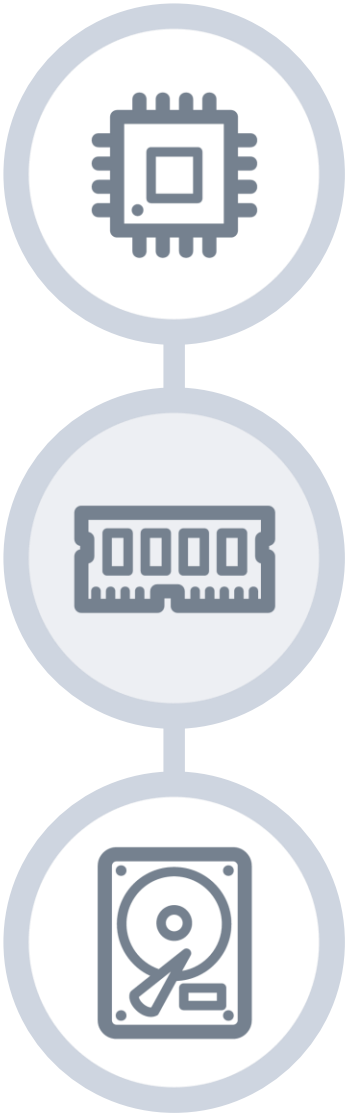
ARCHITECTURAL TECHNIQUES TO ENHANCE DRAM SCALING

Thesis Defense

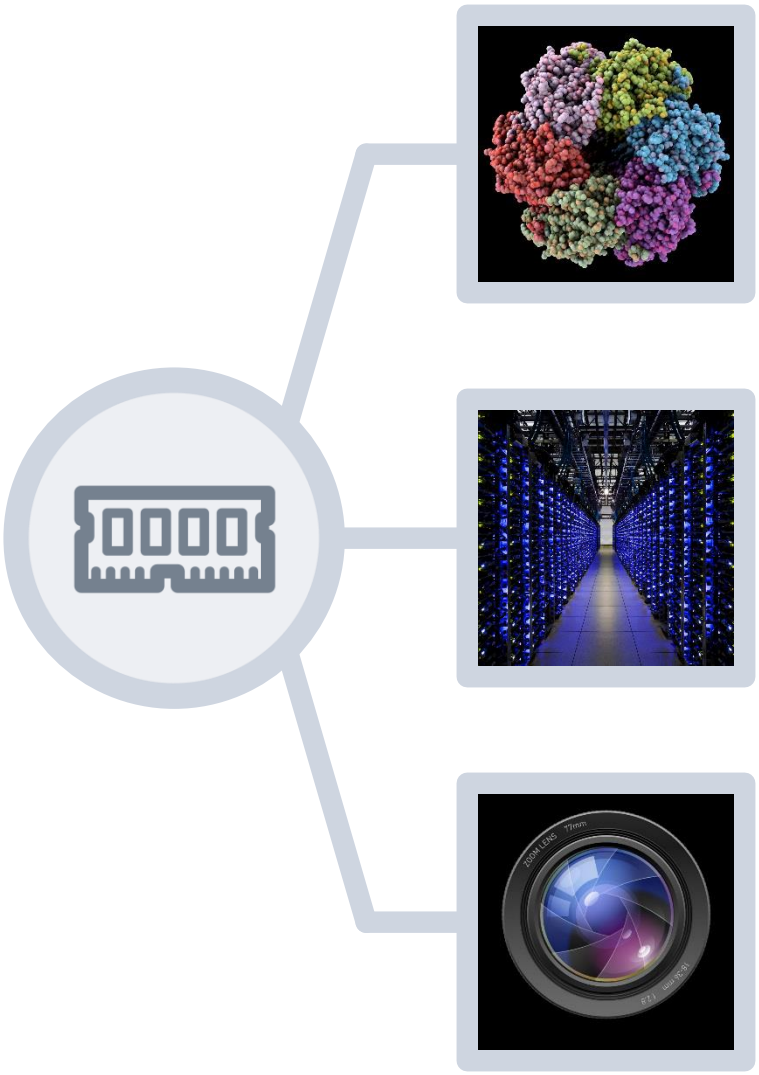
Yoongu Kim



CPU+CACHE

MAIN MEMORY

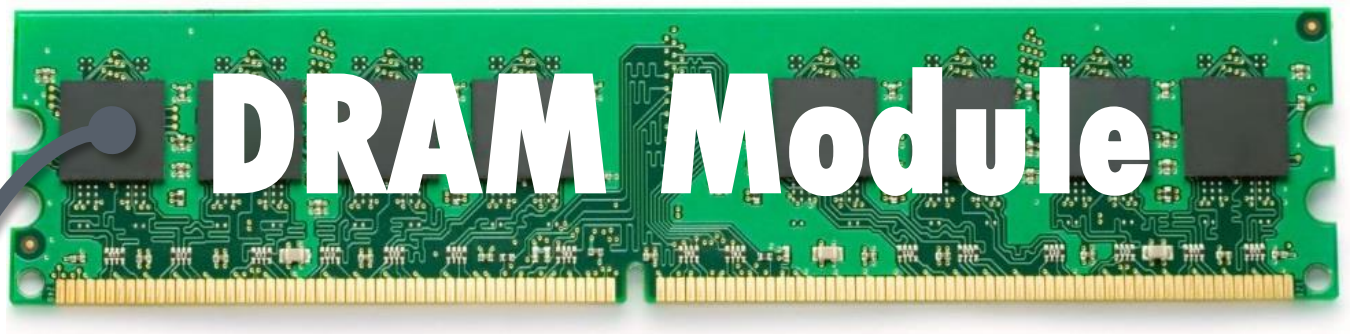
STORAGE



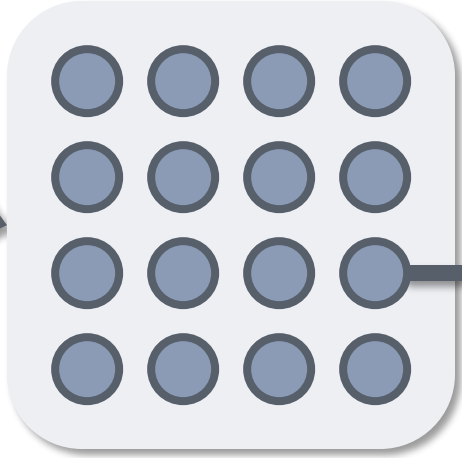
Complex Problems

Large Datasets

High Throughput



DRAM Chip



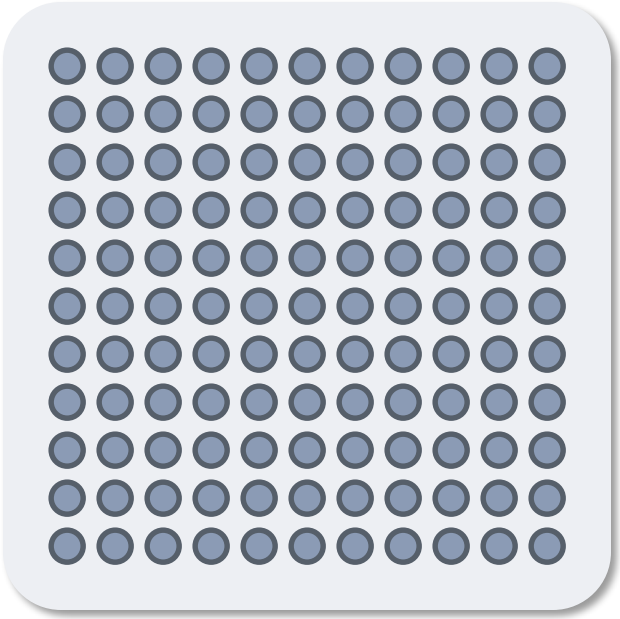
‘1’ ‘0’

DRAM Cell
(Capacitor)

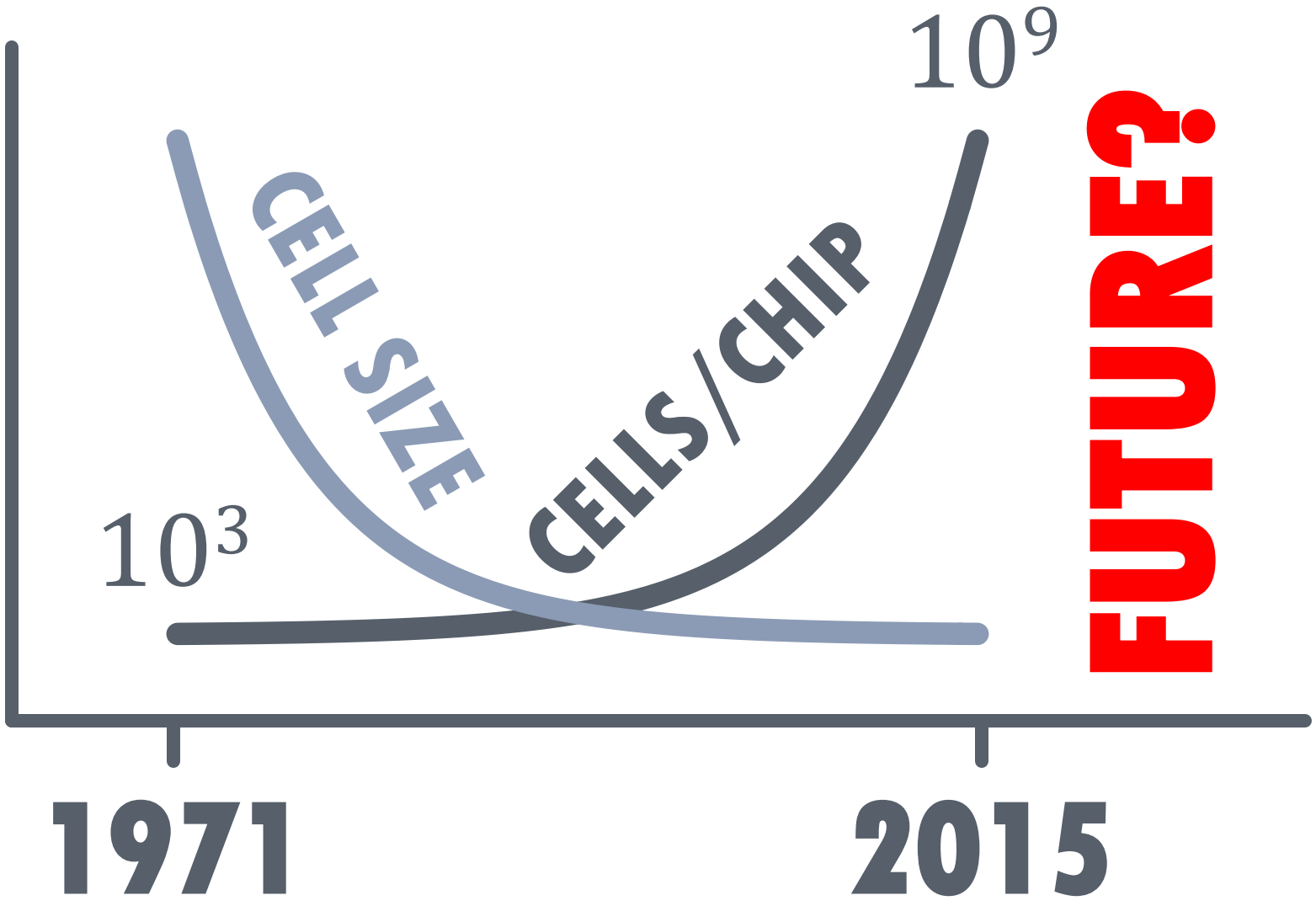
1971

FIRST DRAM CHIP

2015



10^9 Cells


10^9
CELL SIZE
CELLS/CHIP
10^3
FUTURE?
1971
2015

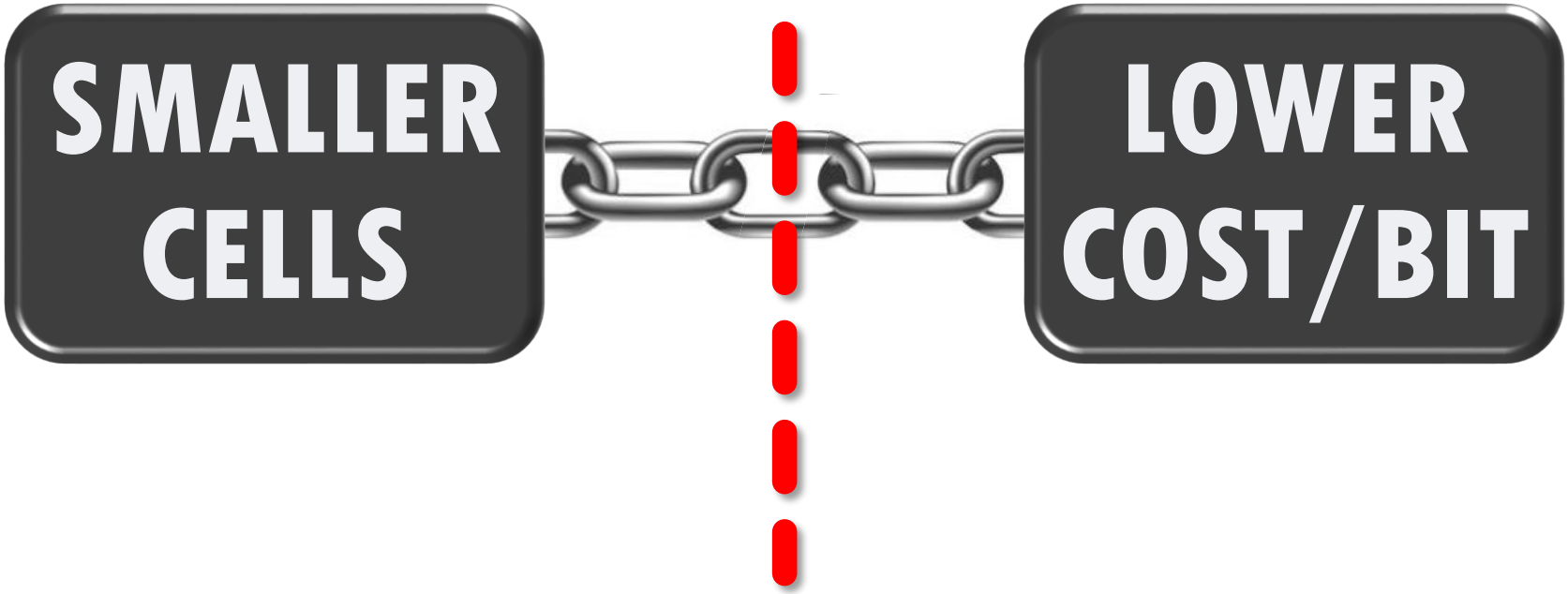
DRAM SCALING

TECHNOLOGICAL FEASIBILITY

Can we make smaller cells?

ECONOMIC VIABILITY

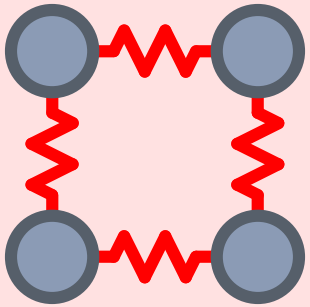
Should we make smaller cells?



1. RELIABILITY TAX

2. PERFORMANCE TAX

1. RELIABILITY

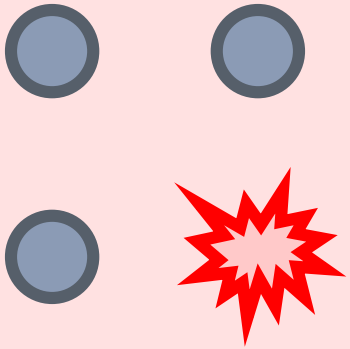


COUPLING BETWEEN NEARBY CELLS

ROW HAMMER (ISCA 2014)

Your DRAM chips are probably broken.

2. PERFORMANCE



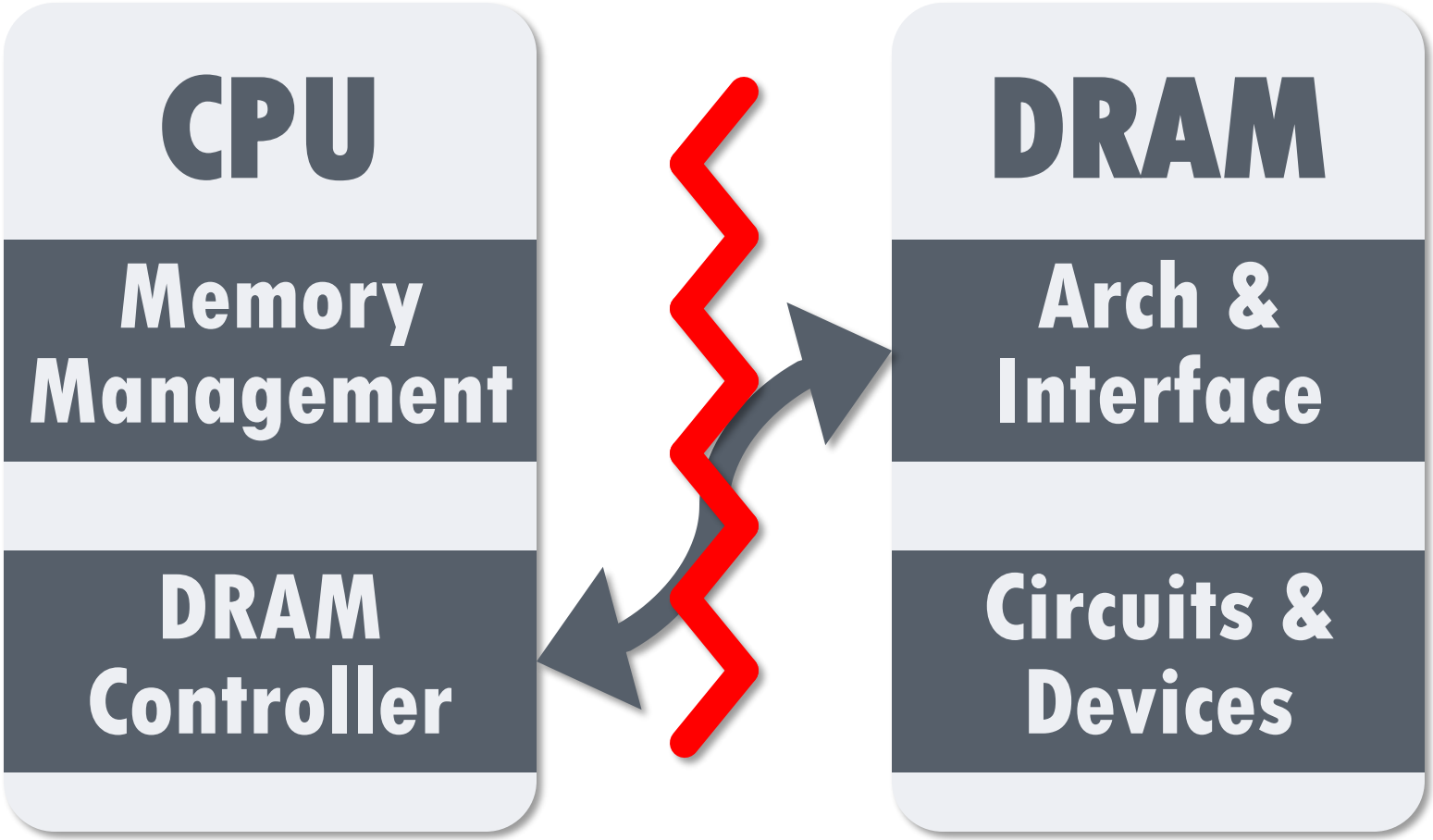
ABNORMALLY SLOW

OUTLIER CELLS

BANK CONFLICTS (ISCA 2012)

Our solution may be adopted by industry.

