18-742: Computer Architecture & Systems

Row Hammer

Prof. Phillip Gibbons

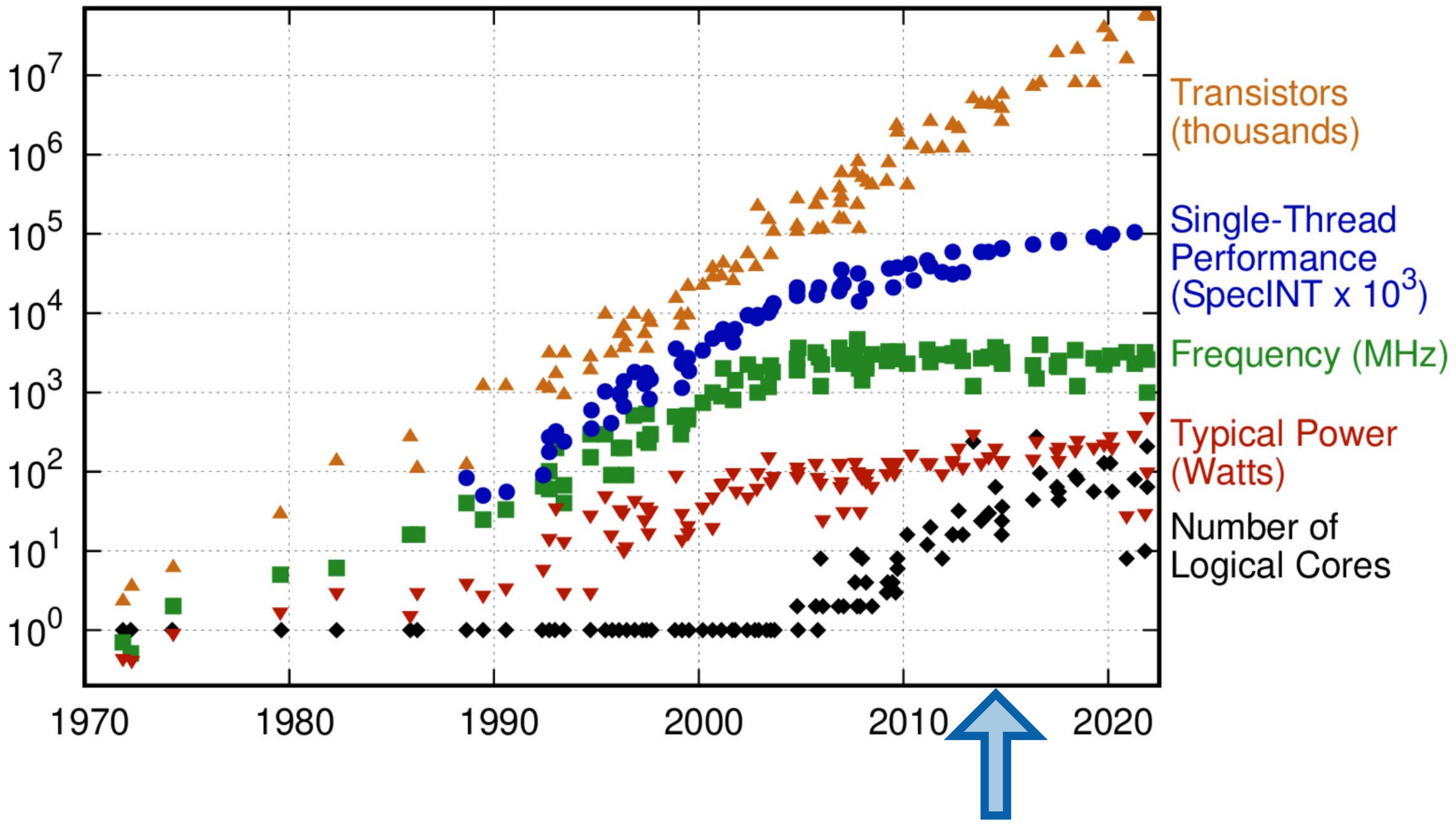
Spring 2025, Lecture 19

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

Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, Onur Mutlu 2014

- **Yoongu:** CMU PhD, now Google
- **Ross:** CMU student, now Stanford PhD
- **Jeremie:** CMU MS, now CMU PhD
- **Chris F:** CMU PhD, now Fastly
- **Ji-Hye:** CMU student, now ??
- **Donghyuk:** CMU PhD, now Nvidia
- **Chris W:** Intel Principal Eng., CMU MS
- **Konrad:** ex-Intel
- **Onur:** CMU prof, now ETH
  - Young Architect Award, Maurice Wilkes Award, ACM/IEEE Fellow

