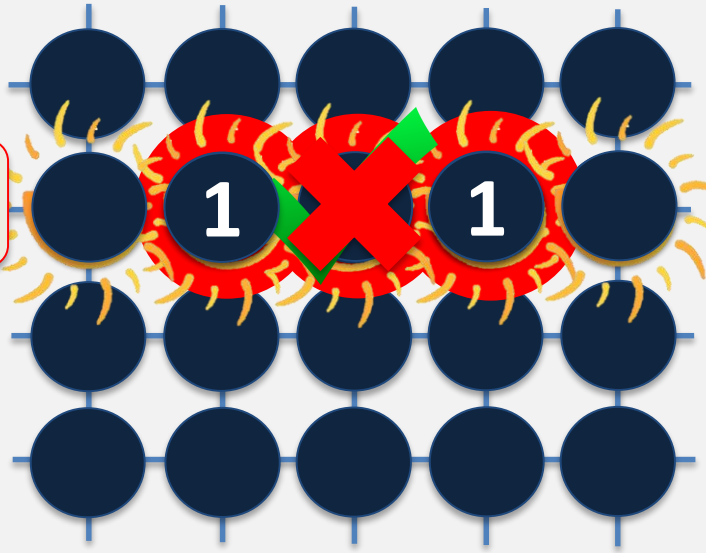
DRAM Cells

- Some retention failures are intermittent
- Two characteristics of intermittent retention failures

1. Data Pattern Sensitivity
2. Variable Retention Time

DATA PATTERN SENSITIVITY

INTERFERENCE

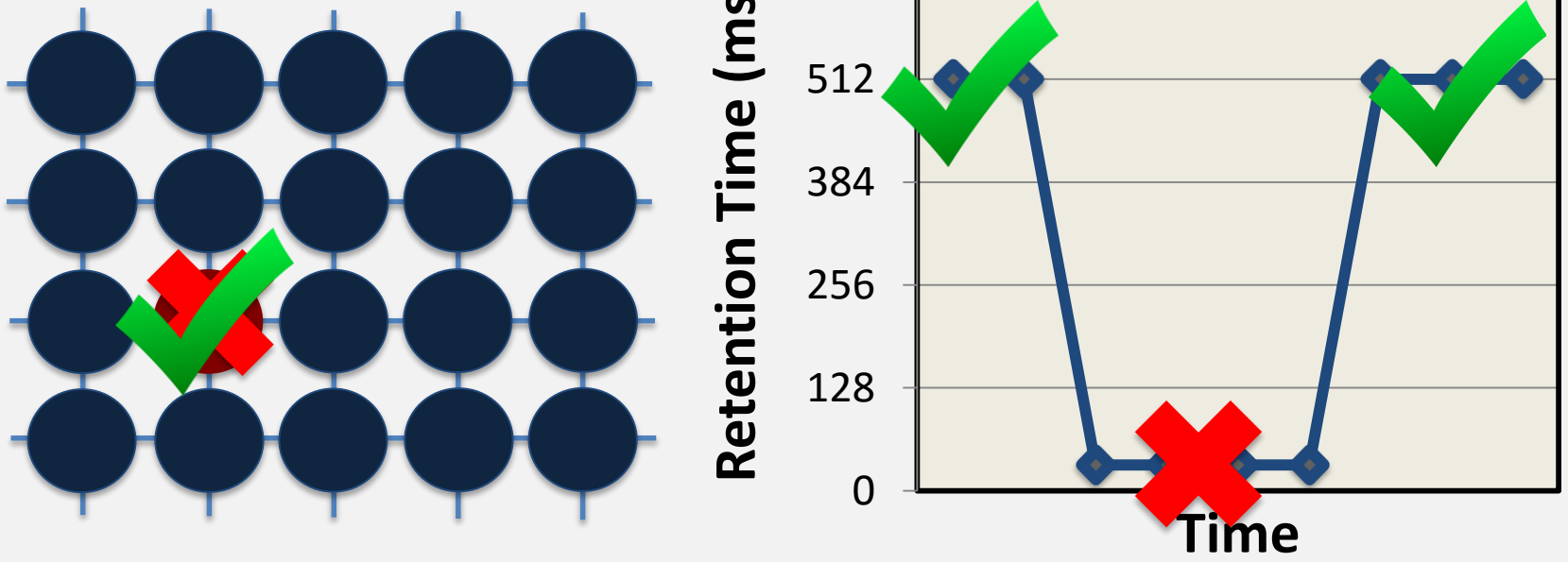


NO FAILURE

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

JSSC'88, MDTD'02

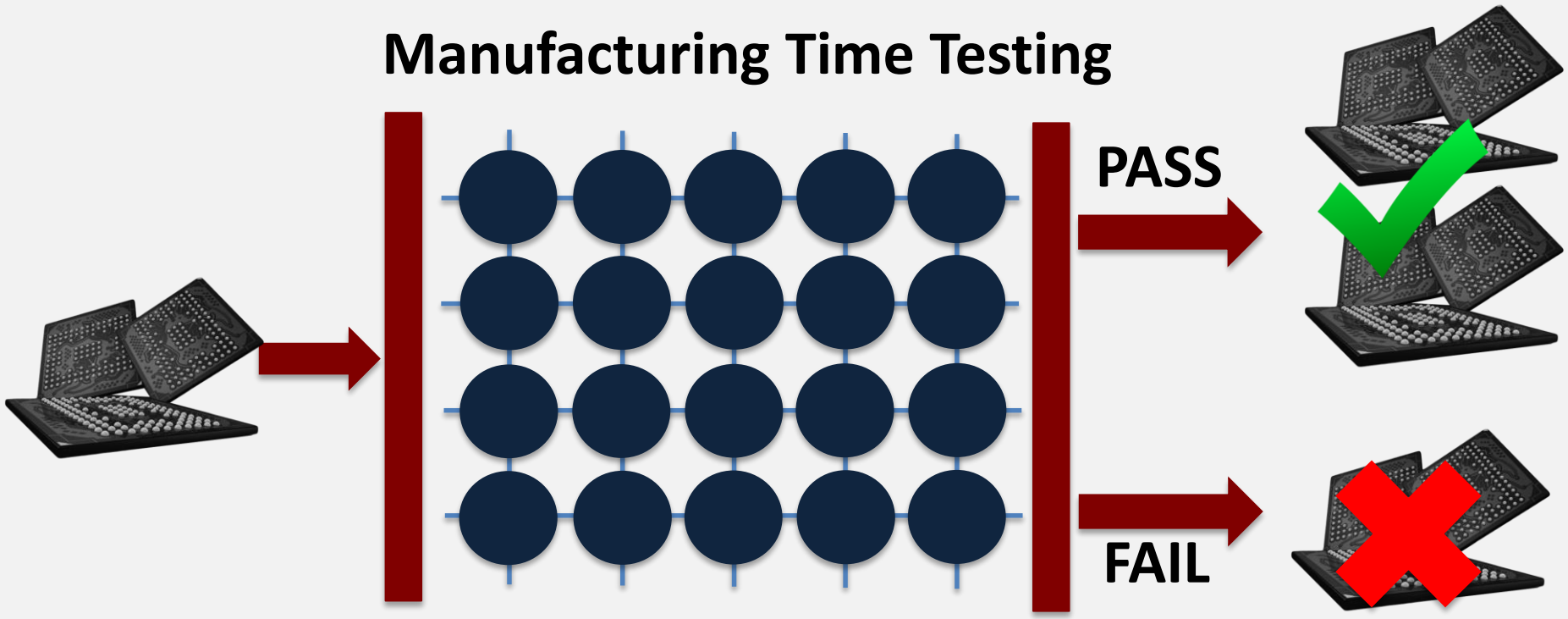
VARIABLE RETENTION TIME



Retention time changes *randomly* in some cells

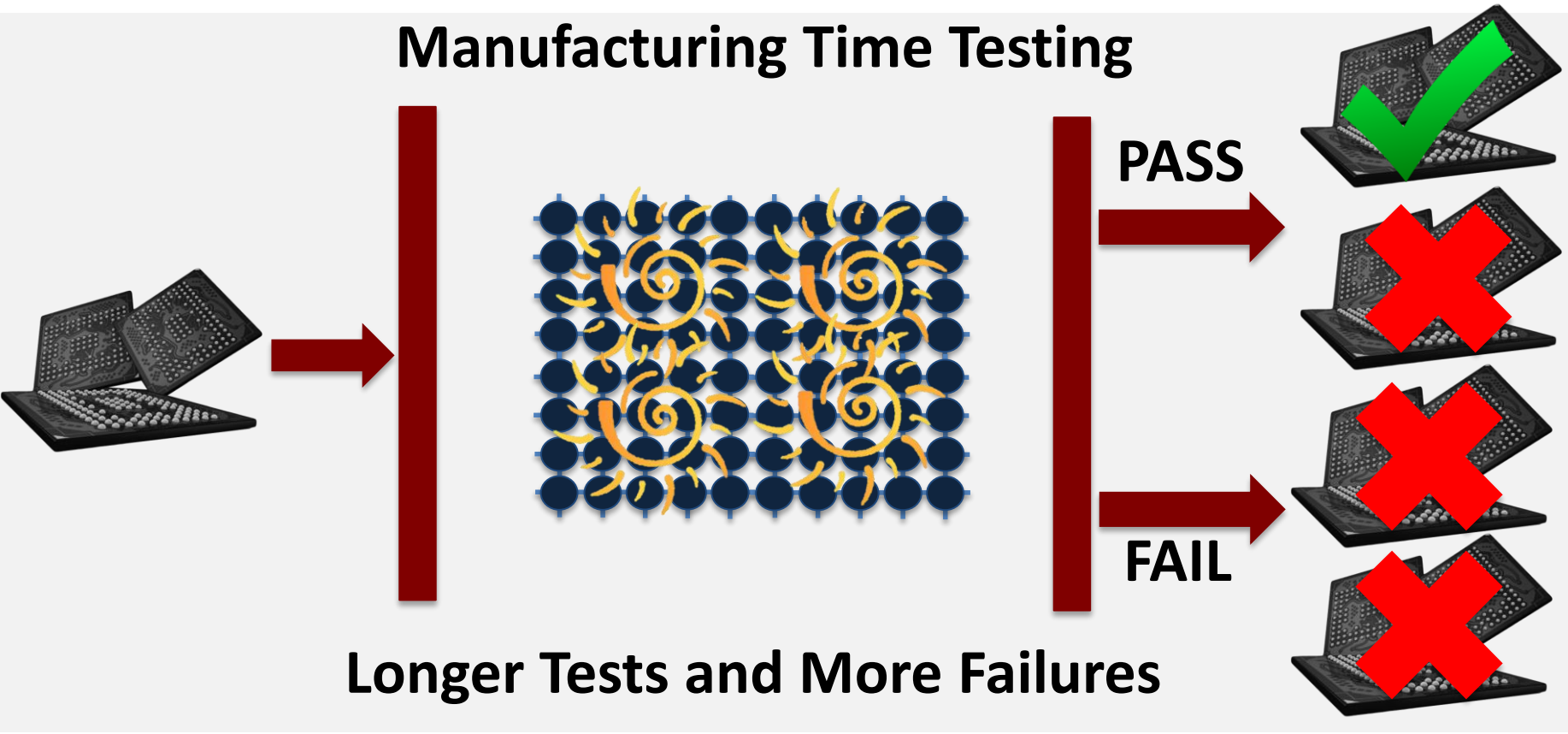
IEDM'87, IEDM'92

CURRENT APROACH TO CONTAIN INTERMITTENT FAILURES



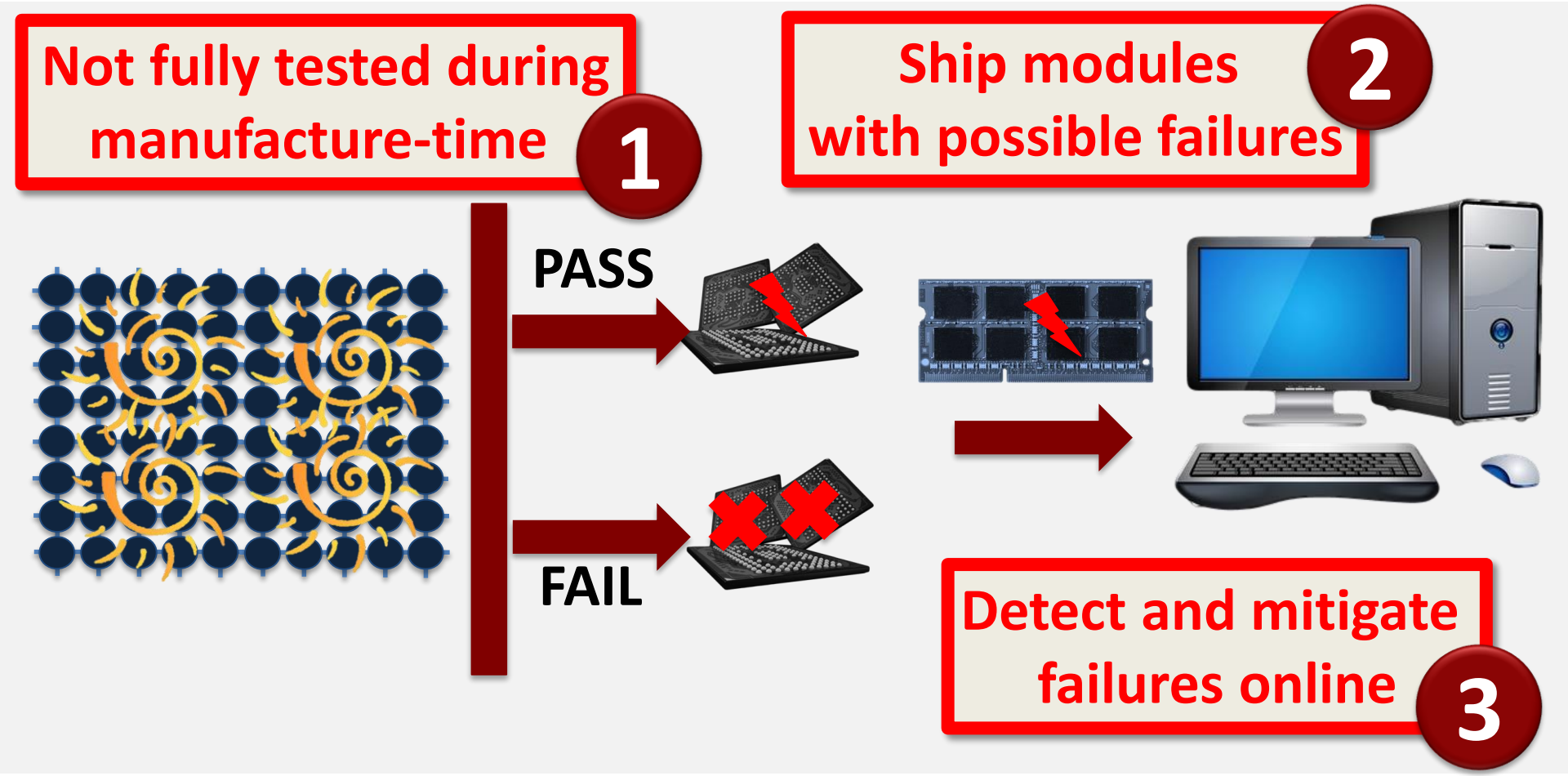
1. Manufacturers perform exhaustive testing of DRAM chips
2. Chips failing the tests are discarded

SCALING AFFECTING TESTING

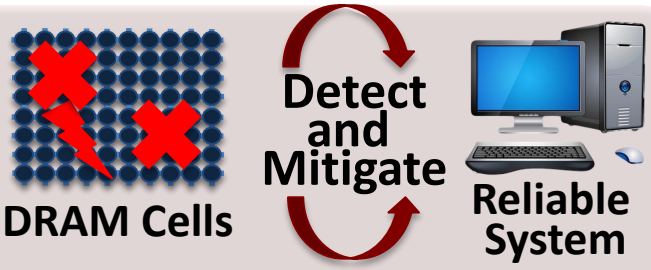


More interference in smaller technology nodes leads to lower yield and higher cost

SYSTEM-LEVEL ONLINE PROFILING



Increases yield, reduces cost, enables scaling



SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

CHALLENGE: INTERMITTENT FAILURES

EFFICACY OF SYSTEM-LEVEL TECHNIQUES WITH REAL DRAM CHIPS

HIGH-LEVEL DESIGN

NEW SYSTEM-LEVEL TECHNIQUES

EFFICACY OF SYSTEM-LEVEL TECHNIQUES

Can we leverage existing techniques?