CO-DESIGN: CPU & DRAM



THESIS STATEMENT

The degradation in DRAM reliability & performance can be effectively mitigated by making low-overhead, non-intrusive modifications to the DRAM chips and the DRAM controller.

MAIN MEMORY:
LARGER, FASTER, RELIABLE, EFFICIENT

ARCHITECTURAL SUPPORT

DRAM SCALING

THREE CONTRIBUTIONS

1. We show that DRAM scaling is negatively affecting reliability.

We expose a new type of DRAM failure, and propose a cost-effective way to address it.

THREE CONTRIBUTIONS

2. We propose a high-performance architecture for DRAM that mitigates its growing latency.

We identify bottlenecks in DRAM's internal design and alleviate them in a cost-effective manner.

THREE CONTRIBUTIONS

3. We develop a new simulator for facilitating rapid design space exploration of DRAM.

The simulator is the fastest, while also being easy to modify due to its modular design.

OUTLINE

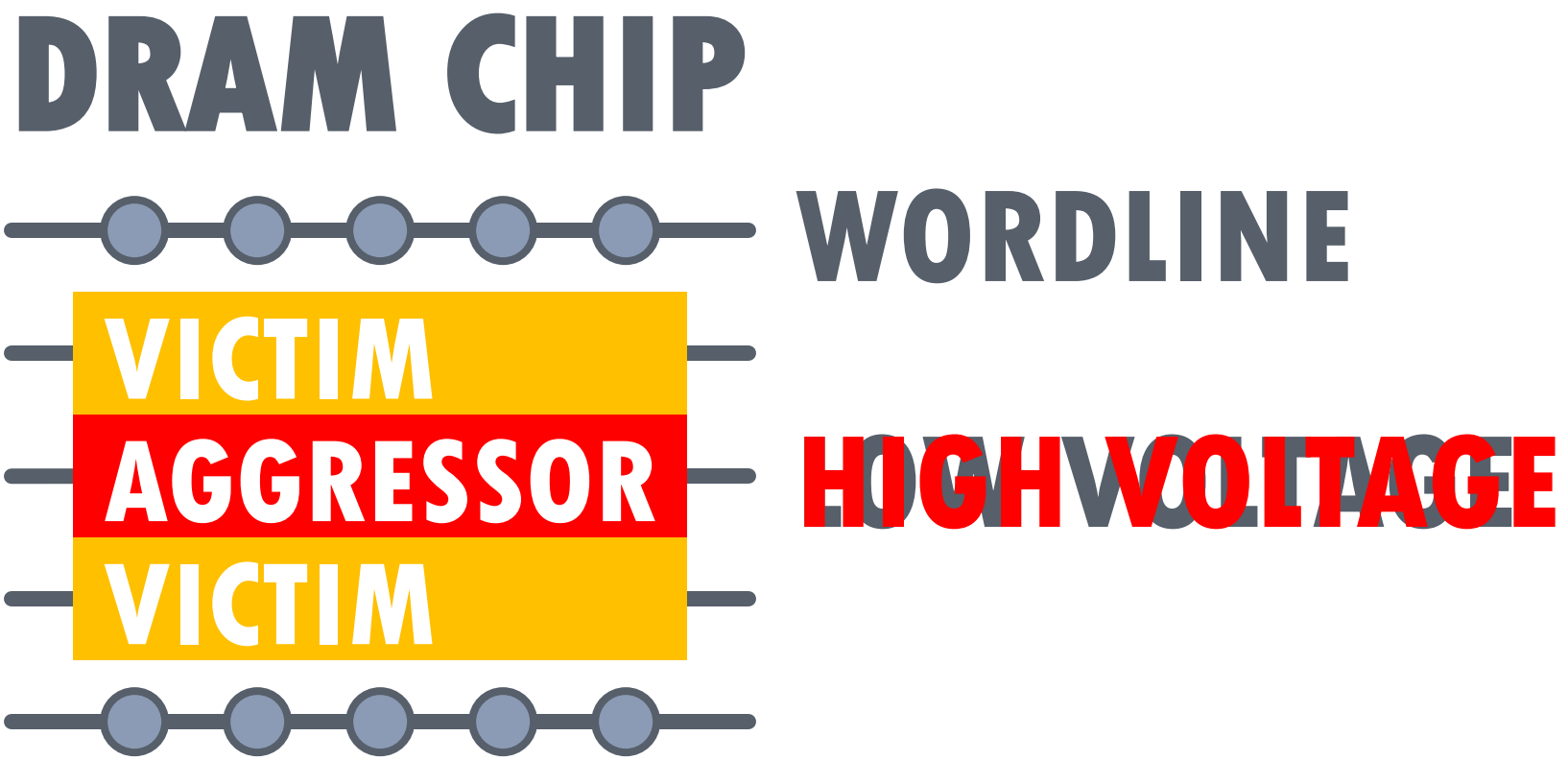
1. RELIABILTY: ROW HAMMER
2. PERF: BANK CONFLICT
3. SIMULATOR: RAMULATOR
4. CONCLUSION

FLIPPING BITS IN MEMORY WITHOUT ACCESSING THEM

ISCA 2014

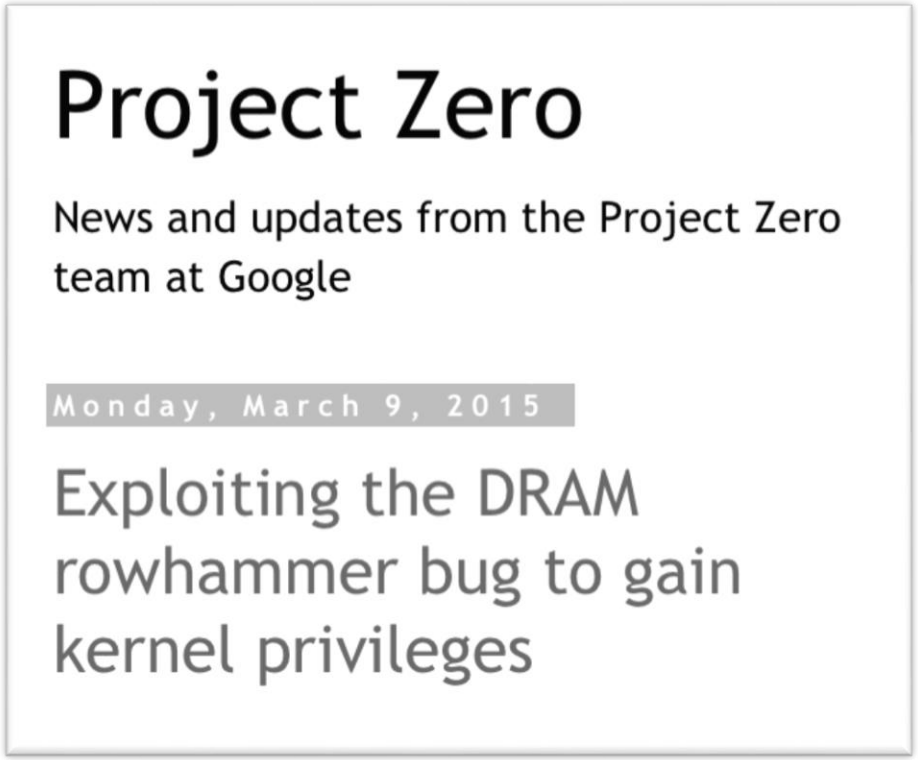

Flipping Bits in Memory Without Accessing Them:
An Experimental Study of DRAM Disturbance Errors

1. ROW HAMMER



READ DATA FROM HERE,
GET ERRORS OVER THERE

GOOGLE'S EXPLOIT


Project Zero

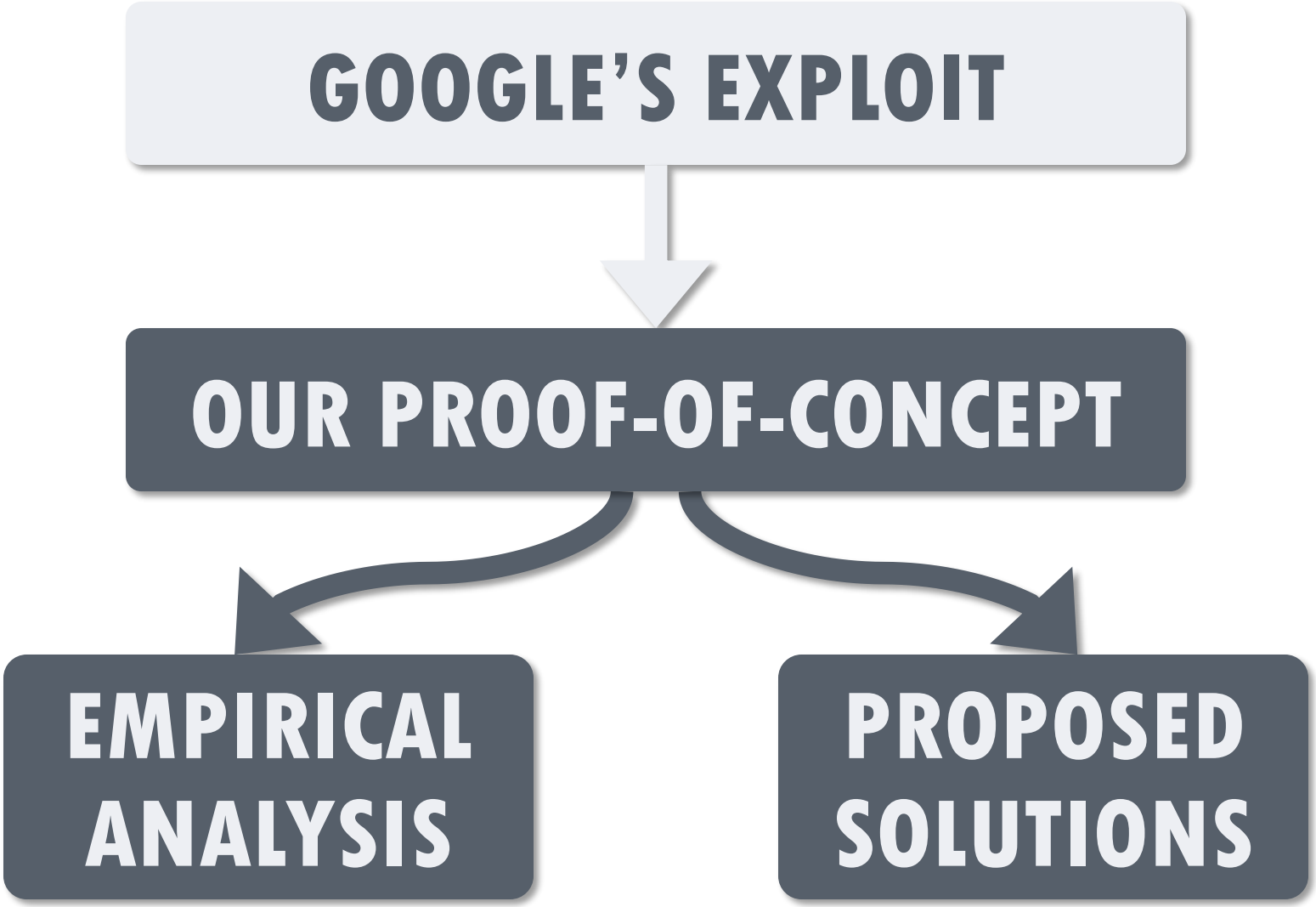
News and updates from the Project Zero team at Google

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

"We learned about rowhammer from Yoongu Kim et al."

http://googleprojectzero.blogspot.com

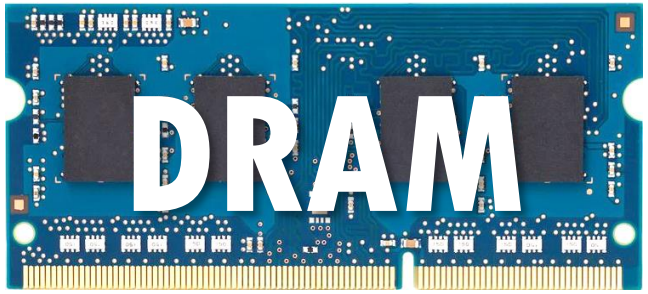

GOOGLE'S EXPLOIT
OUR PROOF-OF-CONCEPT
EMPIRICAL ANALYSIS
PROPOSED SOLUTIONS

REAL SYSTEM

MANY READS TO SAME ADDRESS

≠

OPEN/CLOSE SAME ROW



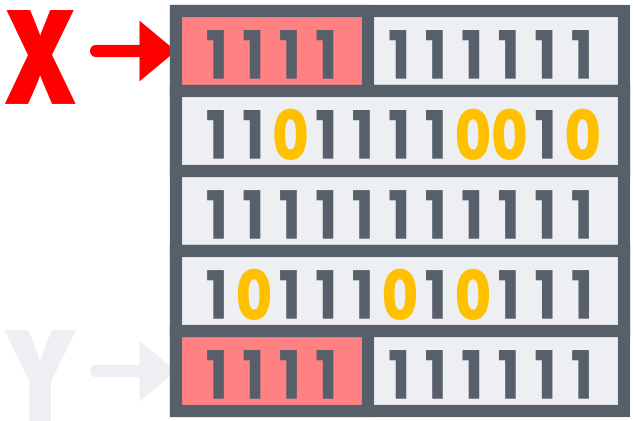
1. CACHE HITS

2. ROW HITS

x86 CPU

```
LOOP:
    mov (X), %reg
    mov (Y), %reg
    clflush (X)
    clflush (Y)
    jmp LOOP
```

DRAM

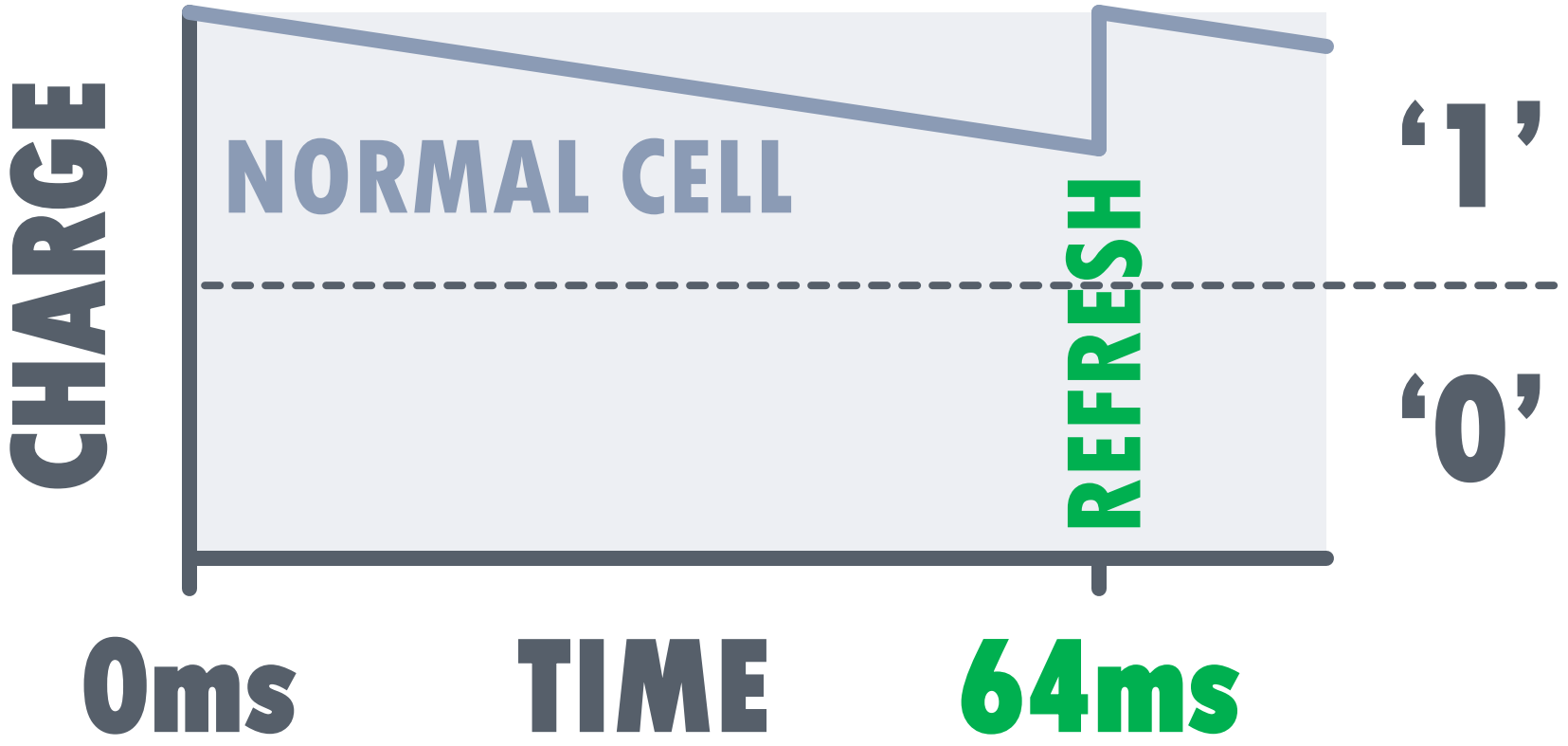


MANY ERRORS!

