18-447
Computer Architecture
Recitation 1

Kevin Chang
Carnegie Mellon University

Spring 2015, 1/23/2015

Agenda for Today

- Quick recap on the previous lectures
- Practice questions
- Q&A on HW1, lab1, and lecture materials

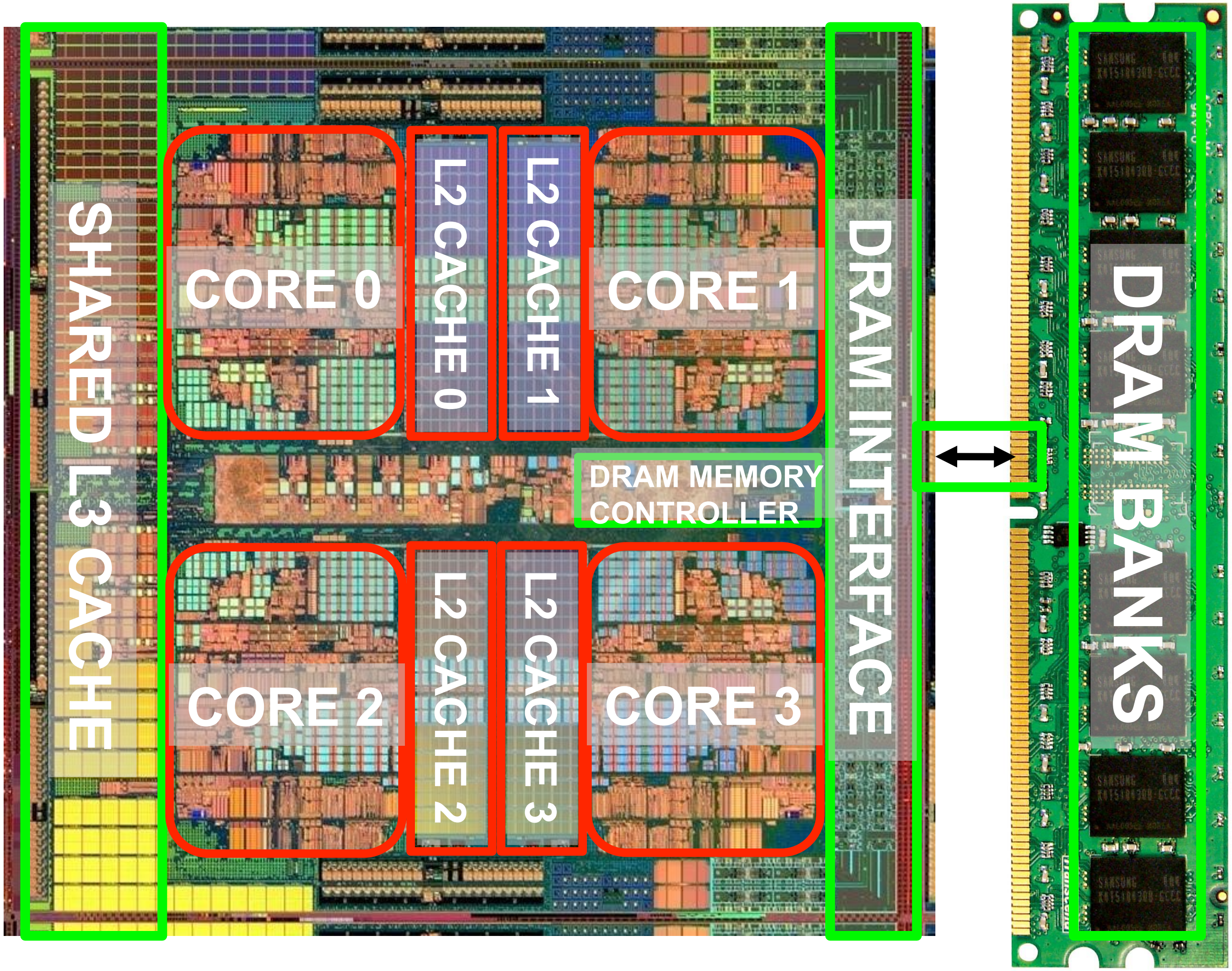
- Important deadlines:
  - **Lab 1** due tonight at 11:59:59 PM. Handin through **AFS.**
  - Wednesday (1/28): **HW 1** due