1. Testing
2. Guardbanding
3. Error Correcting Code
4. Higher Strength ECC

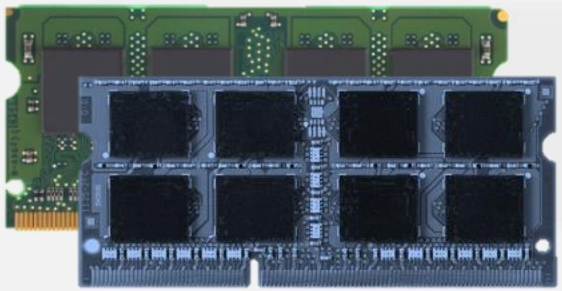
WITHOUT REAL DATA ON FAILURE RATE AND CHARACTERISTICS HARD TO ANALYZE

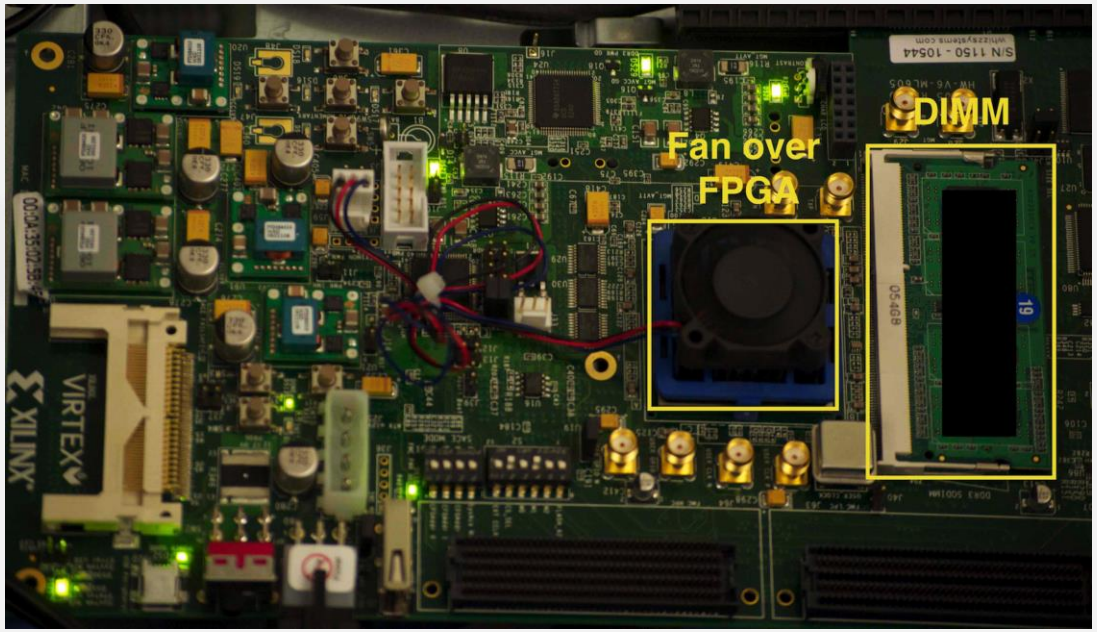
We analyze the effectiveness of these techniques using experimental data from real DRAMs