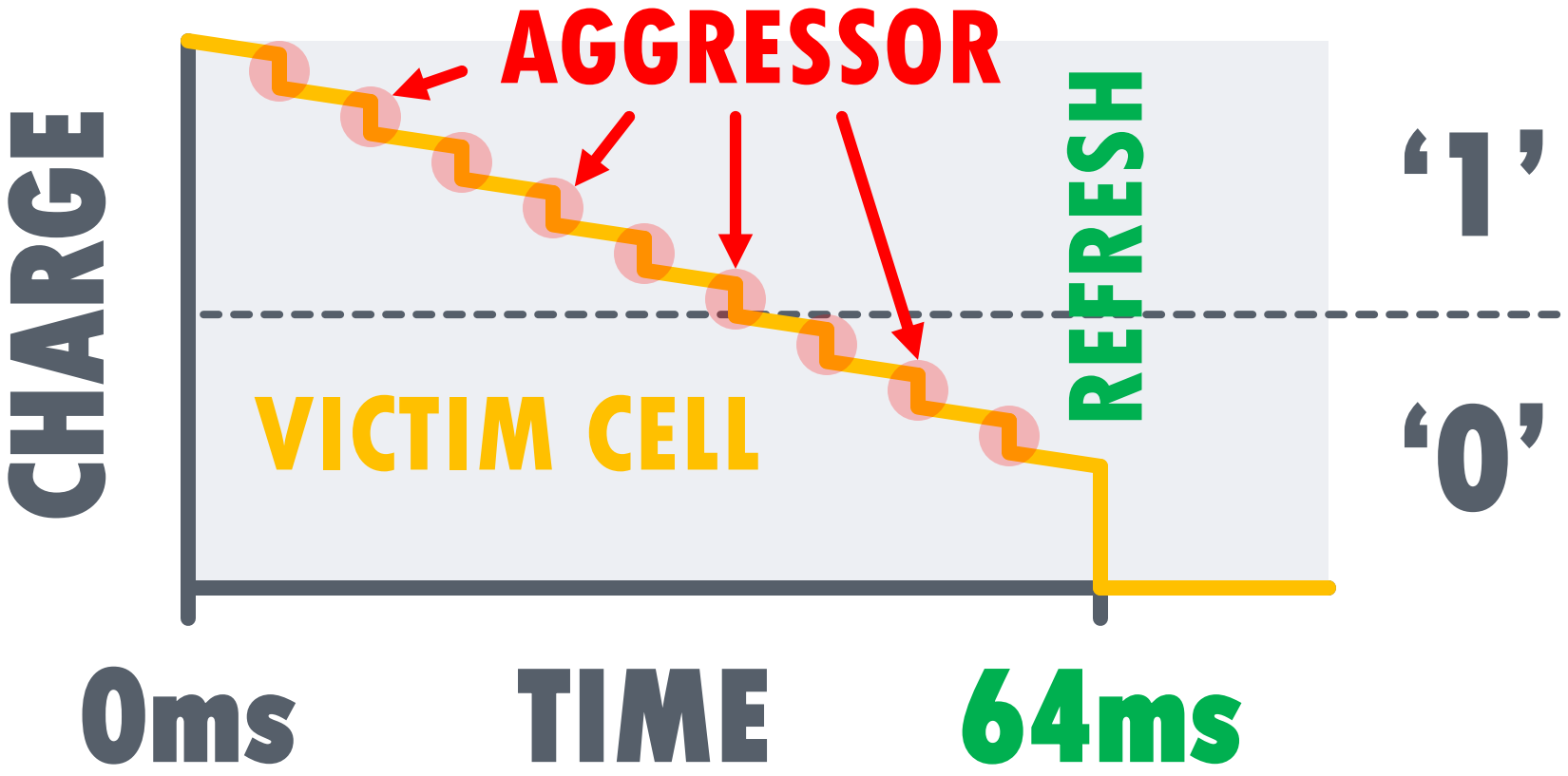
http://www.github.com/CMU-SAFARI/rowhammer

WHY DO THE ERRORS OCCUR?

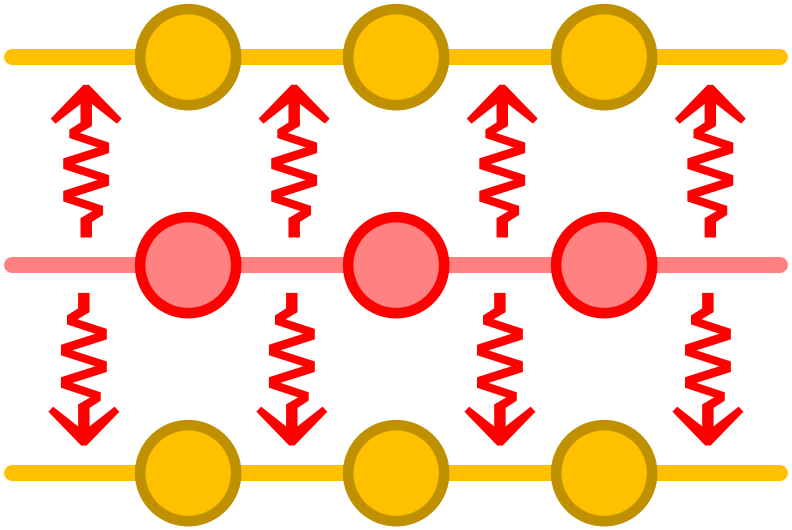
DRAM CELLS ARE LEAKY



DRAM CELLS ARE LEAKY



ROOT CAUSE?



COUPLING

- Electromagnetic
- Tunneling

ACCELERATES CHARGE LOSS

AS DRAM SCALES …

- **CELLS BECOME SMALLER**
  Less tolerance to coupling effects
- **CELLS BECOME PLACED CLOSER**
  Stronger coupling effects

COUPLING ERRORS MORE LIKELY

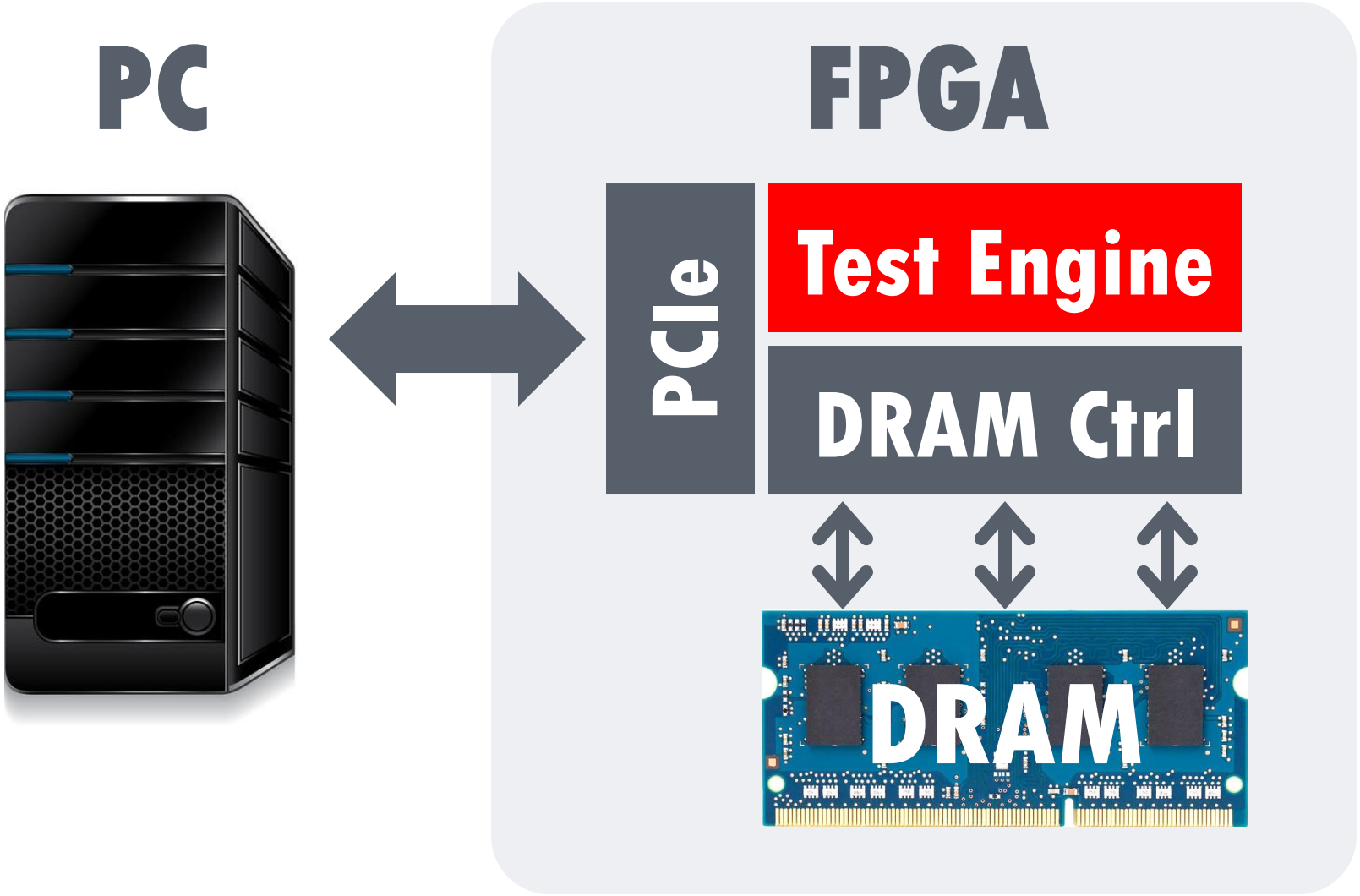
1. ERRORS ARE RECENT

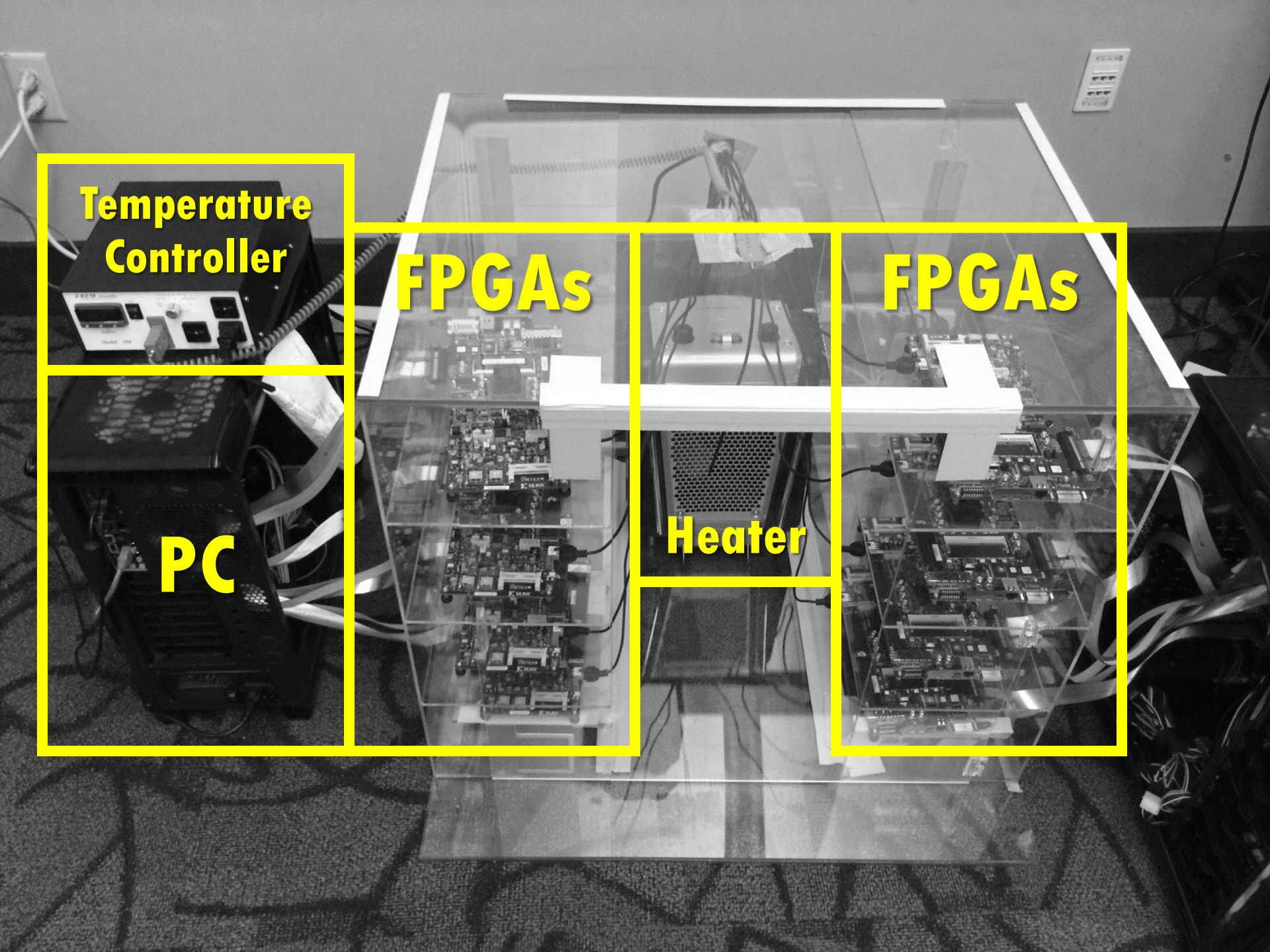
Not found in pre-2010 chips

2. ERRORS ARE WIDESPREAD

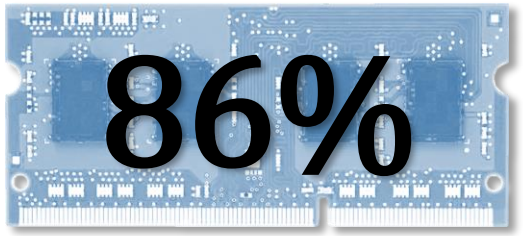
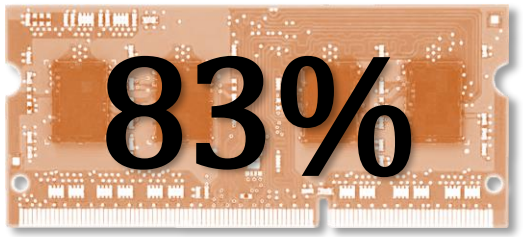
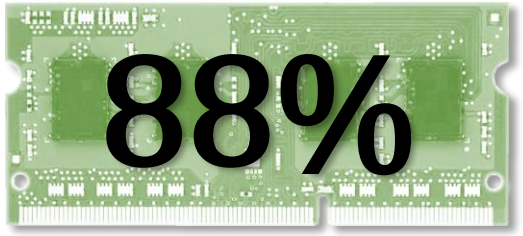
>80% of chips have errors

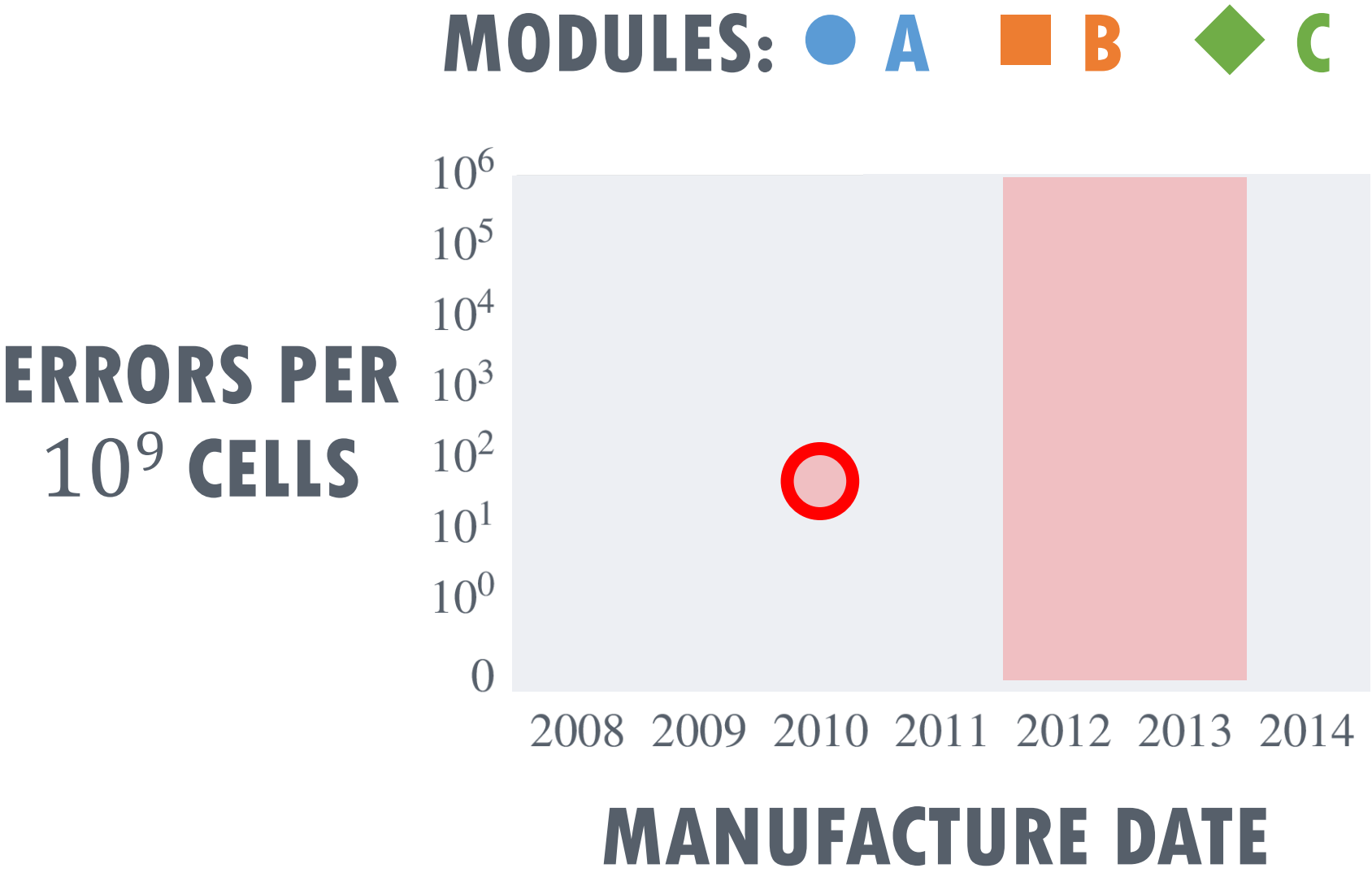
Up to one error per ~1K cells


PC
FPGA
PCIe
Test Engine
DRAM Ctrl
DRAM


Temperature Controller
PC
FPGAs
Heater
FPGAs

MOST MODULES AT RISK

A VENDOR		(37/43)
B VENDOR		(45/54)
C VENDOR		(28/32)


MODULES: A B C
ERRORS PER
10^9 CELLS
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
2008 2009 2010 2011 2012 2013 2014
MANUFACTURE DATE

DISTURBING FACTS

- **AFFECTS ALL VENDORS**
  Not an isolated incident
  Deeper issue in DRAM scaling

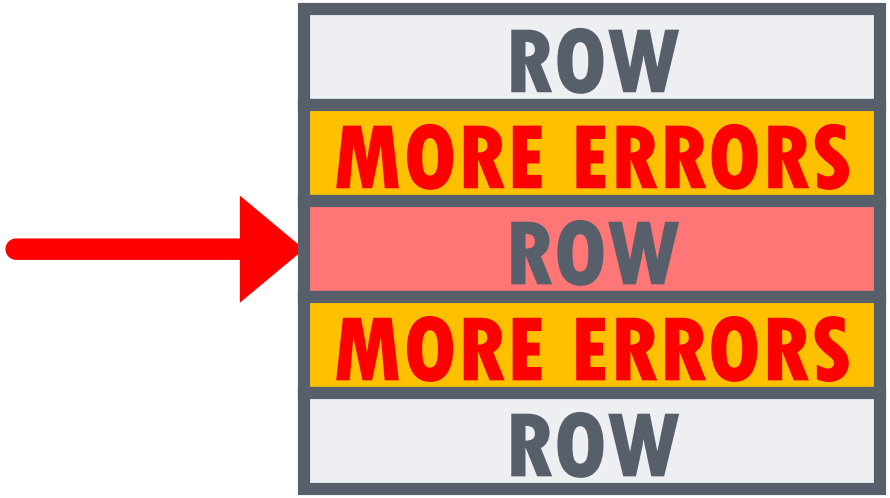
- **UNADDRESSED FOR YEARS**
  Could impact systems in the field

HOW TO PREVENT COUPLING ERRORS?