50 Years of Microprocessor Trend Data

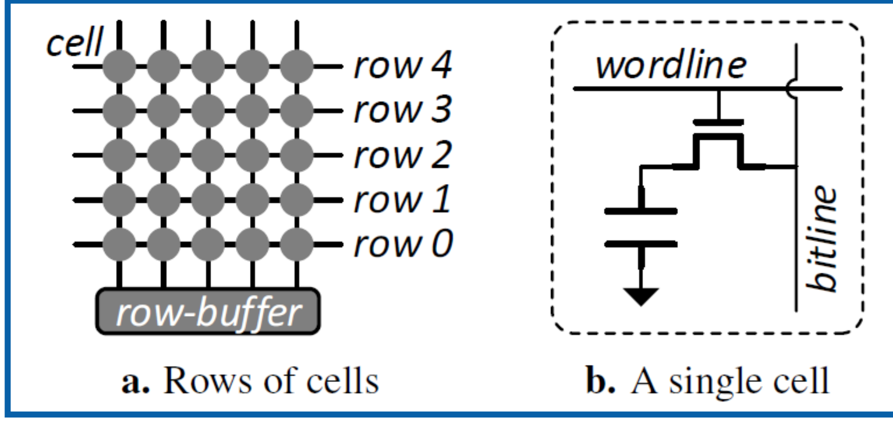


We are here

DRAM Organization

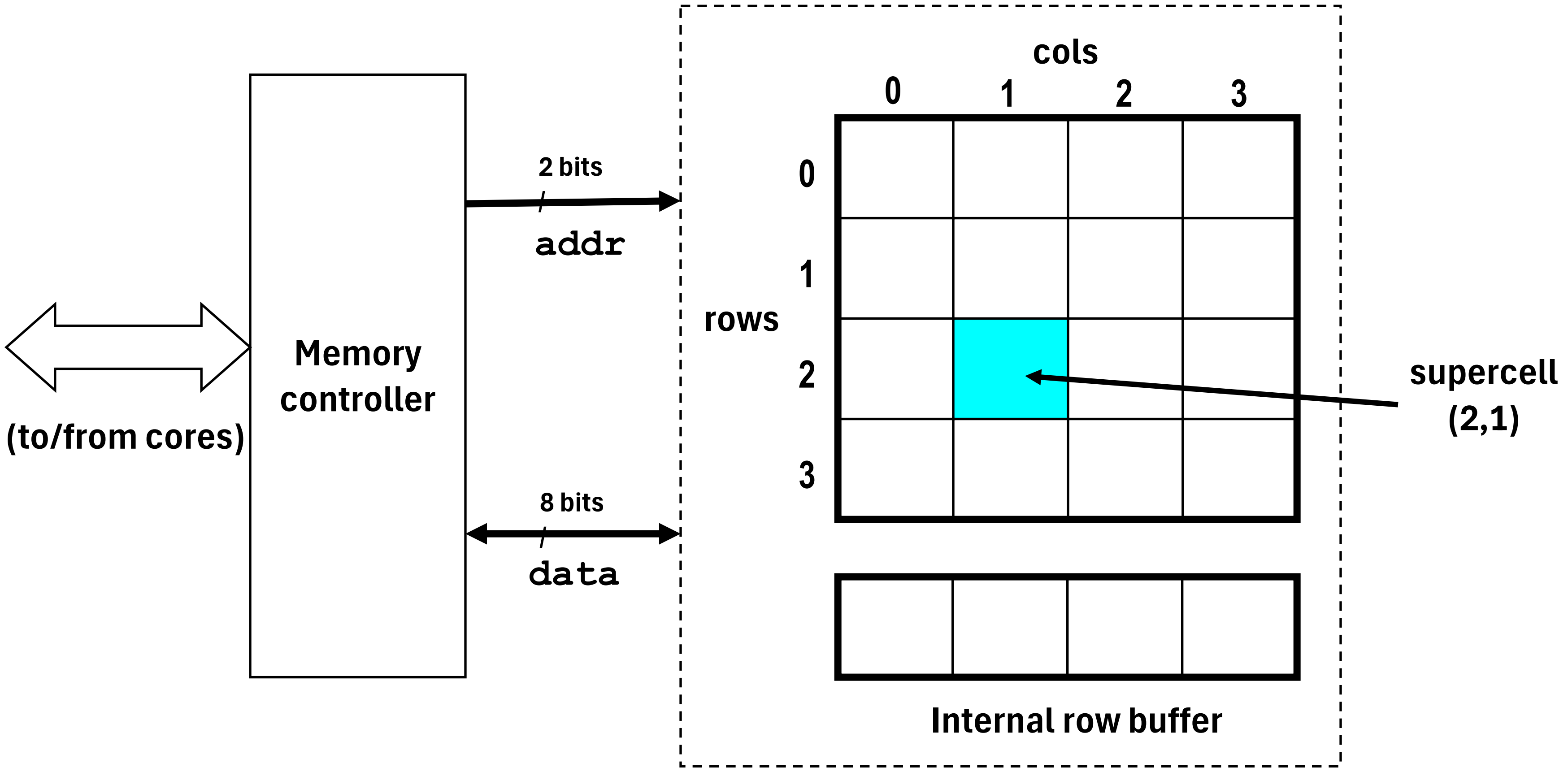
d x *w* DRAM:

- *d* · *w* total bits organized as *d* supercells of size *w* bits

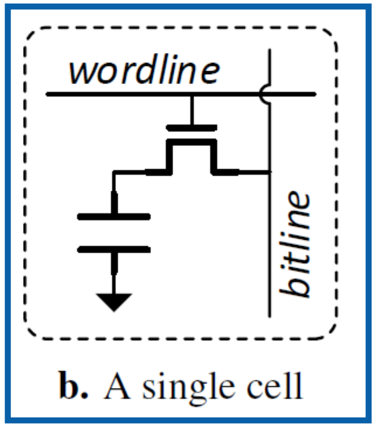


a. Rows of cells **b.** A single cell

16 x 8 DRAM chip



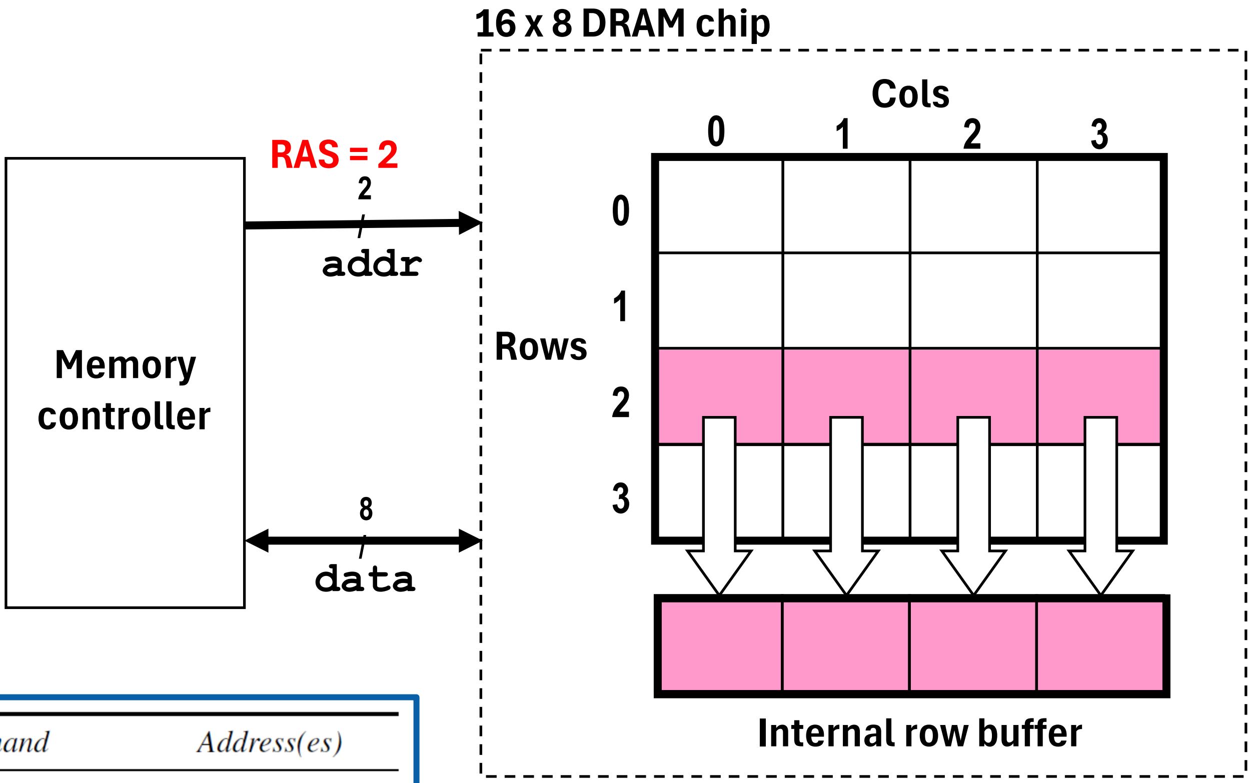
Reading DRAM Supercell (2,1)

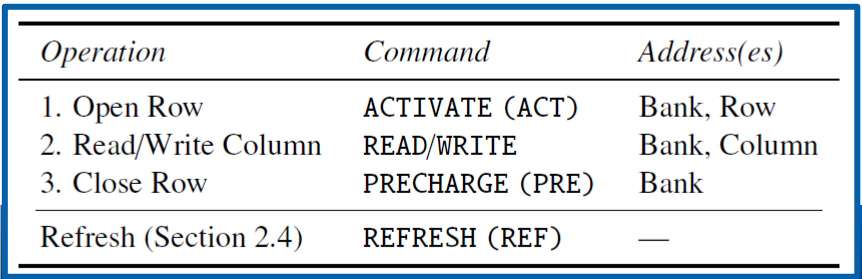


b. A single cell

Step 1(a): Row access strobe (RAS) selects row 2.

Step 1(b): Raising wordline causes Row 2 to be copied from DRAM array to row buffer.