Data set publicly available

METHODOLOGY

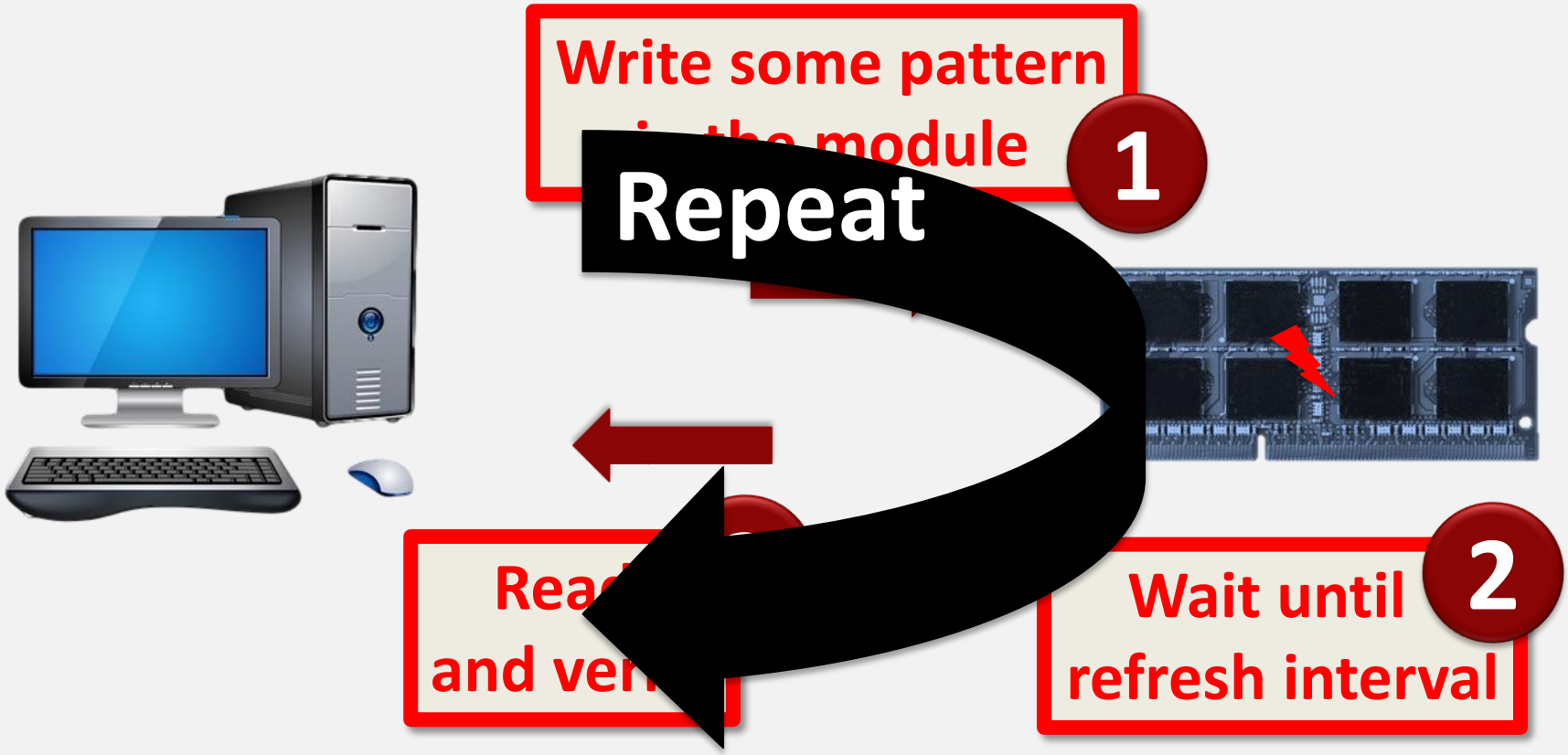
FPGA-based testing infrastructure





Evaluated 96 chips from three major vendors

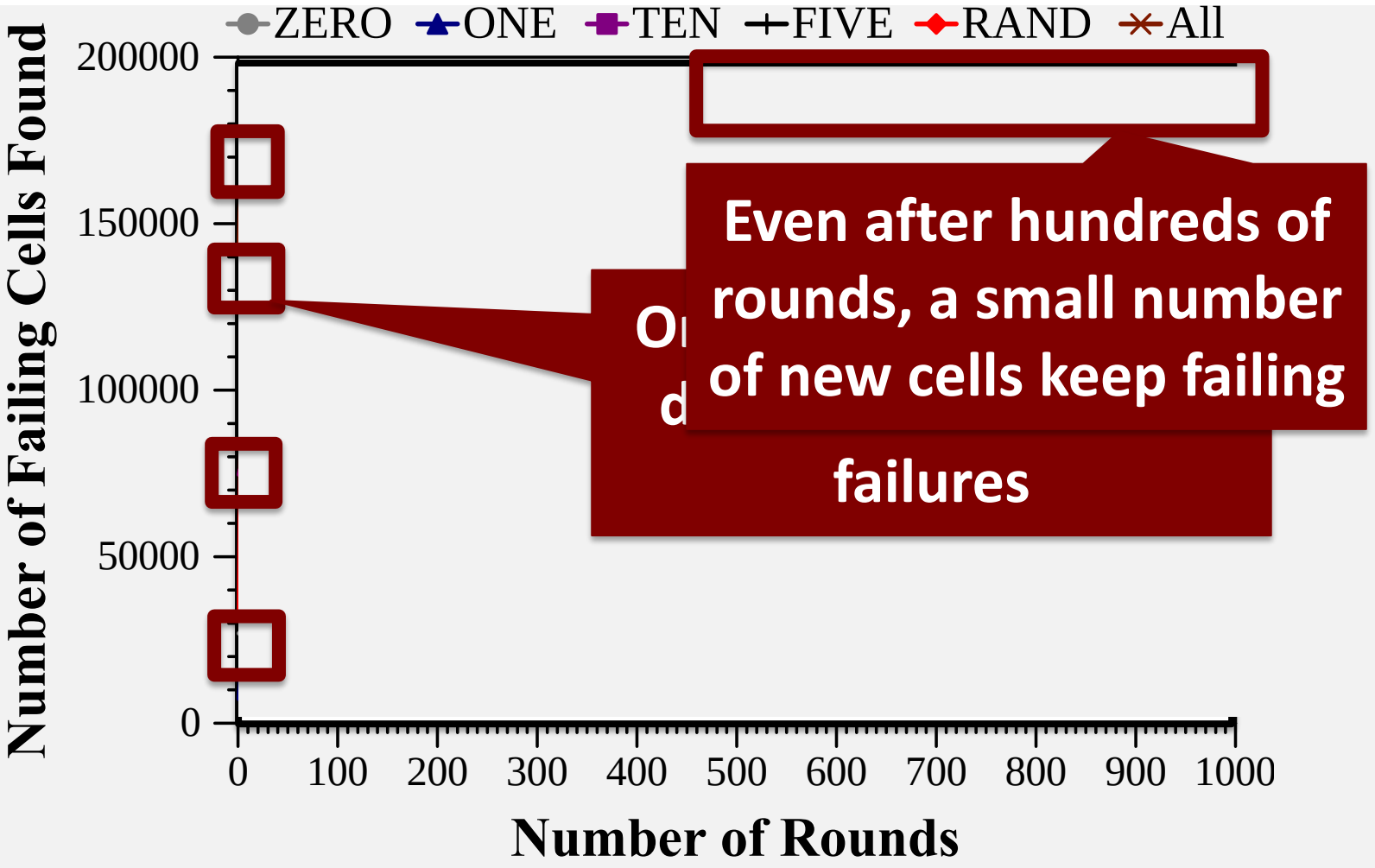
1. TESTING



Test each module with different patterns for many rounds

Zeros (0000), Ones (1111), Tens (1010), Fives (0101), Random

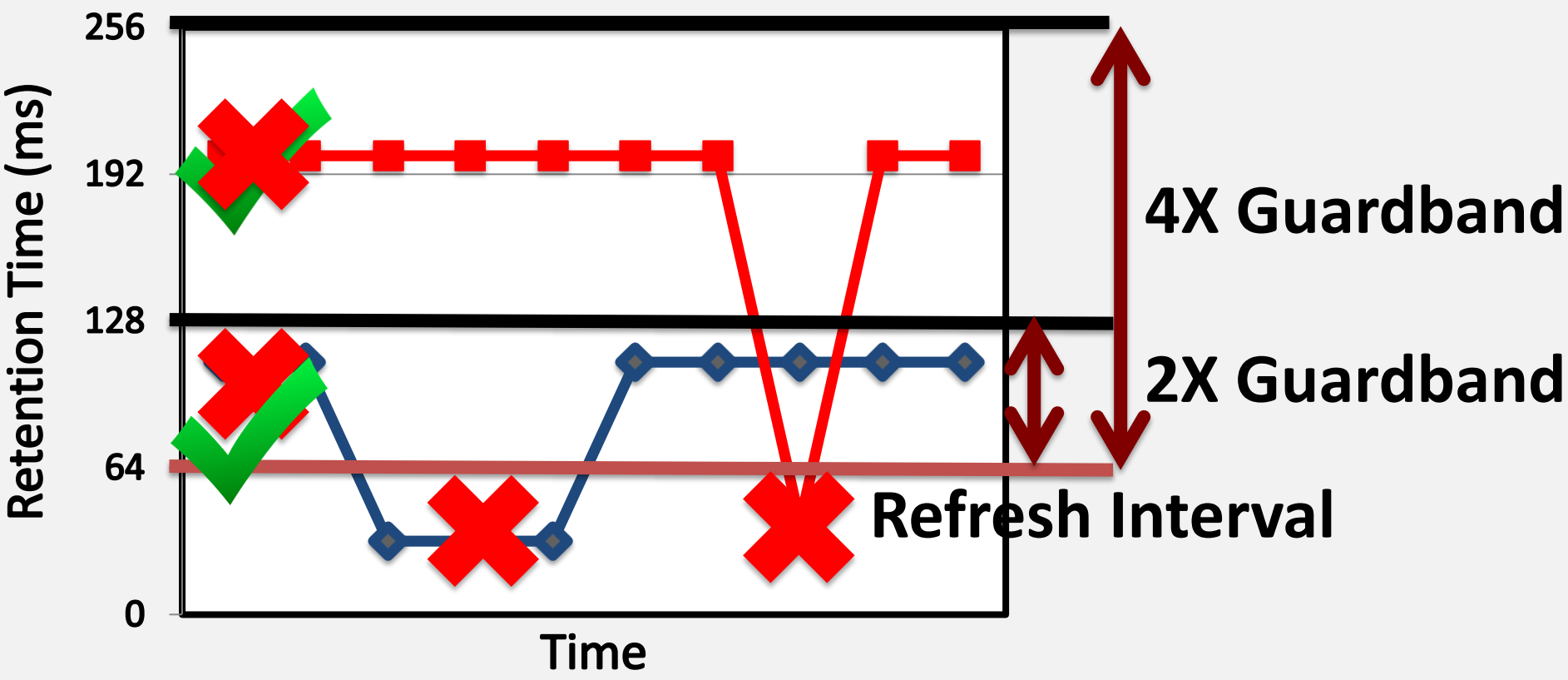
EFFICACY OF TESTING



Conclusion: Testing alone cannot detect all possible failures

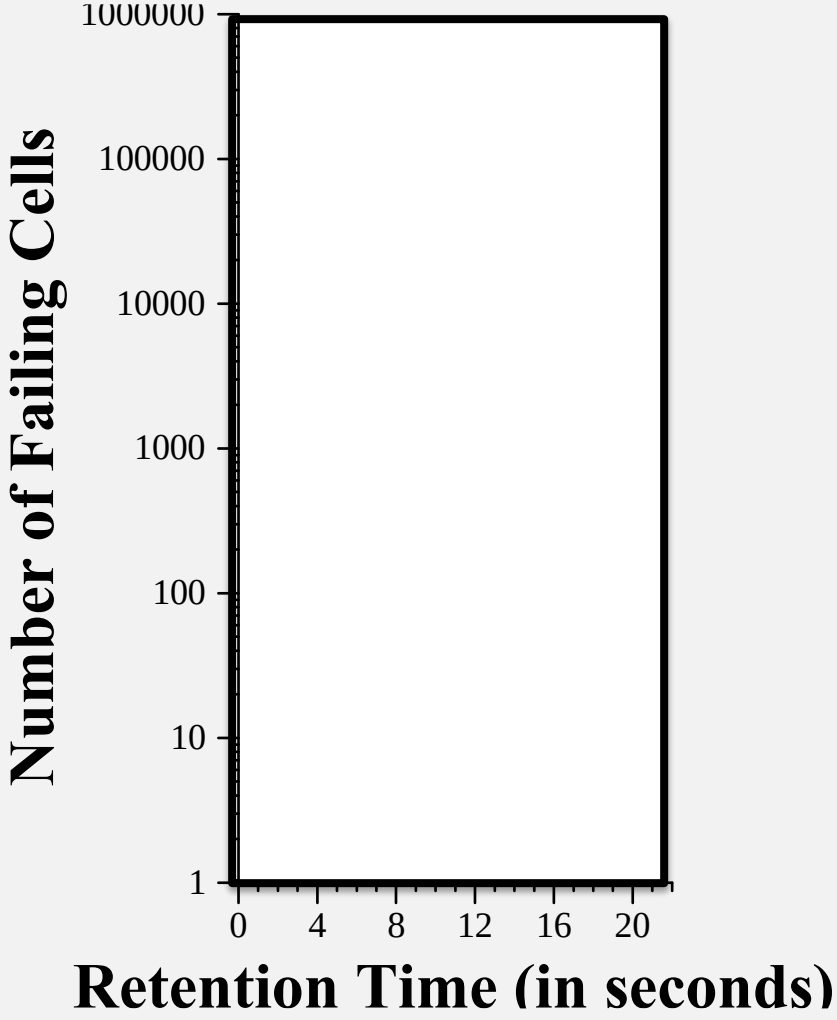
2. GUARDBANDING

- Adding a safety-margin on the refresh interval
- Can avoid VRT failures

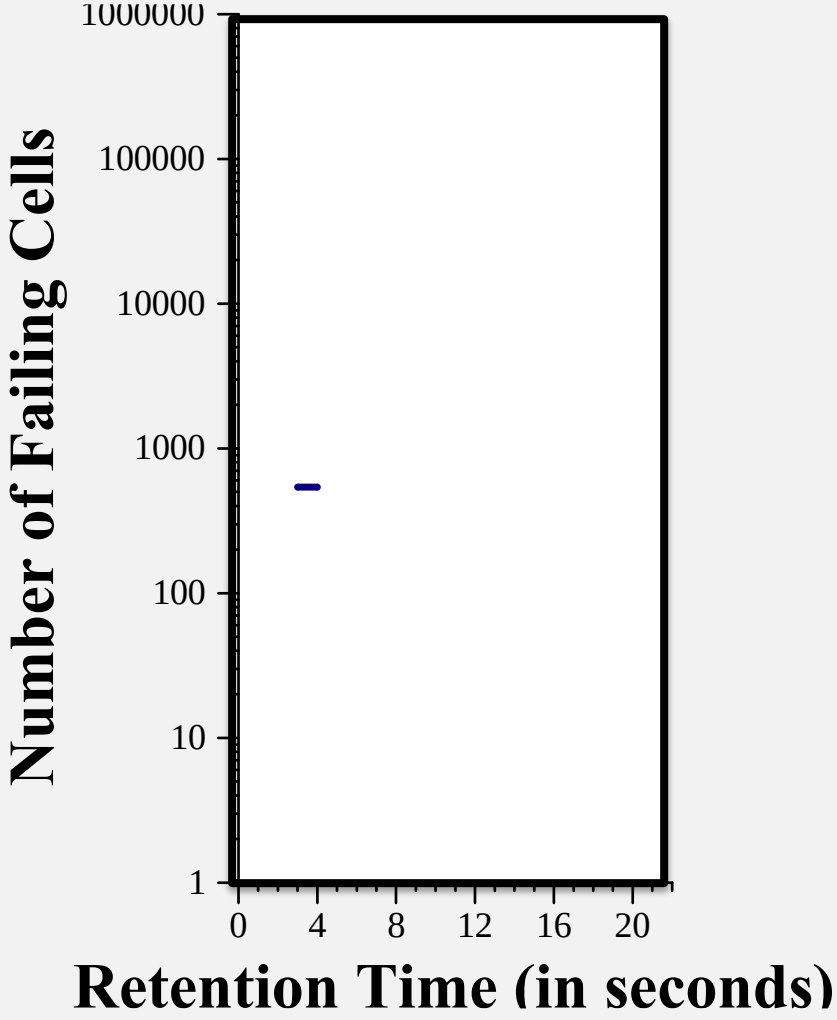


Effectiveness depends on the difference between retention times of a cell

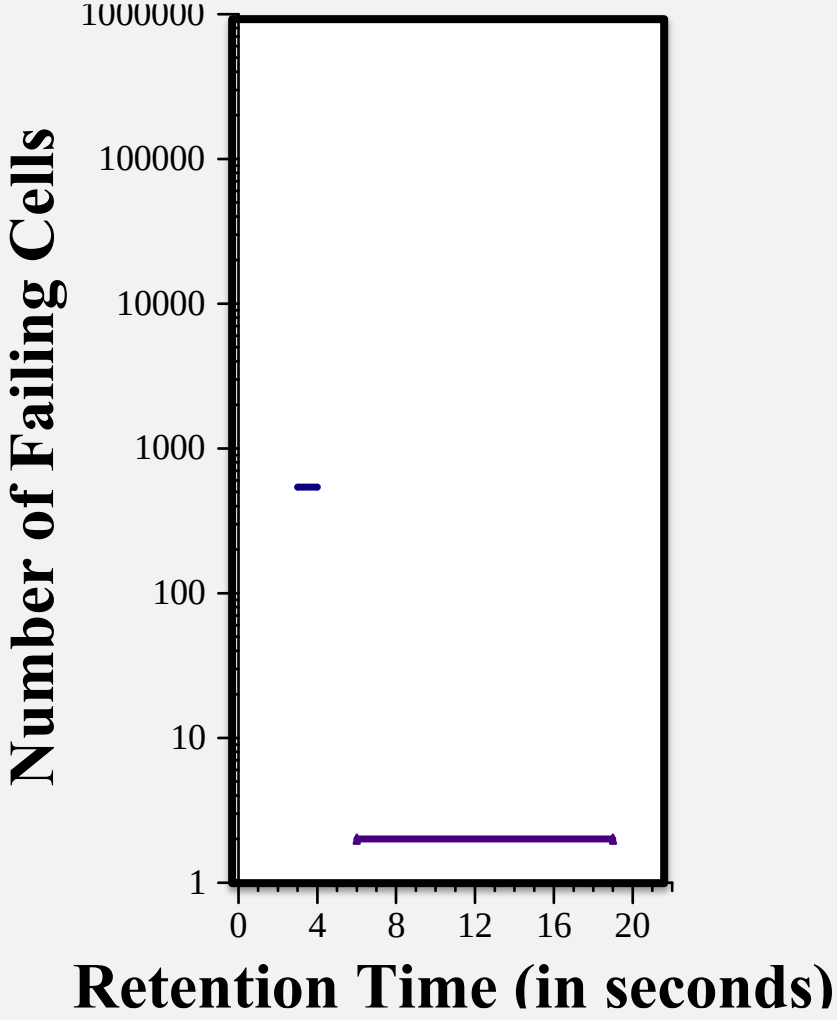
EFFICACY OF GUARDBANDING



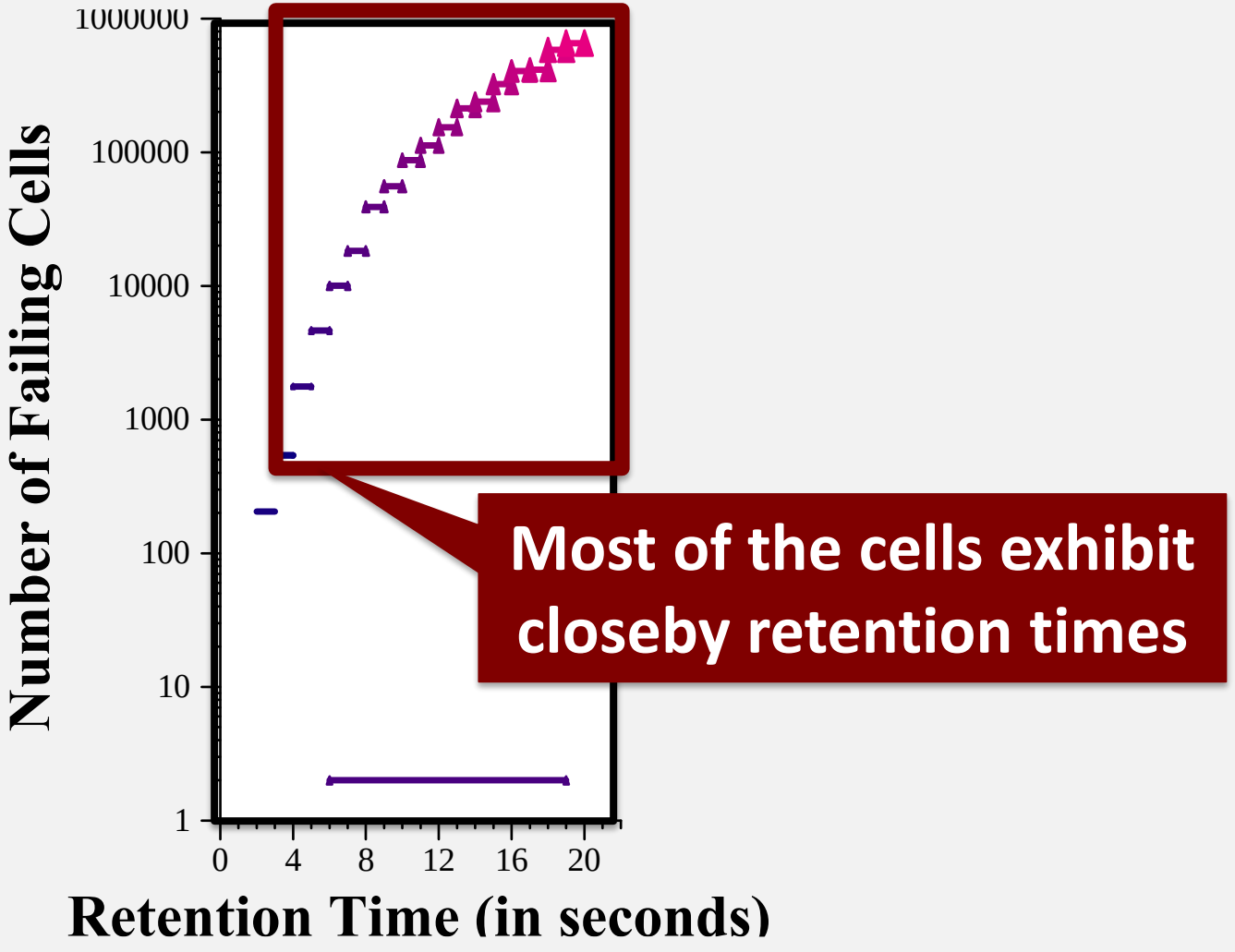
EFFICACY OF GUARDBANDING



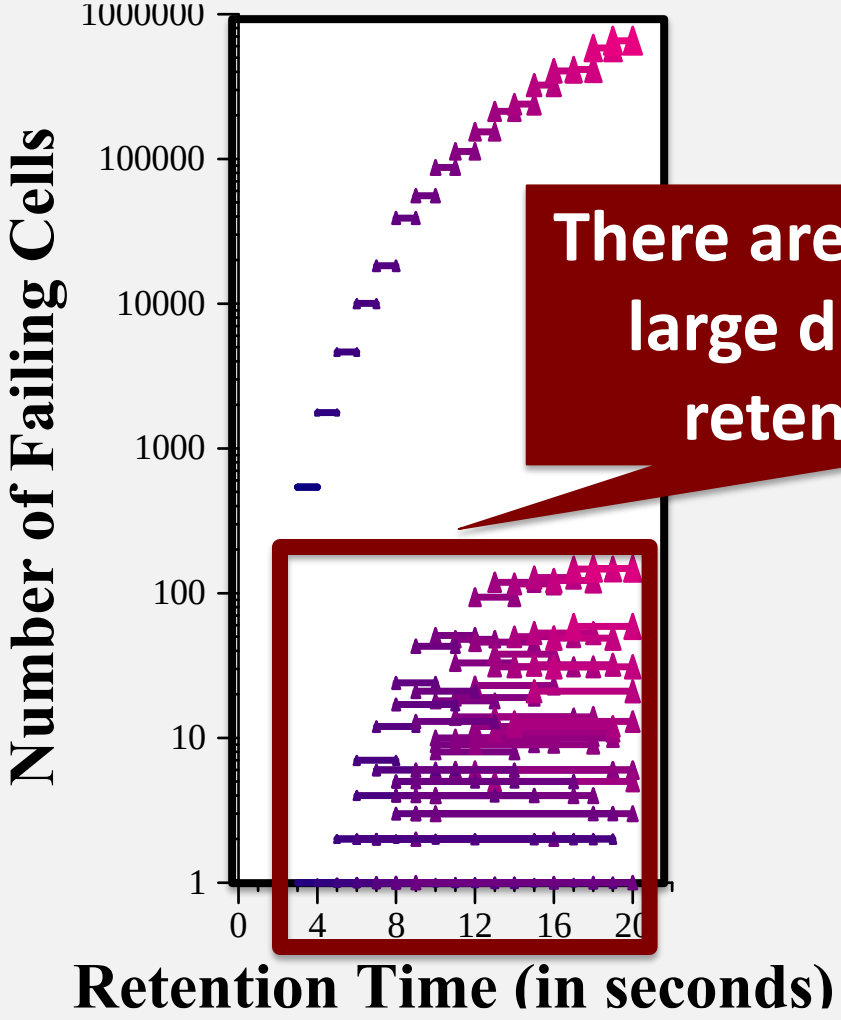
EFFICACY OF GUARDBANDING



EFFICACY OF GUARDBANDING



EFFICACY OF GUARDBANDING

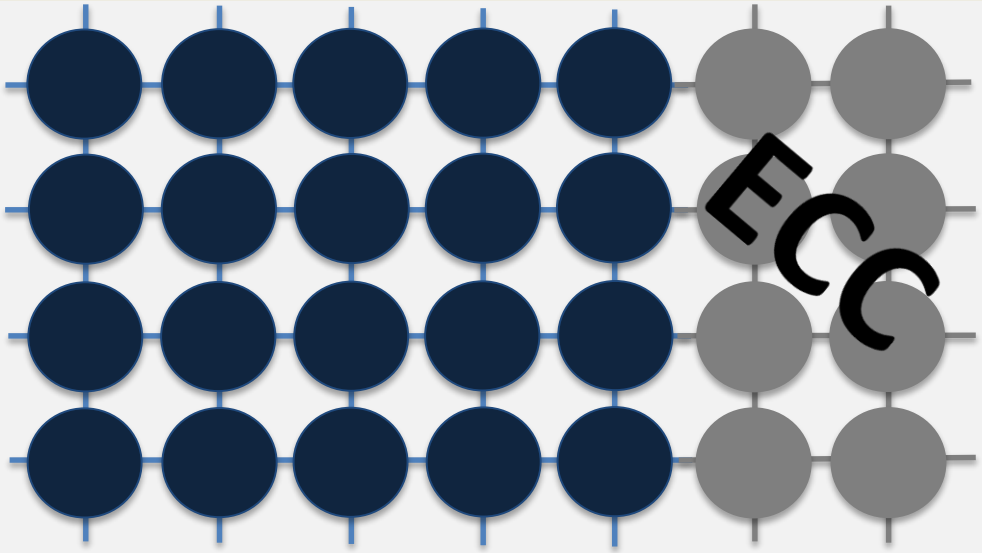


There are few cells with large differences in retention times

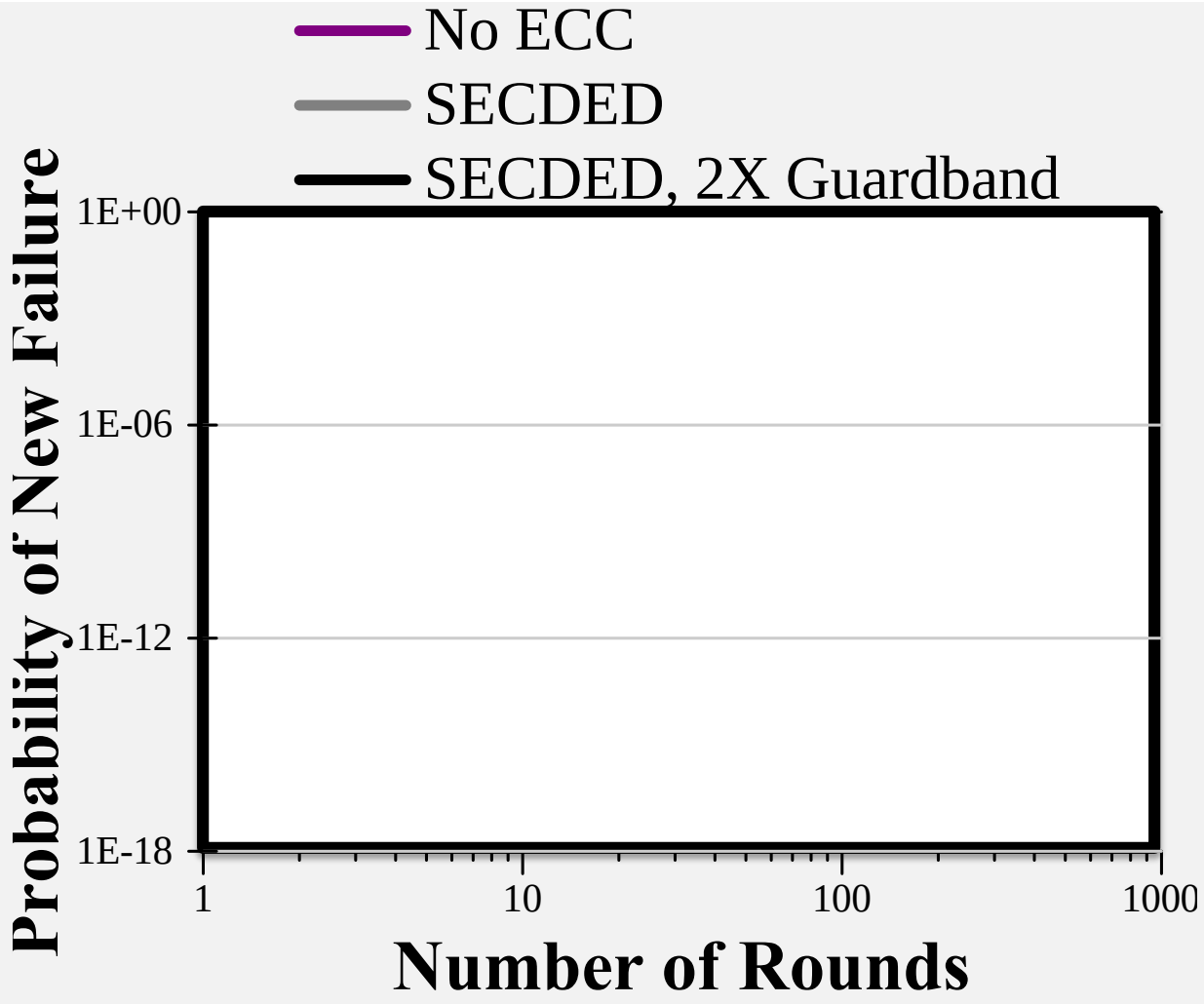
Conclusion: Even a large guardband (5X) cannot detect 5-15% of the failing cells