Quick Review

- DRAM-based memory system
  - Cells, banks, refresh, performance hog, row hammer

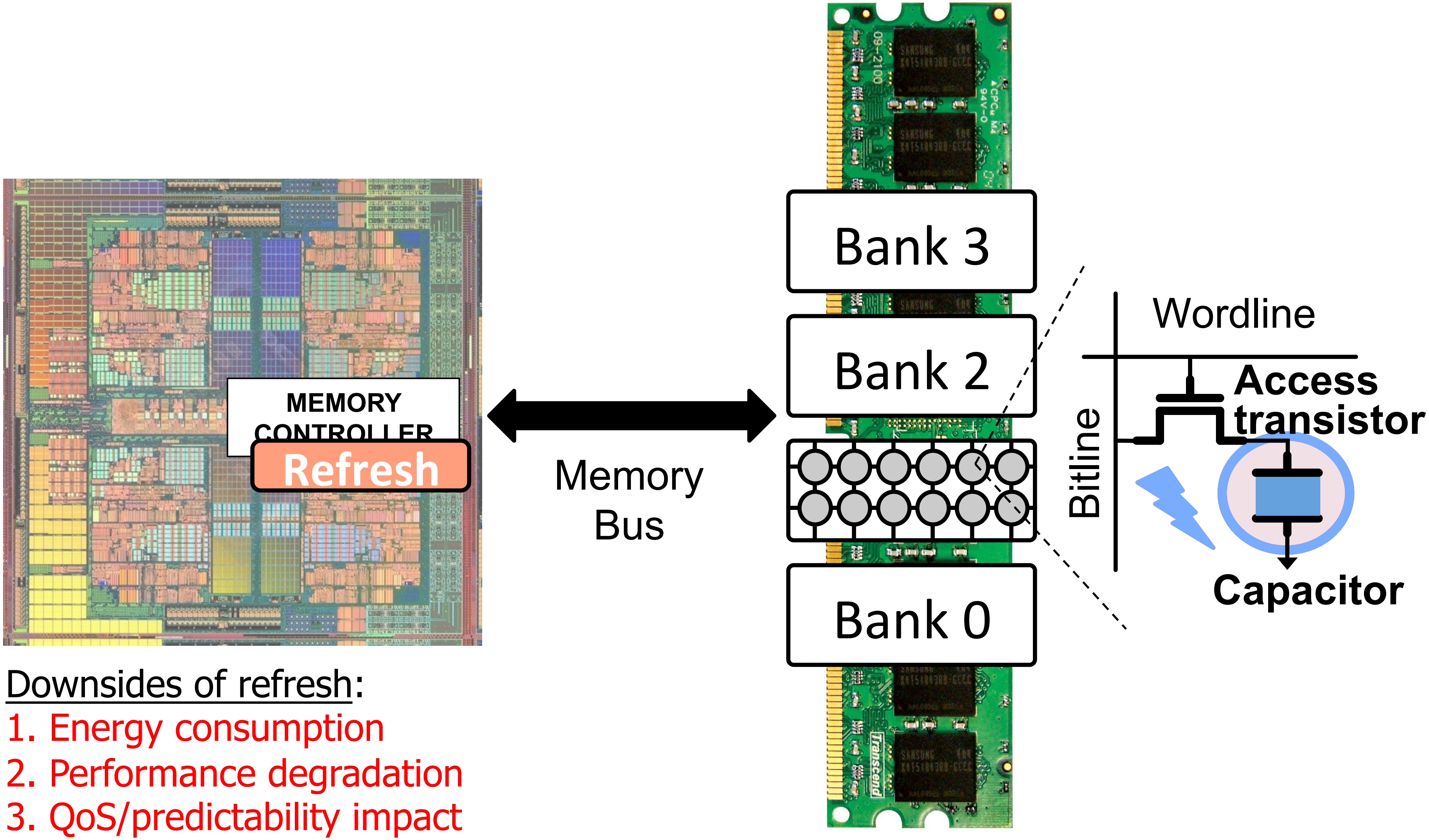
DRAM in the System

