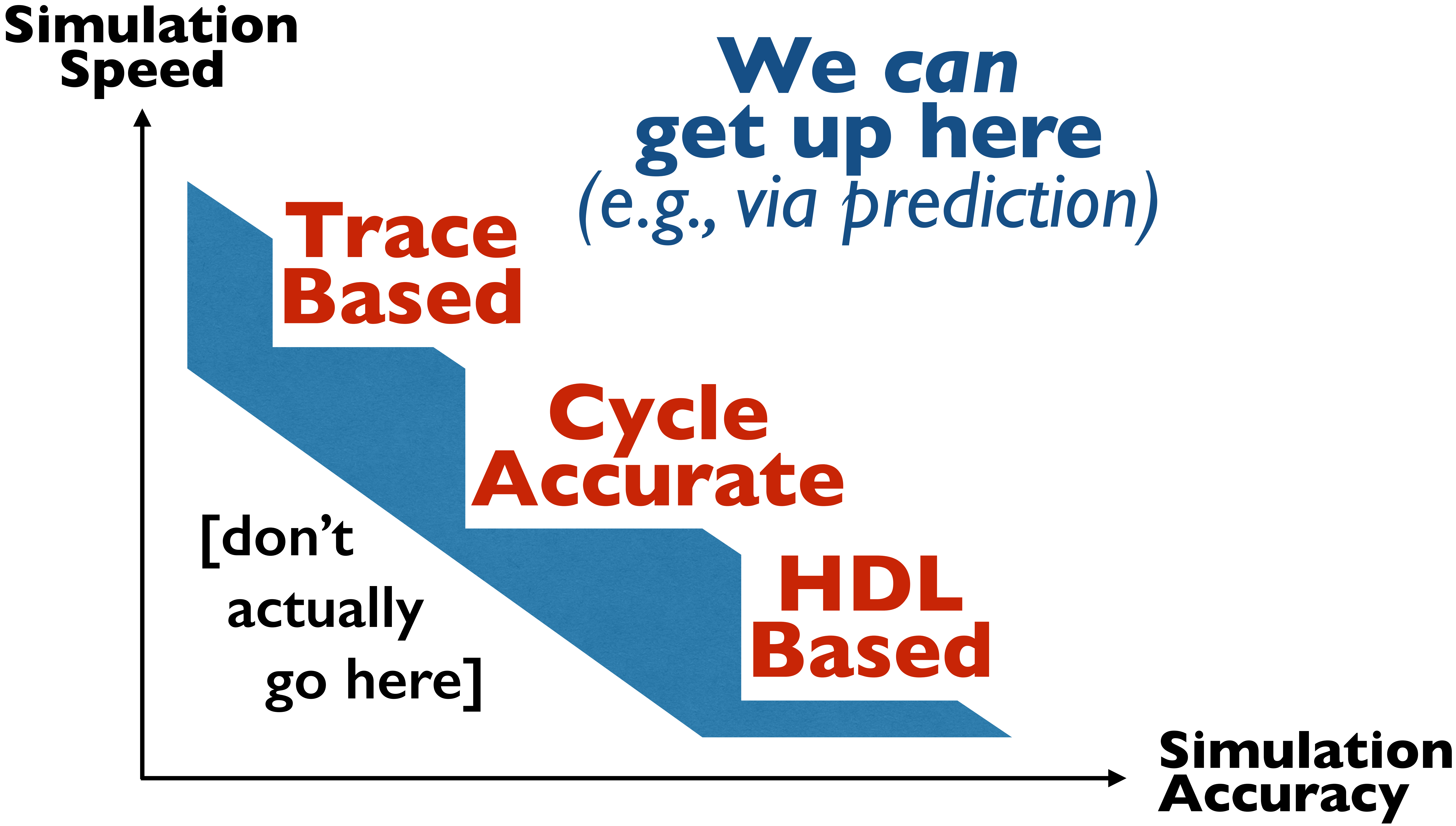
Faster and Accurater

The Future of Memory-System Modeling and Simulation

Bruce Jacob *(with Ph.D. results of Shang Li)*

Keystone Professor
University of Maryland

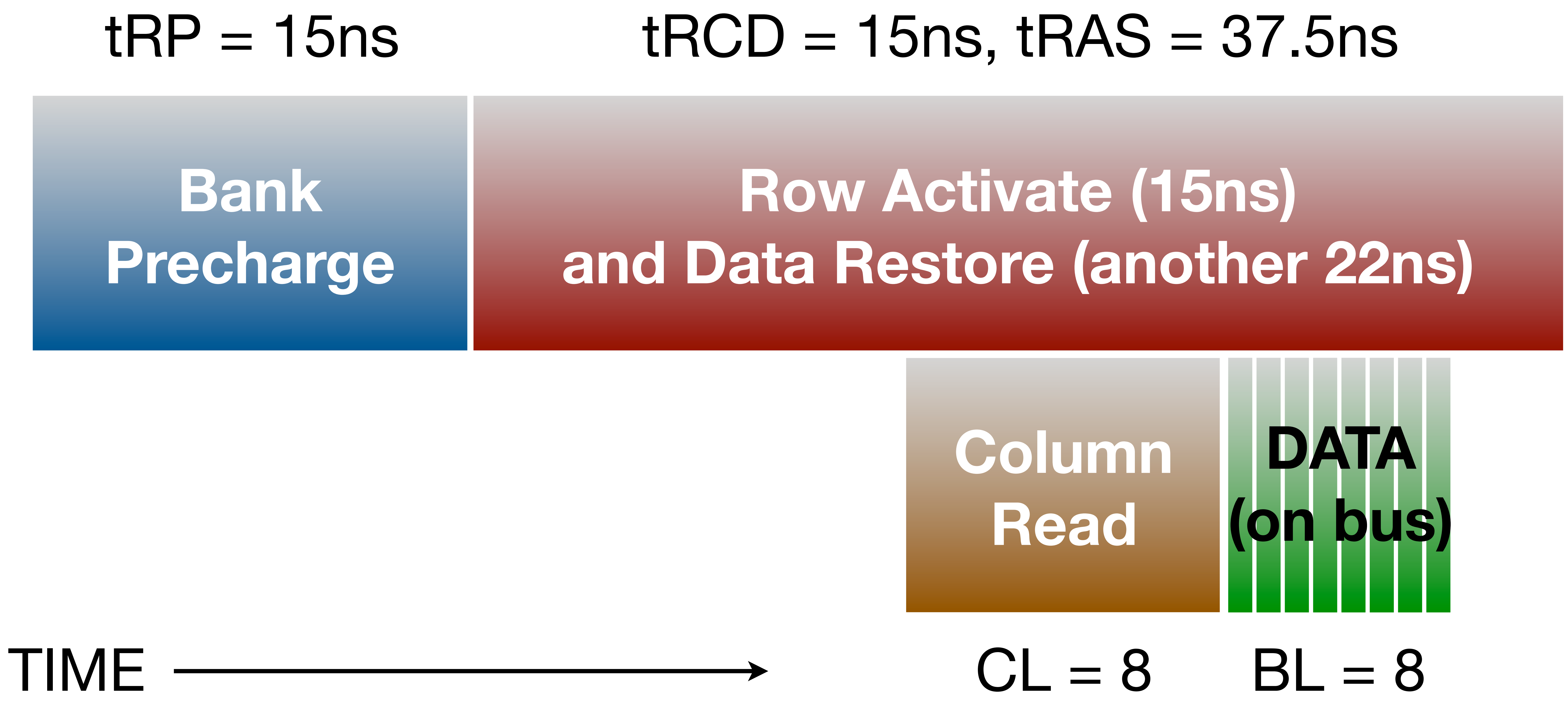
The Bottom Line



We *can* get up here *(e.g., via prediction)*

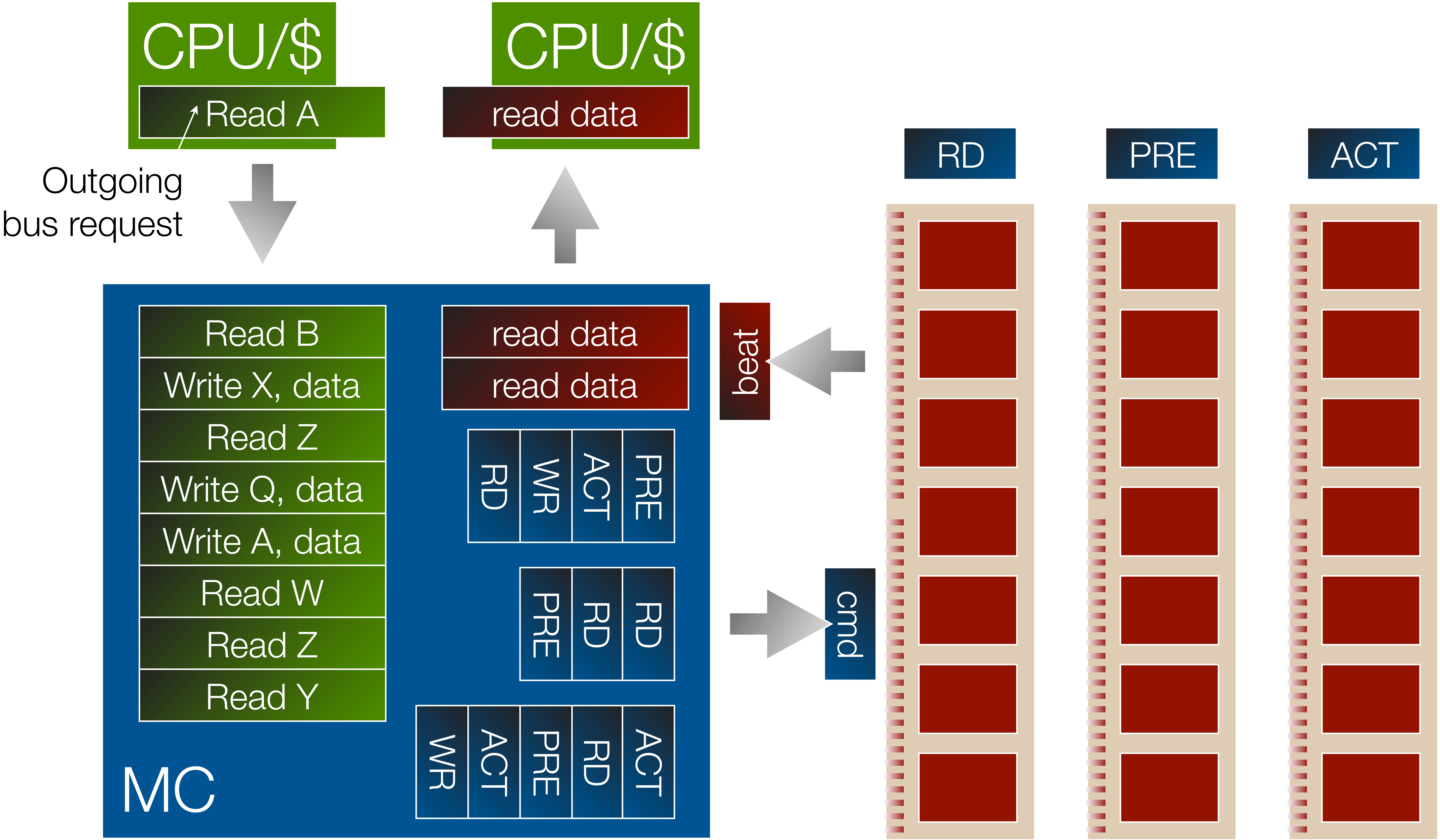
[don't actually go here]

Background