3. ERROR CORRECTING CODE

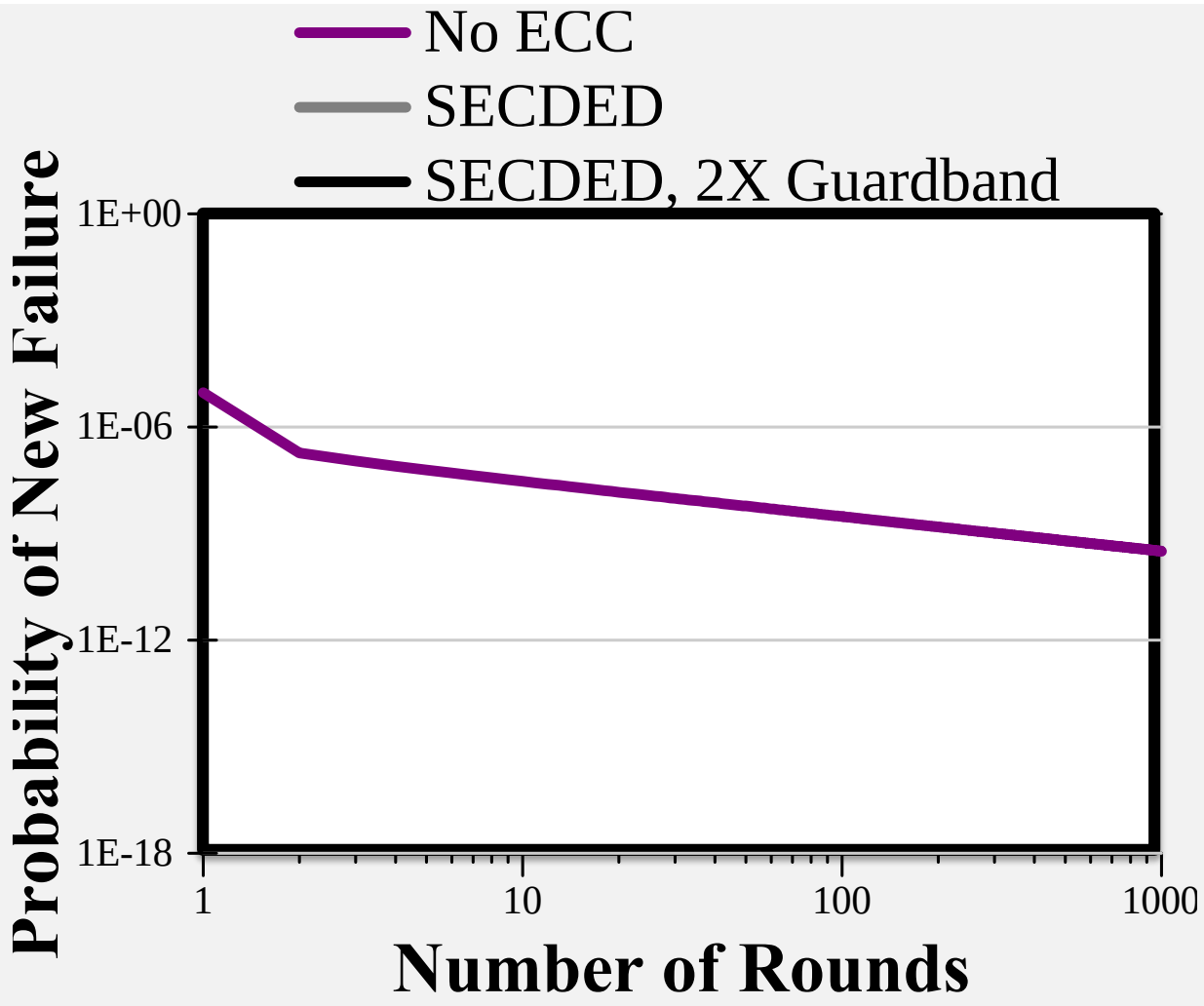
- **Error Correcting Code (ECC)**
  - **Additional information to detect error and correct data**