Multi-Core Chip



*Die photo credit: AMD Barcelona

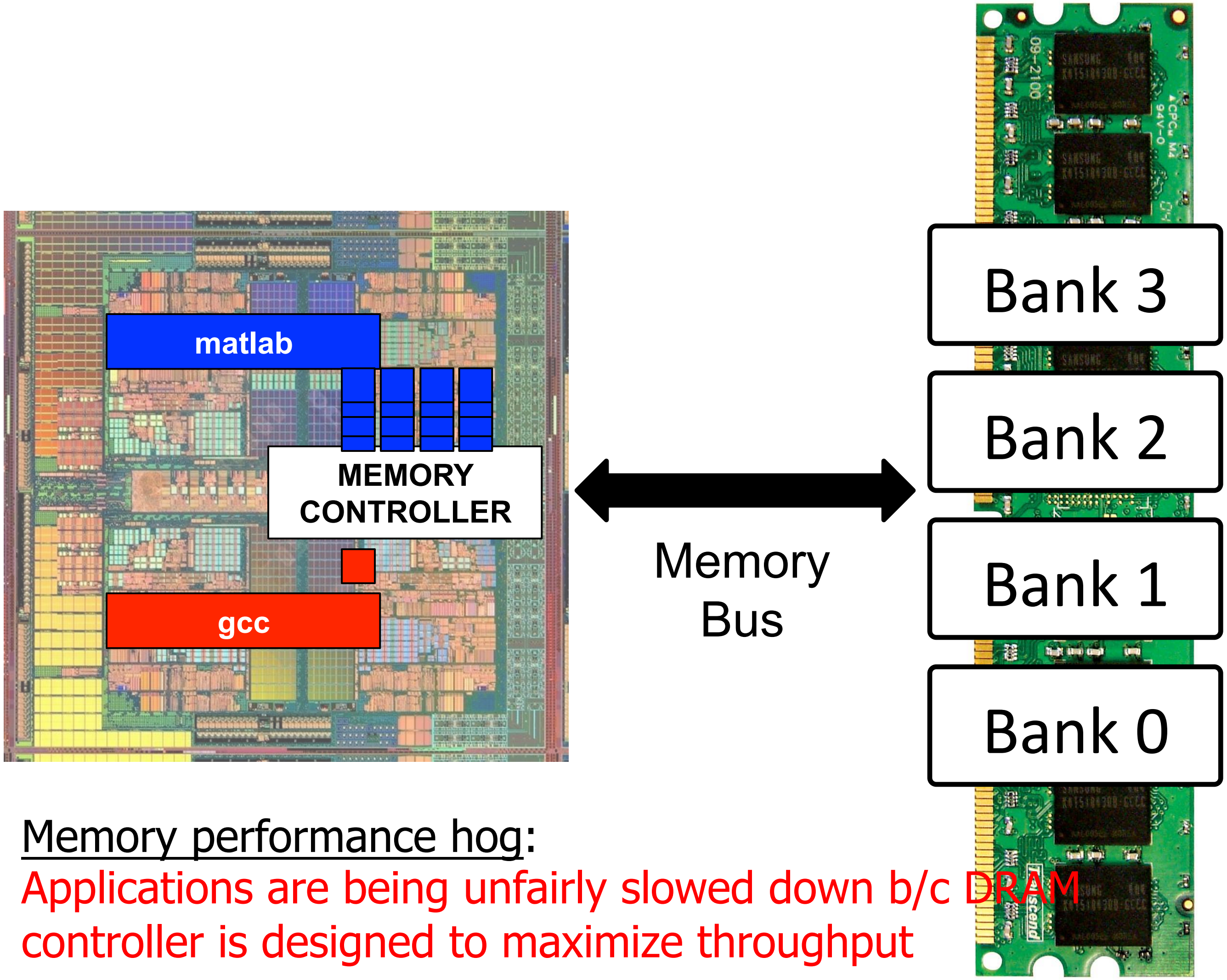
DRAM in the System: Refresh



Downsides of refresh:

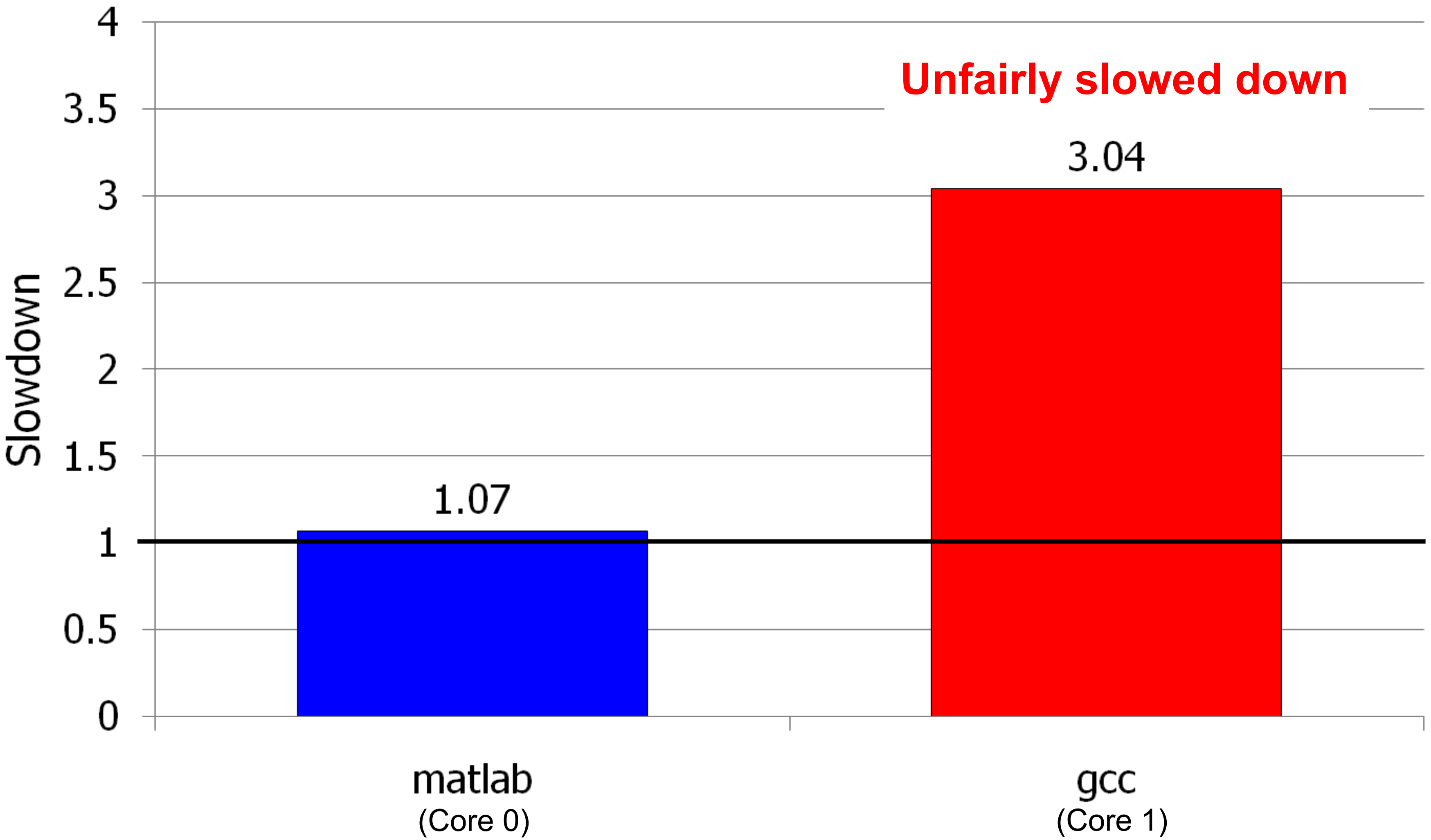
1. Energy consumption
2. Performance degradation
3. QoS/predictability impact
4. Refresh rate limits DRAM capacity scaling