tRP = 15ns
tRCD = 15ns, tRAS = 37.5ns
Bank Precharge
Row Activate (15ns) and Data Restore (another 22ns)
Column Read
DATA (on bus)
TIME
CL = 8
BL = 8

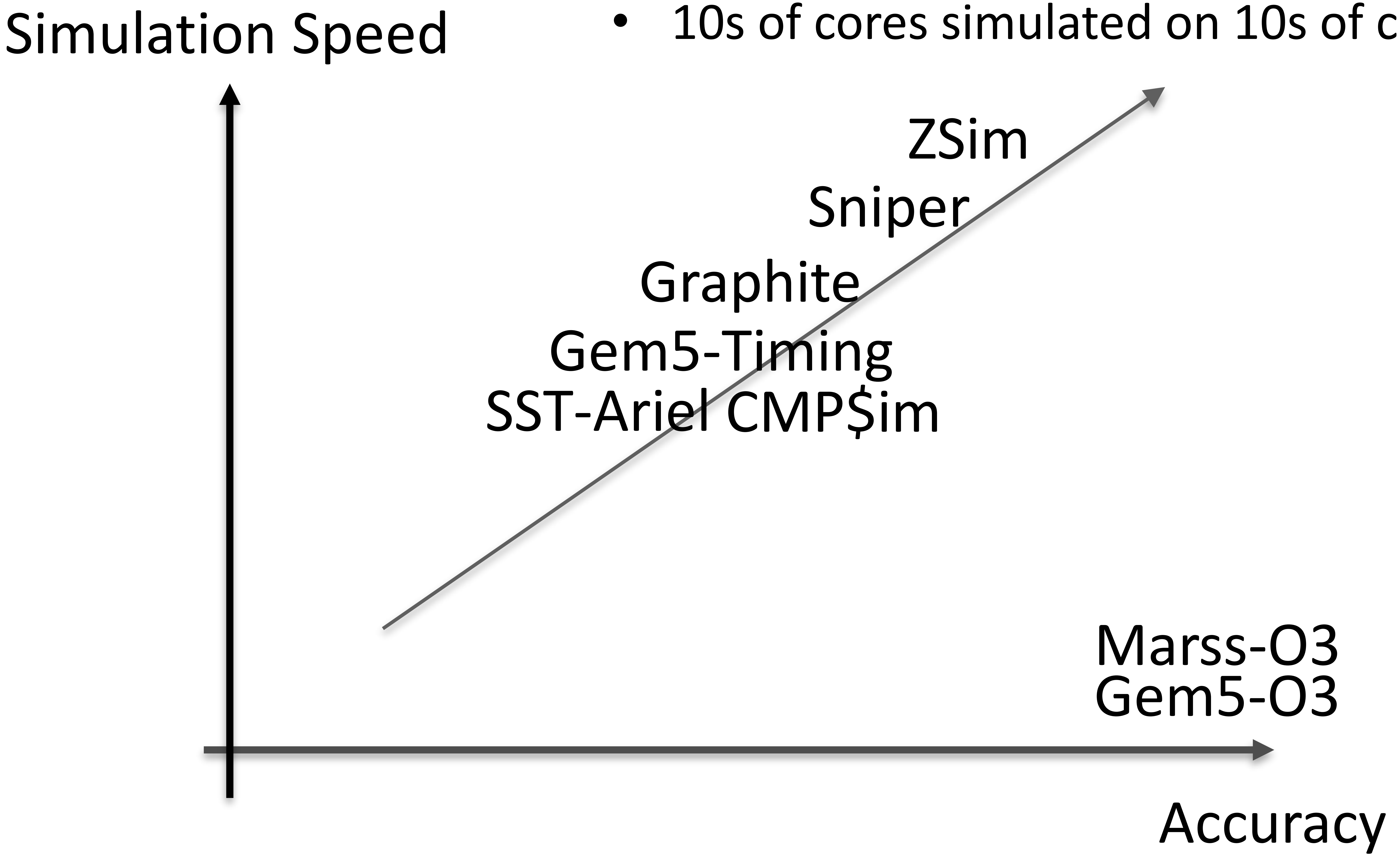
Cost of access is high; requires **significant effort** to amortize this over the (increasingly short) payoff.

Background ('significant effort')



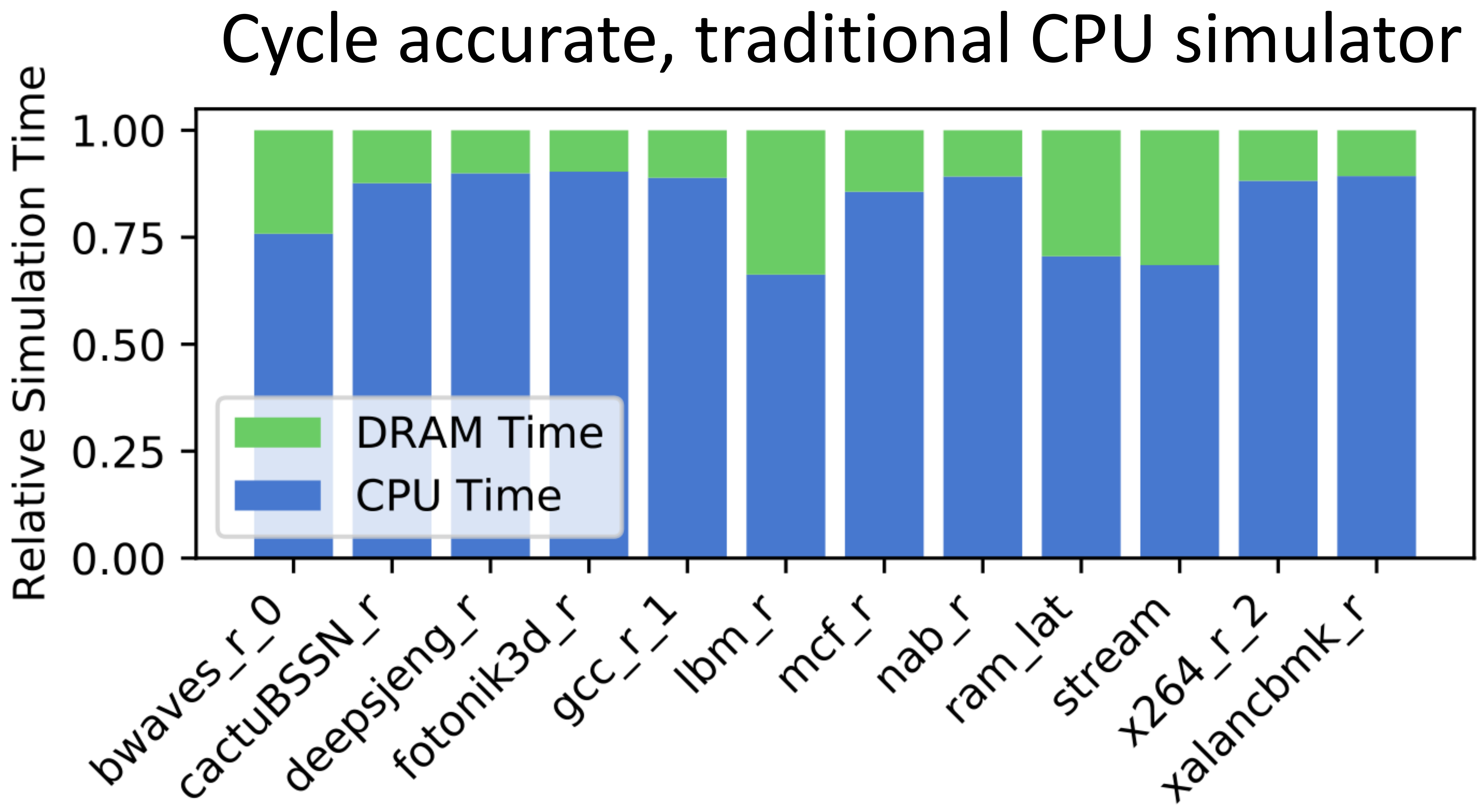
Faster?