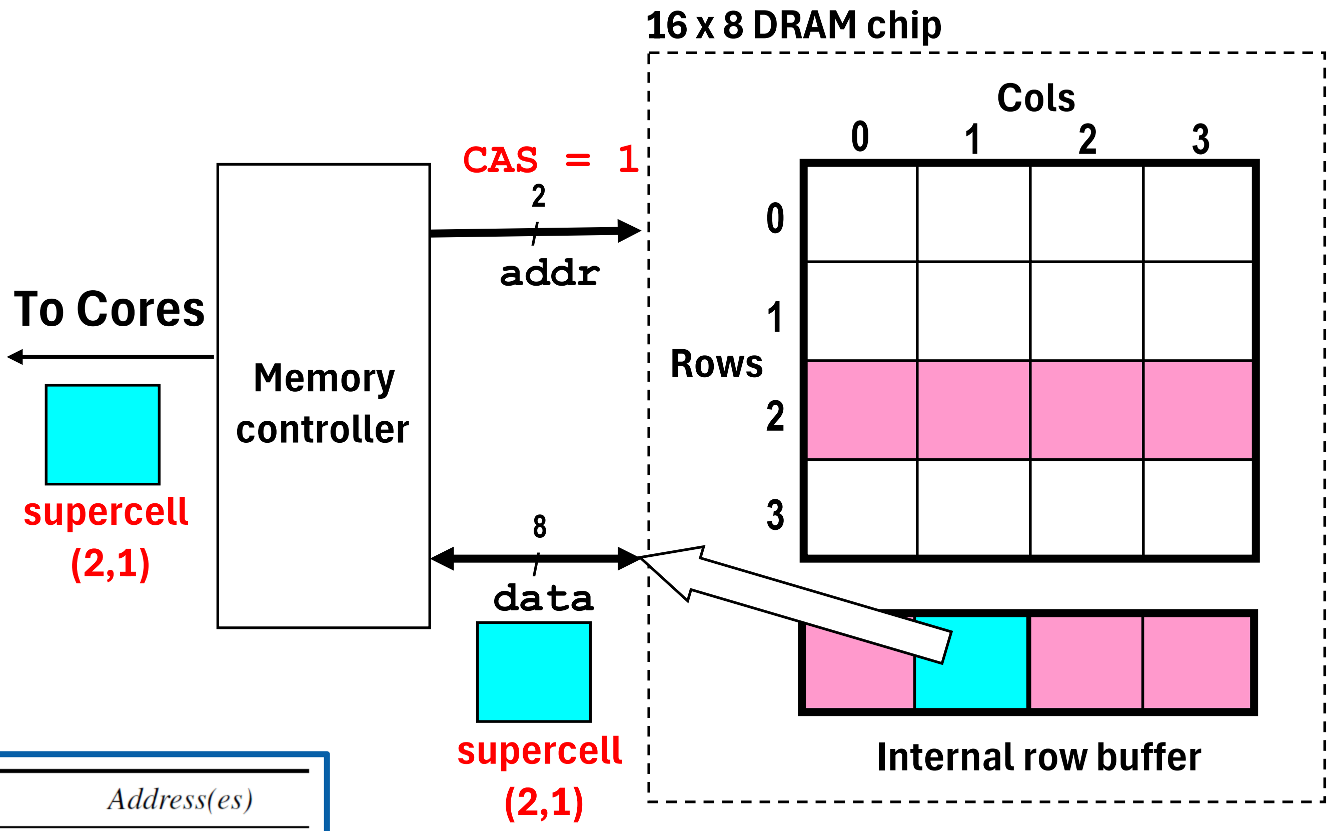


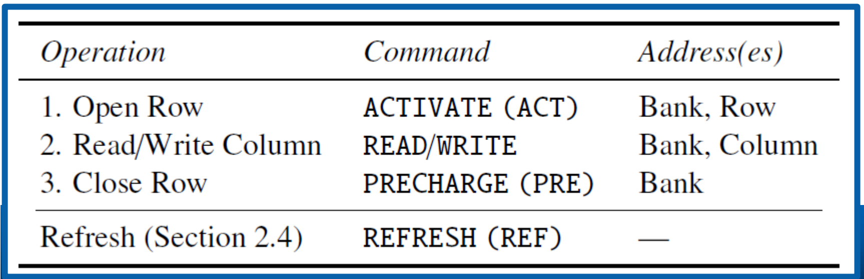
Operation	*Command*	*Address(es)*
1. Open Row	ACTIVATE (ACT)	Bank, Row
2. Read/Write Column	READ/WRITE	Bank, Column
3. Close Row	PRECHARGE (PRE)	Bank
Refresh (Section 2.4)	REFRESH (REF)	—

Reading DRAM Supercell (2,1)

Step 2(a): Column access strobe (CAS) selects column 1.

Step 2(b): Supercell (2,1) copied from buffer to data lines, and back to CPU.

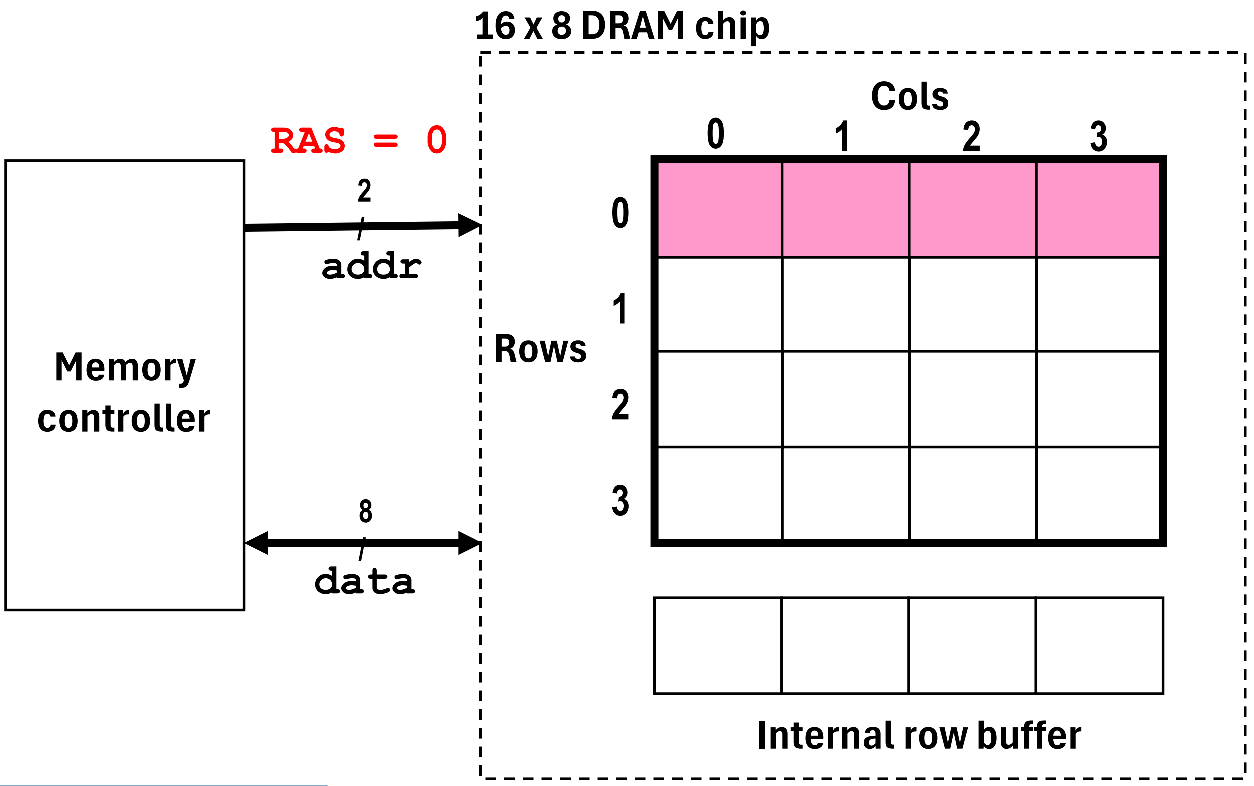




Operation	*Command*	*Address(es)*
1. Open Row	ACTIVATE (ACT)	Bank, Row
2. Read/Write Column	READ/WRITE	Bank, Column
3. Close Row	PRECHARGE (PRE)	Bank
Refresh (Section 2.4)	REFRESH (REF)	—

Reading DRAM Supercell (2,1)

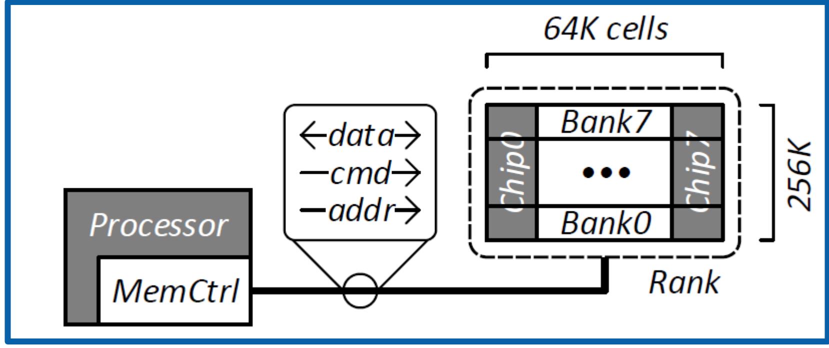
Step 3: To serve a different RAS, first lower wordline to close row 2 and empty row buffer.