DRAM in the System: Performance Hog



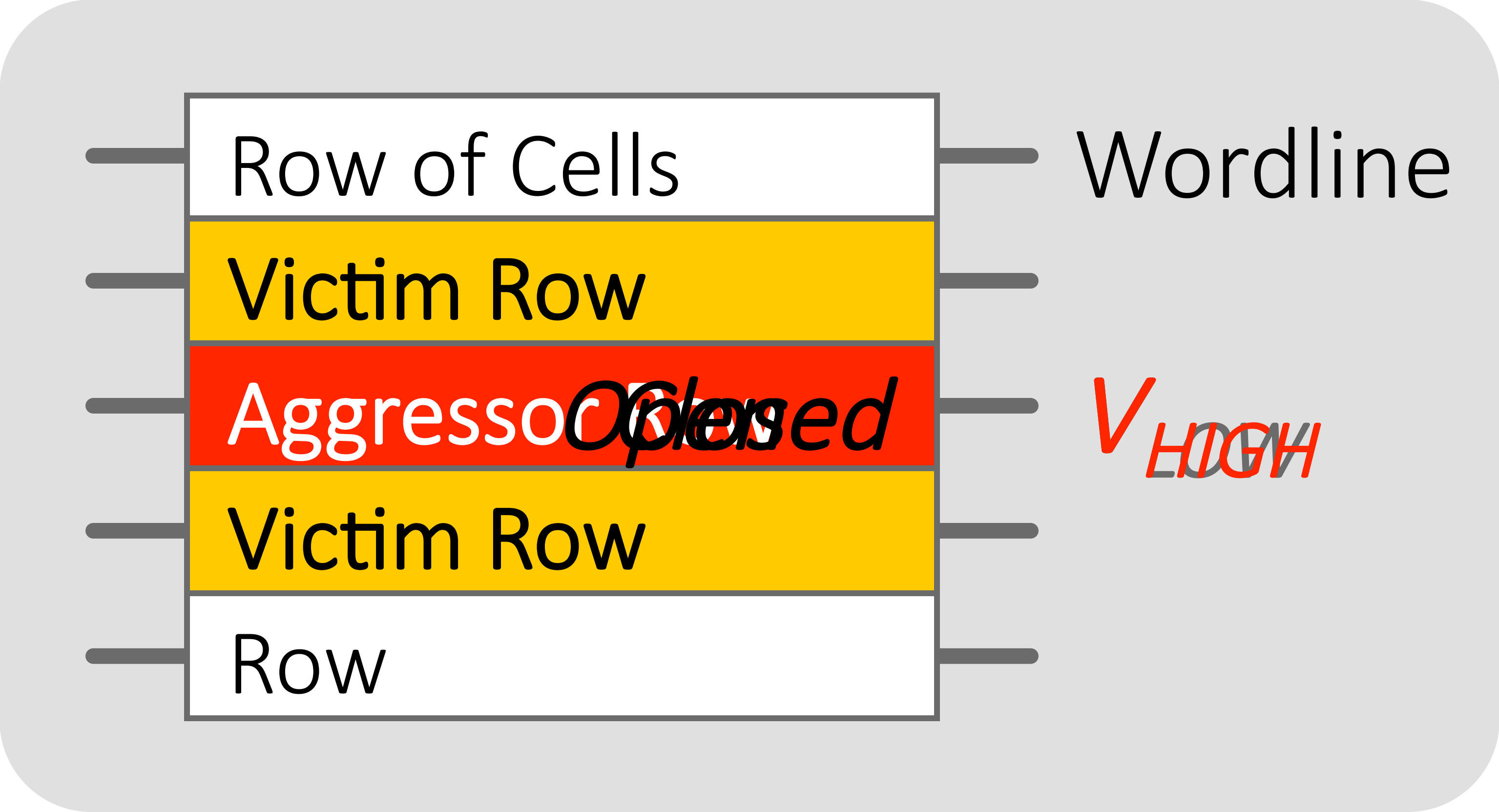
Memory performance hog:
Applications are being unfairly slowed down b/c DRAM controller is designed to maximize throughput

Unexpected Slowdowns in Multi-Core



Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

Disturbance Errors in Modern DRAM



Row of Cells

Victim Row

Aggressor Row

Opened

Closed

Victim Row

Row

Wordline

V_{LOW}

V_{HIGH}

Repeatedly opening and closing a row enough times within a refresh interval induces disturbance errors in adjacent rows in most real DRAM chips you can buy today