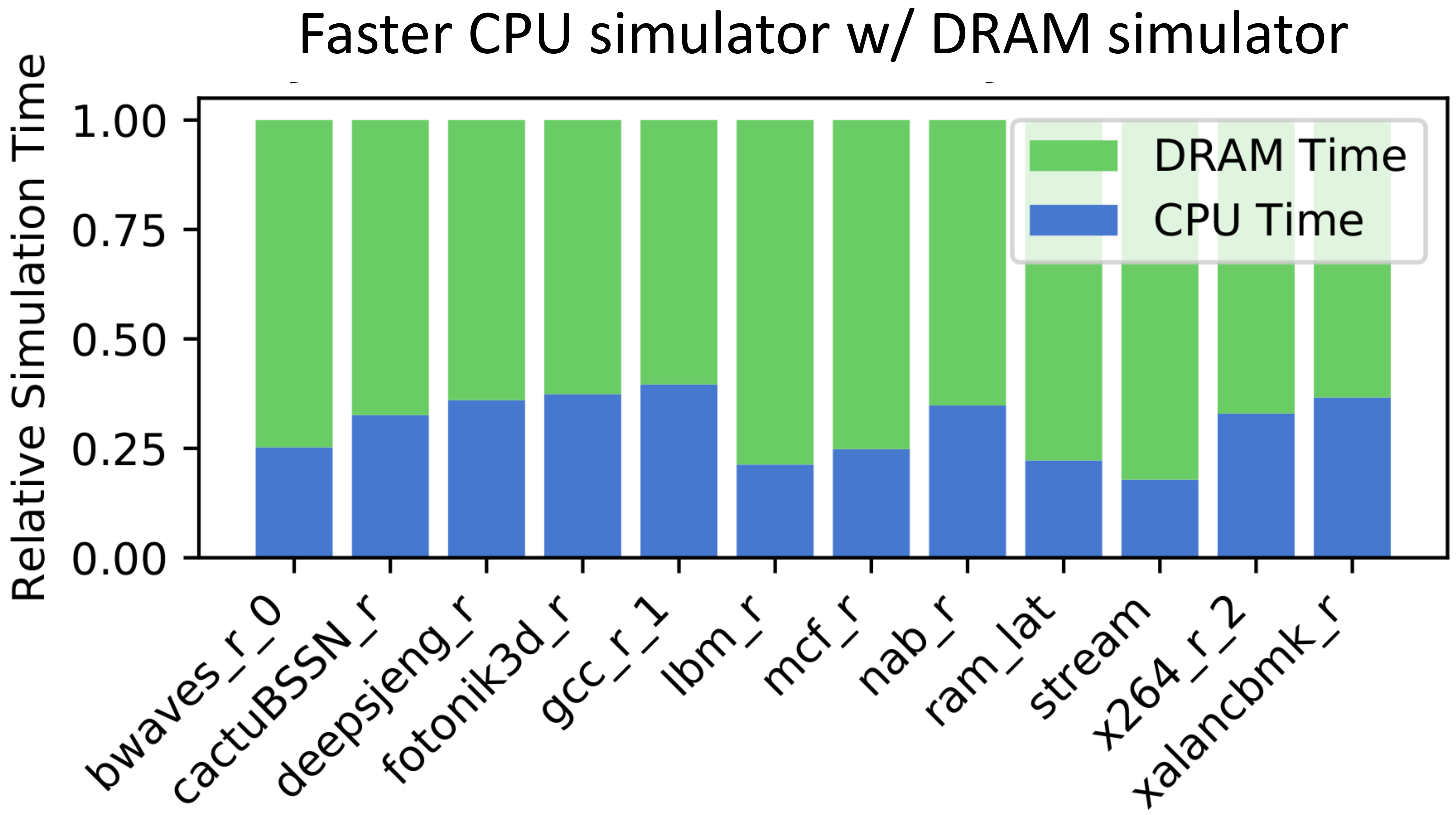
- Simulation speed: 100X faster
- Error: < 20%
- 10s of cores simulated on 10s of cores



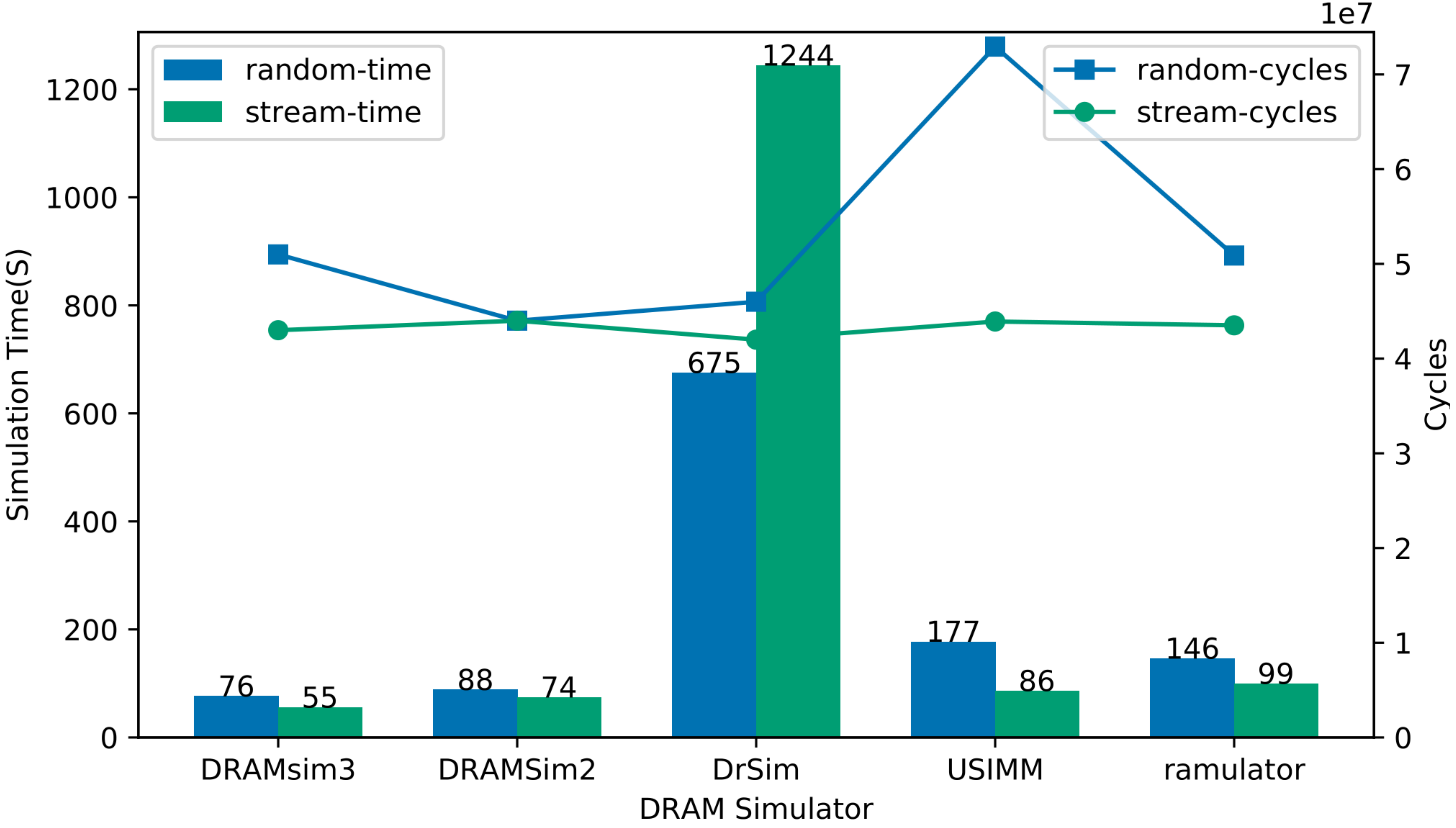
Faster?

Easily Predictable Result: *Memory-System Simulation is now Limiting Factor*


Cycle accurate, traditional CPU simulator


Faster CPU simulator w/ DRAM simulator

I'll just leave this here ...