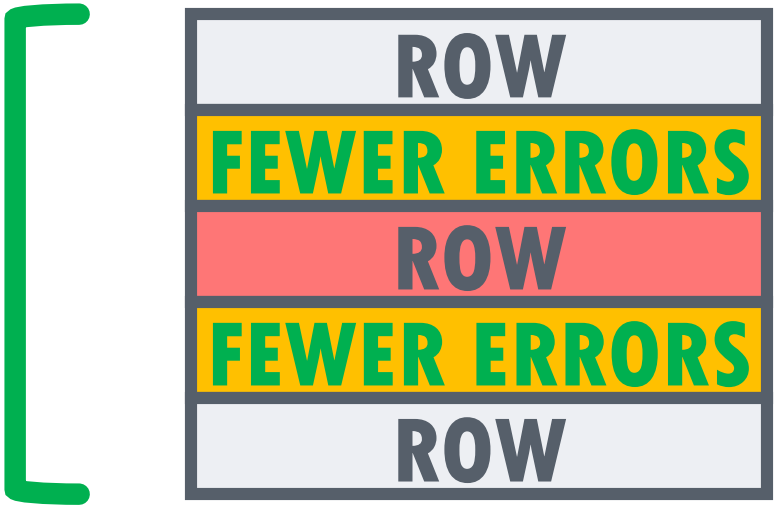
Previous Approaches

1. Make Better Chips: Expensive
2. Rigorous Testing: Takes Too Long

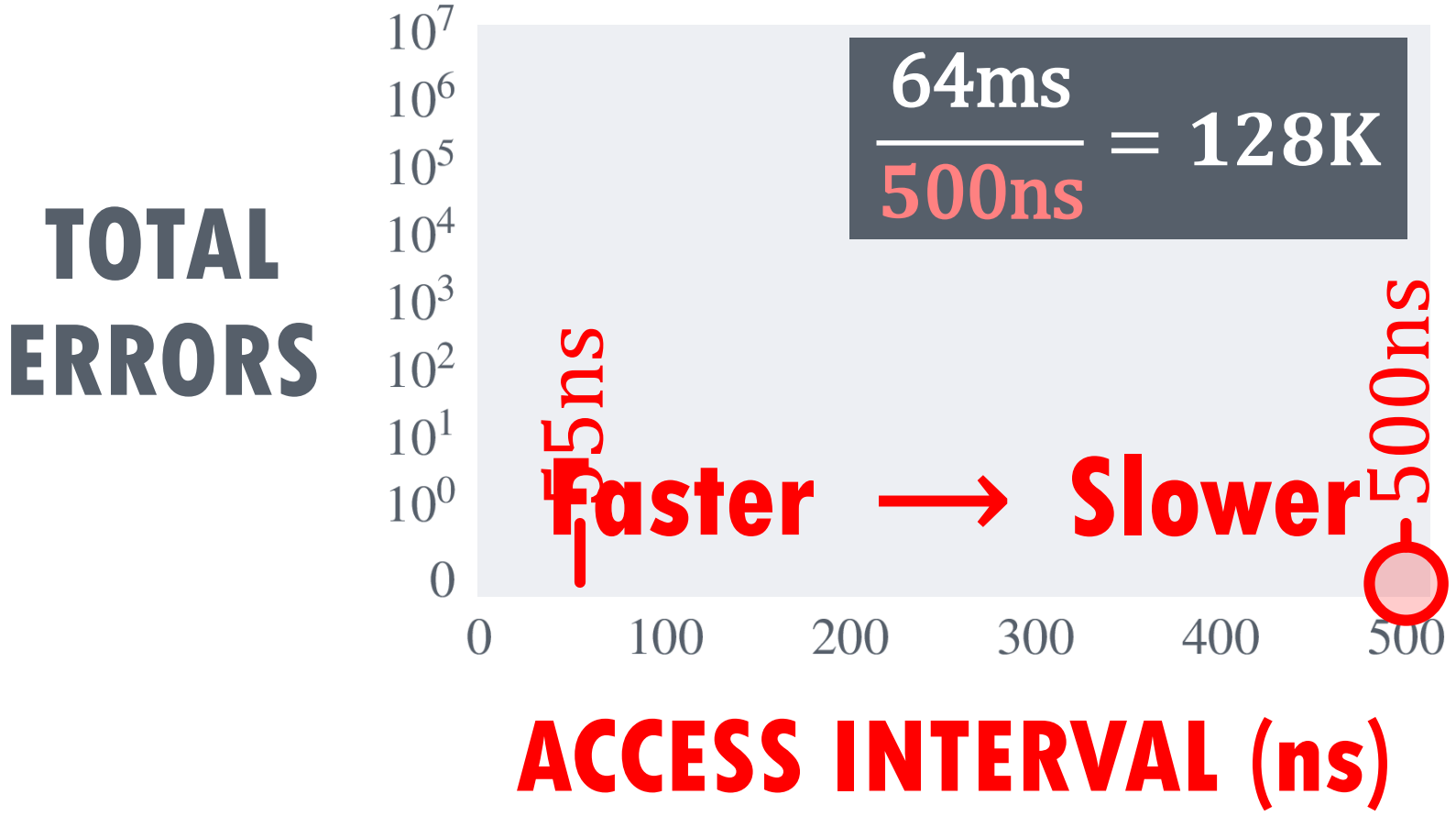
FASTER ACCESS



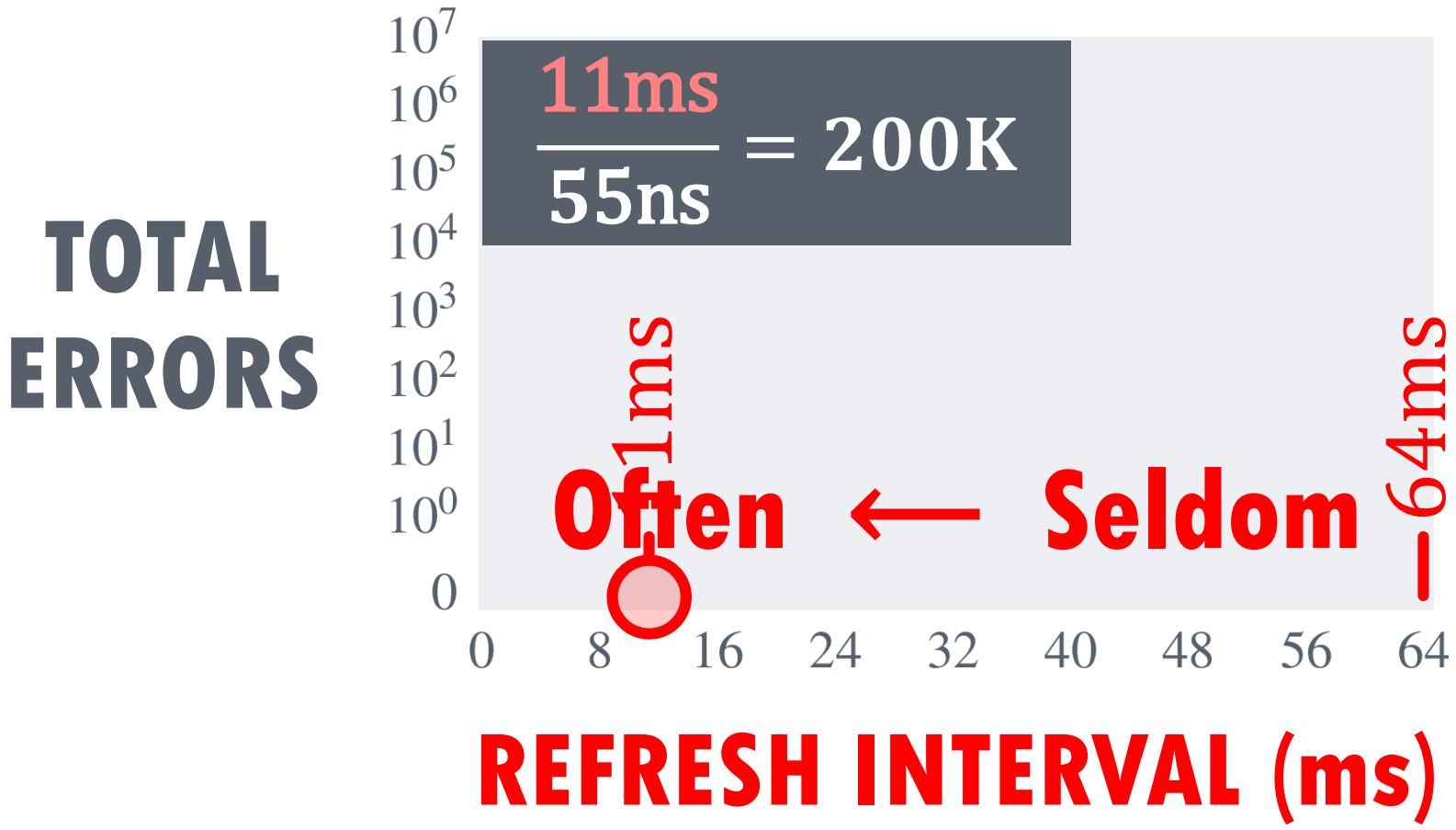
FREQUENT REFRESH



ONE MODULE: ○ A □ B ◇ C



ONE MODULE: ○ A □ B ◇ C



TWO NAIVE SOLUTIONS

1. LIMIT ACCESSES TO ROW

Access Interval $> 500\text{ns}$

2. REFRESH ALL ROWS OFTEN

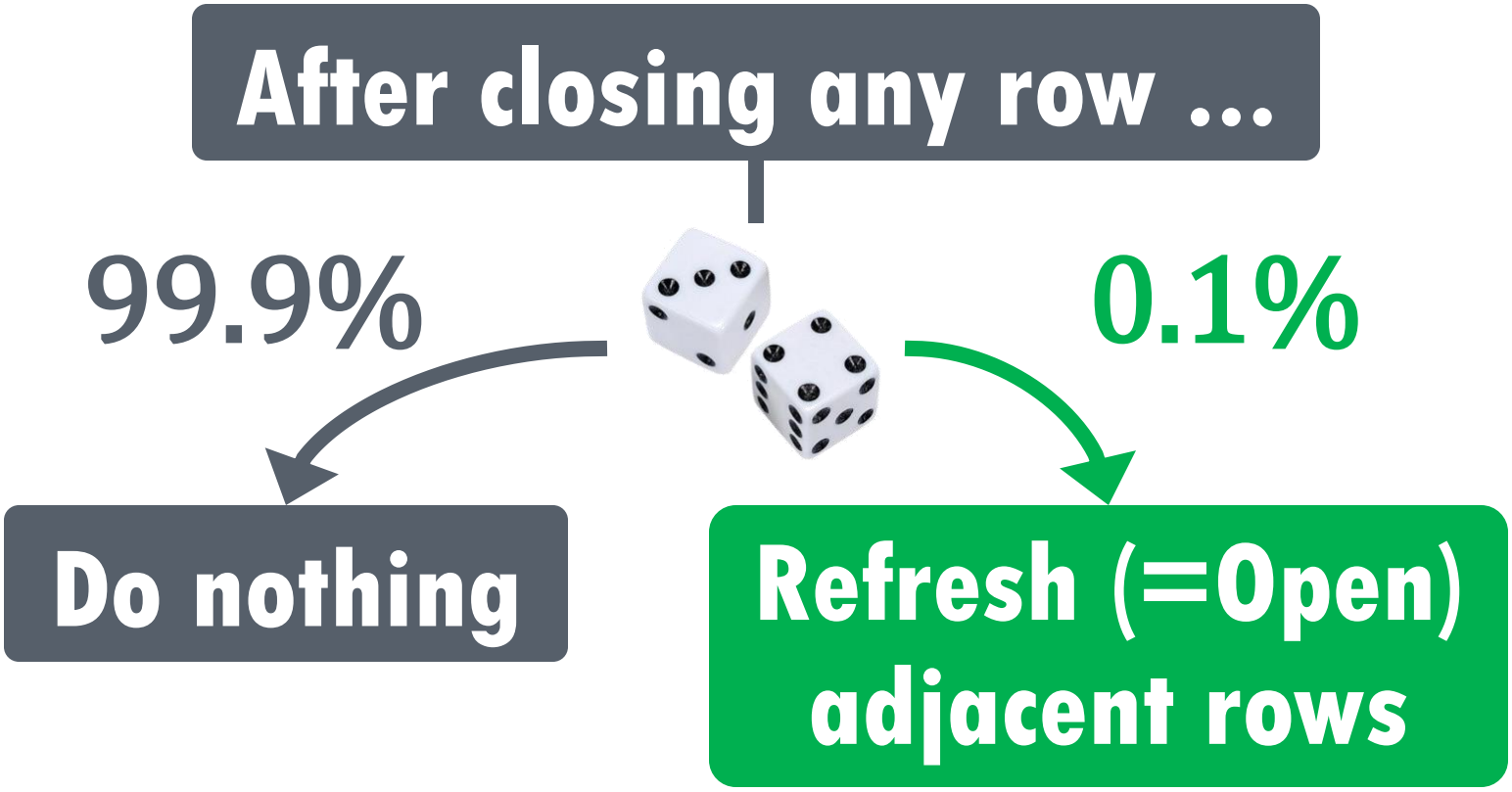
Refresh Interval $< 11\text{ms}$

LARGE OVERHEAD:

PERF, ENERGY, COMPLEXITY

OUR SOLUTION: PARR

Probabilistic Adjacent Row Refresh



PARR: CHANCE OF ERROR

- **NO REFRESHES IN N TRIALS**
  Probability: 0.999^{N}
- **N=128K FOR ERROR (64ms)**
  Probability: $0.999^{128K} = 10^{-56}$

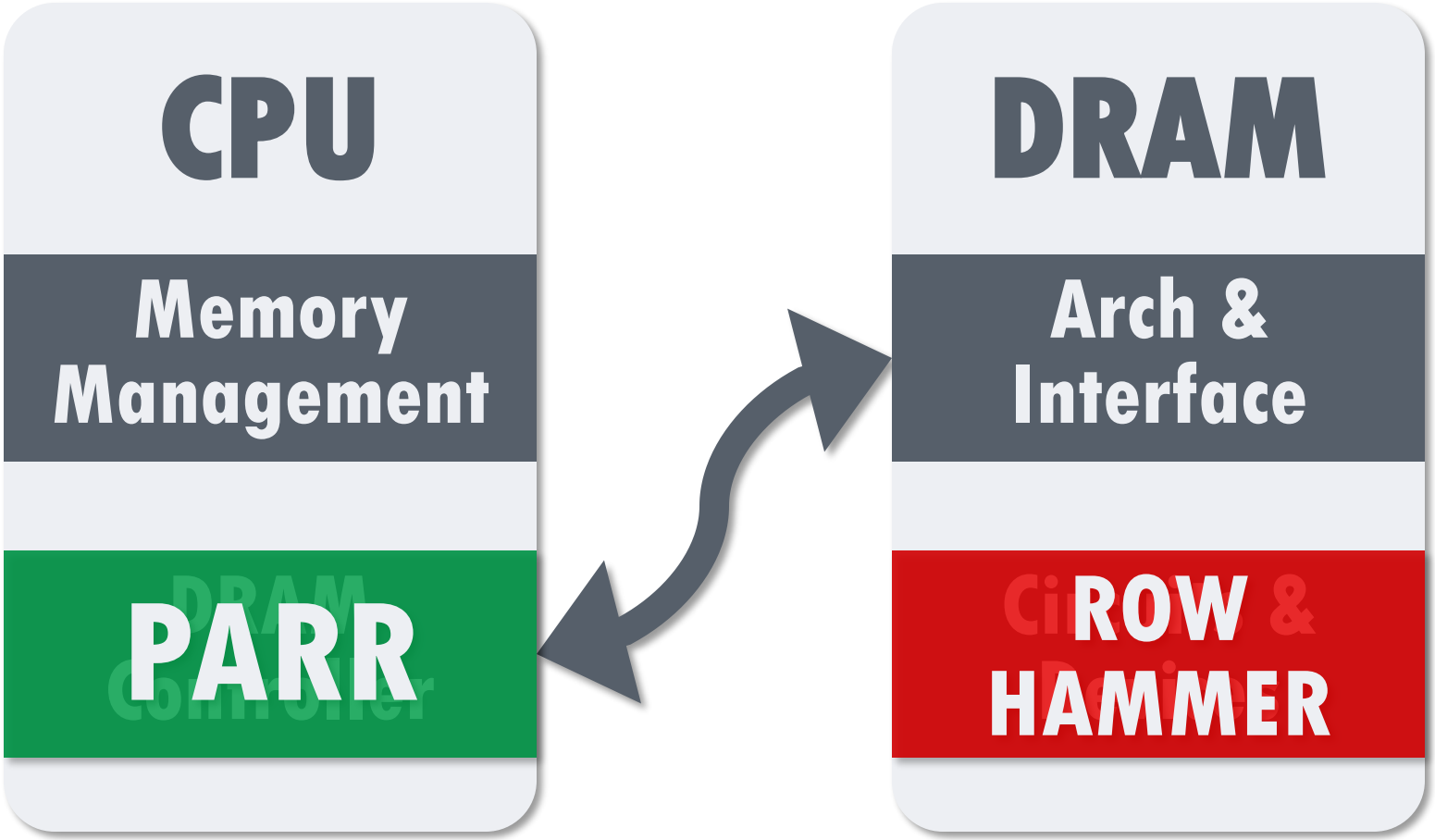
STRONG RELIABILITY GUARANTEE

STRONG RELIABILITY	9.4×10^{-14} Errors/Year
LOW PERF OVERHEAD	0.20% Slowdown
NO STORAGE OVERHEAD	0 Bytes

RELATED WORK

- **Security Exploit** (Seaborn@Google 2015)
- **Industry Analysis** (Kang@SK Hynix 2014)
  "... will be [more] severe as technology shrinks down."
- **Targeted Row Refresh** (JEDEC 2014)
- **DRAM Testing** (e.g., Van de Goor+ 1999)
- **Disturbance in Flash & Hard Disk**