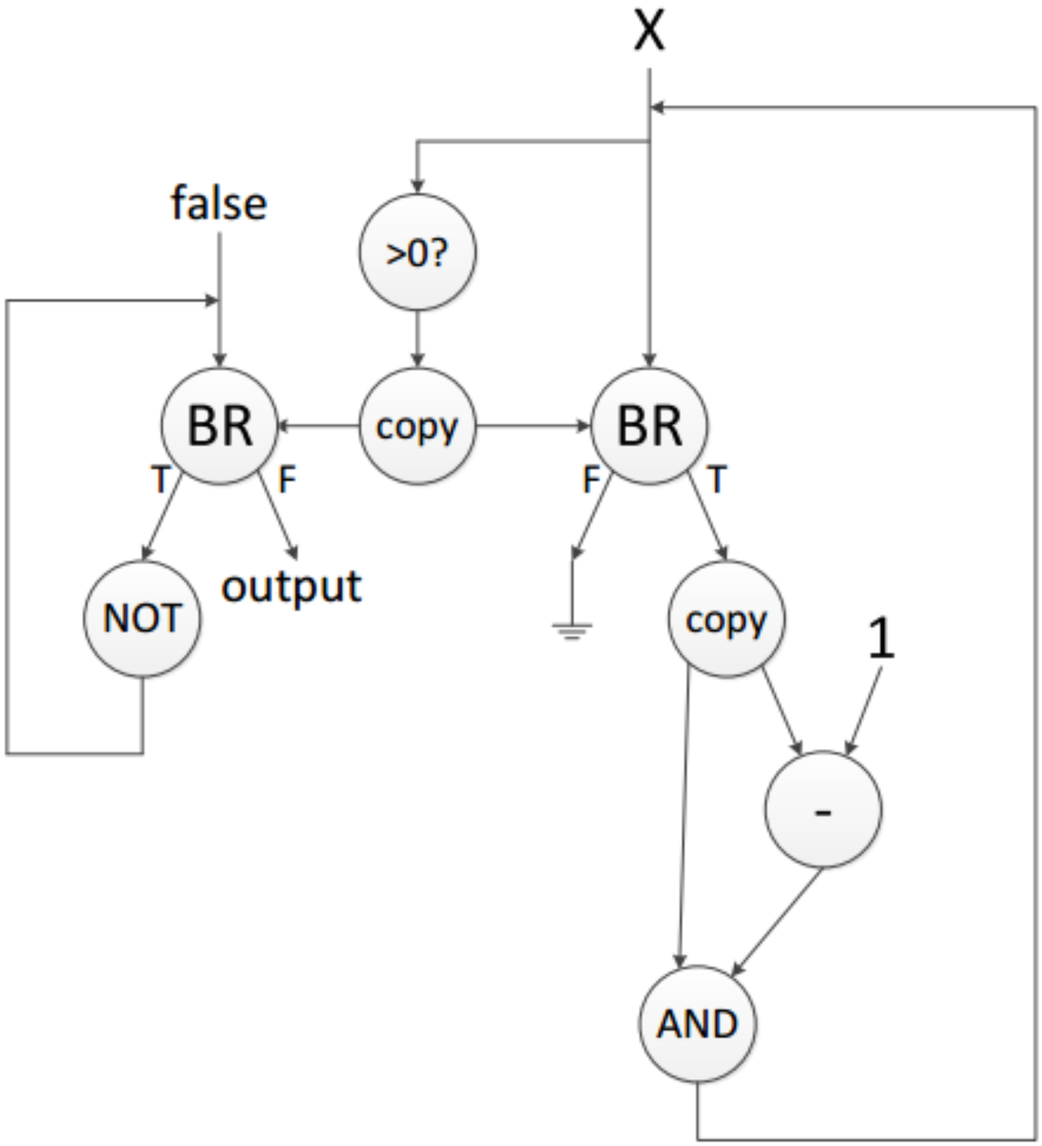
Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

Quick Review

- DRAM-based memory system
  - Cells, banks, refresh, row hammer, performance hog
- Key components of a computer
- The von Neumann vs. dataflow model
- ISA vs. microarchitecture
- Elements of an ISA
  - Instructions: opcodes, data types, registers, formats, etc
  - Memory: address space, addressing modes, alignment, etc
- ISA tradeoffs
  - CISC vs. RISC
  - Semantic gap

Practice Questions

Practice Question 1: Dataflow



Practice Question 2: MIPS ISA

```
int foo(int *A, int n) {
   int s;
   if (n>=2) {
      s=foo(A, n-1);
      s=s+A[n-2];
   }
   else {
      s=1;
   }
   A[n]=s+1;
   return A[n];
}
```

MIPS Assembly →

```
_foo:
  // TODO
_branch:
  // TODO
_true:
  // TODO
_false:
  // TODO
_join:
  // TODO
_done:
  // TODO
```

1. **A** and **n** are passed in to **r4** and **r5**
2. Result should be returned in **r2**, and **r31** stores the return address
3. **r29** (stack ptr), **r8-r15** (caller saved), **r16-r23** (called saved)

Q & A