Trace Simulation:
10M Random requests
10M Stream requests

Even Faster via Prediction

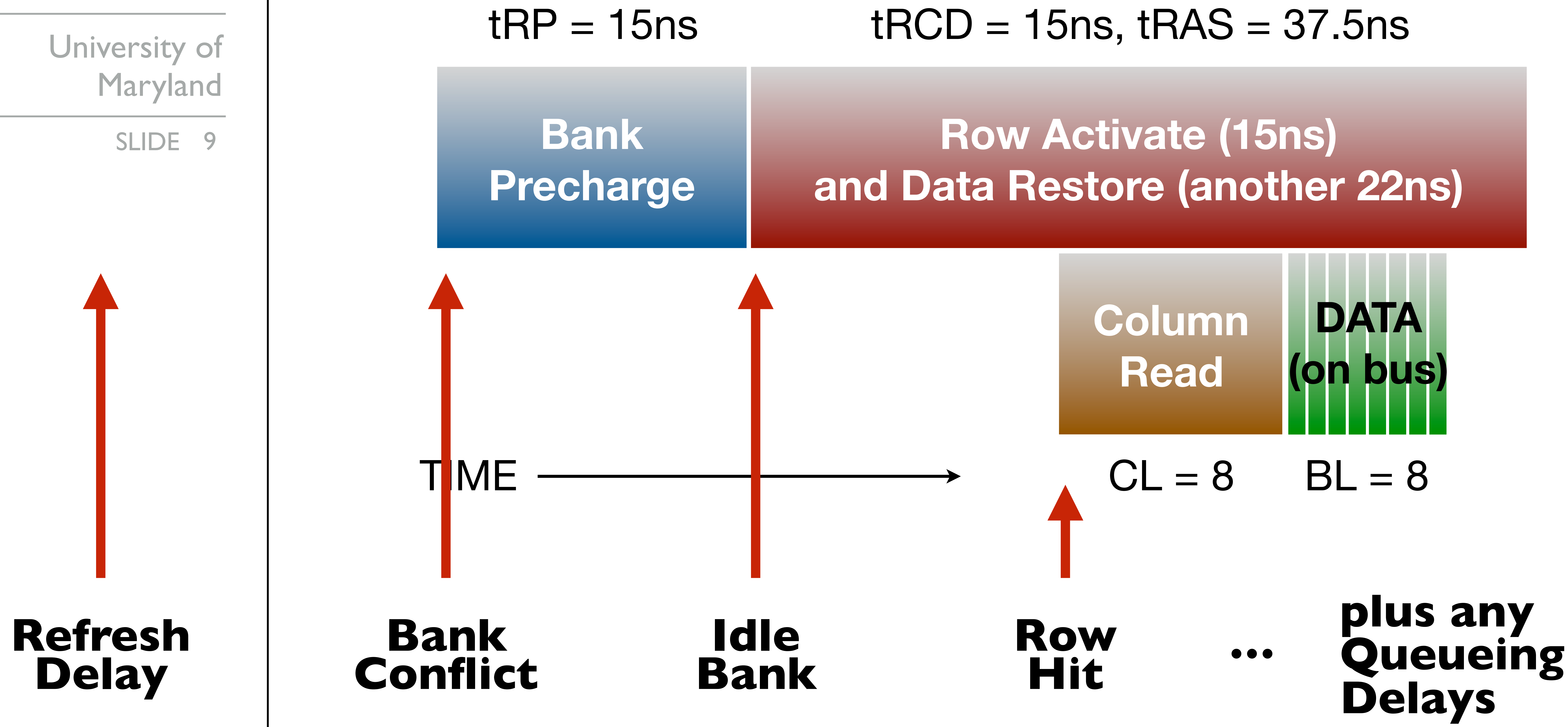
Statistical DRAM Model

Proposed Approach

Turning DRAM timing simulation into a classification problem

Clock	Address	OP		Class		Latency
0	0x01230000	READ	Classification →	Idle	Recovery →	36
12	0x01230020	READ		Row-Hit		22
40	0x0123003C	READ		Row-Hit		22
65	0x06340000	WRITE		Row-Miss		56
...	...	...		...		...

Latency ← Queue Contents



tRP = 15ns

tRCD = 15ns, tRAS = 37.5ns

Bank Precharge

Row Activate (15ns) and Data Restore (another 22ns)

Column Read

DATA (on bus)

TIME

CL = 8

BL = 8

Refresh Delay

Bank Conflict

Idle Bank

Row Hit

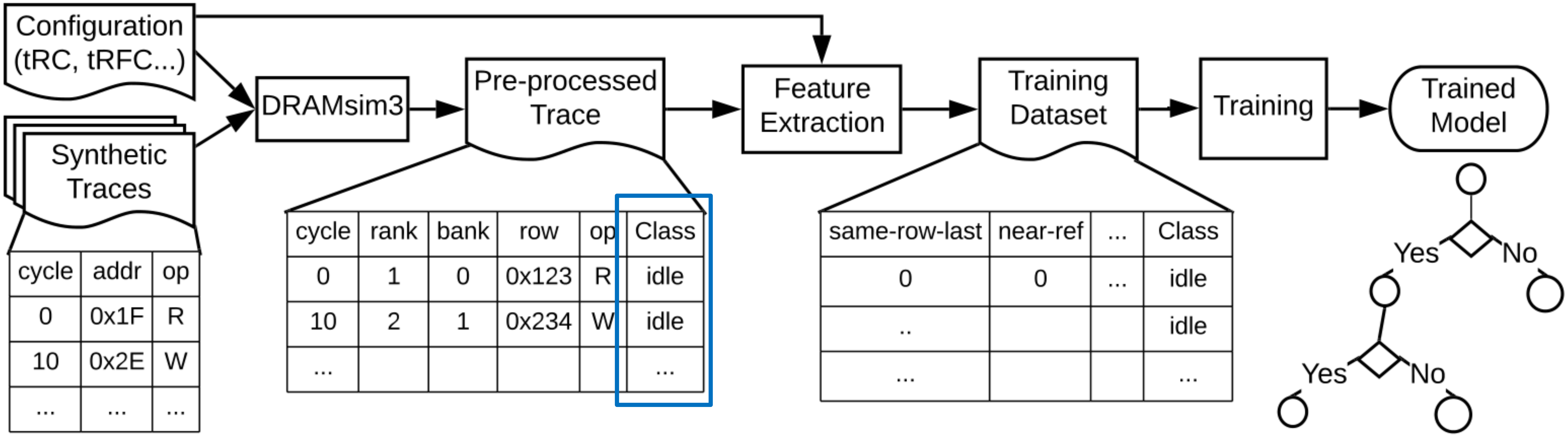
...

plus any Queueing Delays

Training Process

Training: Supervised Learning

Synthetic Trace:
~7000 Requests
Various access patterns, inter-arrival timings to cover all kinds of workloads



cycle	addr	op
0	0x1F	R
10	0x2E	W
...	...	...

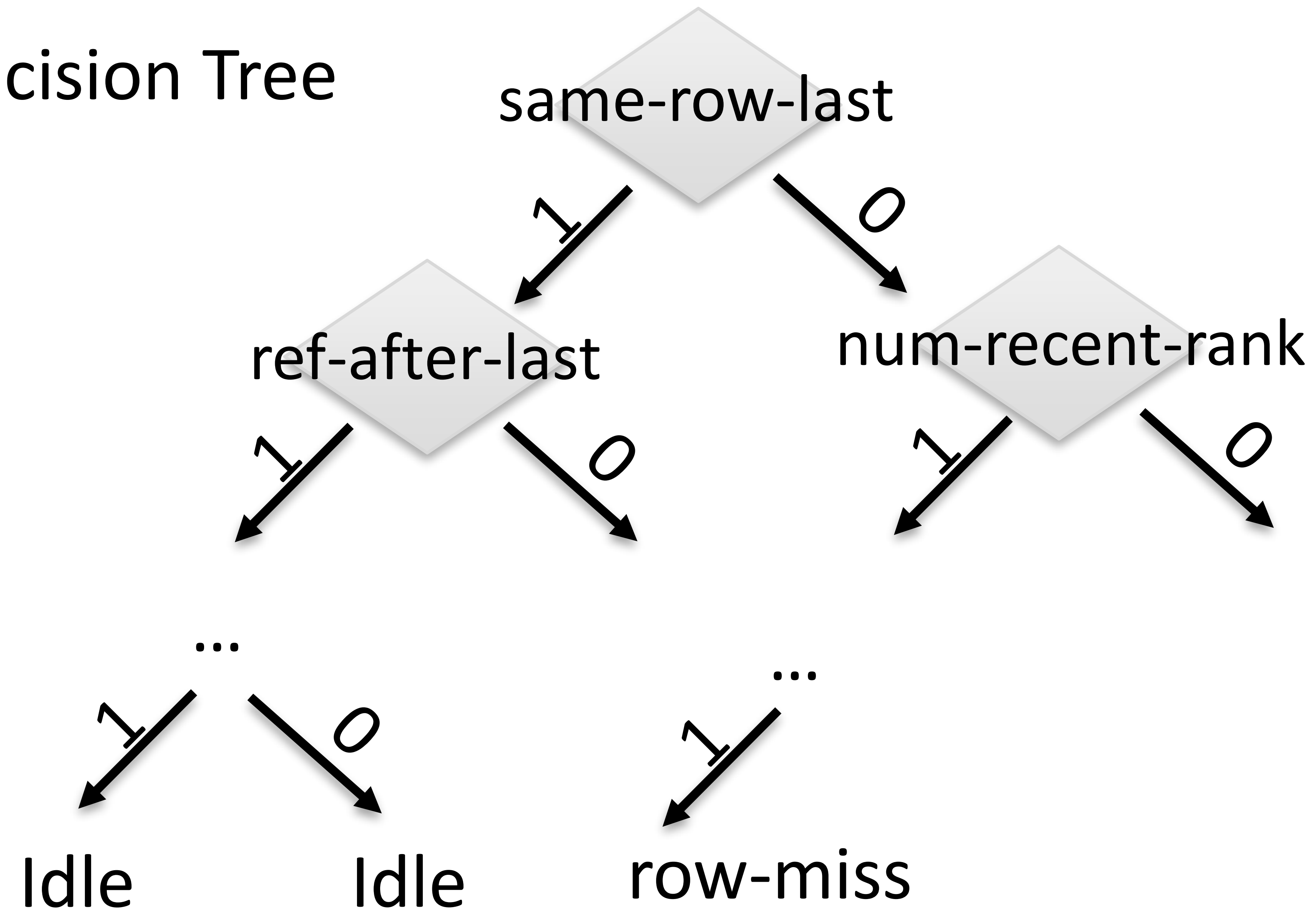
cycle	rank	bank	row	op	Class
0	1	0	0x123	R	idle
10	2	1	0x234	W	idle
...					...