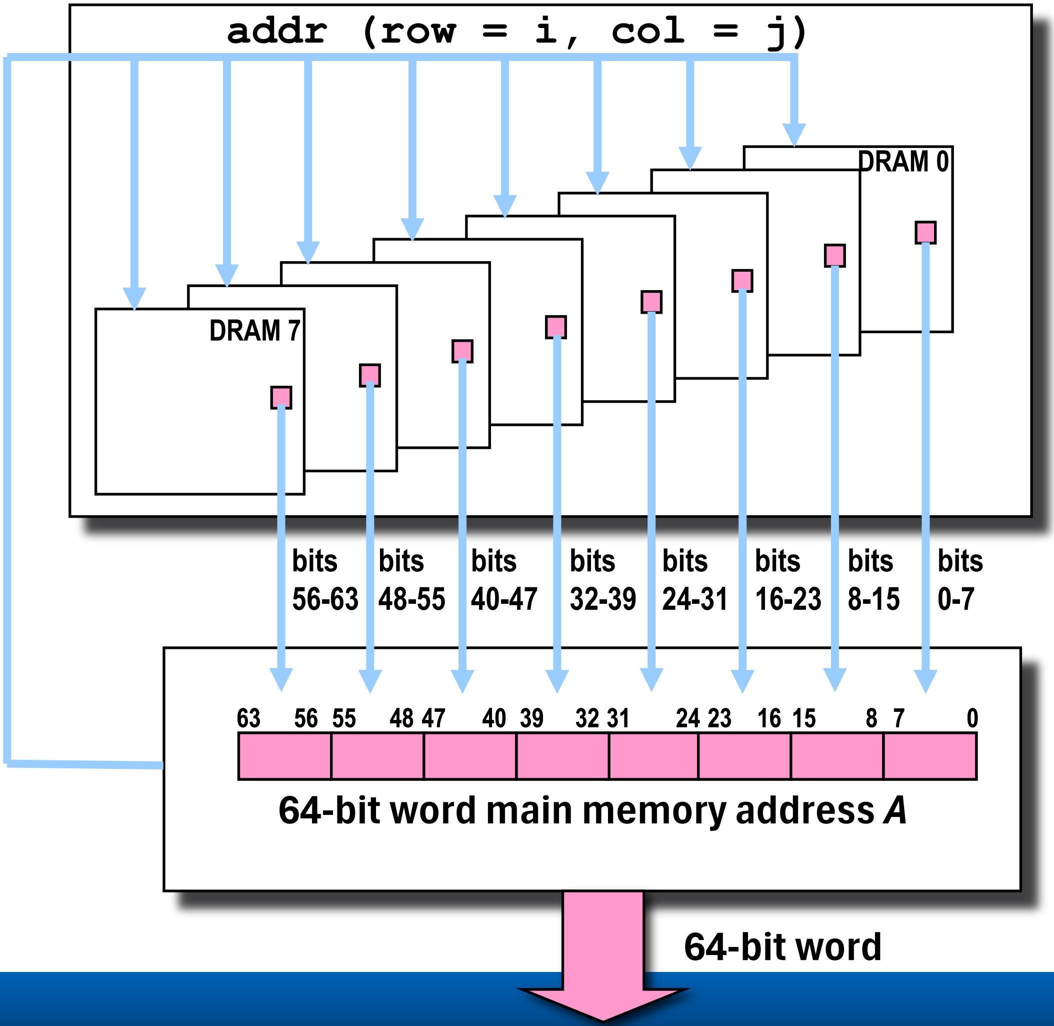


Operation	*Command*	*Address(es)*
1. Open Row	ACTIVATE (ACT)	Bank, Row
2. Read/Write Column	READ/WRITE	Bank, Column
3. Close Row	PRECHARGE (PRE)	Bank
Refresh (Section 2.4)	REFRESH (REF)	—

t_{RC} ~ 50 nanosecs between reopening same row

DRAM Rank





🟪 : supercell (i,j)

64 MB memory module consisting of eight 8Mx8 DRAMs

Memory controller

Shrinking DRAM Process Technology

- **Benefits**
  - Reduces cost-per-bit
  - Desired capacity w/fewer DIMMs: smaller form factor
- **Reliability challenges**
  - Holds limited charge: reduces noise margin, vulnerable to loss
  - Proximity: electromagnetic coupling effects
  - High variation: more cells susceptible to inter-cell crosstalk

Key Findings

- **Disturbance errors are widespread in commodity DRAM**
  - Disturbable cells exist in 110 of 129 tested modules (all of 2012-13)
  - Intel filed Row Hammer patents in 2014 (paper was under review)
- **Simple user-level programs can induce such errors**
- **Root cause is the repeated toggling of a row's wordline**
  - Voltage fluctuation causes nearby rows to rapidly lose charge
  - As few as 139K times suffice
- **Up to 1 in 1.7K cells is disturbable**
- **Propose probabilistic adjacent row refresh as mitigation**

Assembly Code on Intel/AMD Machines

```
code1a:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp code1a
```

a. Induces errors

```
code1b:
  mov (X), %eax
  clflush (X)


  mfence
  jmp code1b
```

b. Does not induce errors

Bit-Flip	Sandy Bridge	Ivy Bridge	Haswell	Piledriver
‘0’ → ‘1’	7,992	10,273	11,404	47
‘1’ → ‘0’	8,125	10,449	11,467	12