RECAP: RELIABILITY



A CASE FOR SUBARRAY PARALLELISM

ISCA 2012

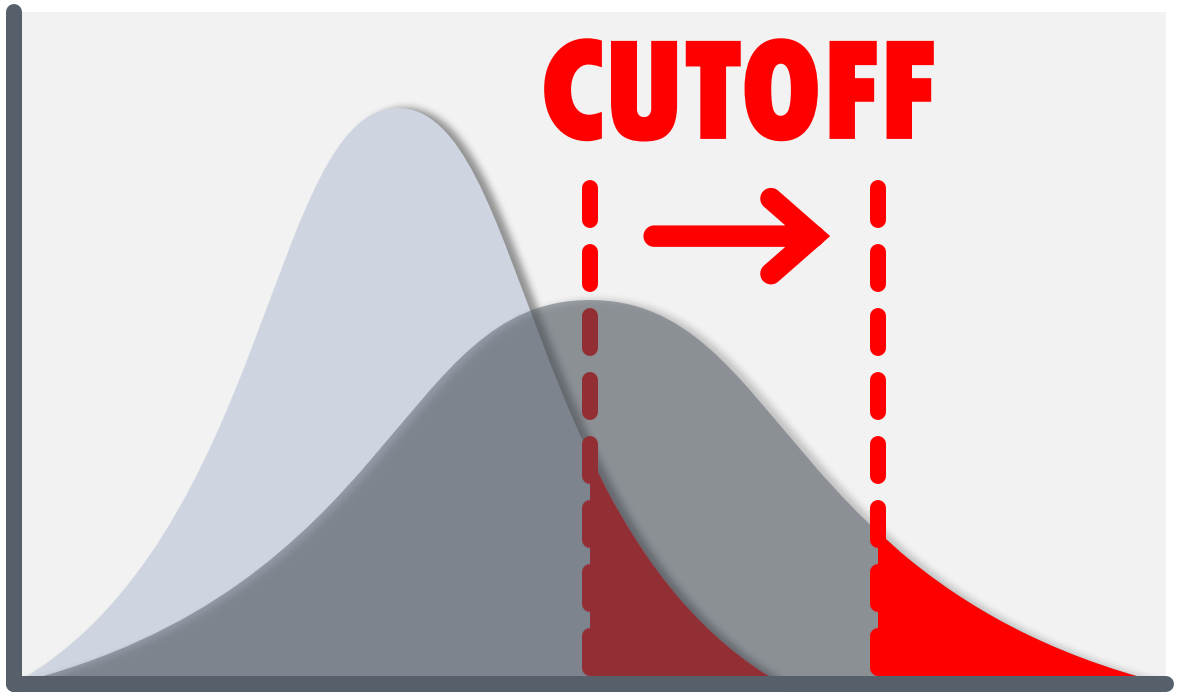

A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM

Carnegie Mellon University

Abstract

2. BANK CONFLICTS

DRAM CELLS

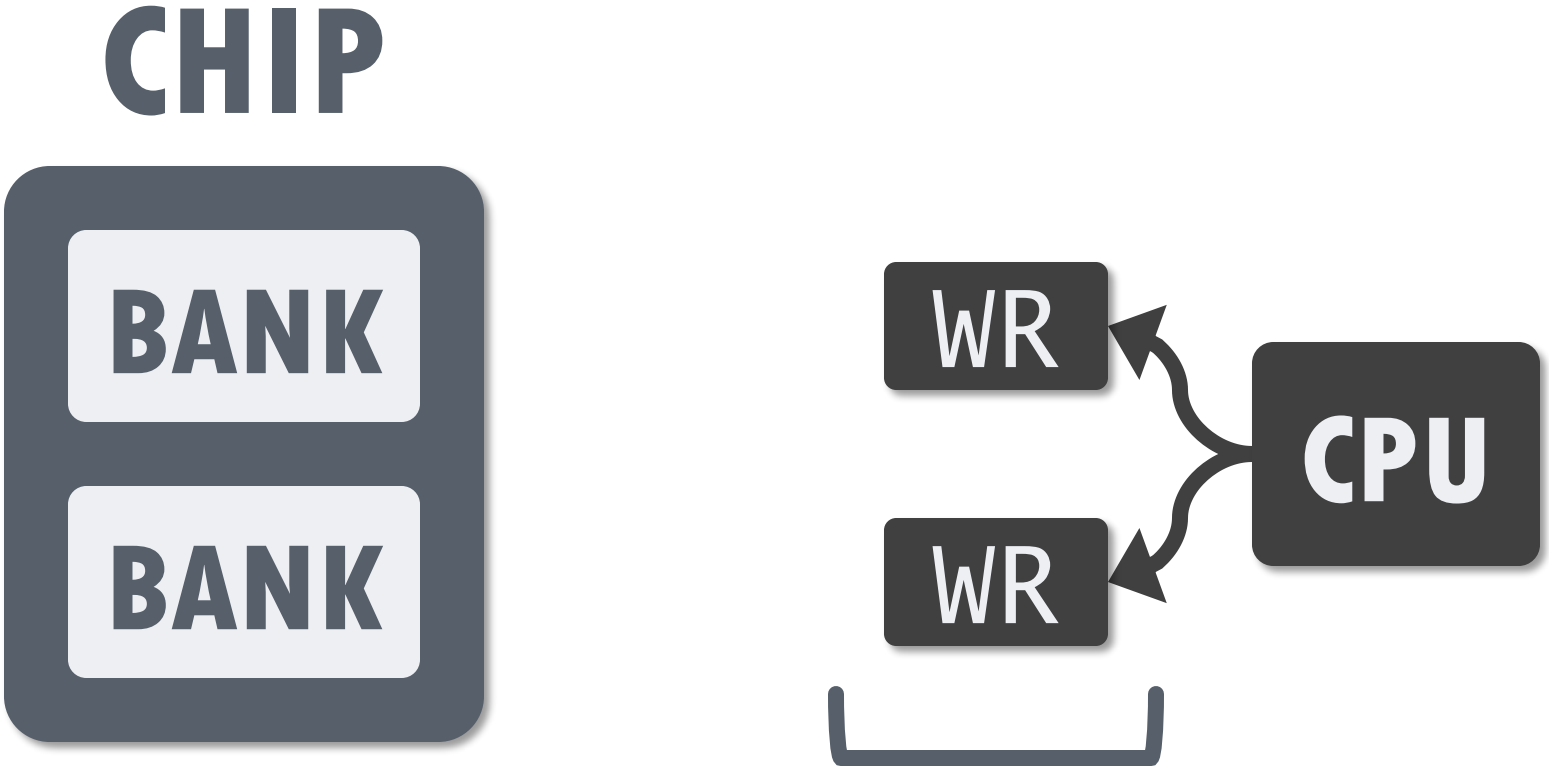


FASTEST SLOWEST

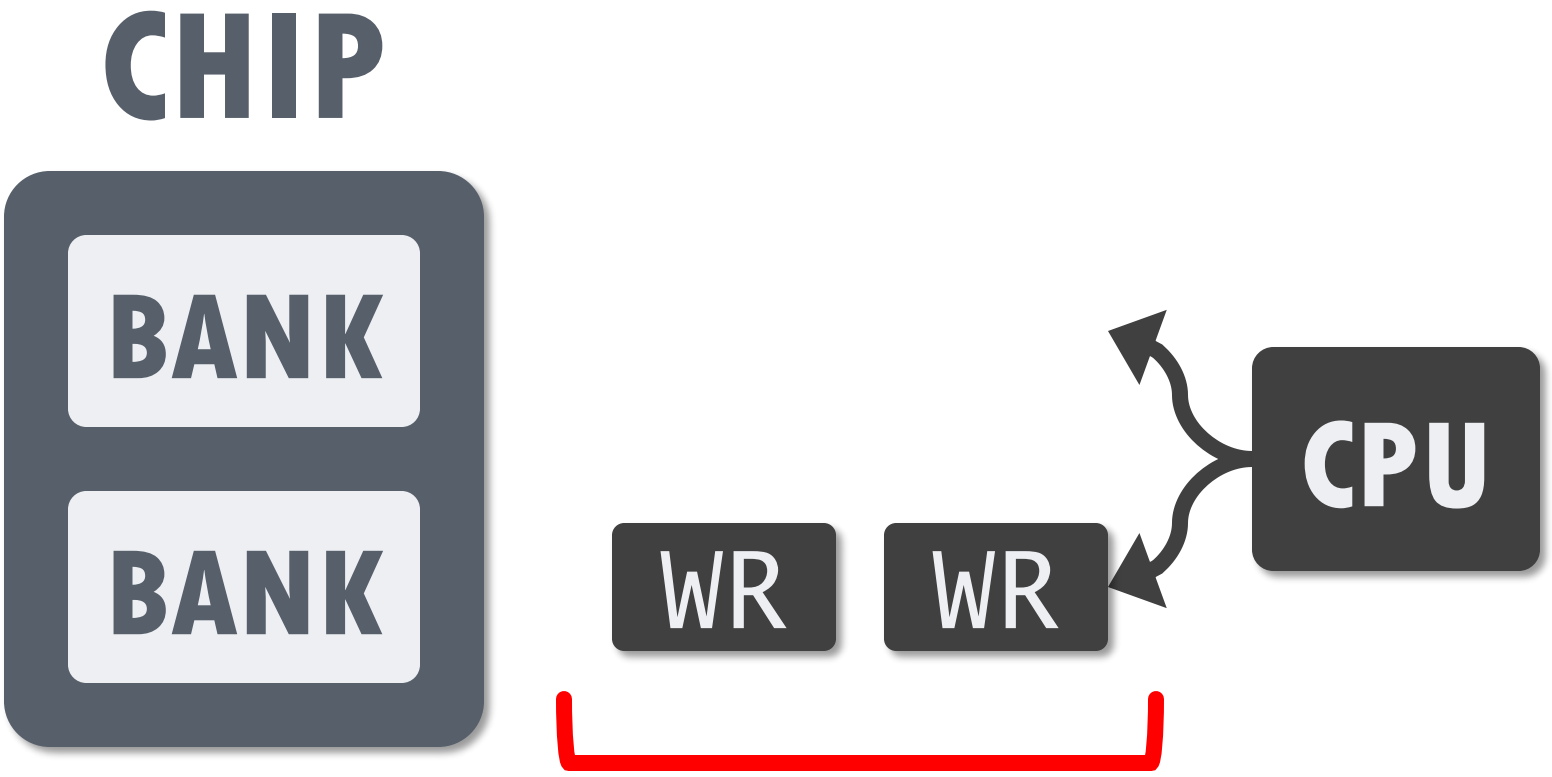
5× "WRITE PENALTY"

Source: Samsung & Intel. The Memory Forum 2014

FIGHT LATENCY WITH MORE PARALLELISM

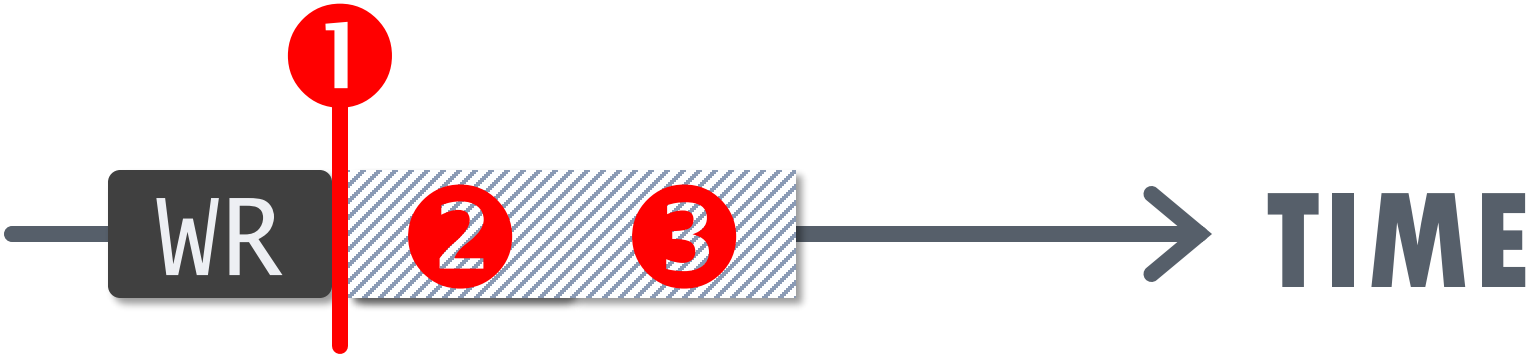


DIFFERENT BANKS:
SMALL LATENCY



BANK CONFLICT:
LARGE LATENCY

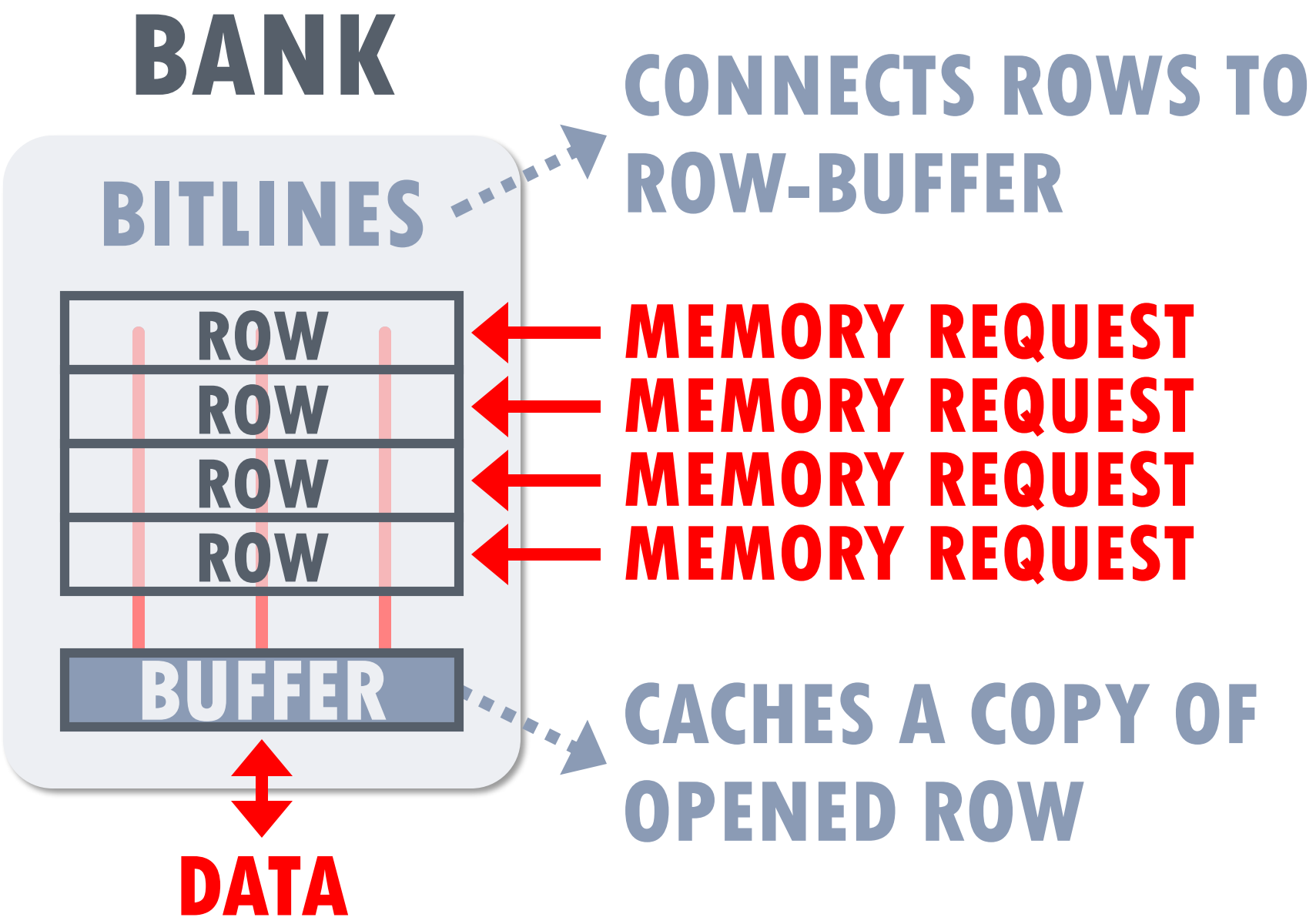
BANK CONFLICT LATENCY



❶ SERIALIZATION

❷ WRITE PENALTY (GETTING WORSE)

❸ "ROW-BUFFER" THRASHING

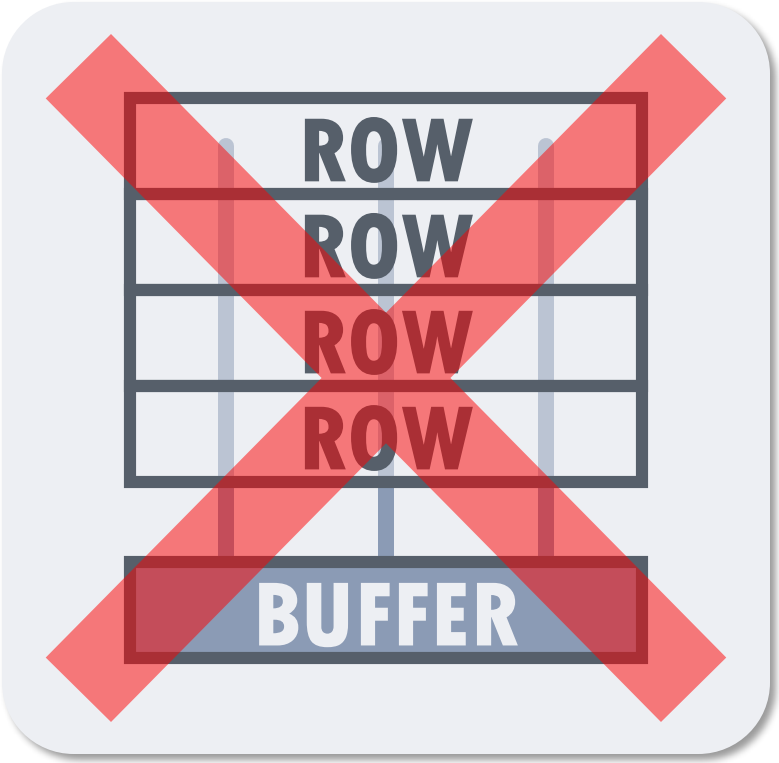

BANK
BITLINES
ROW
ROW
ROW
ROW
BUFFER
DATA
CONNECTS ROWS TO ROW-BUFFER
MEMORY REQUEST
MEMORY REQUEST
MEMORY REQUEST
MEMORY REQUEST
CACHES A COPY OF OPENED ROW

HOW TO PARALLELIZE BANK CONFLICTS?