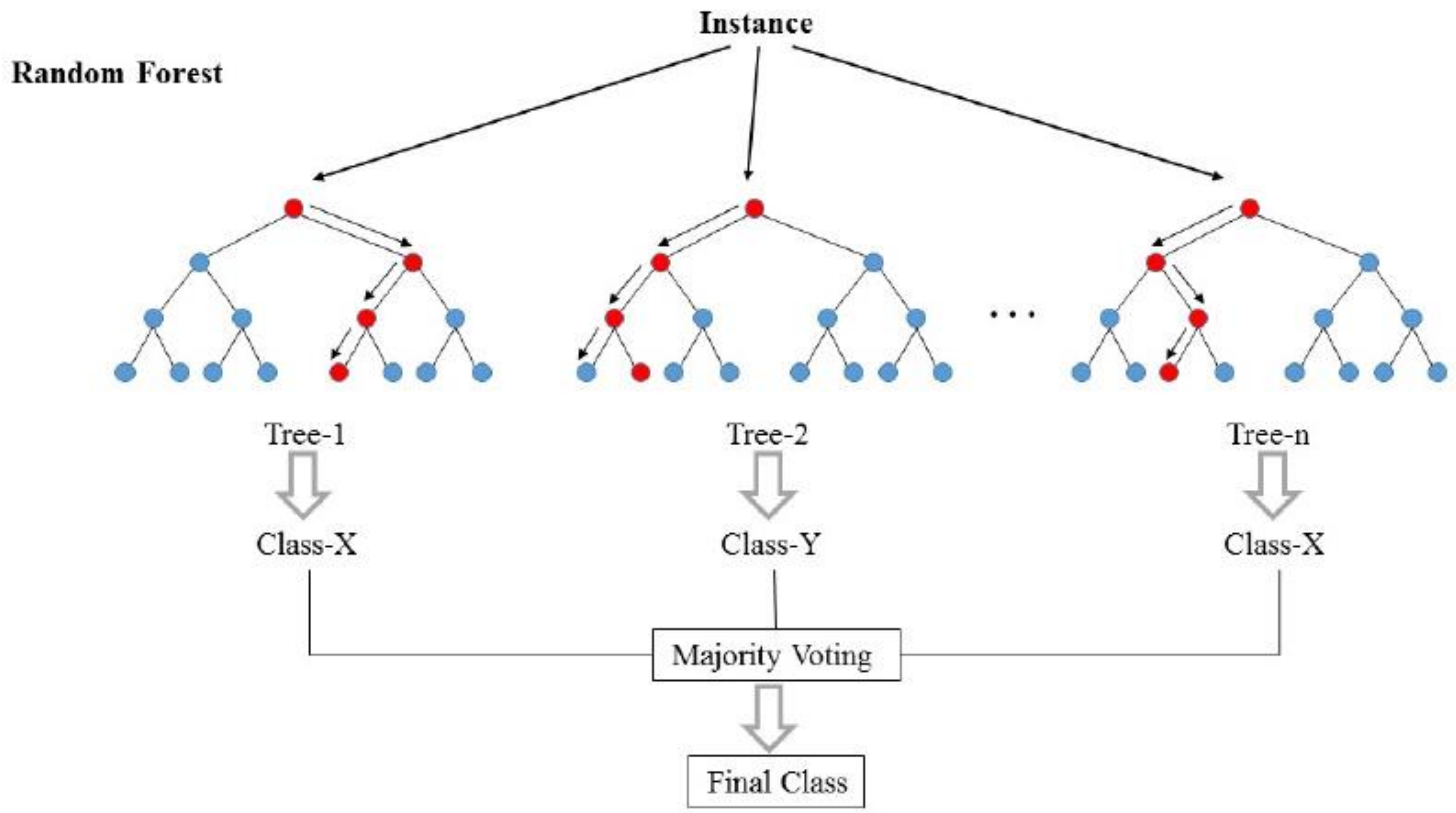
same-row-last	near-ref	...	Class
0	0	...	idle
..			idle
...			...

Models (performed the same)