Errors vs. Manufacturing Date

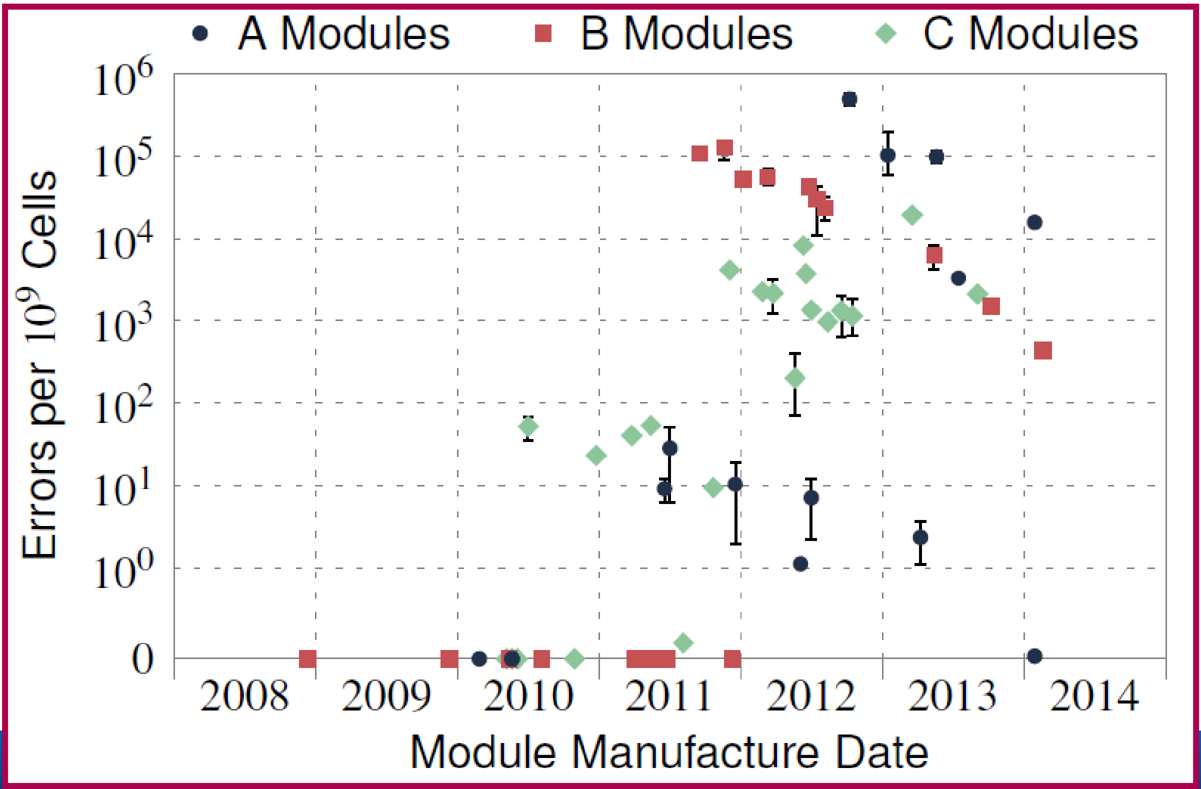
```
TestBulk($AI, RI, DP$)
  setAI($AI$)
  setRI($RI$)
  $N \leftarrow (2 \times RI)/AI$

  writeAll($DP$)
  for $r \leftarrow 0 \cdots ROW_{MAX}$
    for $i \leftarrow 0 \cdots N$
      ACT $r^{th}$ row
      READ $0^{th}$ col.
      PRE $r^{th}$ row
  readAll()
  findErrors()
```

a. Test all rows at once

FPGA experiments to raw DRAM
(Activation Interval, Refresh Interval, Data Pattern)

Access Pattern	Disturbance Errors?
1. *(open–read–close)*^{N}	**Yes**
2. *(open–write–close)*^{N}	**Yes**
3. *open–read*^{N}*–close*	No
4. *open–write*^{N}*–close*	No



Discussion: Summary Question #1

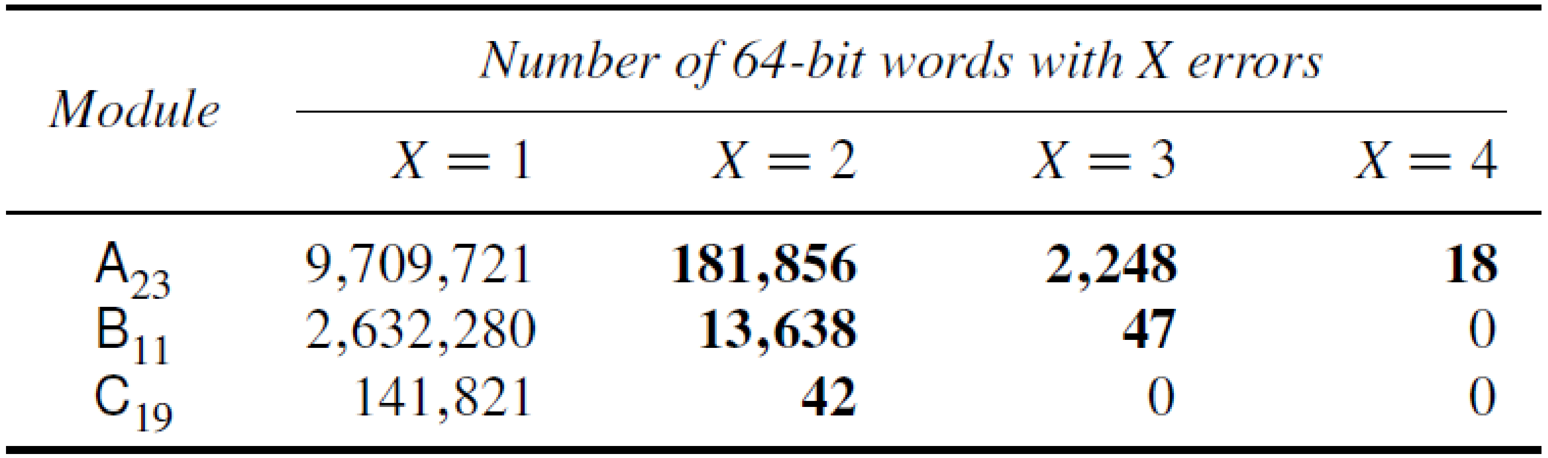
What Did the Paper Get Right?

State the 3 most important things the paper says.

These could be some combination of the motivations, observations, interesting parts of the design, or clever parts of the implementation.

Uncorrectable Multi-Bit Errors

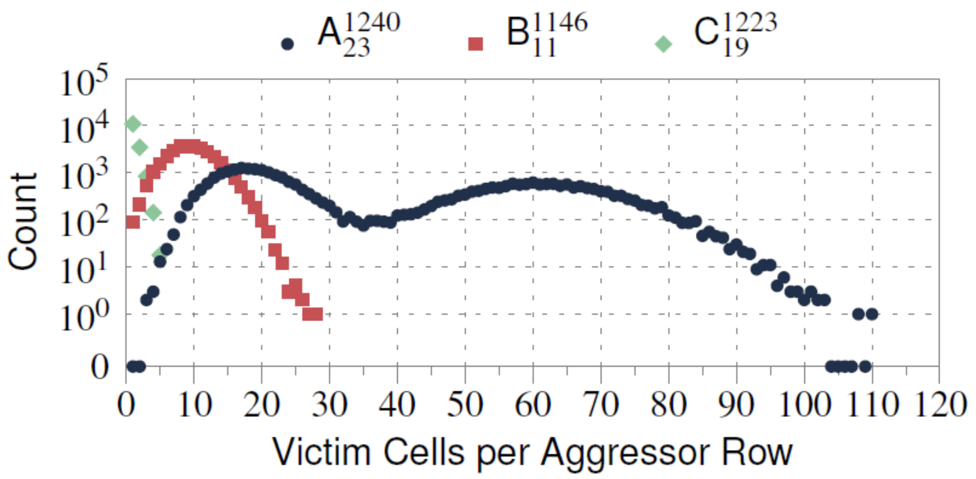
For single error-correction codes on 64-bit words