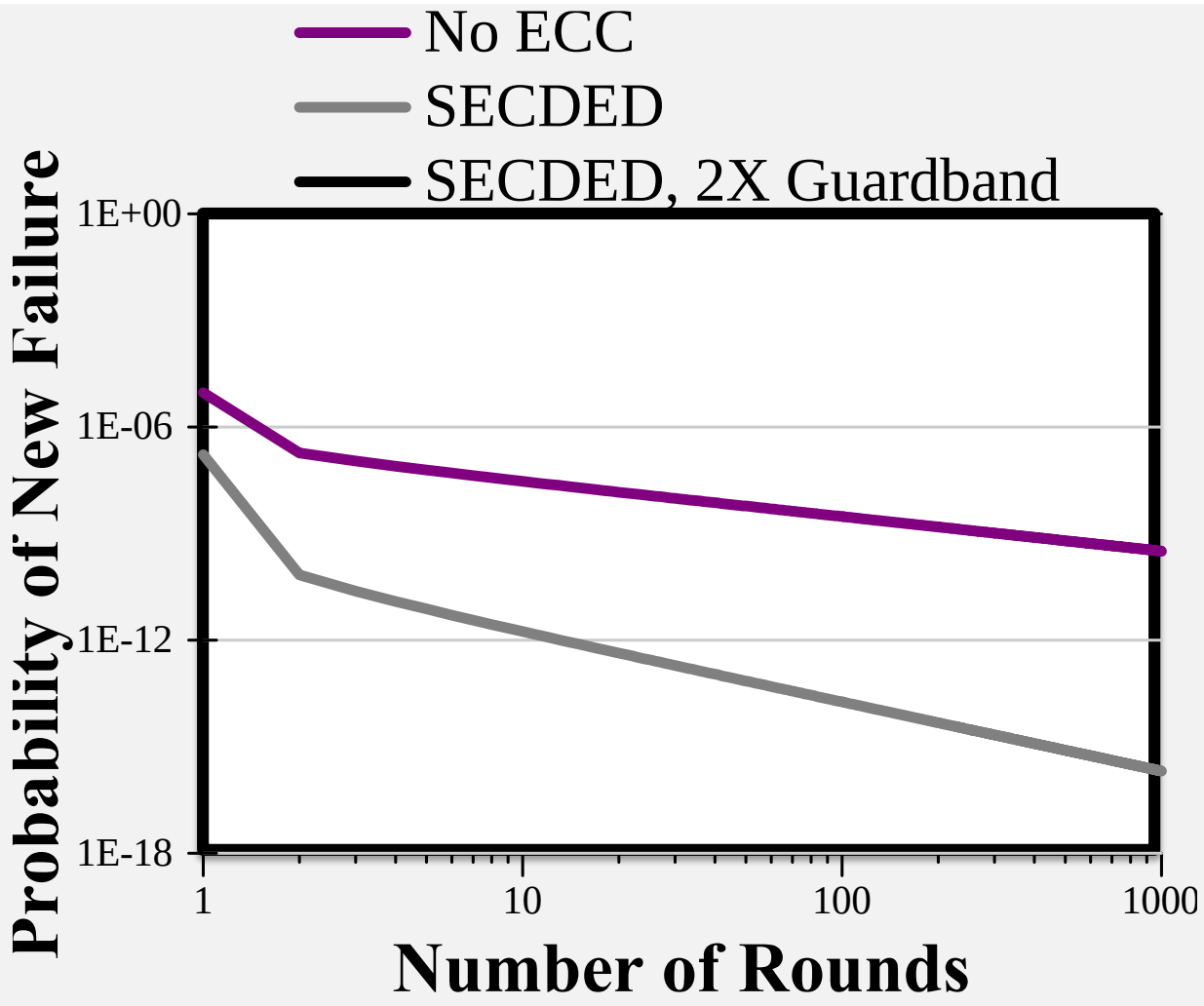
EFFECTIVENESS OF ECC



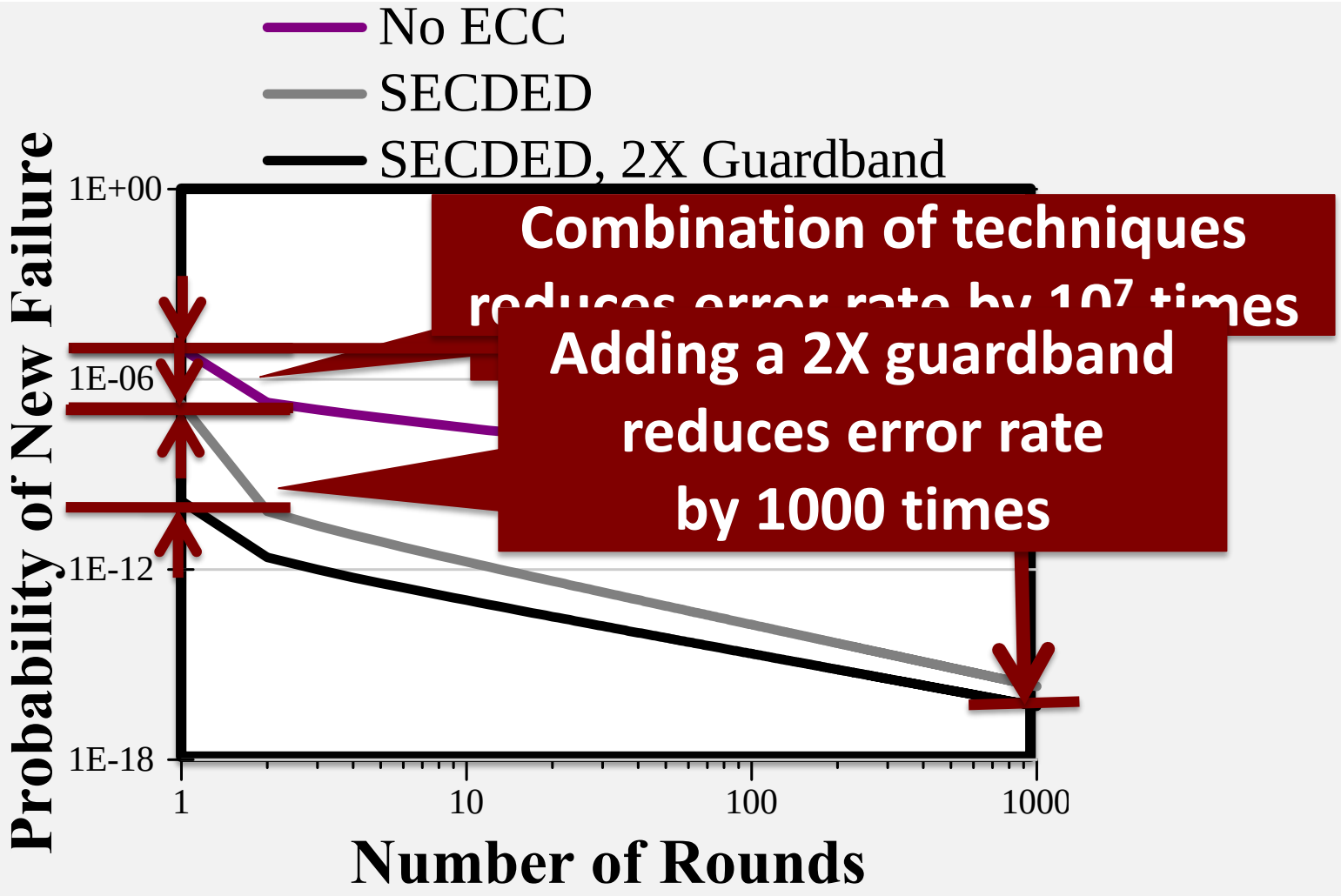
EFFECTIVENESS OF ECC



EFFECTIVENESS OF ECC



EFFECTIVENESS OF ECC

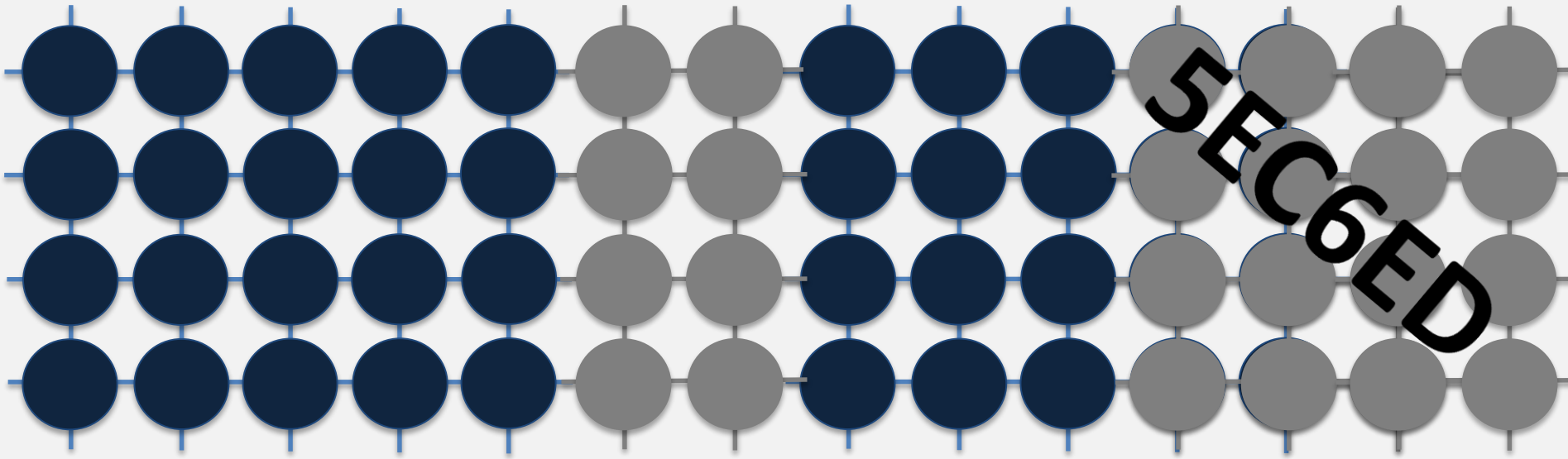


Conclusion: A combination of mitigation techniques is much more effective

4. HIGHER STRENGTH ECC (HI-ECC)

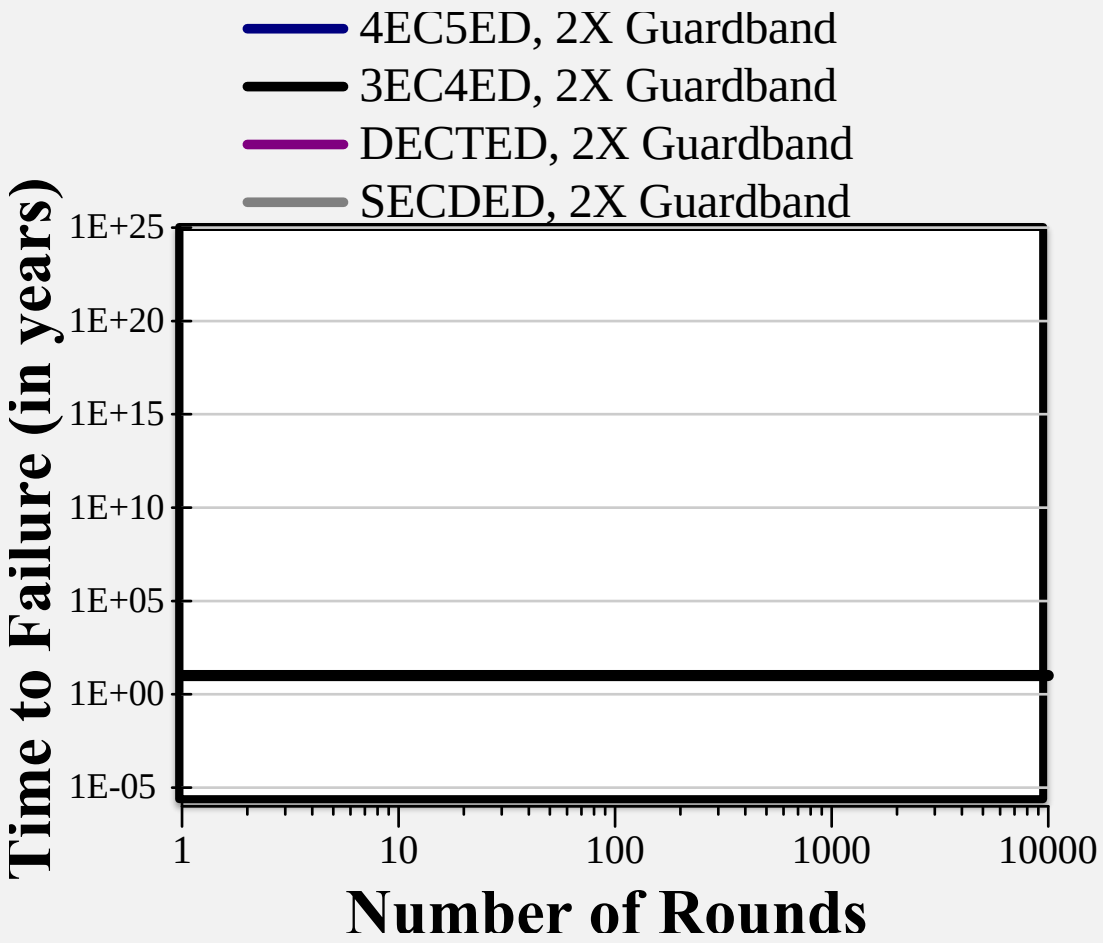
No testing, use strong ECC

But amortize cost of ECC over larger data chunk



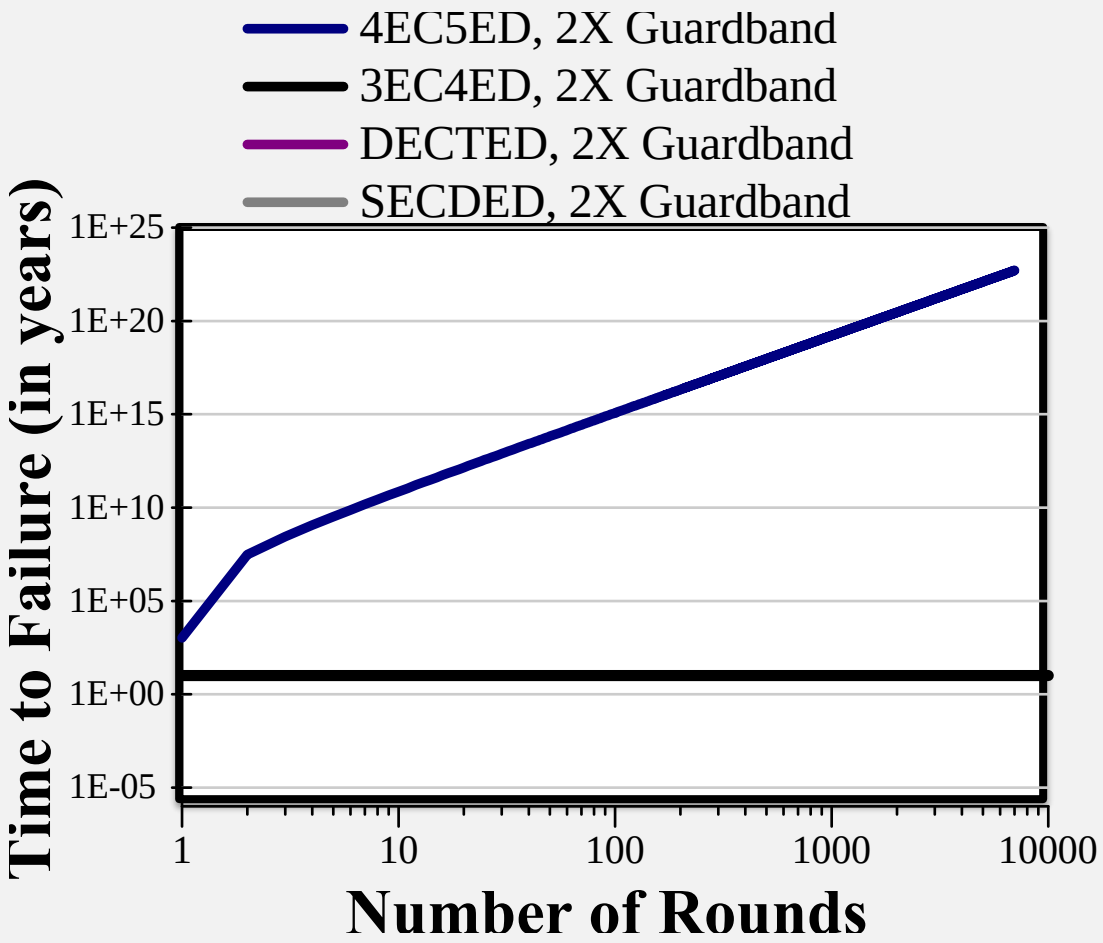
Can potentially tolerate errors at the cost of higher strength ECC

EFFICACY OF HI-ECC



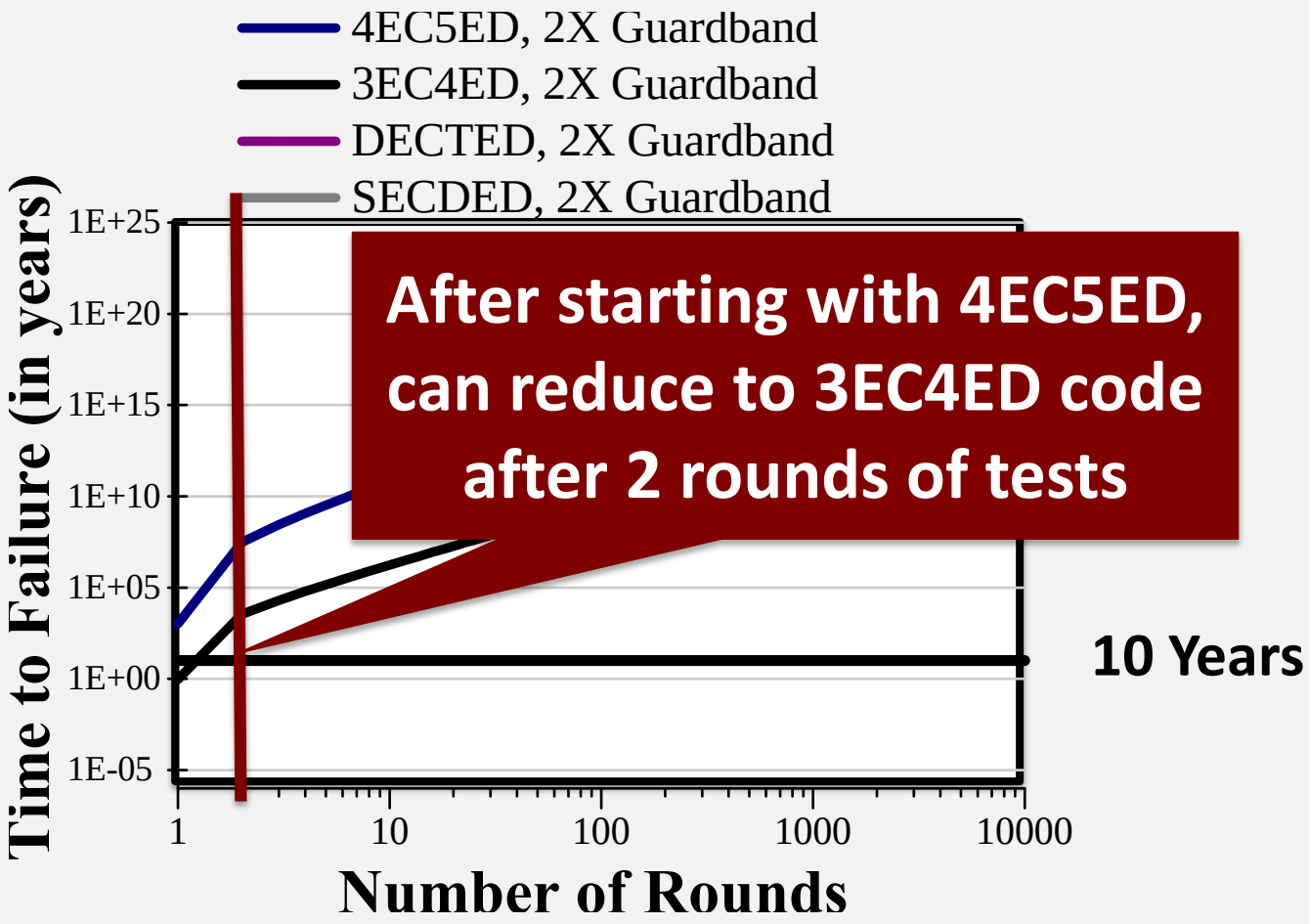
10 Years

EFFICACY OF HI-ECC

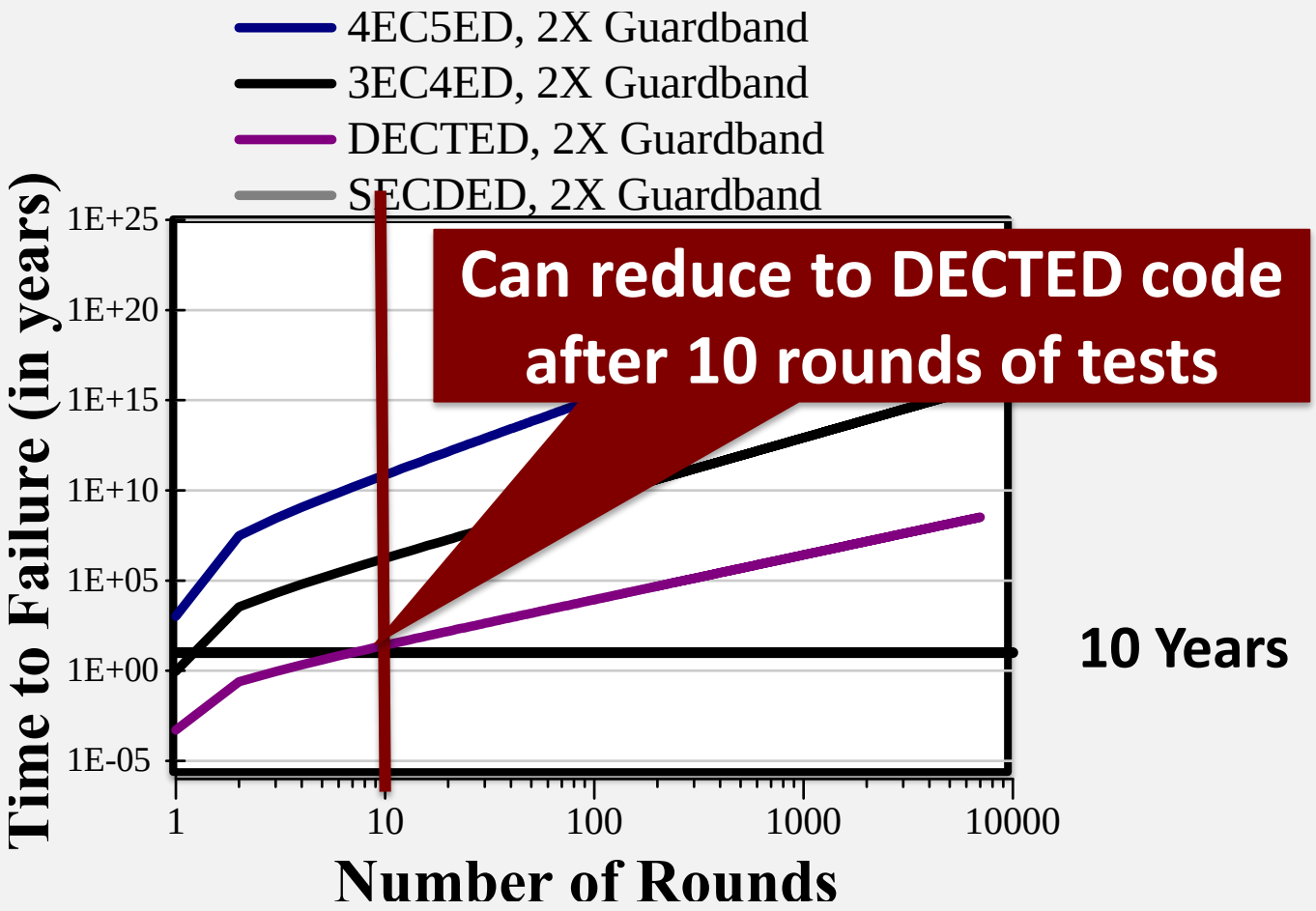


10 Years

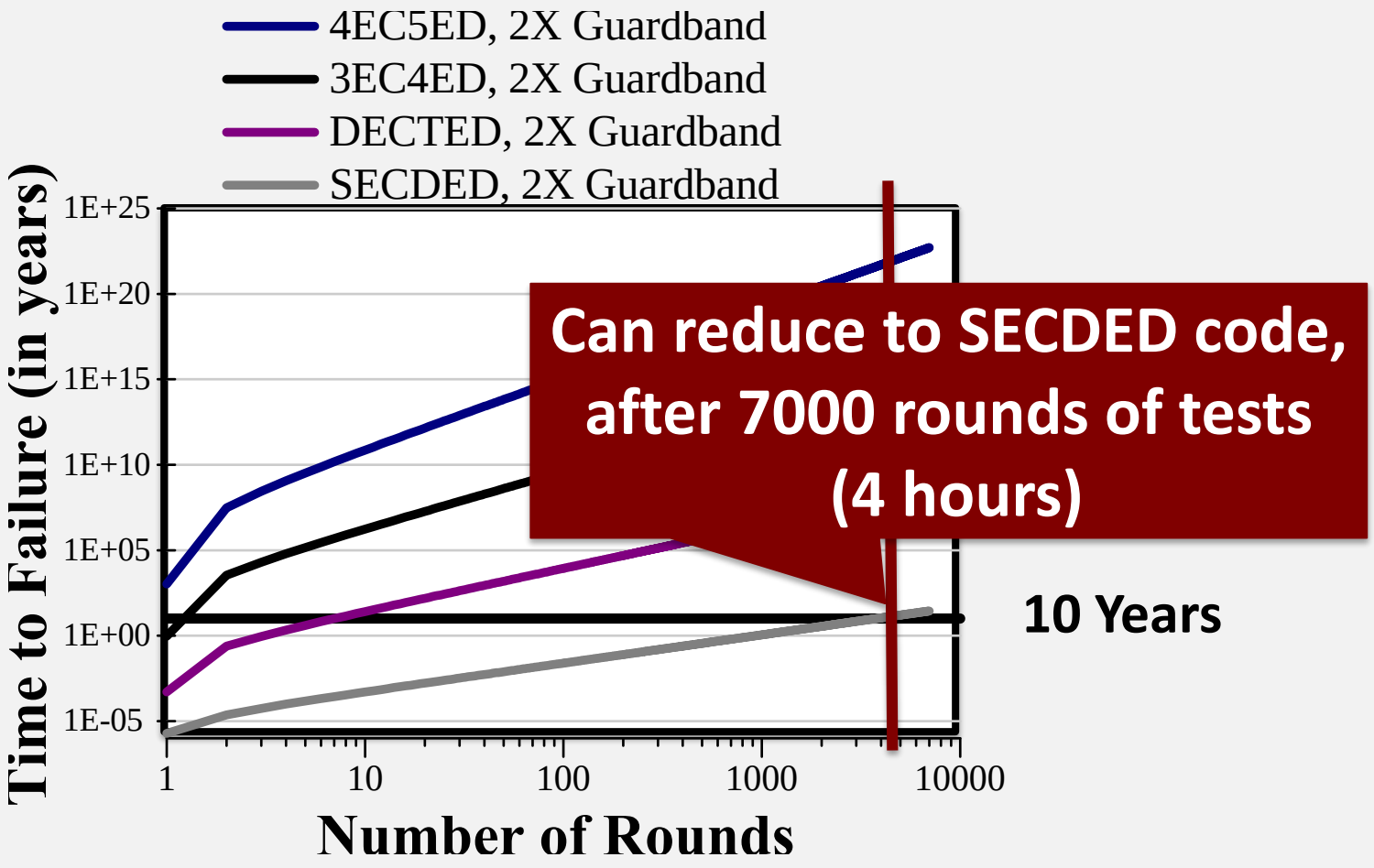
EFFICACY OF HI-ECC



EFFICACY OF HI-ECC



EFFICACY OF HI-ECC

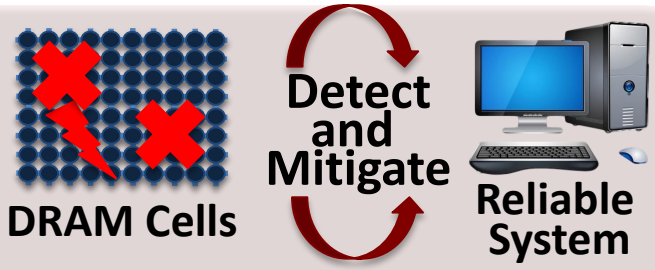


Conclusion: Testing can help to reduce the ECC strength, but *blocks memory for hours*

CONCLUSIONS SO FAR

Key Observations:

- **Testing alone cannot detect all possible failures**
- **Combination of ECC and other mitigation techniques is much more effective**
- **Testing can help to reduce the ECC strength**
  - **Even when starting with a higher strength ECC**
  - **But degrades performance**



SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

CHALLENGE: INTERMITTENT FAILURES

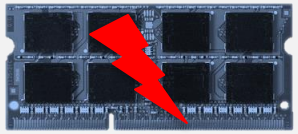
EFFICACY OF SYSTEM-LEVEL TECHNIQUES WITH REAL DRAM CHIPS

HIGH-LEVEL DESIGN

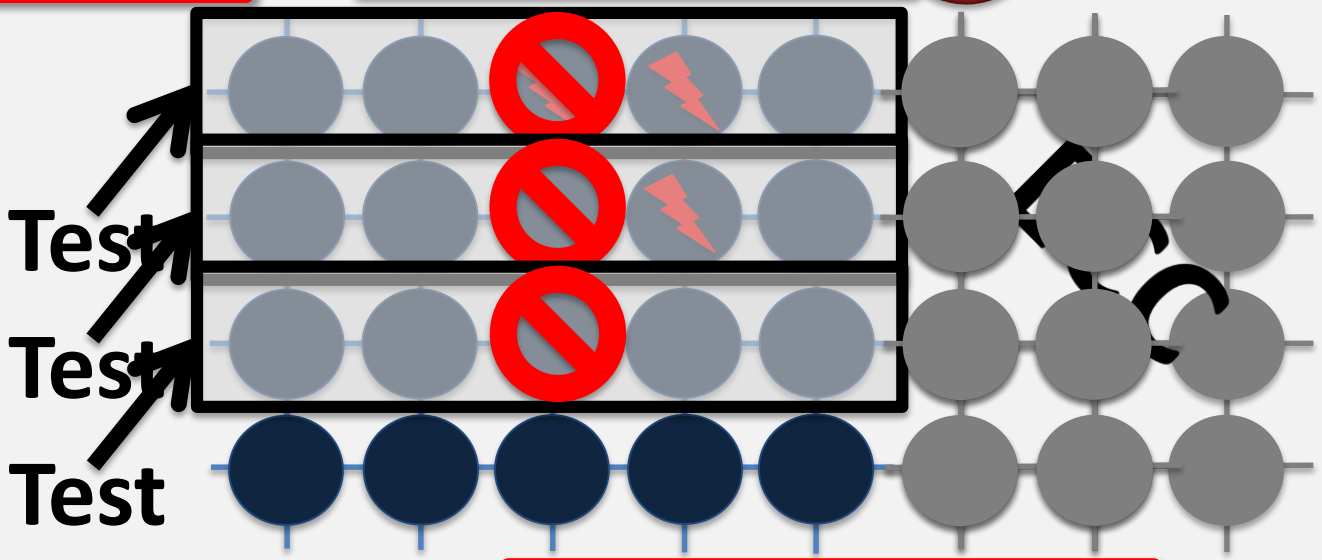
NEW SYSTEM-LEVEL TECHNIQUES

TOWARDS AN ONLINE PROFILING SYSTEM

1 Initially Protect DRAM with Strong ECC

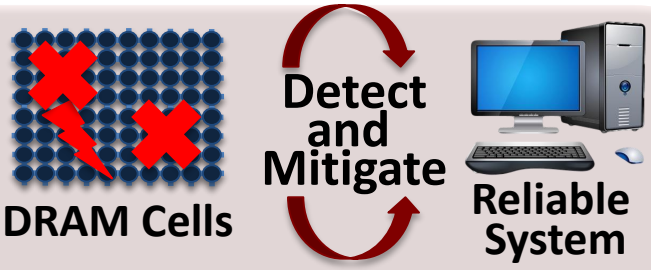


2 Periodically Test Parts of DRAM



3 Mitigate errors and reduce ECC

Run tests periodically after a short interval at smaller regions of memory



SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

CHALLENGE: INTERMITTENT FAILURES

EFFICACY OF SYSTEM-LEVEL TECHNIQUES WITH REAL DRAM CHIPS

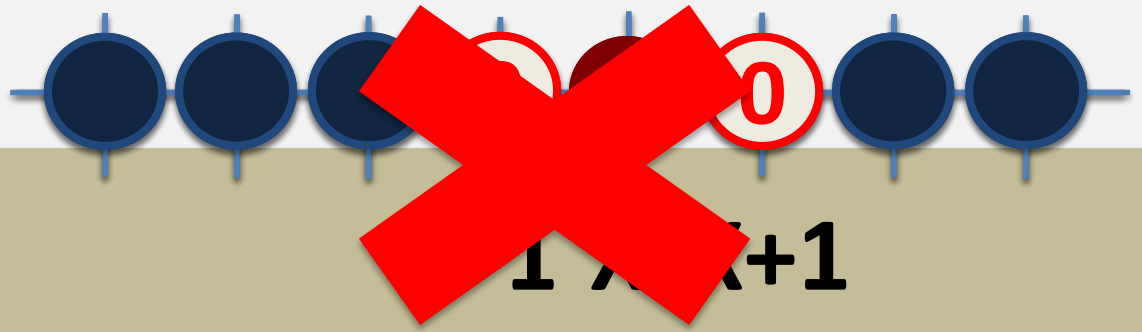
HIGH-LEVEL DESIGN

NEW SYSTEM-LEVEL TECHNIQUES

WHY SO MANY ROUNDS OF TESTS?

DATA-DEPENDENT FAILURE

Fails when specific pattern in the neighboring cell



Even many rounds of random patterns cannot detect all failures

DETERMINE THE LOCATION OF PHYSICALLY ADJACENT CELLS

SCRAMBLED ADDRESS

X-? X X+?

NAÏVE SOLUTION

For a given failure X,

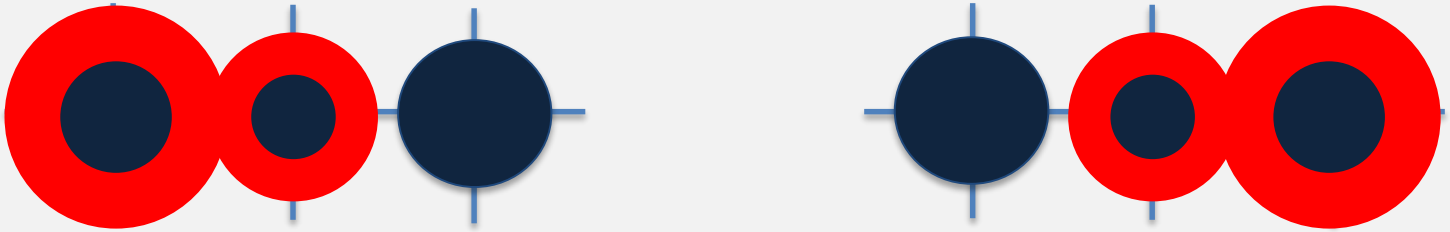
test every combination of two bit addresses in the row

$O(n^2)$

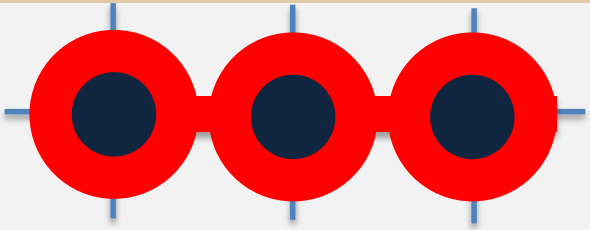
8192*8192 tests, 49 days for a row with 8K cells

Our goal is to reduce the test time

STRONGLY VS. WEAKLY DEPENDENT CELLS



STRONGLY DEPENDENT
Fails even if only one neighbor data changes

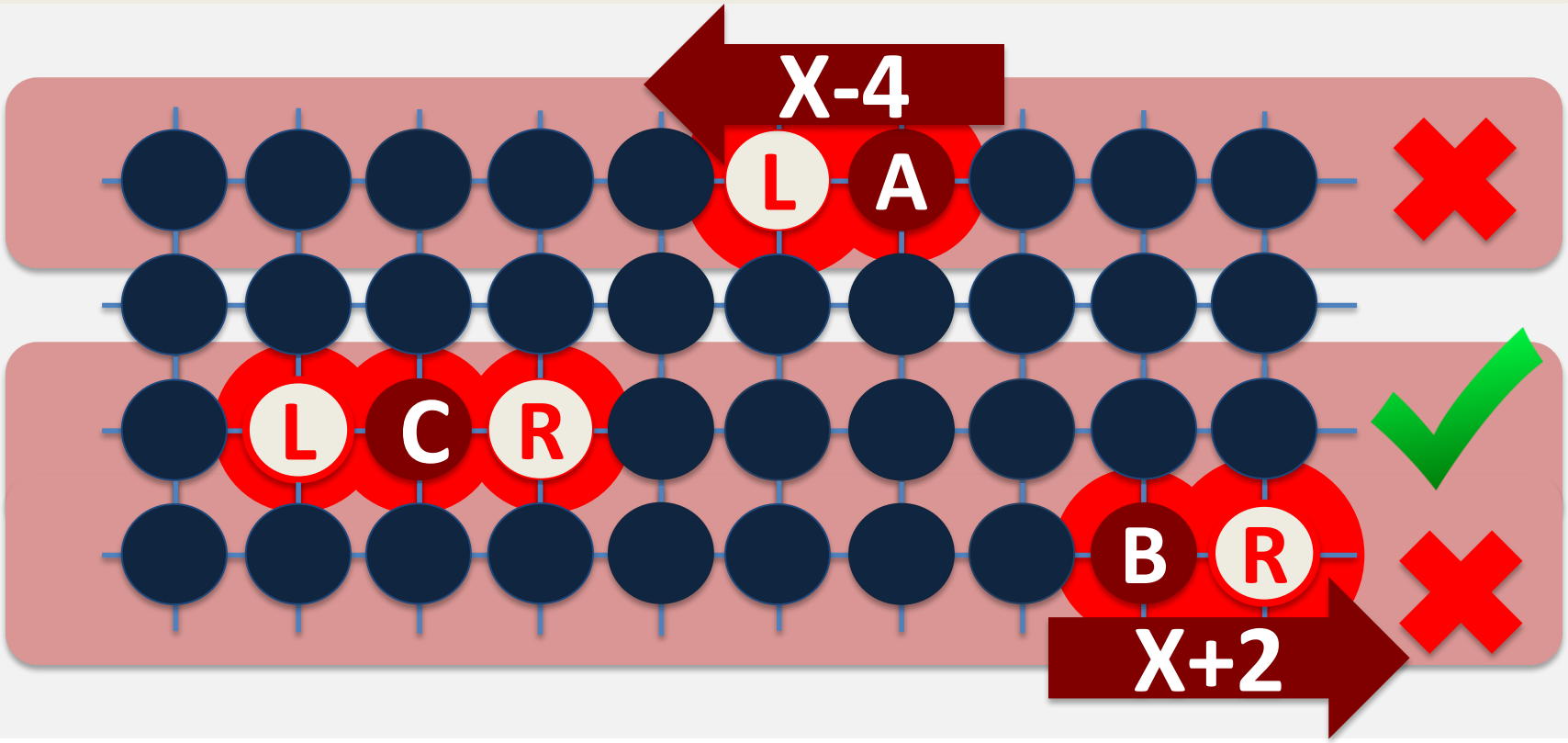


WEAKLY DEPENDENT
Fails if both neighbor data change

Can detect neighbor location in strongly dependent cells by testing every bit address 0, 1, … , X, X+1, X+2, … n

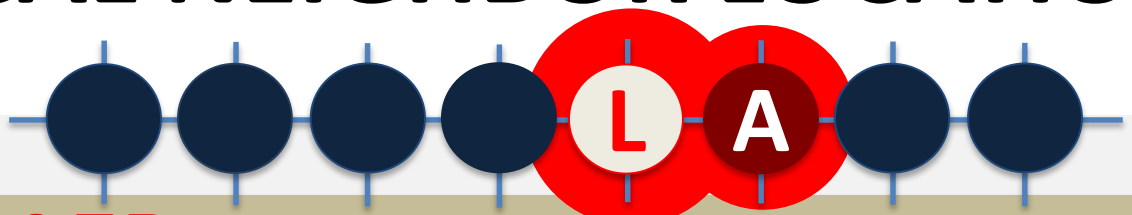
PHYSICAL NEIGHBOR LOCATION TEST

Testing every bit address will detect only one neighbor
Run parallel tests in different rows



Aggregate the locations from different rows

PHYSICAL NEIGHBOR LOCATION TEST



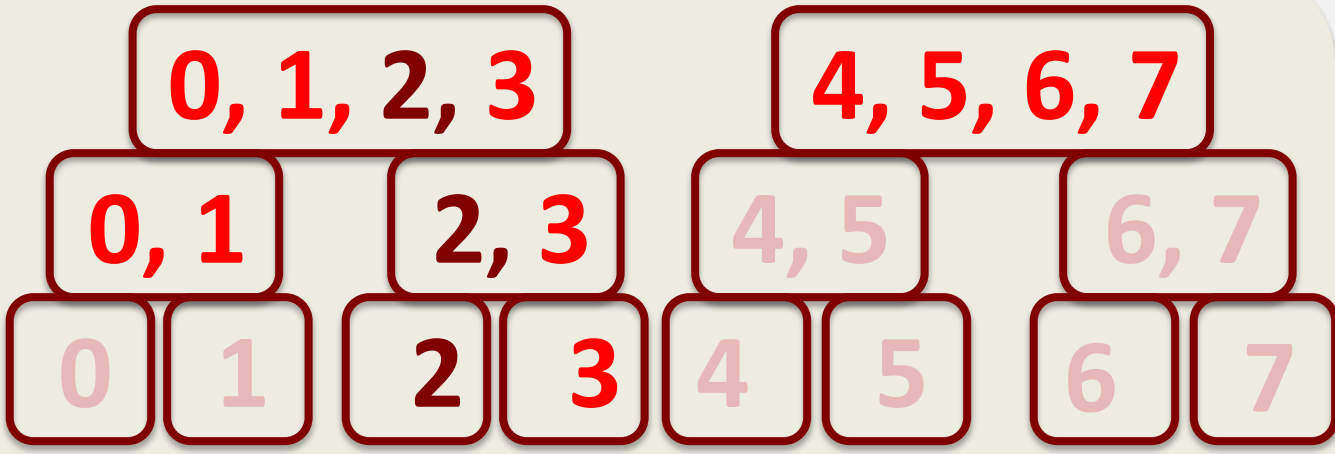
SCRAMBLED ADDRESS

X-4 X
2 6

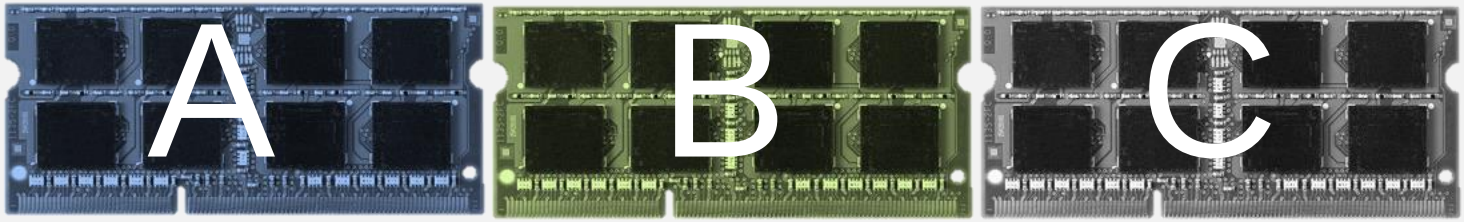
LINEAR TESTING

0 1 2 3 4 5 6 7

RECURSIVE TESTING



PHYSICAL NEIGHBOR-AWARE TEST



	A	B	C
NUM TEST REDUCED	745654X	1016800X	745654X
EXTRA FAILURES	27%	11%	14%

Detects more failures with small number of tests leveraging neighboring information

NEW SYSTEM-LEVEL TECHNIQUES

REDUCE FAILURE MITIGATION OVERHEAD

Mitigation for worst vs. common case

Reduces mitigation cost around 3X-10X

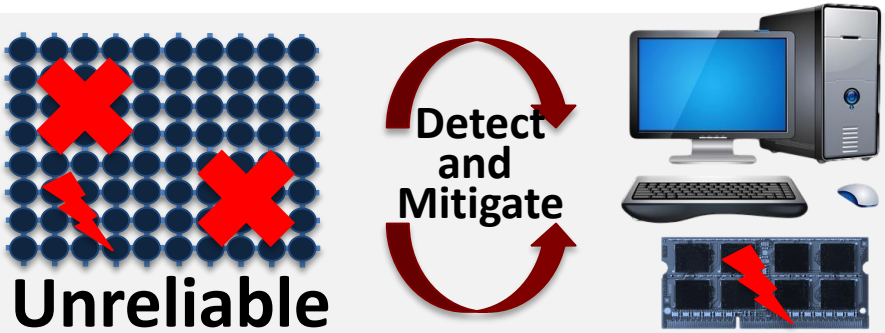
ONGOING

Variable refresh

Leverages adaptive refresh to mitigate failures

DSN'15

SUMMARY

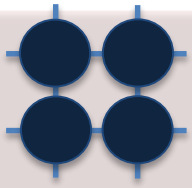


Proposed online profiling to enable scaling

Analyzed efficacy of system-level detection and mitigation techniques

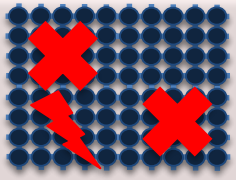
Found that combination of techniques is much more effective, but blocks memory