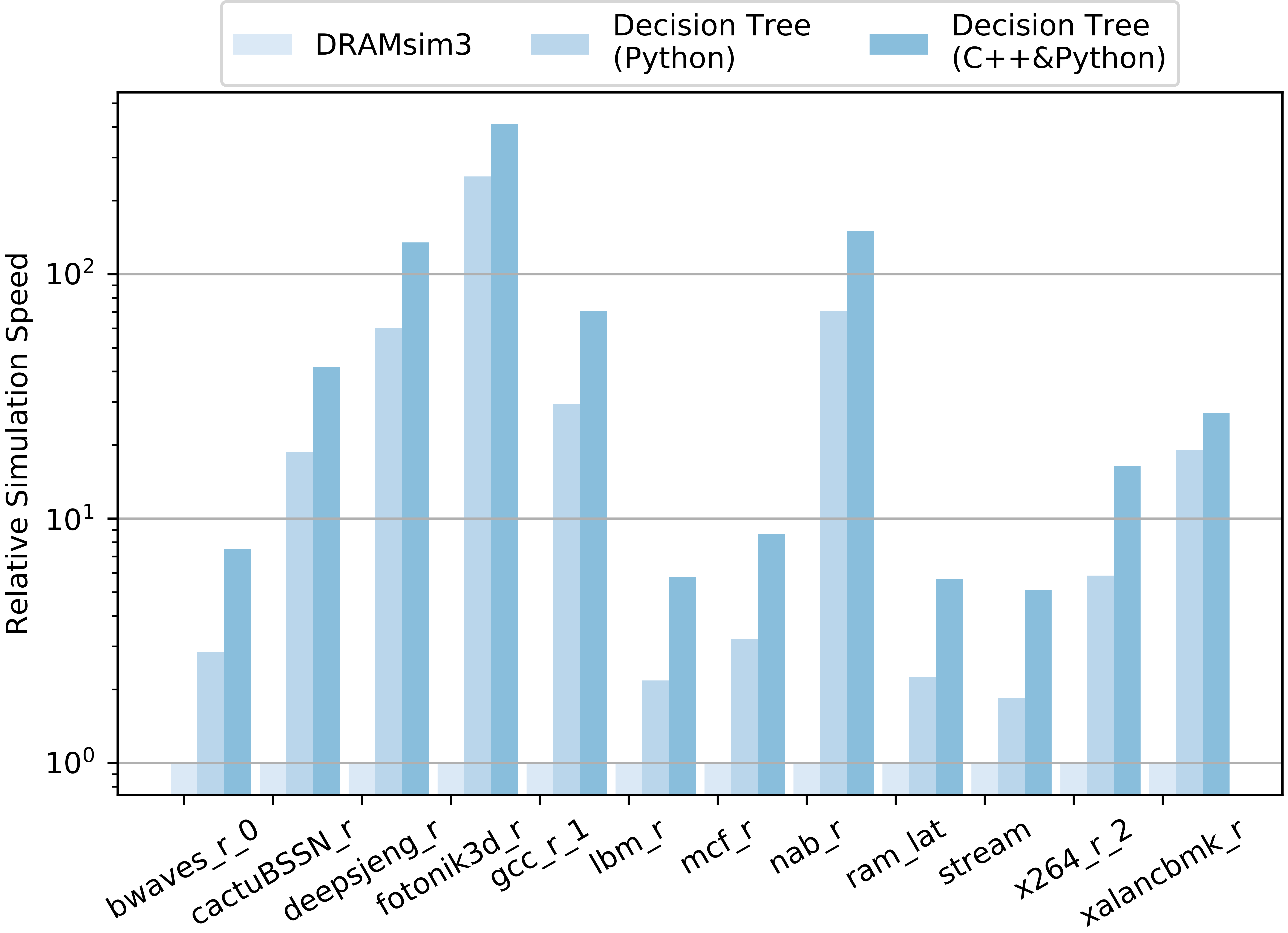
Models: Decision Tree & Random Forest

Decision Tree

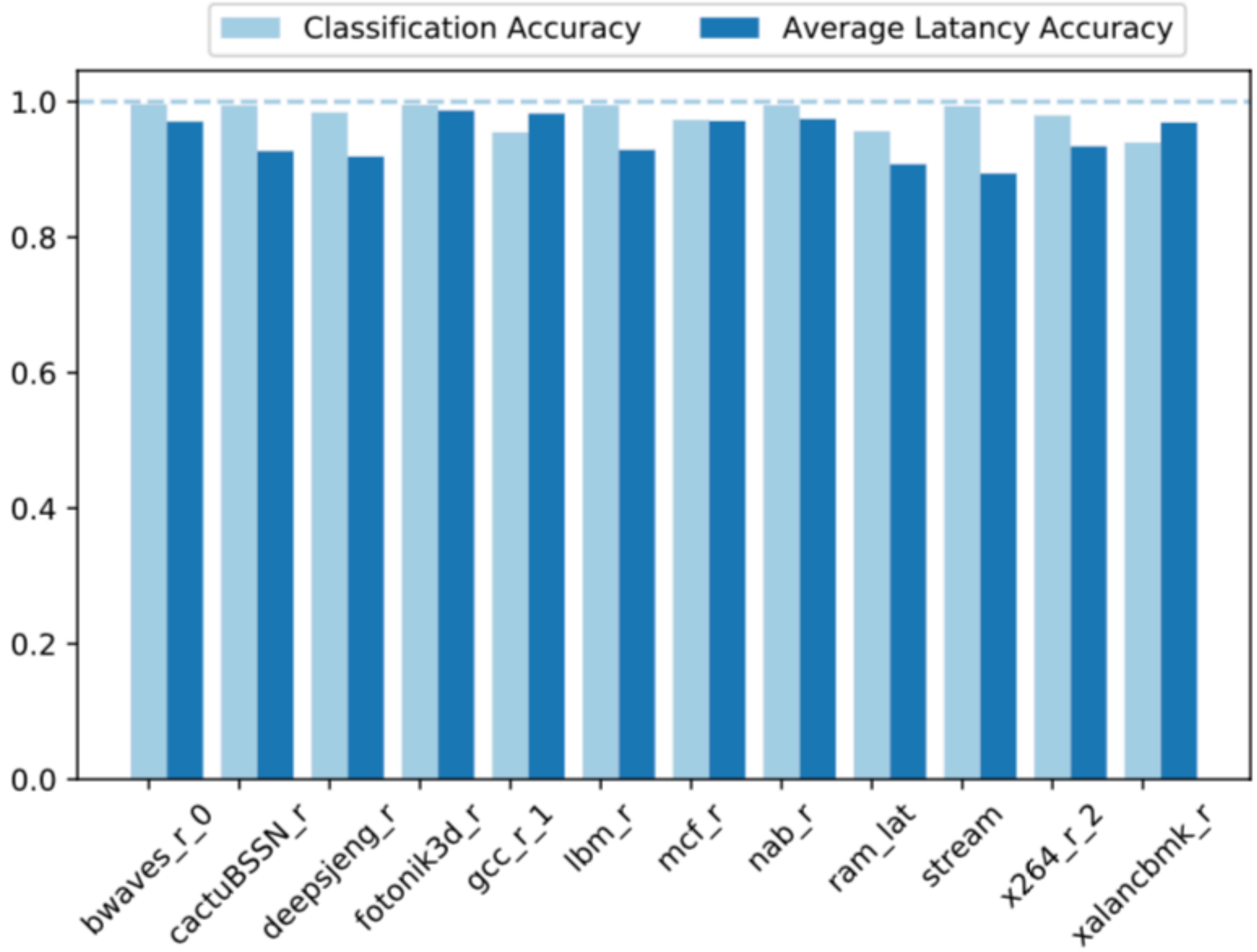




Results: Way Faster

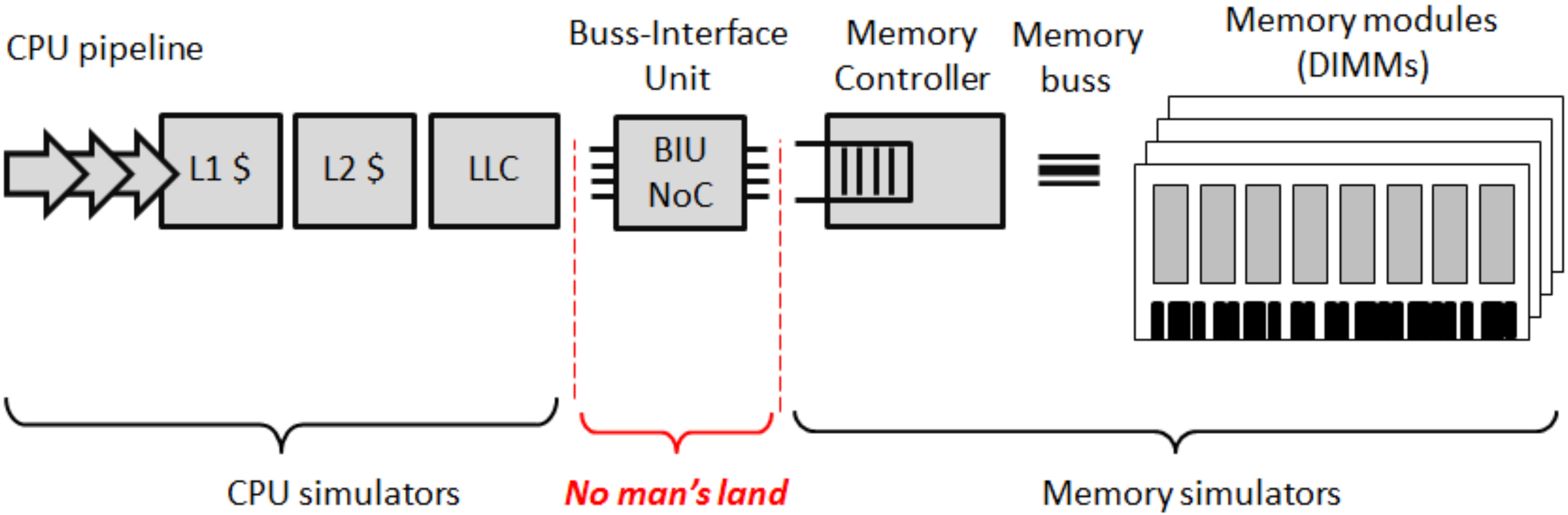


Results: Way Faster



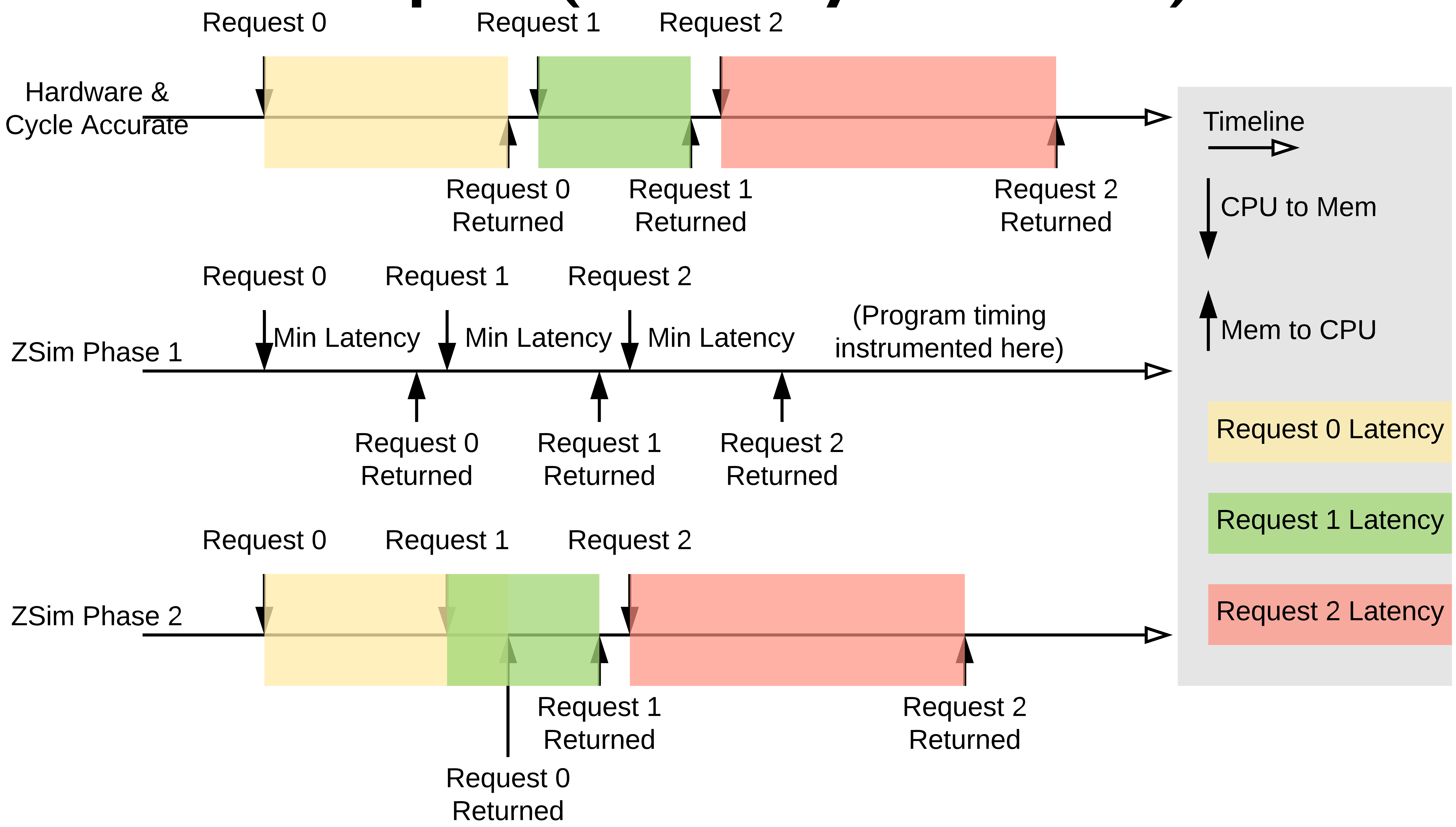
But Wait — and Accurater?