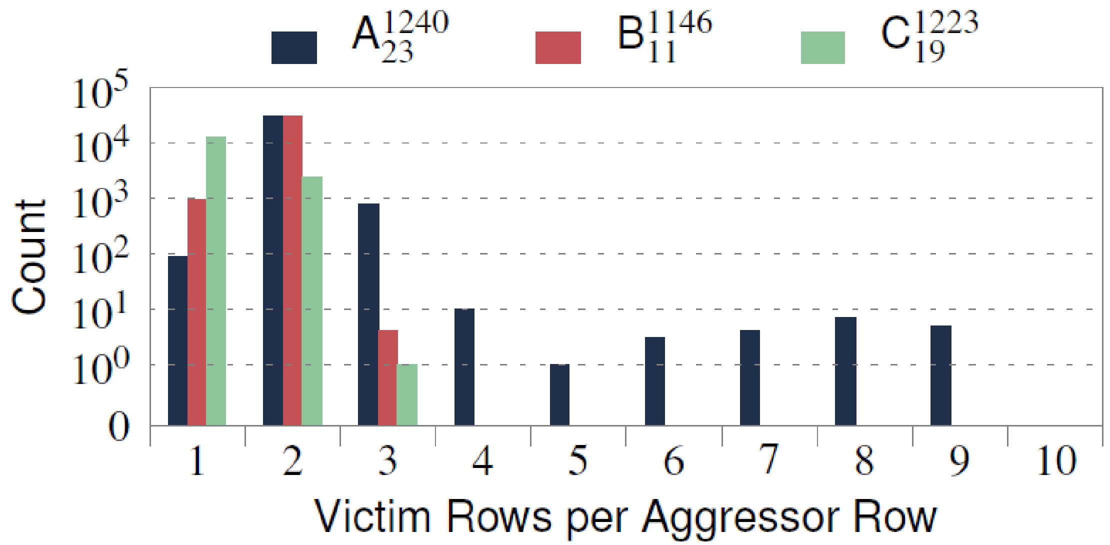


Module	*Number of 64-bit words with X errors*			
	$X = 1$	$X = 2$	$X = 3$	$X = 4$
A_{23}	9,709,721	**181,856**	**2,248**	**18**
B_{11}	2,632,280	**13,638**	**47**	0
C_{19}	141,821	**42**	0	0

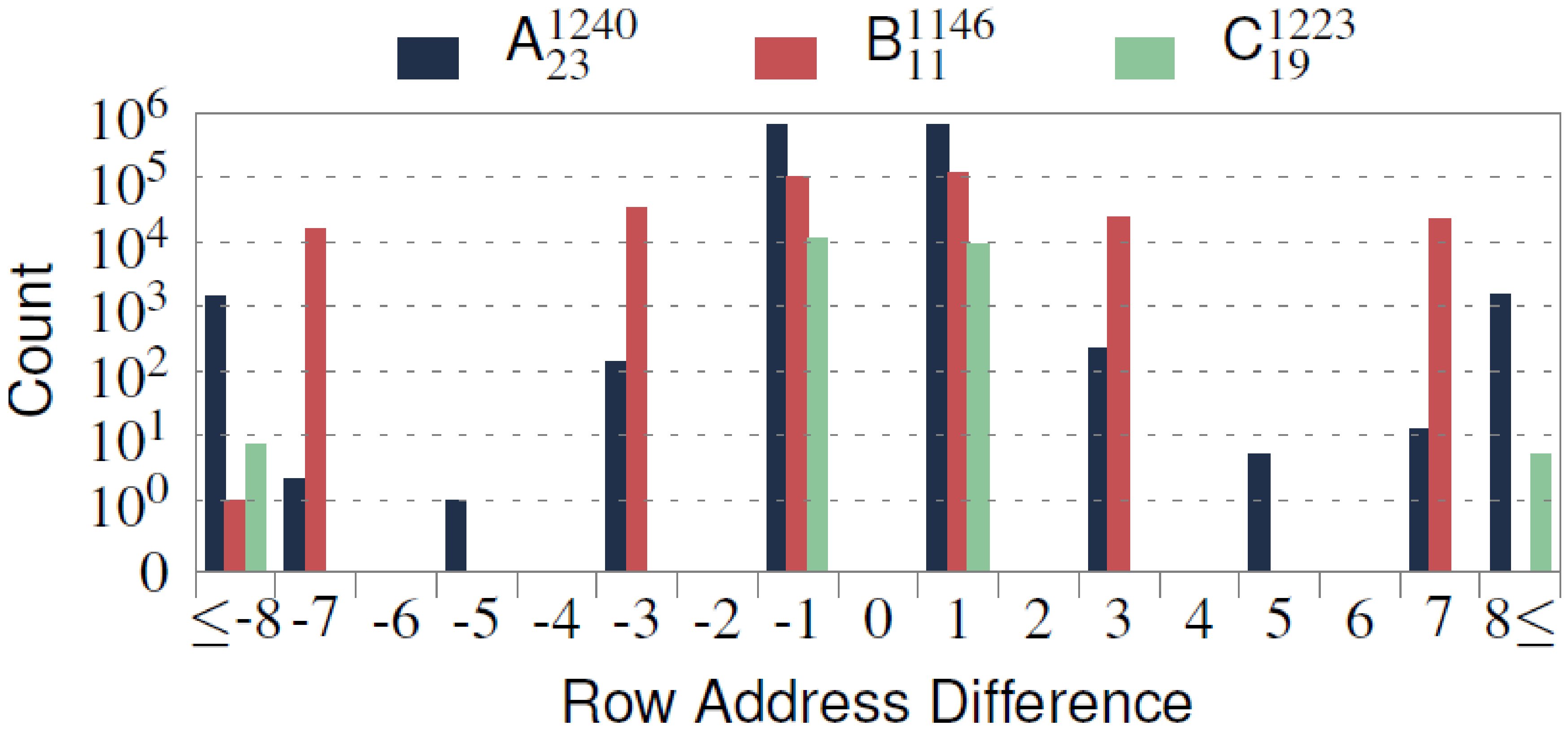
Table 5. Uncorrectable multi-bit errors (in bold)

Victim Cells/Rows per Aggressor Row



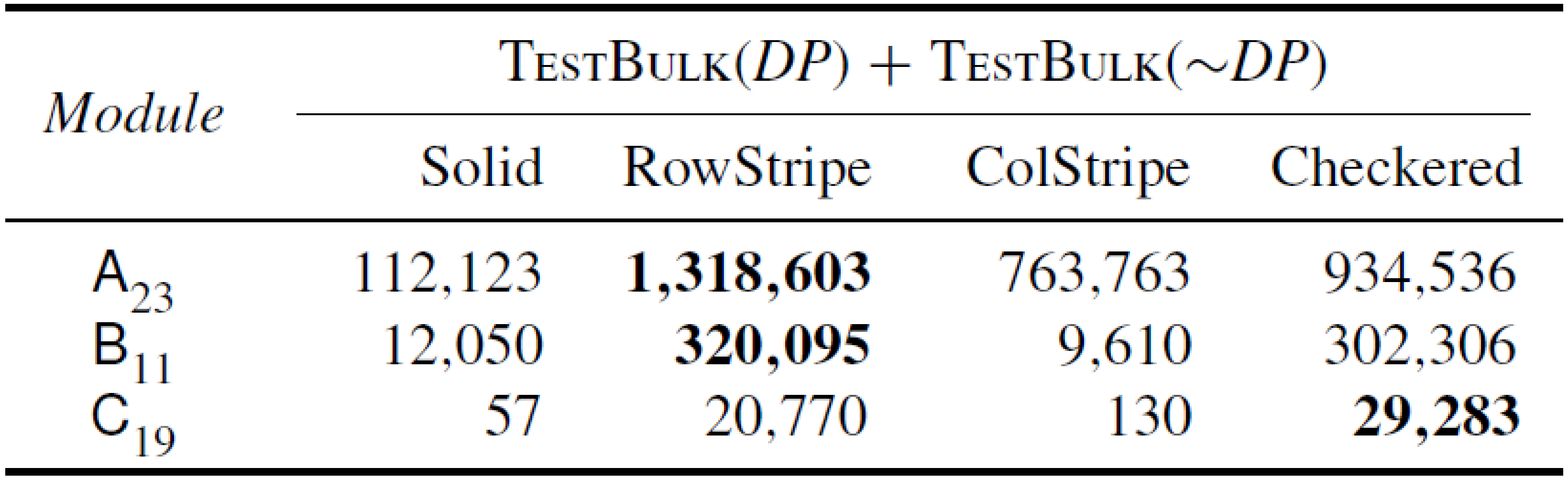


Which Rows Affected by Aggressor Row



Causes for ^{-1,1}: Manufacturer-dependent mappings, remappings

Num Errors for Different Data Patterns



Module	TestBulk(DP) + TestBulk($\sim DP$)			
	Solid	RowStripe	ColStripe	Checkered
A_{23}	112,123	**1,318,603**	763,763	934,536
B_{11}	12,050	**320,095**	9,610	302,306
C_{19}	57	20,770	130	**29,283**

Error is always in discharge direction,
which can be 1->0 (for true-cells) or 0->1 (for anti-cells).