Previous Approaches

1. Have More Banks: Expensive
2. Bank Interleaving: Non-Solution

BANK (LOGICAL VIEW)

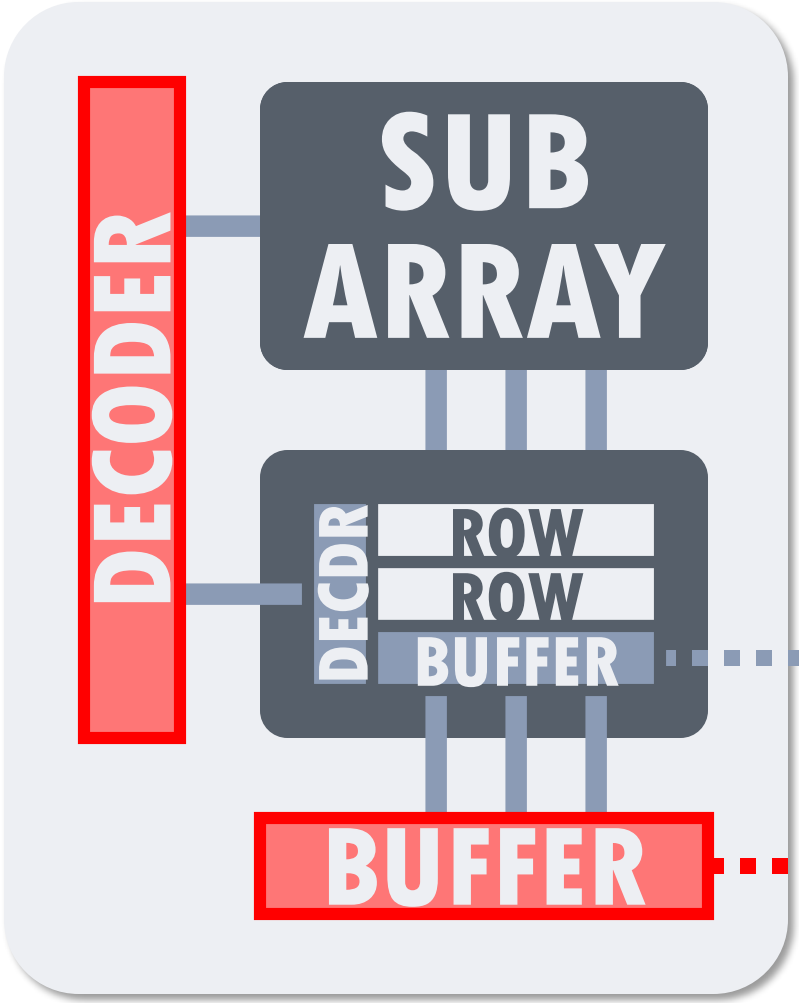


MANY ROWS
(~100K)

JUST ONE

CANNOT DRIVE LONG BITLINES

BANK (PHYSICAL VIEW)

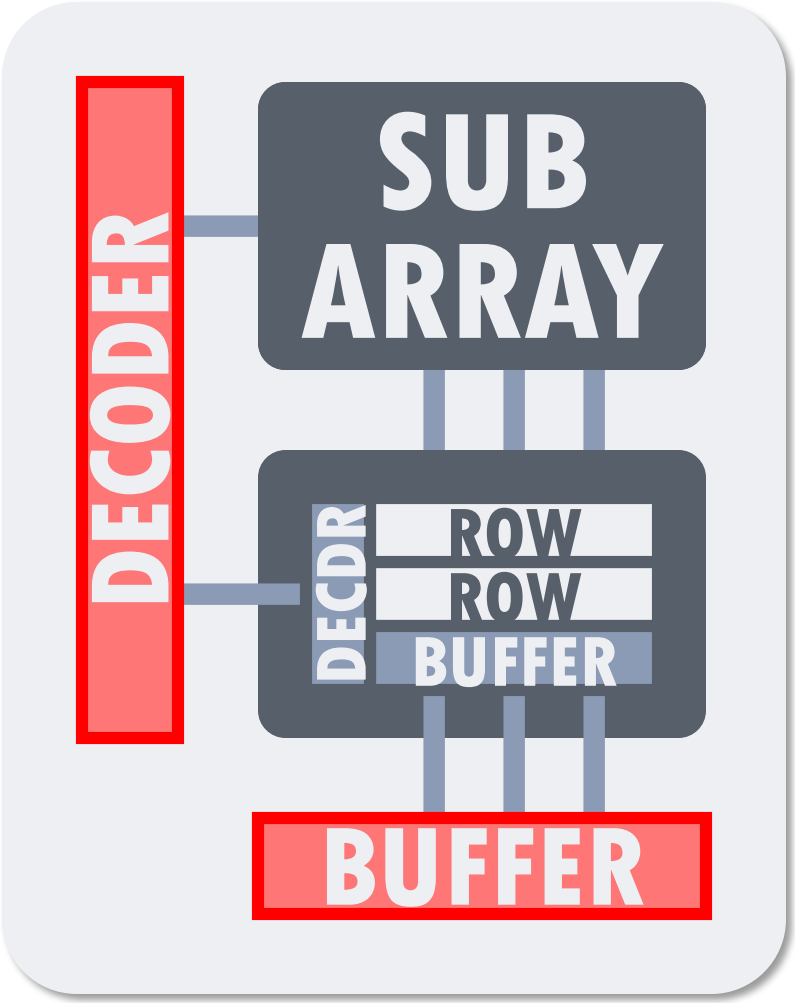


FEWER ROWS (~1K)

LOCAL (SUBARRAY)

GLOBAL (BANK)

SHARING = ROOT OF EVIL



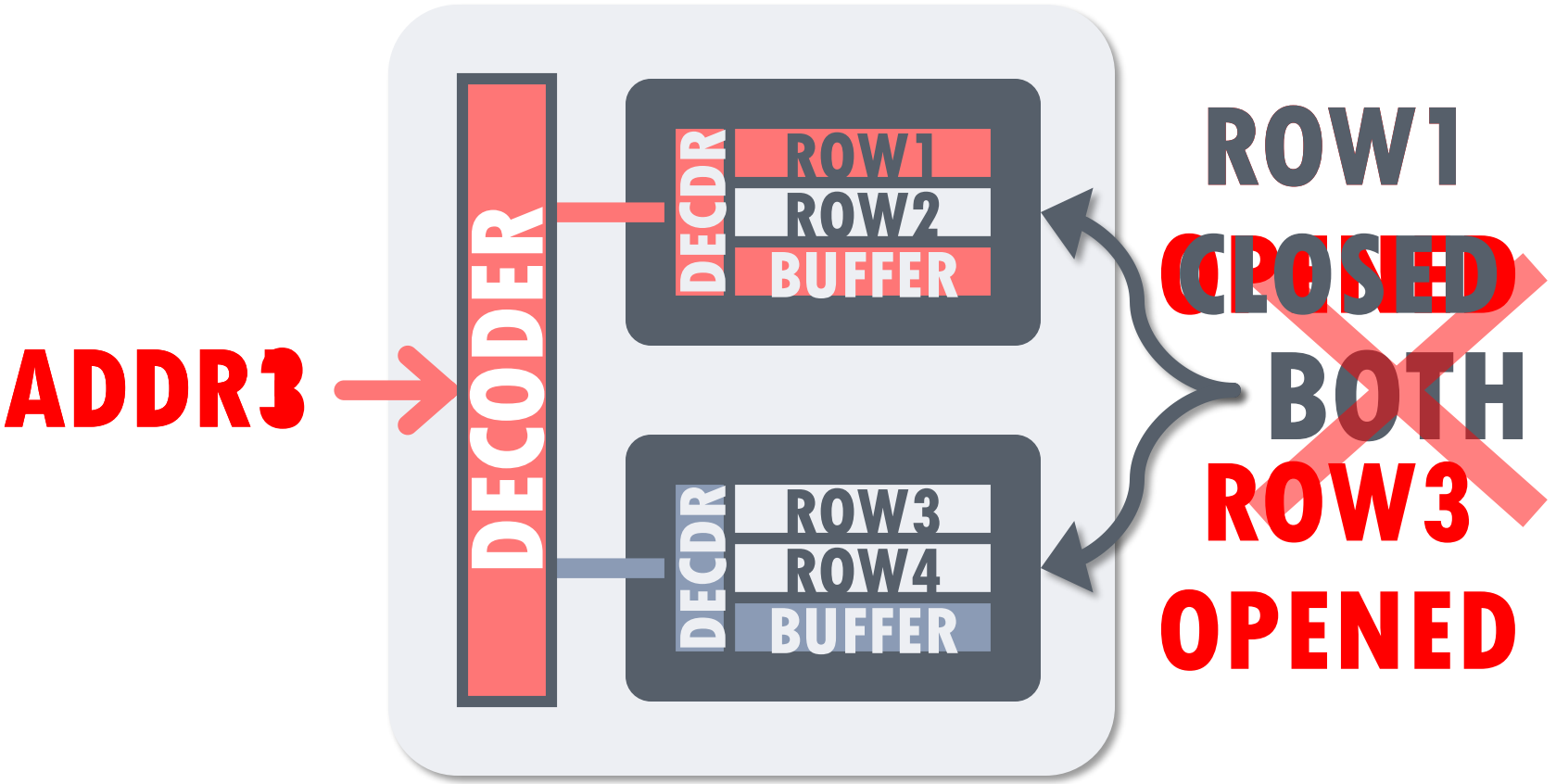
SHARED:

1. GLOBAL DEC.

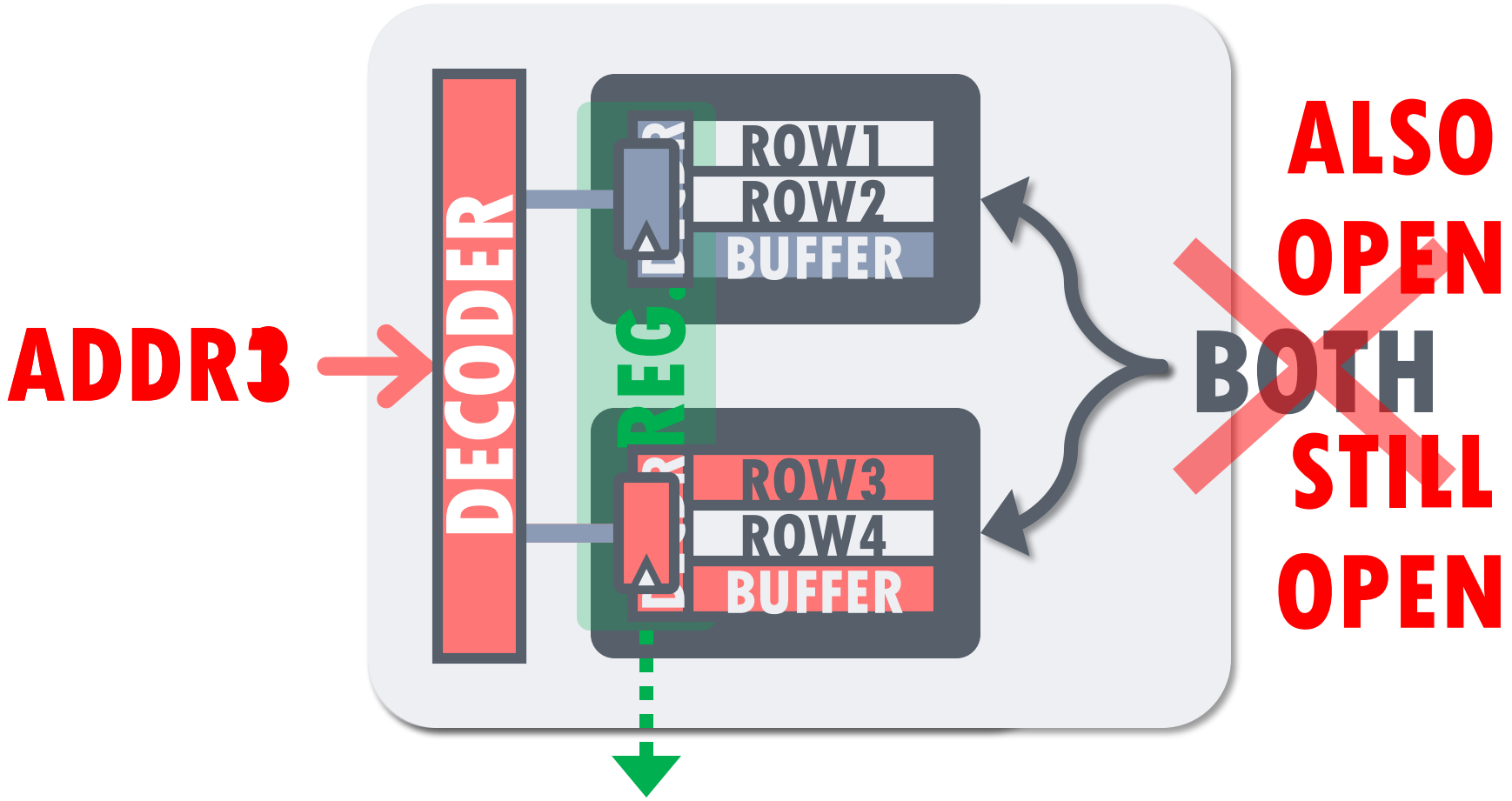
2. GLOBAL BUF.