A Little Background:



But Wait — and Accurater?

The Real Culprit (took 2 yrs to find):



ZSim 2-phase memory model timeline diagram compared with real hardware/cycle accurate model.

Three back-to-back memory requests (0, 1, 2) are issued to the memory model.

First phase of memory access aggressively schedules reqs for performance; second phase fails to take into account dependence information.

But Wait — and Accurater?

What Programmers WANT:

(and if you can do it → accurate, parallel sims)

```
if (INSTR.isMemOp) {
    if (L1_cache_miss(INSTR.dAddr)) {
        if (L2_cache_miss(INSTR.dAddr)) {
            INSTR.valid = now +
                       DRAM_request(INSTR.dAddr);
        }
    }
}
```

But Wait — and Accurater?

What Programmers WANT:

(and if you can do it → accurate, parallel sims)

```
if (INSTR.isMemOp) {
    if (L1_cache_miss(INSTR.dAddr)) {
        if (L2_cache_miss(INSTR.dAddr)) {
            INSTR.valid = now +
                     DRAM_request(INSTR.dAddr);
        }
    }
}
```

Prediction gives it to them

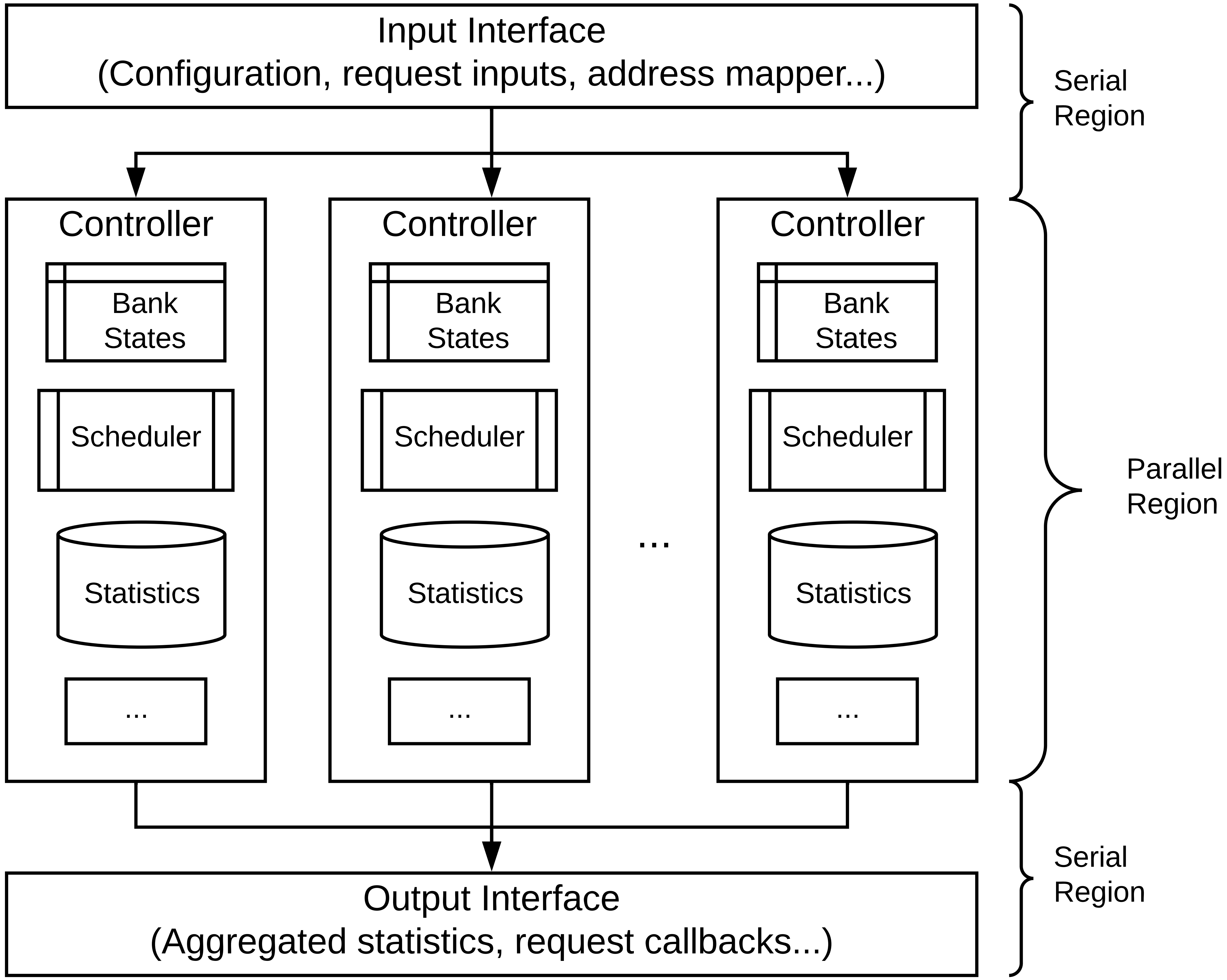
Bottom Line (scalability)

The Future:

Large parallel simulations enabled, wherein each CPU model can have its own memory-system predictor to provide estimates of main memory-system latency.

None of the memory models need interact to provide their predictions.

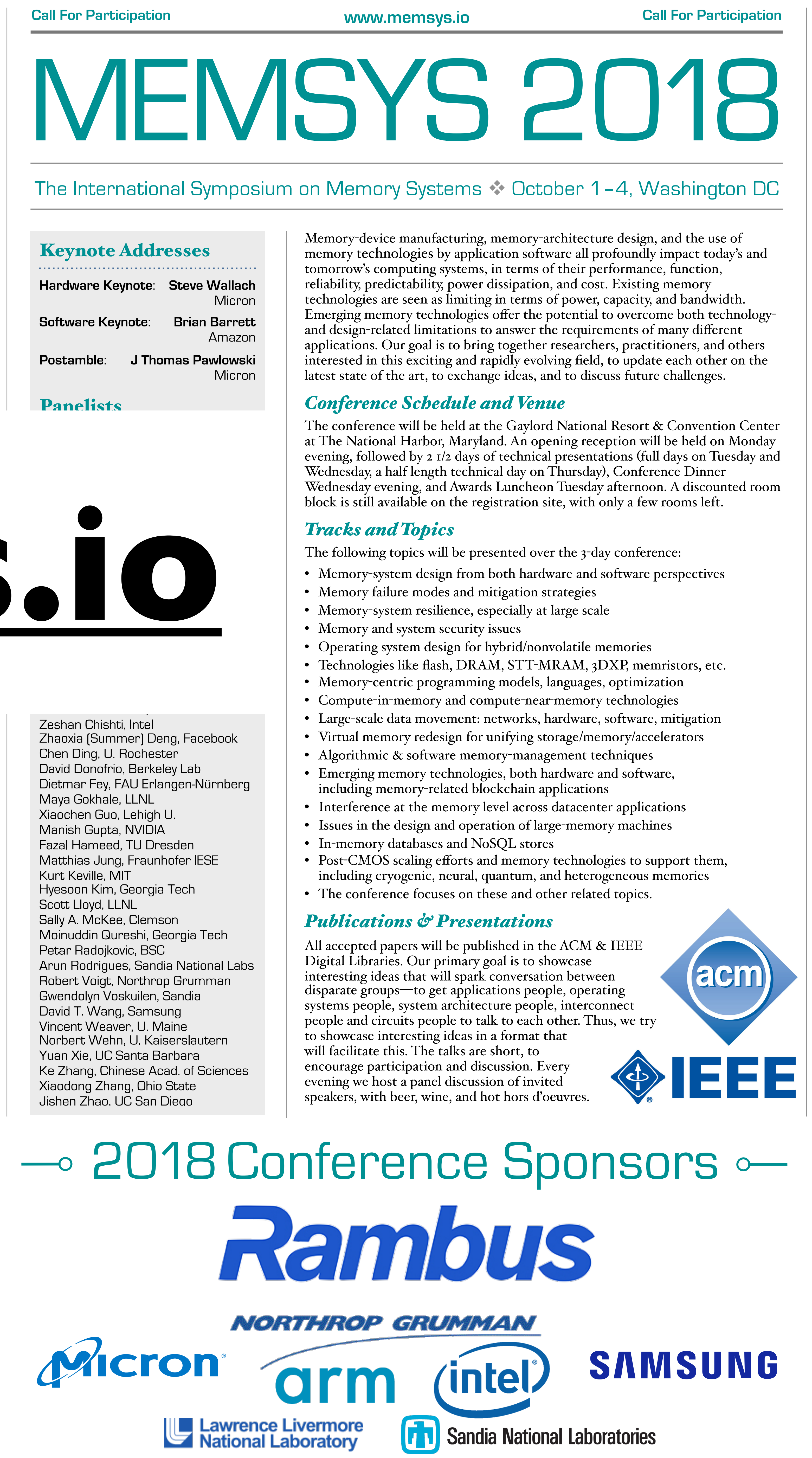
Moreover, the CPU models can be written in a FAR simpler way than they are now, making them faster and less likely to contain “gotcha” assumptions.