DP matters in some complicated way.

Sensitivity Results

- Error are mostly repeatable
- Victim cells $\neq$ Weak (leakier) cells
- Not strongly effected by temperature

Discussion: Summary Question #2

What Did the Paper Get Wrong?

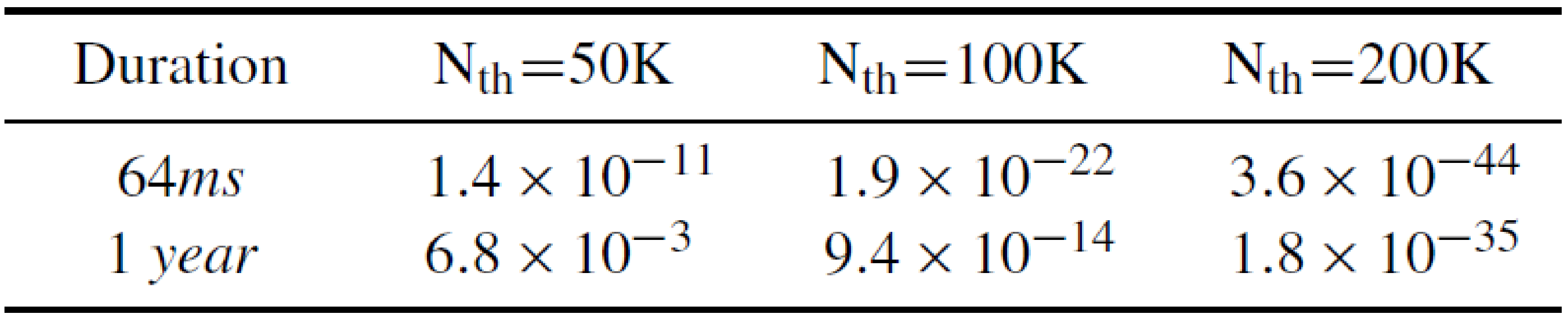
Describe the paper's single most glaring deficiency.

Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.

Possible Row Hammer Mitigations

- Make better chips
- Correct errors
- Refresh all rows frequently
- Retire cells (manufacturer)
- Retire cells (end-user)
- Identify “hot” rows & refresh neighbors
- PARA – next slide

PARA: Probabilistic Adjacent Row Refresh



Duration	N_{th}=50K	N_{th}=100K	N_{th}=200K
$64ms$	1.4×10^{-11}	1.9×10^{-22}	3.6×10^{-44}
1 *year*	6.8×10^{-3}	9.4×10^{-14}	1.8×10^{-35}

Table 7. Error probabilities for PARA when p=0.001

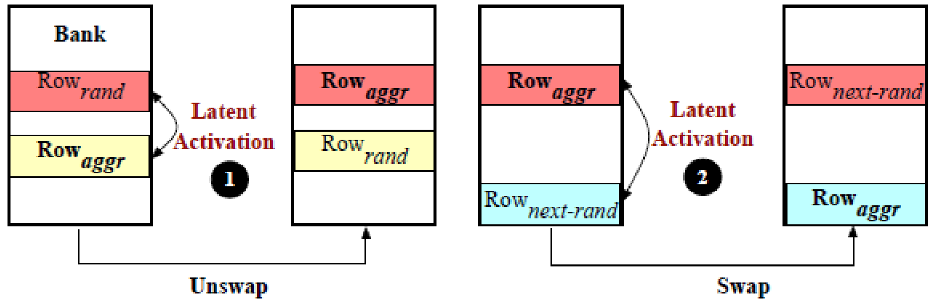
Requires manufacturers to expose their mappings & remappings

Higher & higher *p* to keep up with lower N_{th} =>
Frequent refreshes are an attack on their neighbors!

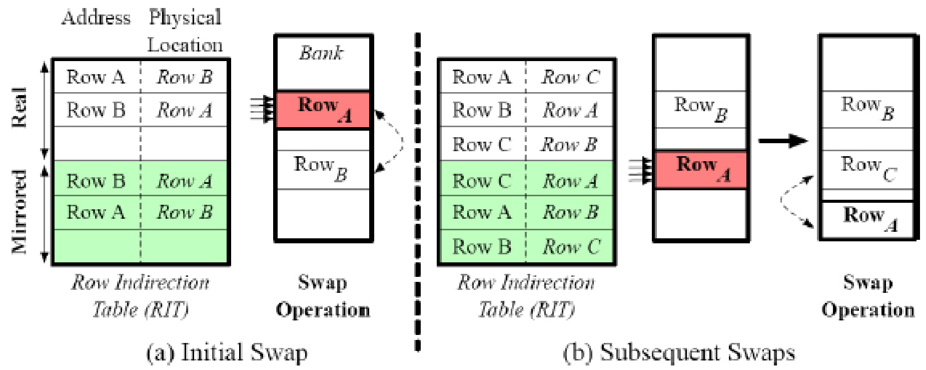
"Scalable and Secure Row-Swap: Efficient and Safe Row Hammer Mitigation in Memory Systems"

Jeonghyun Woo, Gururaj Saileshwar, Prashant J. Nair 2023

Randomized Row-Swap (prior SOTA) is not secure



Instead, don't unswap

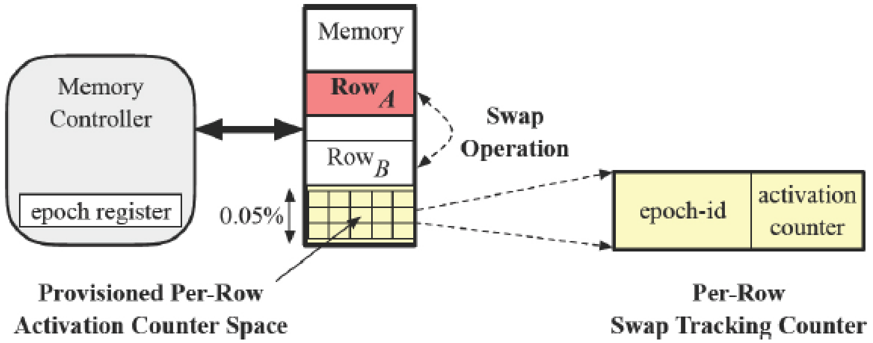


Then do periodic lazy eviction of RIT

“Scalable and Secure Row-Swap: Efficient and Safe Row Hammer Mitigation in Memory Systems”