NO SUBARRAY PARALLELISM

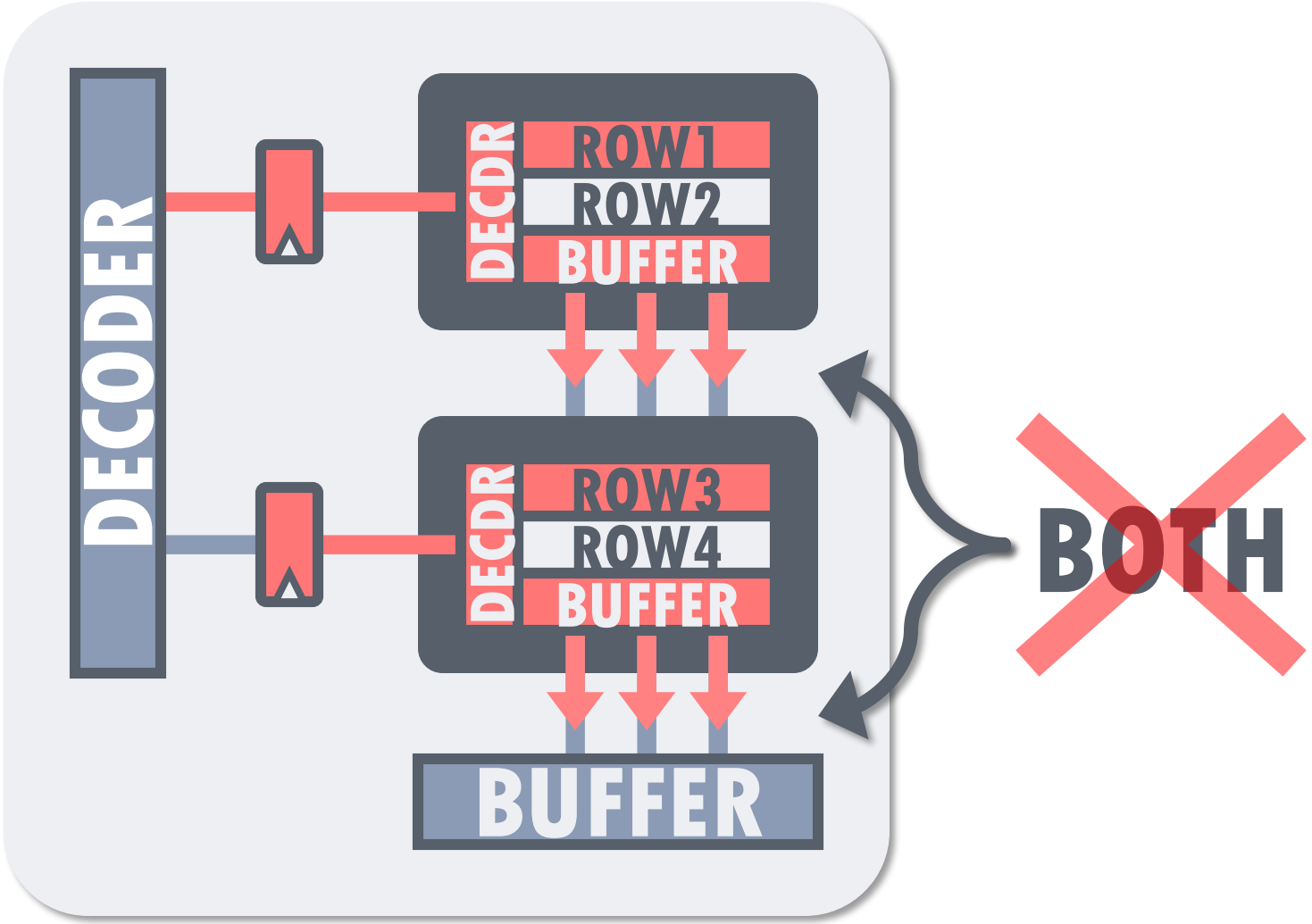
PROBLEM #1: DECODER



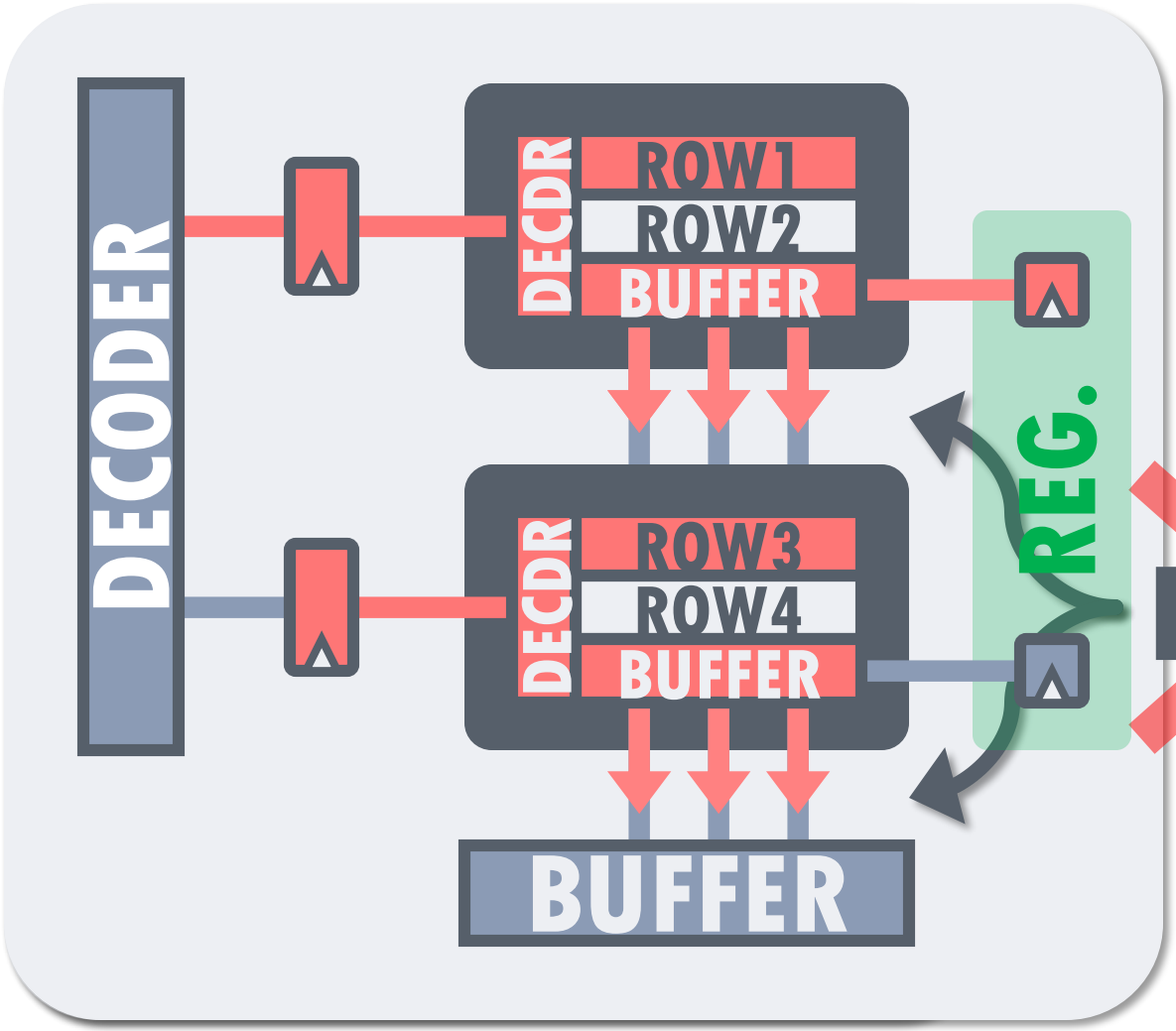
OUR SOLUTION



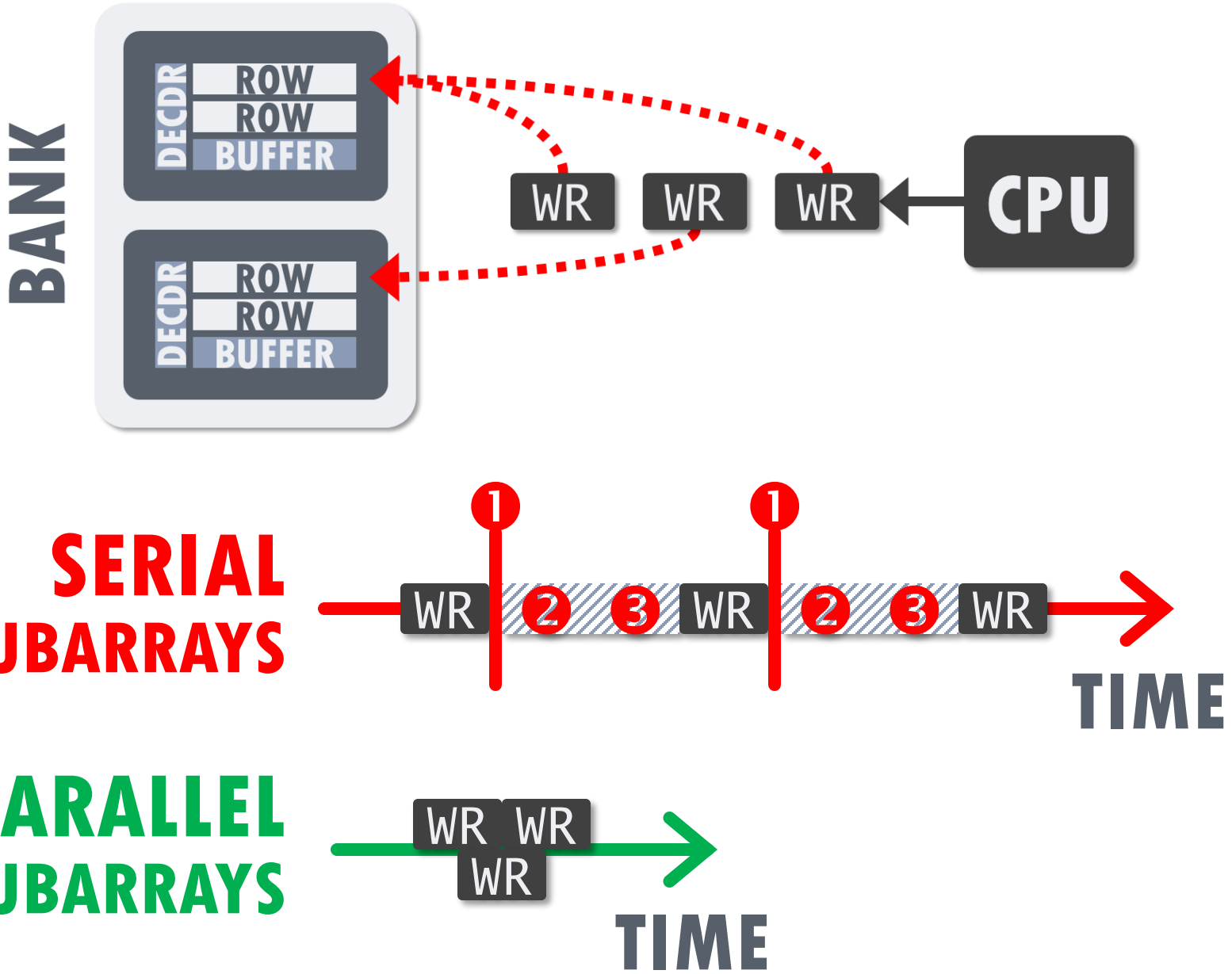
PROBLEM #2: BUFFER

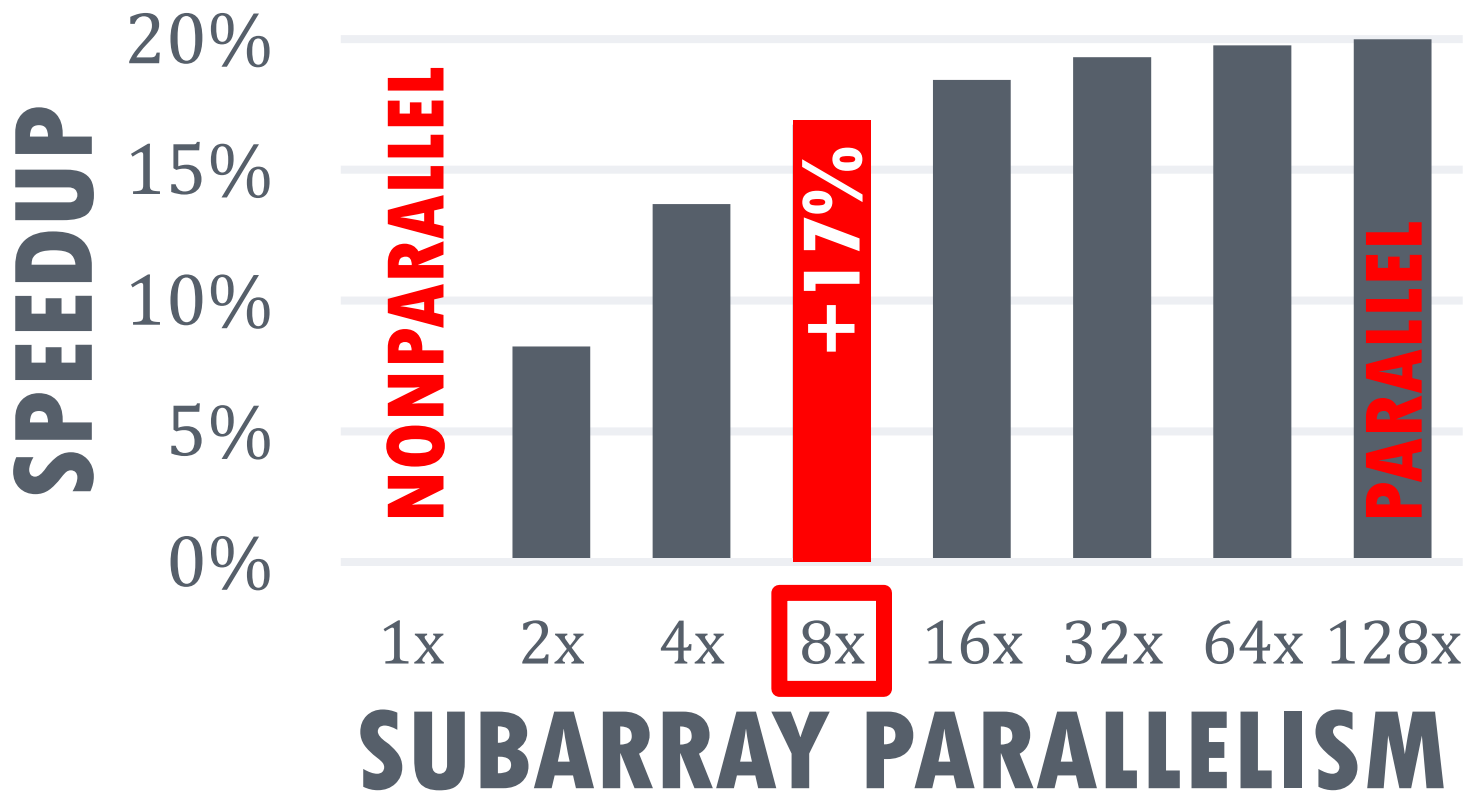


OUR SOLUTION



SELECTS SUBARRAY FOR ACCESS


BANK
DECDR
ROW
ROW
BUFFER
DECDR
ROW
ROW
BUFFER
WR
WR
WR
CPU
SERIAL
SUBARRAYS
1
WR
2
3
WR
1
2
3
WR
TIME
PARALLEL
SUBARRAYS
WR
WR
WR
TIME



- Simulation: Out-of-Order CPU + DDR3-1066
- Benchmarks: SPEC/TPC/STREAM/RANDOM

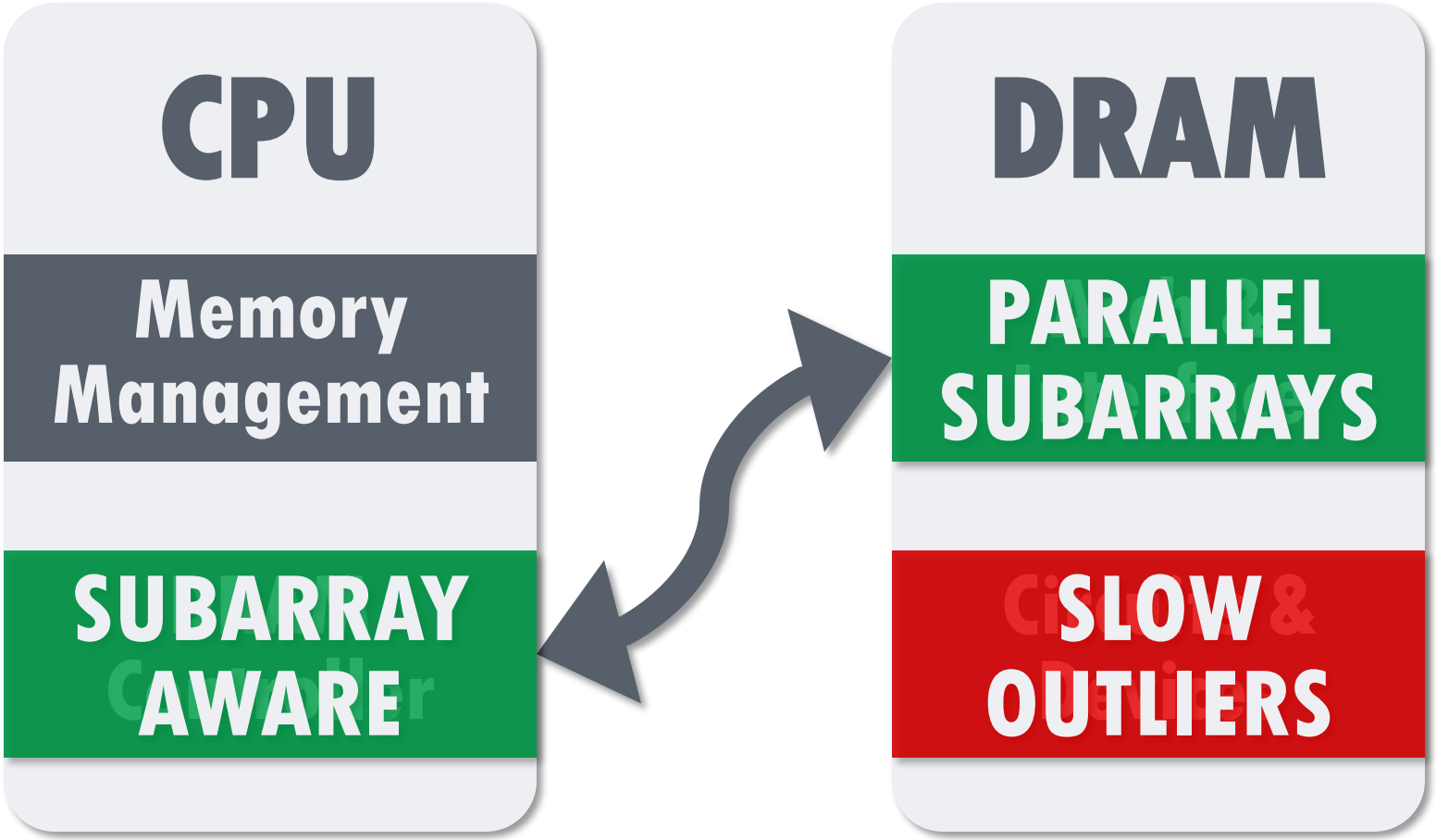
SUBARRAYS	vs.	BANKS
8x	PARALLELISM	8x
+17%	SPEEDUP	+20%
+0.2%	CHIP-SIZE	+36%

Estimated using DRAM area model from Rambus

RELATED WORK

- **Module Partitioning** (e.g., Zheng+ 2008)
  Divide module into small, independent subsets
  Narrower data-bus → Higher unloaded latency
- **Hierarchical Bank** (Yamauchi+ 1997)
  Parallelizes accesses to different subarrays
  Does not utilize multiple local row-buffers
- **Cached DRAM** (e.g., Hidaka+ 1990)
- **Low-Latency DRAM** (e.g., Sato+ 1998)

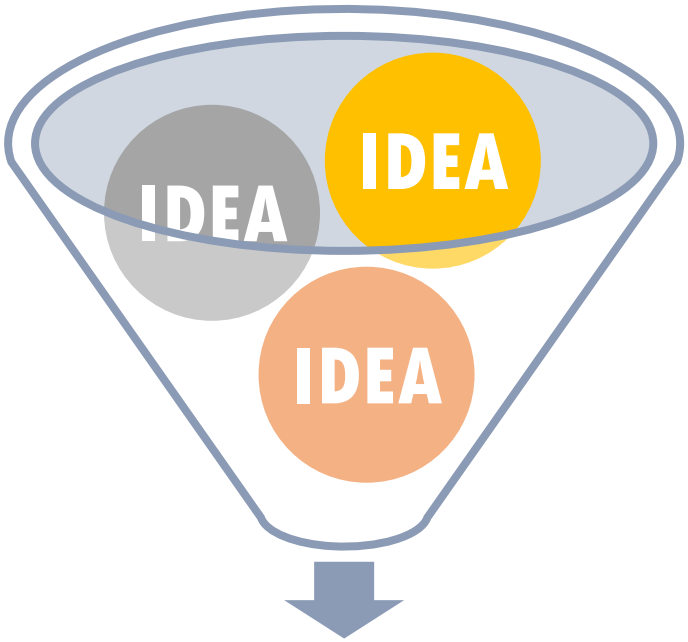
RECAP: PERFORMANCE



RAMULATOR:
A FAST AND EXTENSIBLE DRAM SIMULATOR
IEEE CAL 2015


Ramulator: A Fast and Extensible DRAM Simulator

3. RAMULATOR



NEW IDEAS FOR BETTER DRAM ...

DRAM SIMULATOR

MUST BE VETTED BY SIMULATION

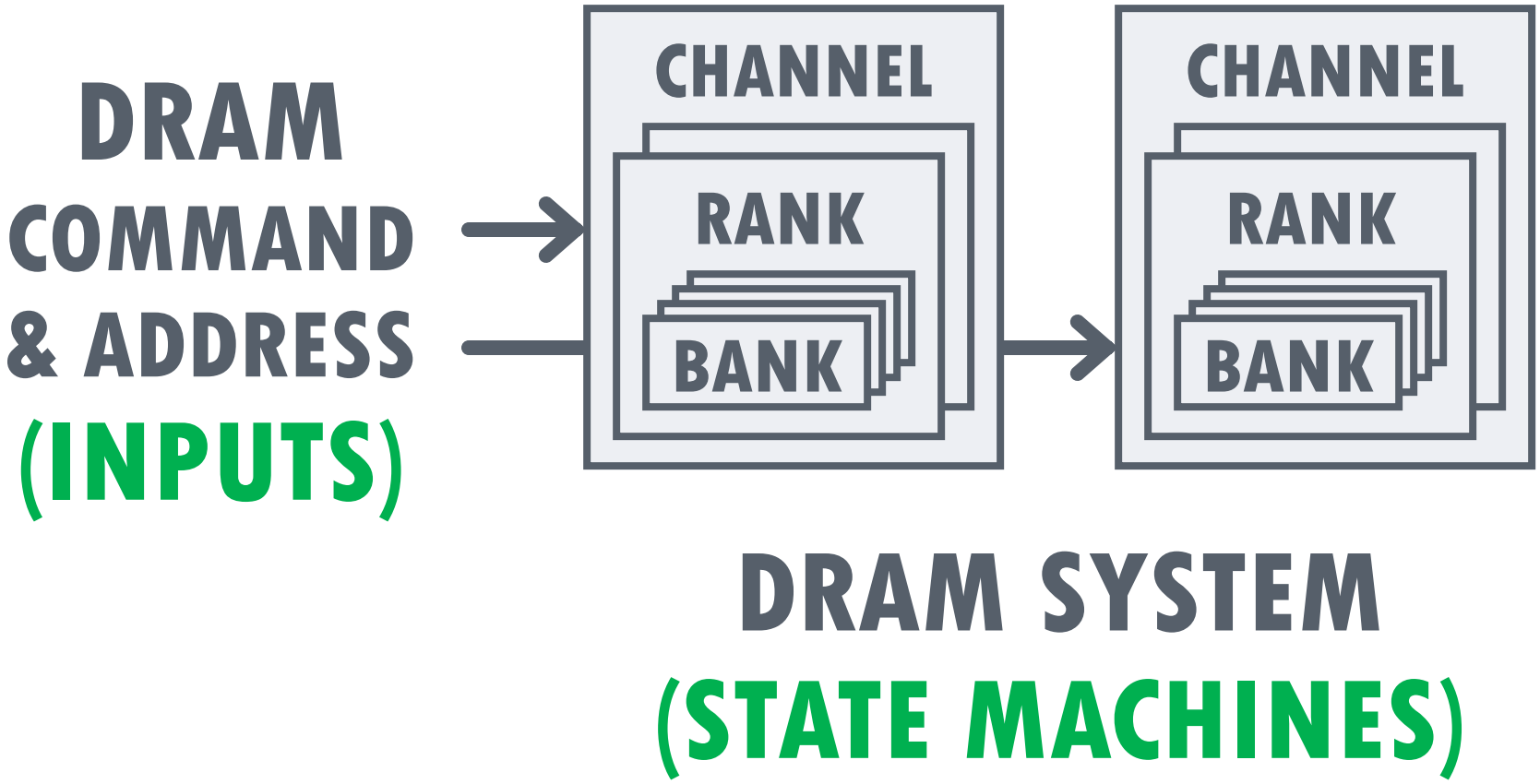
PREVIOUS SIMULATORS

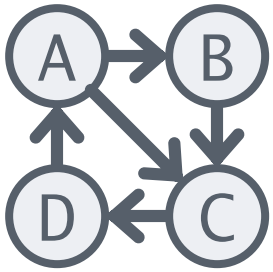
- **PRO: HIGH FIDELITY**
  - Cycle-accurate DRAM models
- **CON: LOW FLEXIBILITY**
  - Hardcoded for DDR3/DDR4 DRAM
  - Difficult to extend to others
    E.g., LPDDRx, WIOx, GDDRx, RLDRAMx, HBM, HMC

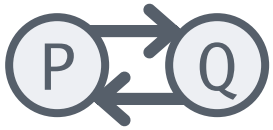
OUR RAMULATOR

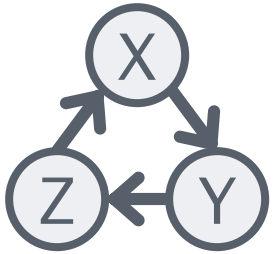
- **BUILT FOR EXTENSIBILITY**
  - Easy to incorporate new ideas
- **HIGH SIMULATION SPEED**
  - 2.5x faster than the next fastest
- **PORTABLE: SIMPLE C++ API**