Proposed new system-level techniques to reduce detection and mitigation overhead



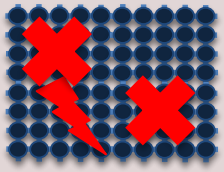
DRAM Cells

Technology Scaling



DRAM Cells

DRAM SCALING CHALLENGE



DRAM Cells

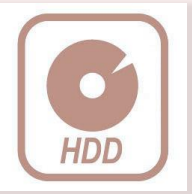
Detect and Mitigate

Reliable System

SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

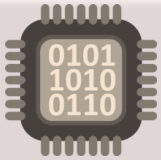
Non-Volatile Memory

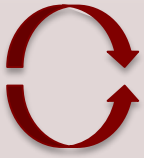
UNIFY

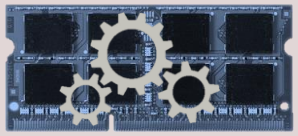


Storage

NON-VOLATILE MEMORIES: UNIFIED MEMORY & STORAGE

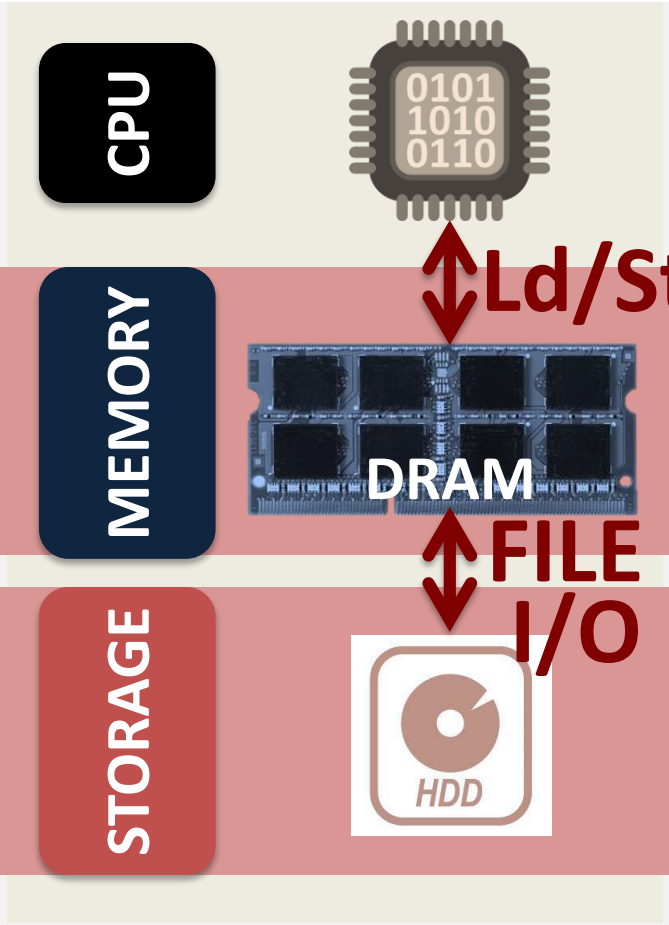






PAST AND FUTURE WORK

TWO-LEVEL STORAGE MODEL



VOLATILE

FAST

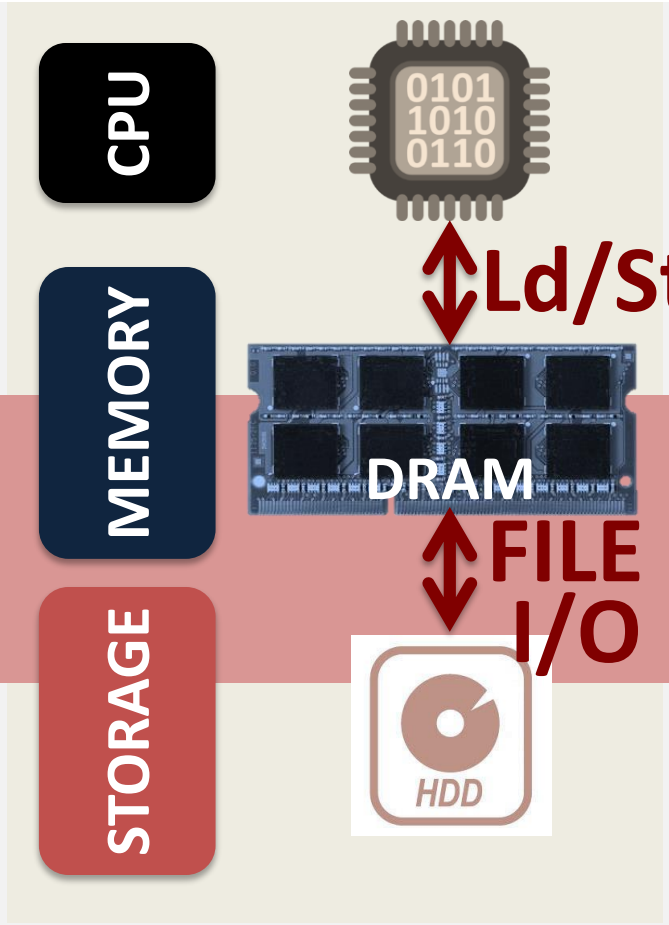
BYTE ADDR

NONVOLATILE

SLOW

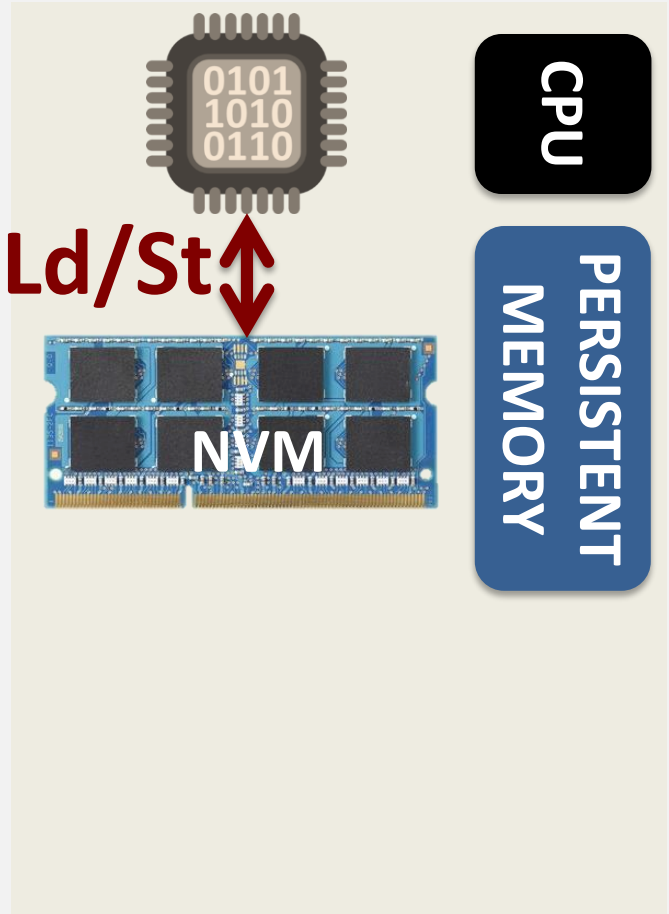
BLOCK ADDR

TWO-LEVEL STORAGE MODEL



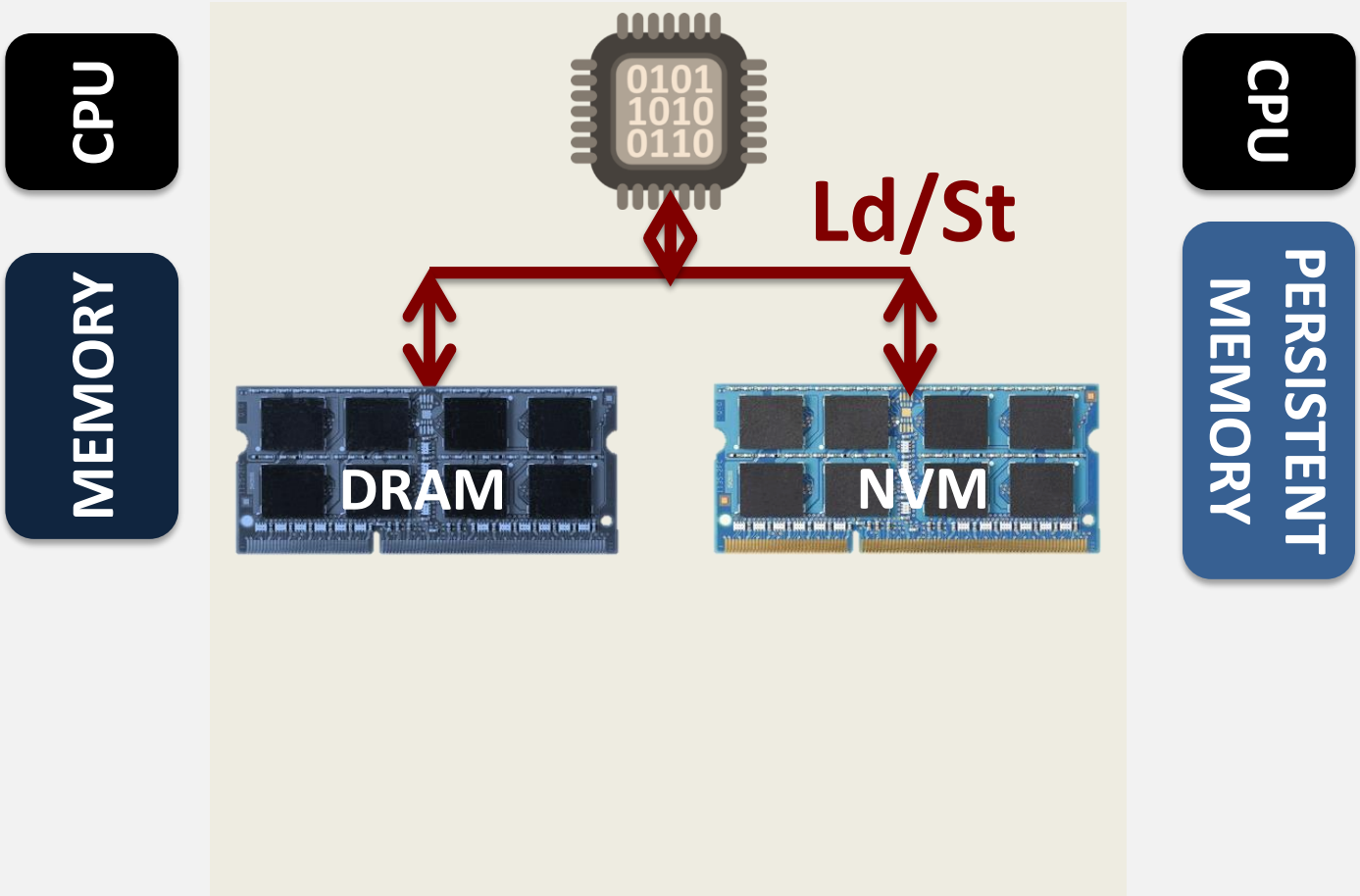
Non-volatile memories combine characteristics of memory and storage

VISION: UNIFY MEMORY AND STORAGE



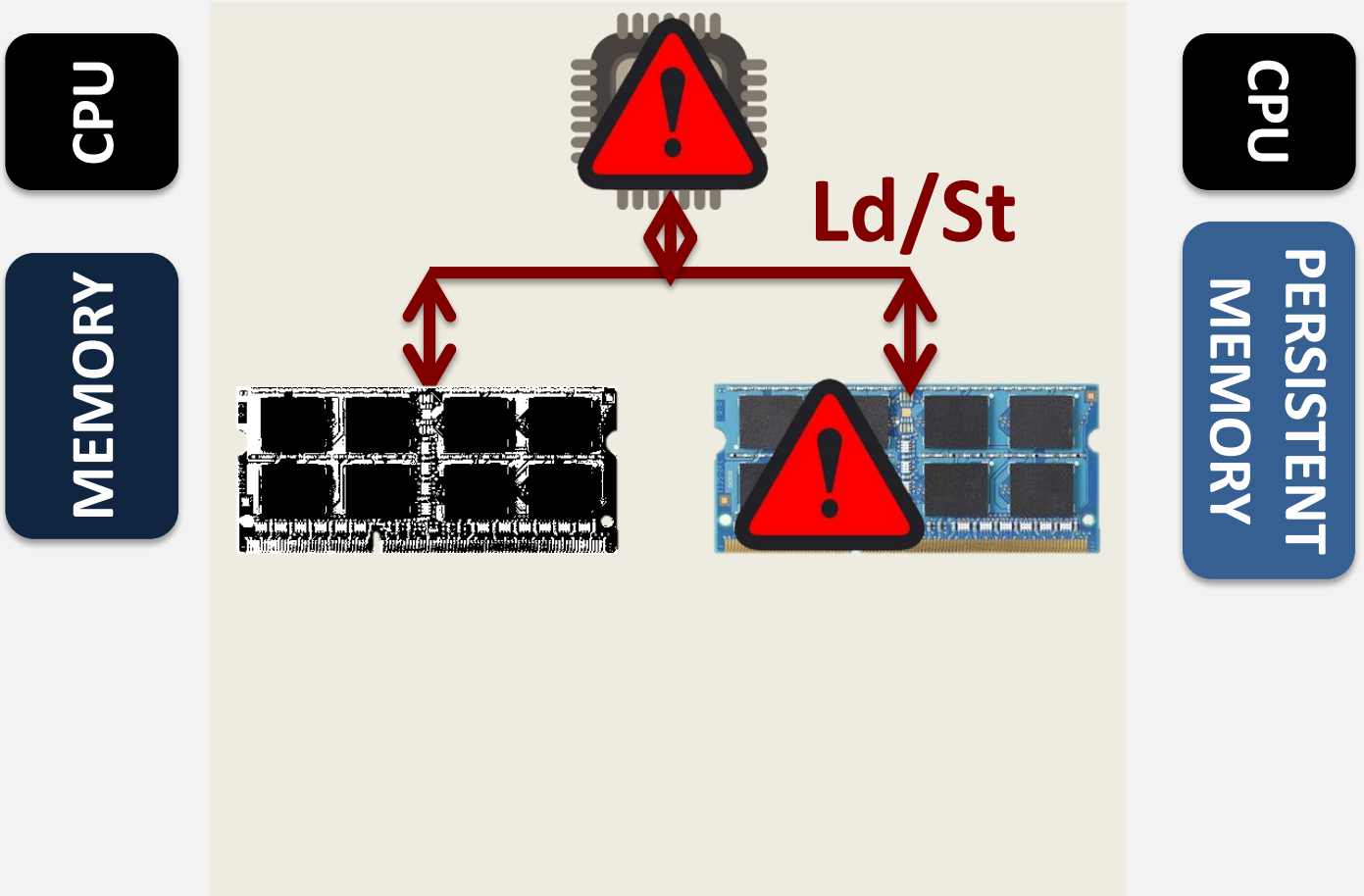
Provides an opportunity to manipulate persistent data directly

DRAM IS STILL FASTER



A hybrid unified memory-storage system

CHALLENGE: DATA CONSISTENCY



System crash can result in permanent data corruption in NVM

CURRENT SOLUTIONS

Explicit interfaces to manage consistency

– **NV-Heaps** [ASPLOS'11], **BPFS** [SOSP'09], **Mnemosyne** [ASPLOS'11]

```
AtomicBegin {
Insert a new node;
} AtomicEnd;
```

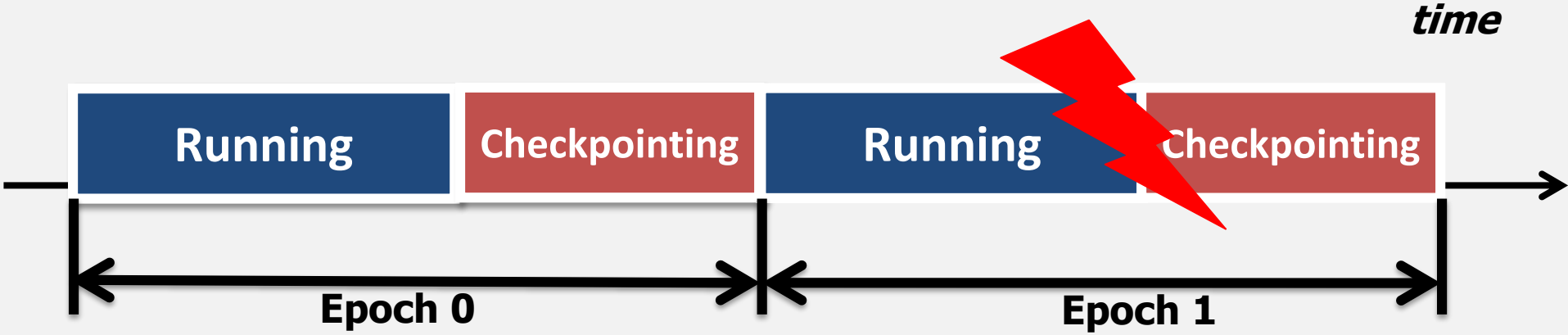
Limits adoption of NVM

Have to rewrite code with clear partition between volatile and non-volatile

Burden on the programmers

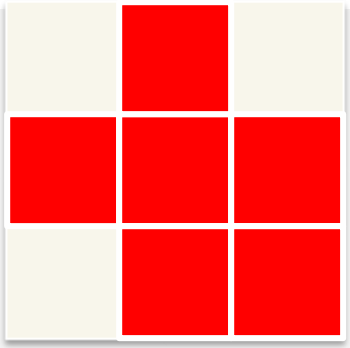
OUR GOAL AND APPROACH

Goal:
Provide efficient transparent consistency in hybrid systems



Periodic Checkpointing of dirty data
Transparent to application and system
Hardware checkpoints and recovers data