Jeonghyun Woo, Gururaj Saileshwar, Prashant J. Nair 2023

Finally, track swap counts

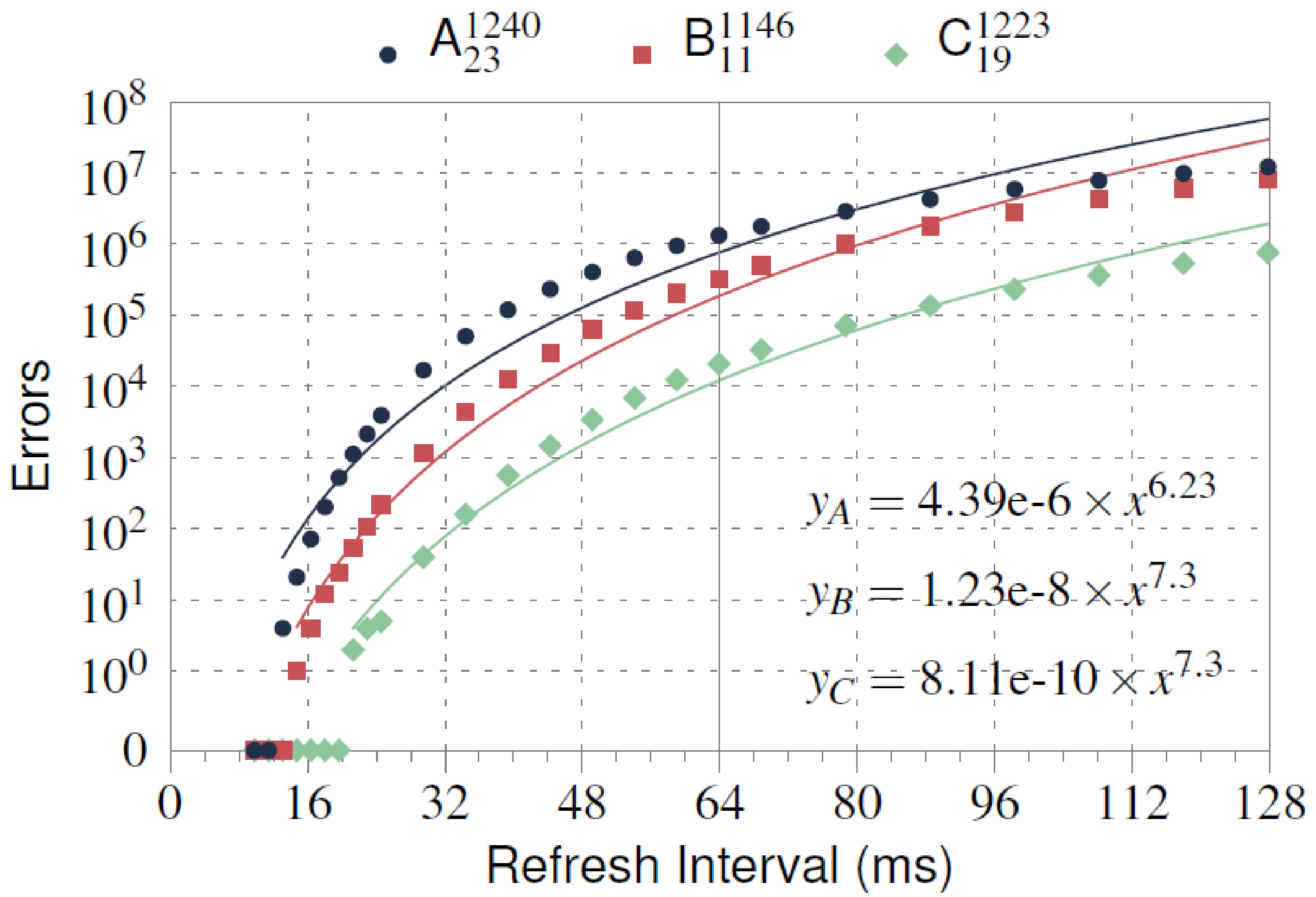


When too much swapping, pin row in LLC for refresh epoch

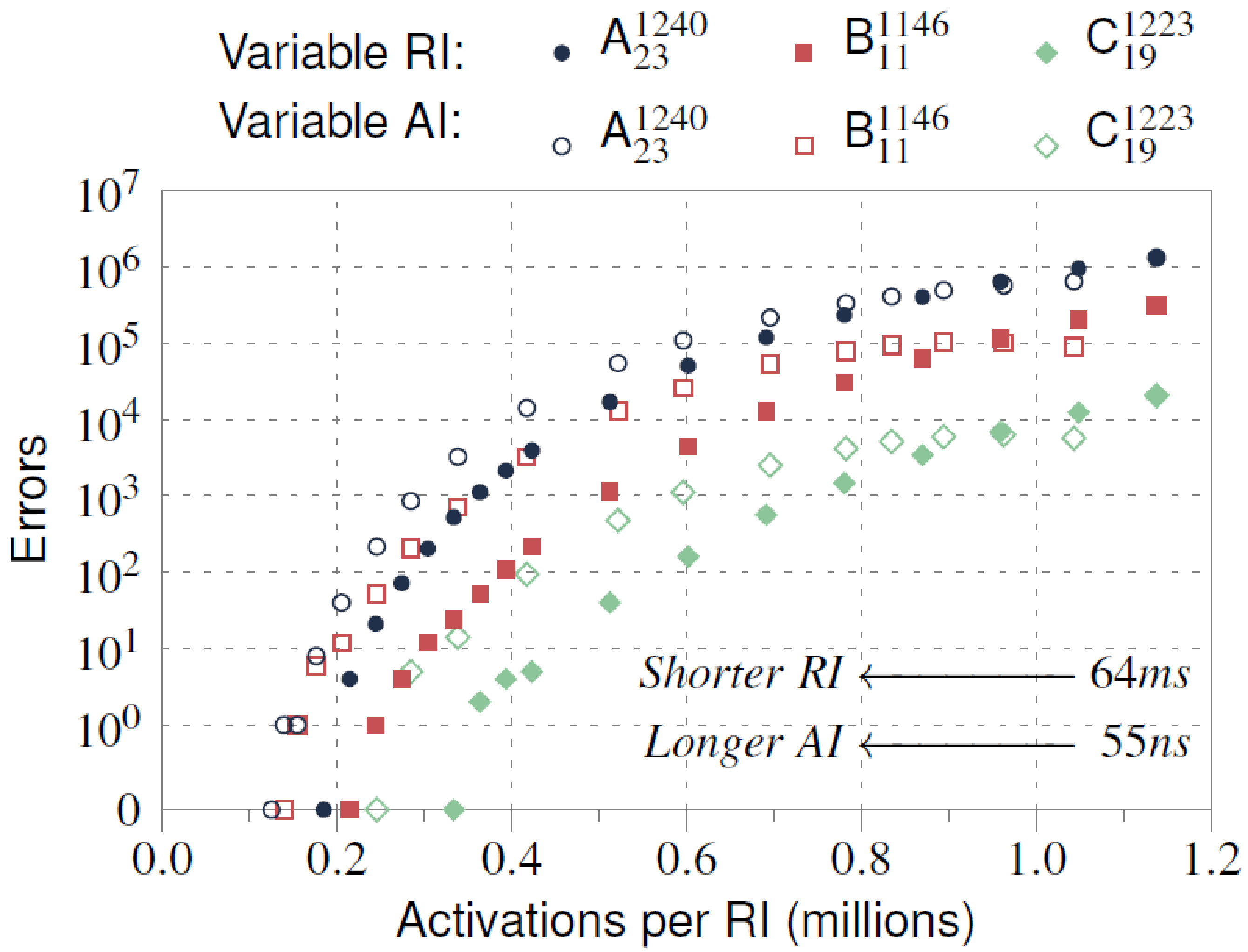
All accesses hit in LLC and don’t go to memory

BACKUP SLIDES

Num Errors Varying Refresh Interval



Num Errors Varying Activations/RI


Variable RI: ● A_{23}^{1240} ■ B_{11}^{1146} ◆ C_{19}^{1223}
Variable AI: ○ A_{23}^{1240} □ B_{11}^{1146} ◇ C_{19}^{1223}
Errors
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
Activations per RI (millions)
Shorter RI ← 64ms
Longer AI ← 55ns