RAMULATOR'S APPROACH



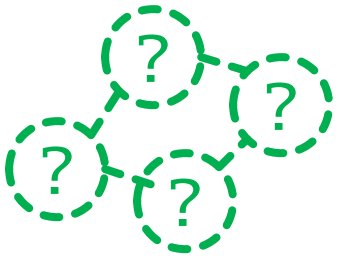




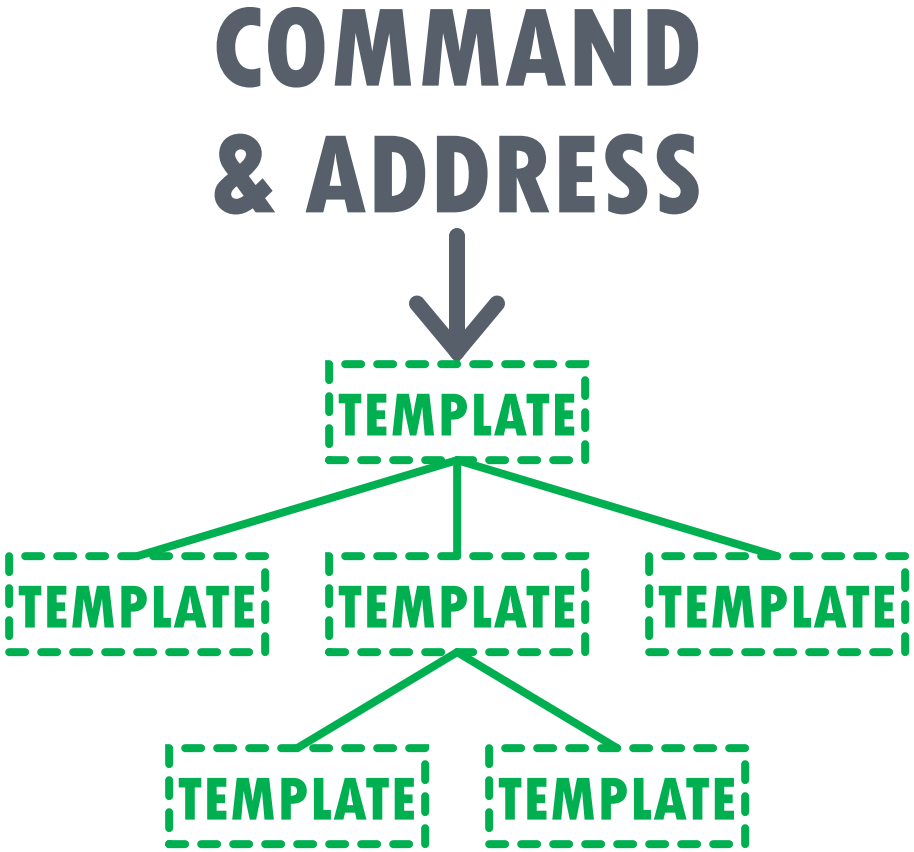


STATE MACHINES ARE DEFINED BY:

- STATES
- EDGES



RAMULATOR: RECONFIGURABLE STATE MACHINE



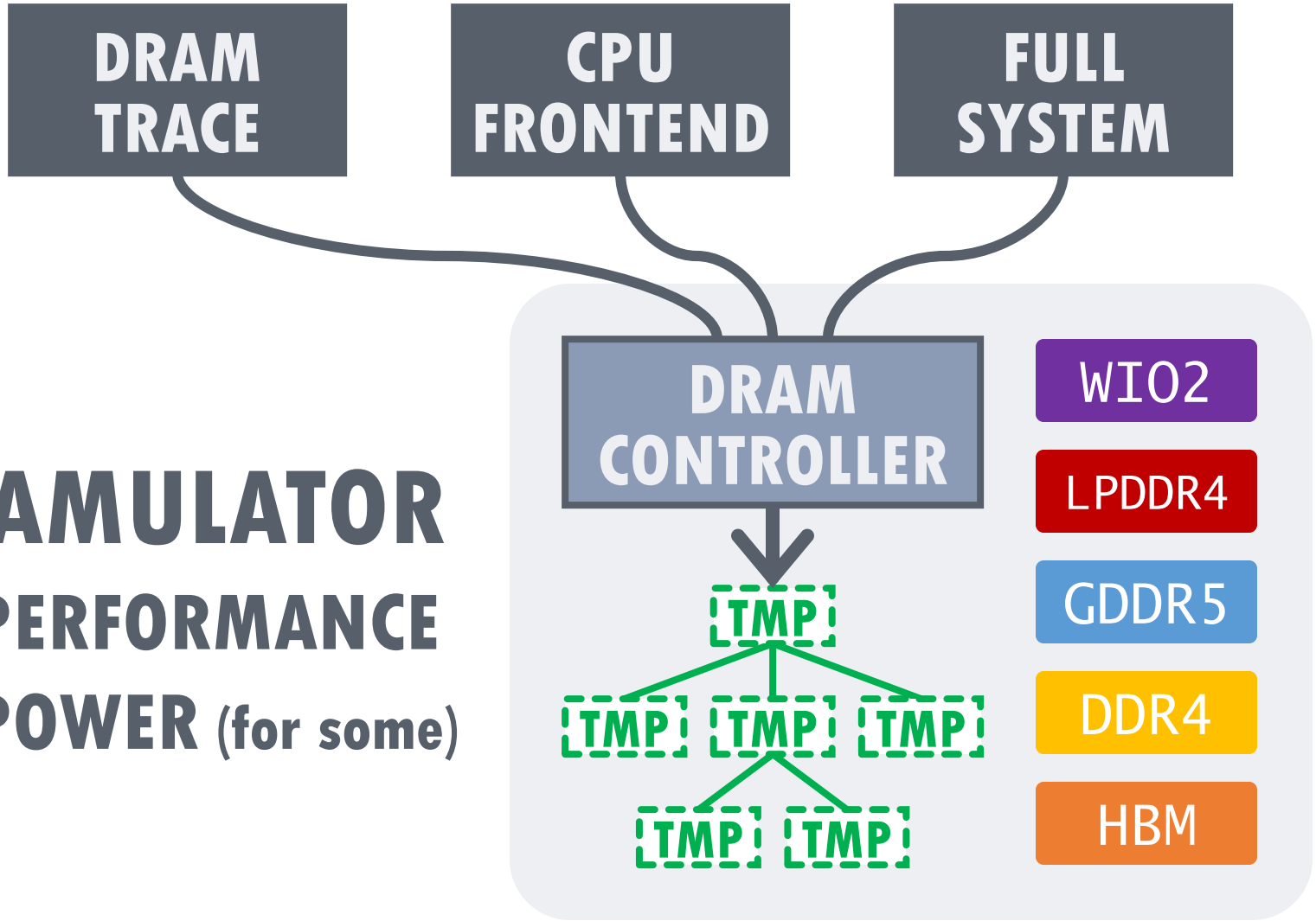
GDDR5
- Hierarchy
- States
- Edges

DDR4
- Hierarchy
- States
- Edges

HBM
- Hierarchy
- States
- Edges

RAMULATOR

- **PERFORMANCE**
- **POWER** (for some)



http://www.github.com/CMU-SAFARI/ramulator

	Supported DRAM Specifications	Simulation Speed (DDR3)
Ramulator	**DDR3/4, LPDDR3/4, GDDR5, HBM, WIO1/2, Subarray Parallelism, etc.**	**2.70x**
DRAMSim2 (Rosenfeld et al.)	DDR2/3	1x
USIMM (Chatterjee et al.)	DDR3	1.08x
DrSim (Jeong et al.)	DDR2/3, LPDDR2	0.11x
NVMain (Poremba and Xie)	DDR3, LPDDR3/4	0.30x

4. CONCLUSION

SUMMARY

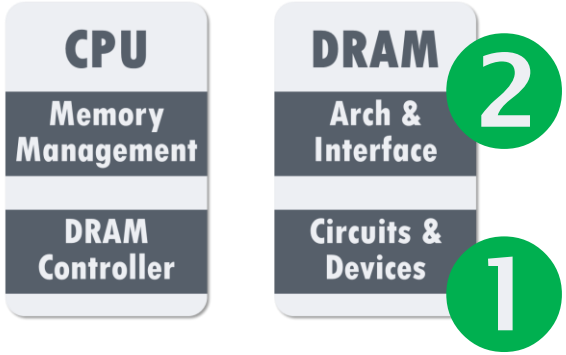
Traditional DRAM Scaling at Risk

Architectural Techniques for Coping with Degrading DRAM

Regain Reliability & Performance

CONTRIBUTIONS

1. We show that DRAM scaling is negatively affecting reliability.
2. We propose a high-performance, cost-effective DRAM architecture.
3. We develop a new simulator for facilitating DRAM research.



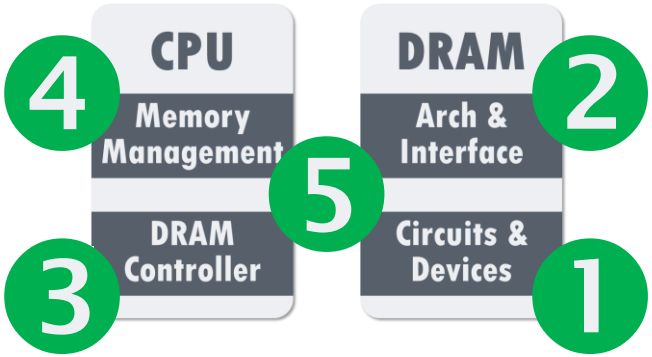
RESEARCH

❶ RELIABILITY

- **Row Hammer**
  Kim et al., ISCA '14
- **Retention Failures**
  Liu et al., ISCA '13
  Khan et al., SIGMETRICS '14
- **Speed vs. Reliability**
  Lee et al., HPCA '15

❷ EFFICIENCY

- **Bank Conflicts**
  Kim et al., ISCA '12
- **In-DRAM Page Copy**
  Seshadri et al., MICRO '13
- **Tiered Latency**
  Lee et al., HPCA '13
- **Fine-Grained Refreshes**
  Chang et al., HPCA '14



RESEARCH

❹ COMPRESSION

- **Simple & Effective**
  Pekhimenko et al., MICRO '13

❸ SCHEDULING

- **High Throughput**
  Kim et al., HPCA '10
- **Quality of Service**
  Kim et al., MICRO '10
  Kim et al., IEEE Micro Top Picks '11
  Subramanian et al., HPCA '13

❺ SIMULATION

- **Fast & Extensible**
  Kim et al., IEEE CAL '15

OUTDATED ASSUMPTIONS

- DRAM SCALING WILL TAKE CARE OF MAIN MEMORY
- LATEST AND CHEAPEST DRAM IS THE GREATEST

NEW TECHNOLOGIES

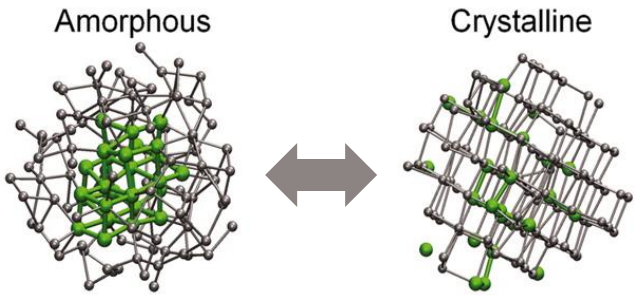


Image: Loke et al., Science 2012

NONVOLATILE MEMORY (NVM)

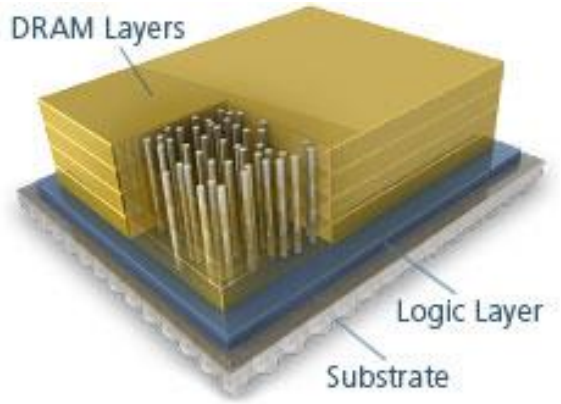
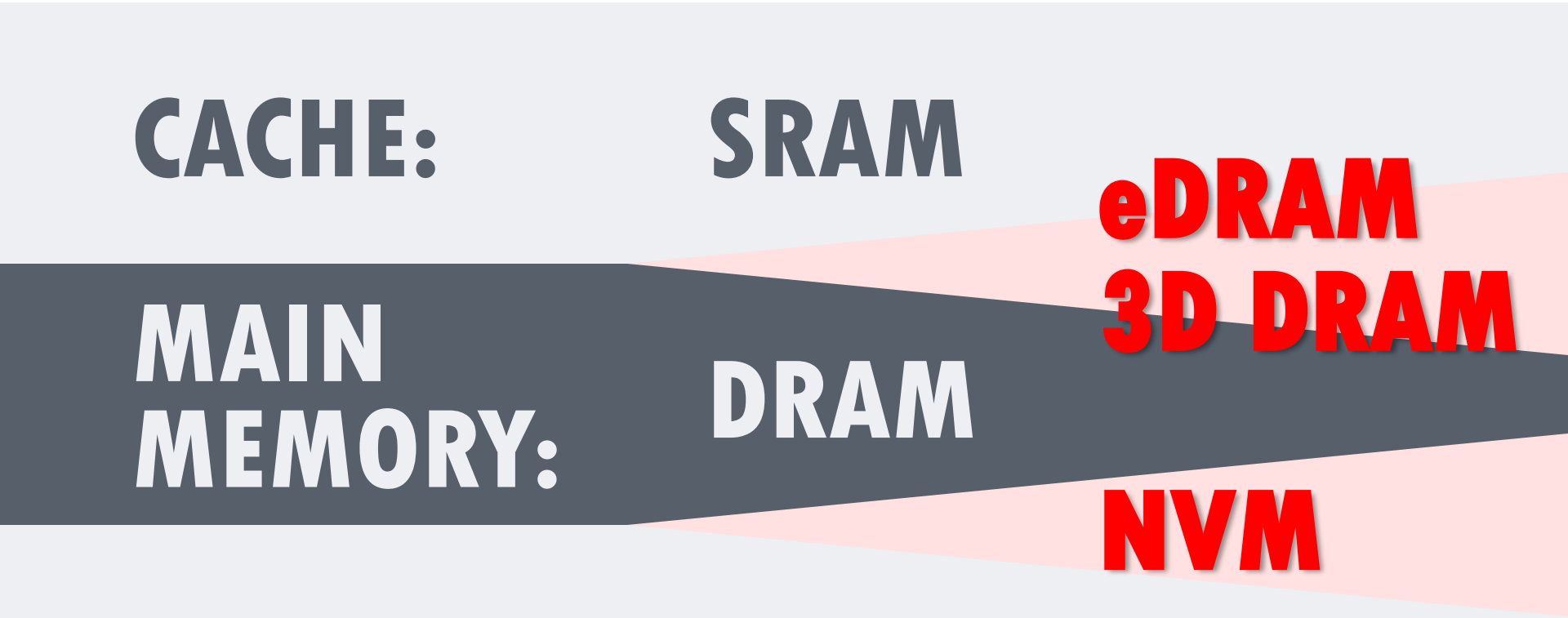


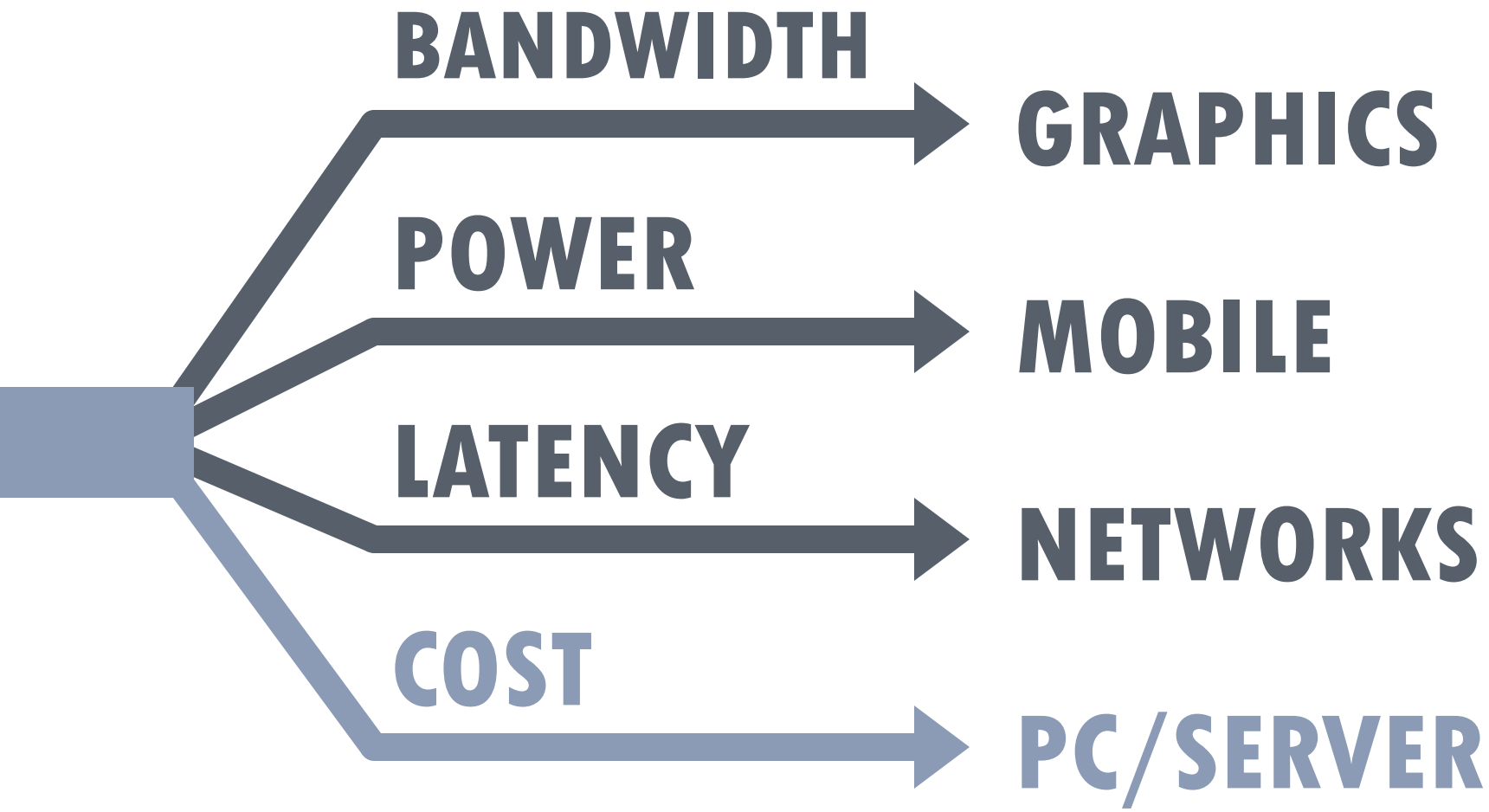
Image: Micron Technology, 2015

3-DIMENSIONAL DIE STACKING

TREND: HETEROGENEITY



TREND: SPECIALIZATION



HOW TO REFORMULATE THE MEMORY HIERARCHY?

- **NEW SOFTWARE ABSTRACTIONS**
  Primitives for managing heterogeneity

- **NEW HARDWARE TRADE-OFFS**
  Defining appropriate roles for each tier

- **RICH MEMORY CAPABILITIES**
  Memory as more than a “bag of bits”

ARCHITECTURAL TECHNIQUES TO ENHANCE DRAM SCALING

Thesis Defense

Yoongu Kim