CHECKPOINTING GRANULARITY



PAGE

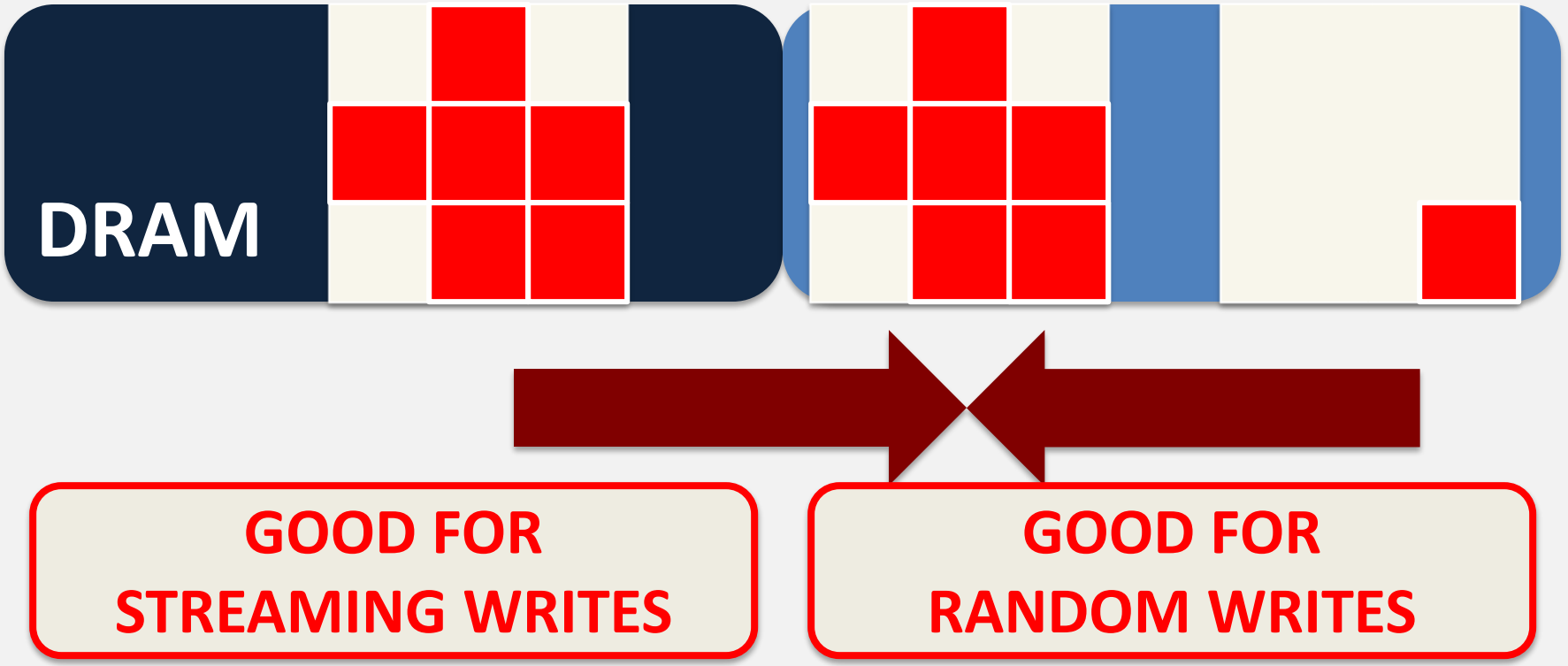
DIRTY CACHE BLOCK

EXTRA WRITES
SMALL METADATA

NO EXTRA WRITE
HUGE METADATA

**High write locality pages in DRAM,
low write locality pages in NVM**

DUAL GRANULARITY CHECKPOINTING



Can adapt to access patterns

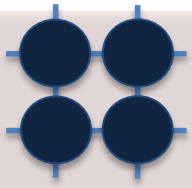
TRANSPARENT DATA CONSISTENCY

Cost of consistency compared to systems with zero-cost consistency

UNMODIFIED LEGACY CODE

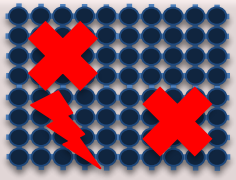
DRAM	NVM
-3.5%	+2.7%

Provides consistency without significant performance overhead



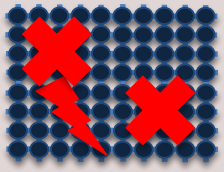
DRAM Cells

Technology Scaling



DRAM Cells

DRAM SCALING CHALLENGE



DRAM Cells

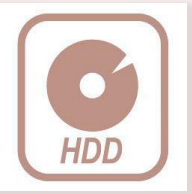
Detect and Mitigate

Reliable System

SYSTEM-LEVEL TECHNIQUES TO ENABLE DRAM SCALING

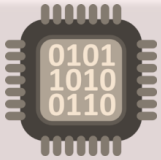
Non-Volatile Memory

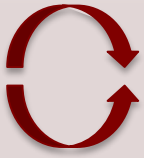
UNIFY

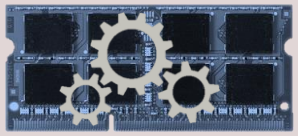


Storage

NON-VOLATILE MEMORIES: UNIFIED MEMORY & STORAGE

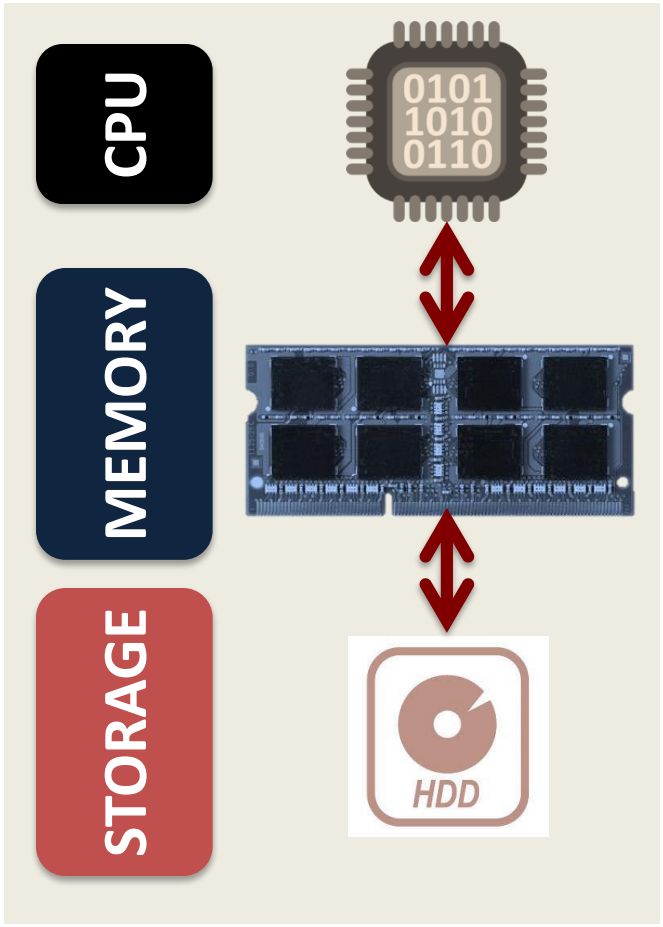






PAST AND FUTURE WORK

OTHER WORK



IMPROVING CACHE PERFORMANCE

MICRO'10, PACT'10, ISCA'12, HPCA'12, HPCA'14

EFFICIENT LOW VOLTAGE PROCESSOR OPERATION

HPCA'13, INTEL TECH JOURNAL'14

NEW DRAM ARCHITECTURE

HPCA'15, ONGOING

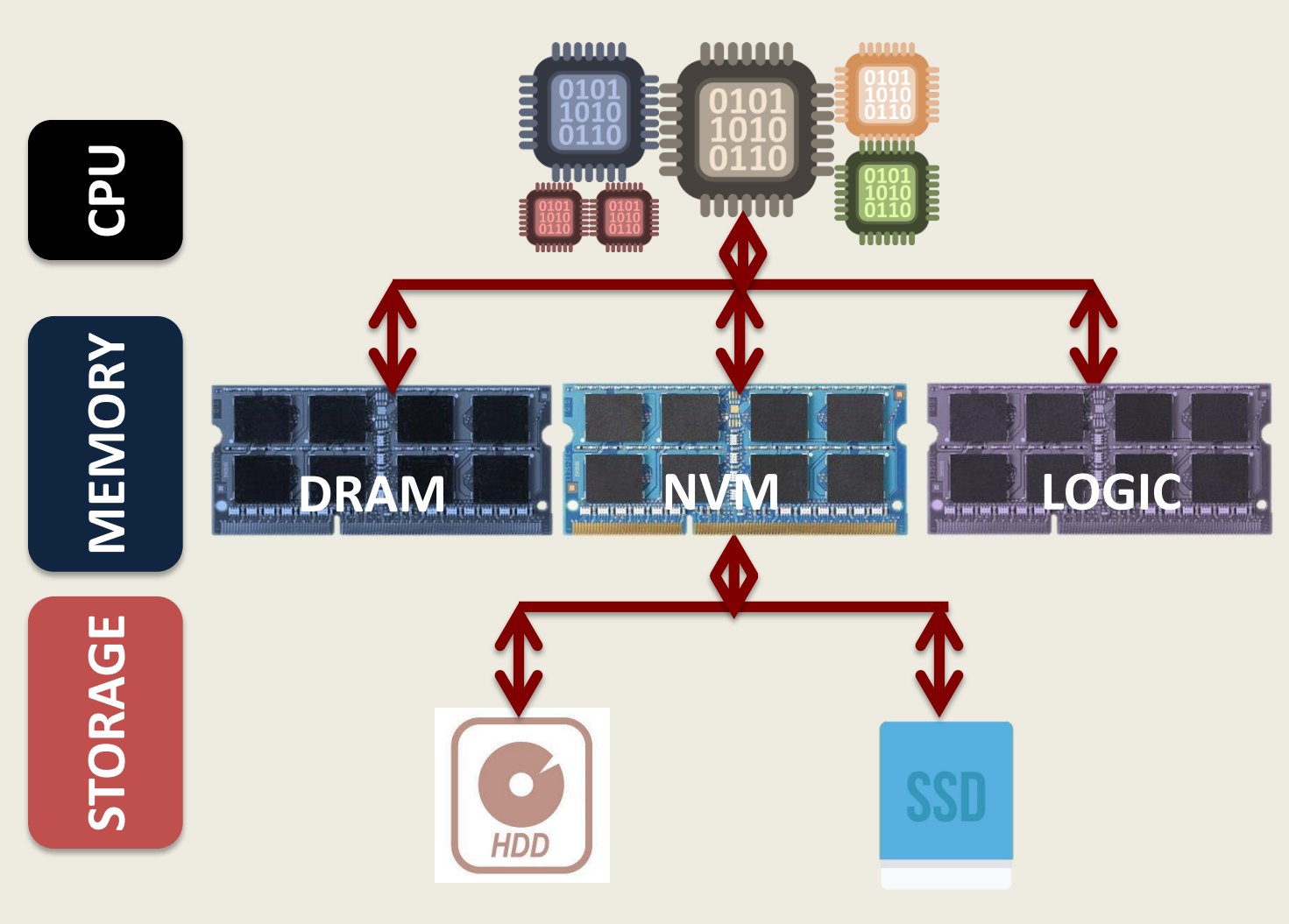
ENABLING DRAM SCALING

SIGMETRICS'14, DSN'15, ONGOING

UNIFYING MEMORY & STORAGE

WEED'13, ONGOING

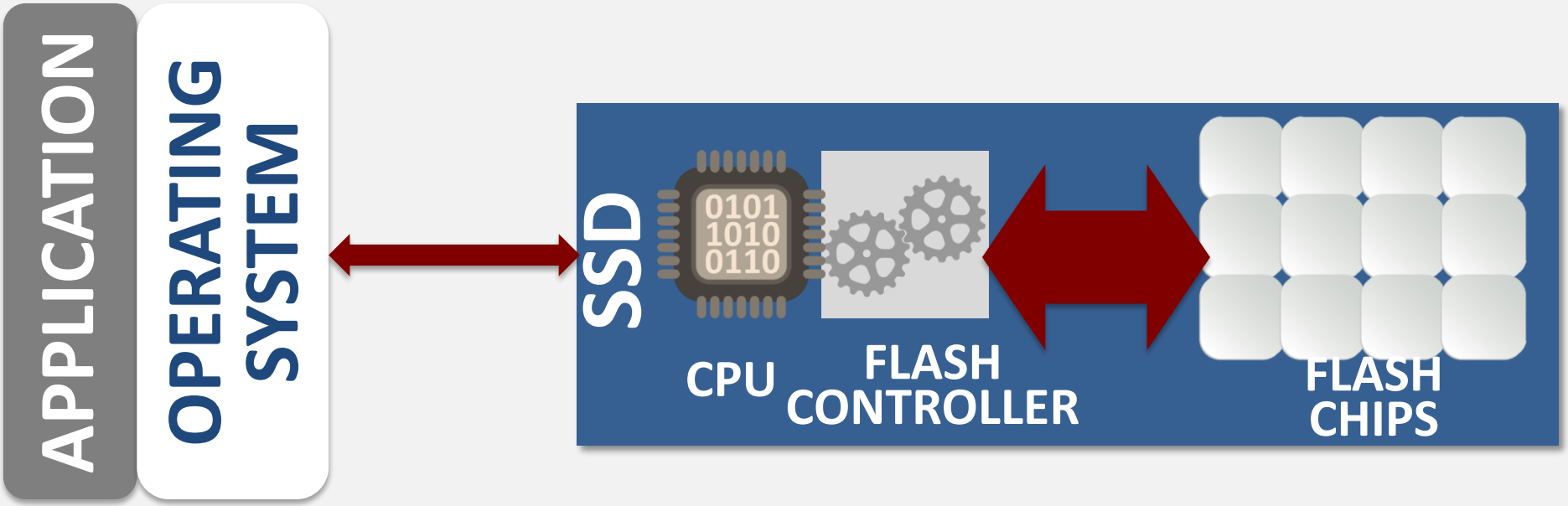
FUTURE WORK: TRENDS



APPROACH

DESIGN AND BUILD BETTER SYSTEMS
WITH NEW CAPABILITIES
BY REDEFINING FUNCTIONALITIES
ACROSS DIFFERENT LAYERS
IN THE SYSTEM STACK

RETHINKING STORAGE



APPLICATION, OPERATING SYSTEM, DEVICES

What is the best way to design a system to take advantage of the SSDs?

ENHANCING SYSTEMS WITH NON-VOLATILE MEMORY

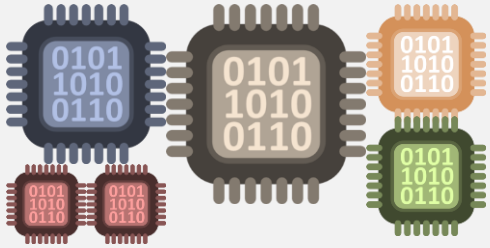
Recover and migrate

NVM

PROCESSOR, FILE SYSTEM, DEVICE, NETWORK, MEMORY

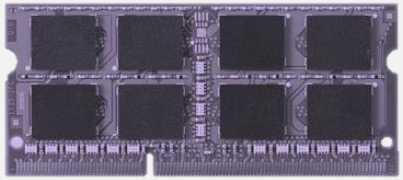
How to provide efficient instantaneous system recovery and migration?

DESIGNING SYSTEMS WITH MEMORY AS AN ACCELERATOR



SPECIALIZED CORES

MANAGE DATA FLOW



MEMORY WITH LOGIC

APPLICATION, PROCESSOR, MEMORY

How to manage data movement when applications run on different accelerators?

THANK YOU

QUESTIONS?

SOLVING THE DRAM SCALING CHALLENGE:

RETHINKING THE INTERFACE BETWEEN CIRCUITS, ARCHITECTURE, AND SYSTEMS

Samira Khan

Carnegie Mellon