Shameless Plug

www.memsys.io

Washington DC
Sep 30 – Oct 3, 2019



MEMSYS 2018

The International Symposium on Memory Systems ❖ October 1 – 4, Washington DC

Keynote Addresses

Hardware Keynote:	Steve Wallach Micron
Software Keynote:	Brian Barrett Amazon
Postamble:	J Thomas Pawlowski Micron

Panelists

Zeshan Chishti, Intel
Zhaoxia (Summer) Deng, Facebook
Chen Ding, U. Rochester
David Donofrio, Berkeley Lab
Dietmar Fey, FAU Erlangen-Nürnberg
Maya Gokhale, LLNL
Xiaochen Guo, Lehigh U.
Manish Gupta, NVIDIA
Fazal Hameed, TU Dresden
Matthias Jung, Fraunhofer IESE
Kurt Keville, MIT
Hyesoon Kim, Georgia Tech
Scott Lloyd, LLNL
Sally A. McKee, Clemson
Moinuddin Qureshi, Georgia Tech
Petar Radojkovic, BSC
Arun Rodrigues, Sandia National Labs
Robert Voigt, Northrop Grumman
Gwendolyn Voskuilen, Sandia
David T. Wang, Samsung
Vincent Weaver, U. Maine
Norbert Wehn, U. Kaiserslautern
Yuan Xie, UC Santa Barbara
Ke Zhang, Chinese Acad. of Sciences
Xiaodong Zhang, Ohio State
Jishen Zhao, UC San Diego

Memory-device manufacturing, memory-architecture design, and the use of memory technologies by application software all profoundly impact today's and tomorrow's computing systems, in terms of their performance, function, reliability, predictability, power dissipation, and cost. Existing memory technologies are seen as limiting in terms of power, capacity, and bandwidth. Emerging memory technologies offer the potential to overcome both technology- and design-related limitations to answer the requirements of many different applications. Our goal is to bring together researchers, practitioners, and others interested in this exciting and rapidly evolving field, to update each other on the latest state of the art, to exchange ideas, and to discuss future challenges.

Conference Schedule and Venue

The conference will be held at the Gaylord National Resort & Convention Center at The National Harbor, Maryland. An opening reception will be held on Monday evening, followed by 2 1/2 days of technical presentations (full days on Tuesday and Wednesday, a half length technical day on Thursday), Conference Dinner Wednesday evening, and Awards Luncheon Tuesday afternoon. A discounted room block is still available on the registration site, with only a few rooms left.

Tracks and Topics

The following topics will be presented over the 3-day conference:

- Memory-system design from both hardware and software perspectives
- Memory failure modes and mitigation strategies
- Memory-system resilience, especially at large scale
- Memory and system security issues
- Operating system design for hybrid/nonvolatile memories
- Technologies like flash, DRAM, STT-MRAM, 3DXP, memristors, etc.
- Memory-centric programming models, languages, optimization
- Compute-in-memory and compute-near-memory technologies
- Large-scale data movement: networks, hardware, software, mitigation
- Virtual memory redesign for unifying storage/memory/accelerators
- Algorithmic & software memory-management techniques
- Emerging memory technologies, both hardware and software, including memory-related blockchain applications
- Interference at the memory level across datacenter applications
- Issues in the design and operation of large-memory machines
- In-memory databases and NoSQL stores
- Post-CMOS scaling efforts and memory technologies to support them, including cryogenic, neural, quantum, and heterogeneous memories
- The conference focuses on these and other related topics.

Publications & Presentations

All accepted papers will be published in the ACM & IEEE Digital Libraries. Our primary goal is to showcase interesting ideas that will spark conversation between disparate groups—to get applications people, operating systems people, system architecture people, interconnect people and circuits people to talk to each other. Thus, we try to showcase interesting ideas in a format that will facilitate this. The talks are short, to encourage participation and discussion. Every evening we host a panel discussion of invited speakers, with beer, wine, and hot hors d'oeuvres.

2018 Conference Sponsors

Rambus
NORTHROP GRUMMAN
Micron
arm
intel
SAMSUNG
Lawrence Livermore National Laboratory
Sandia National Laboratories

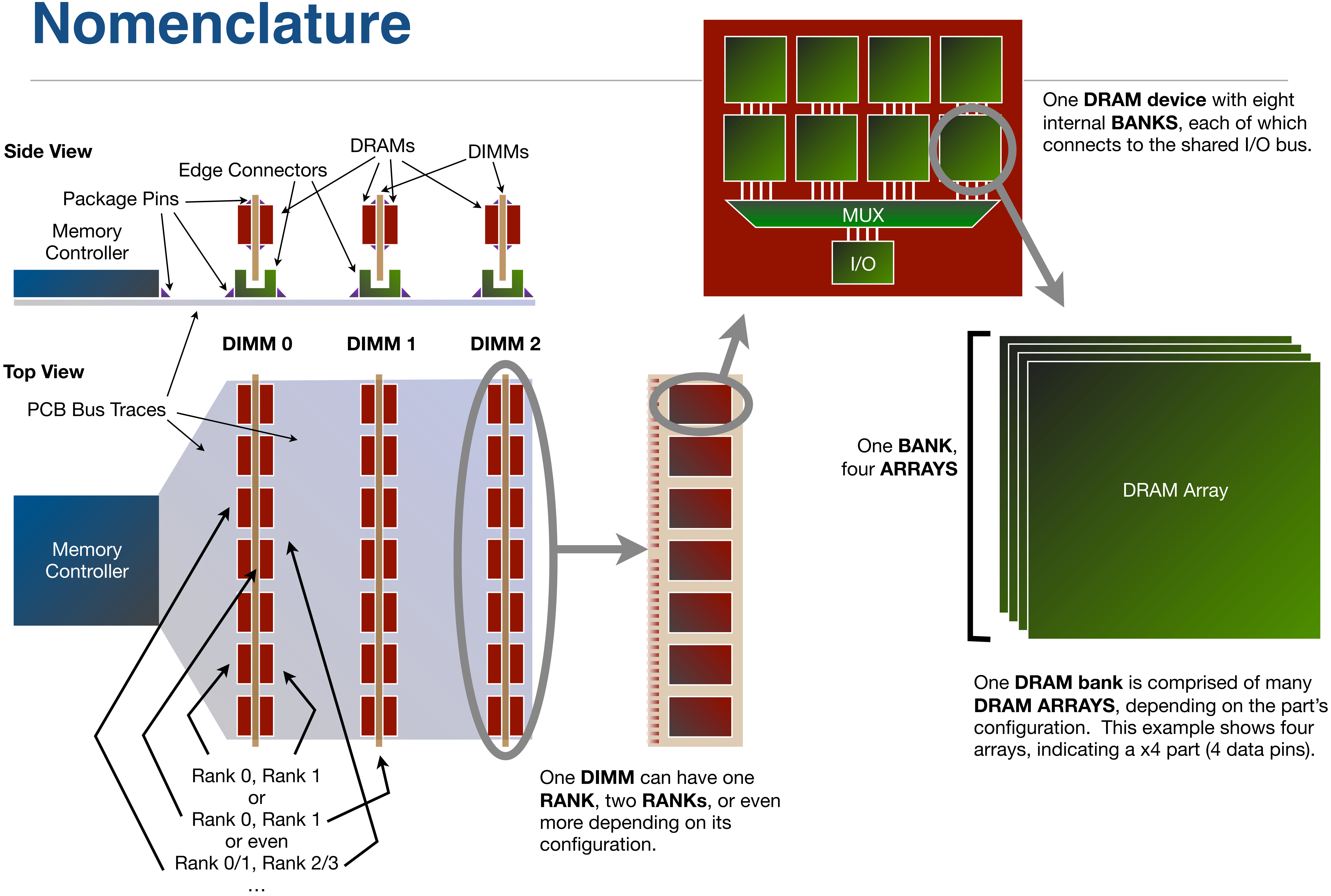
Thank You!

Bruce Jacob

blj@umd.edu
www.ece.umd.edu/~blj

Backup Slides

Nomenclature

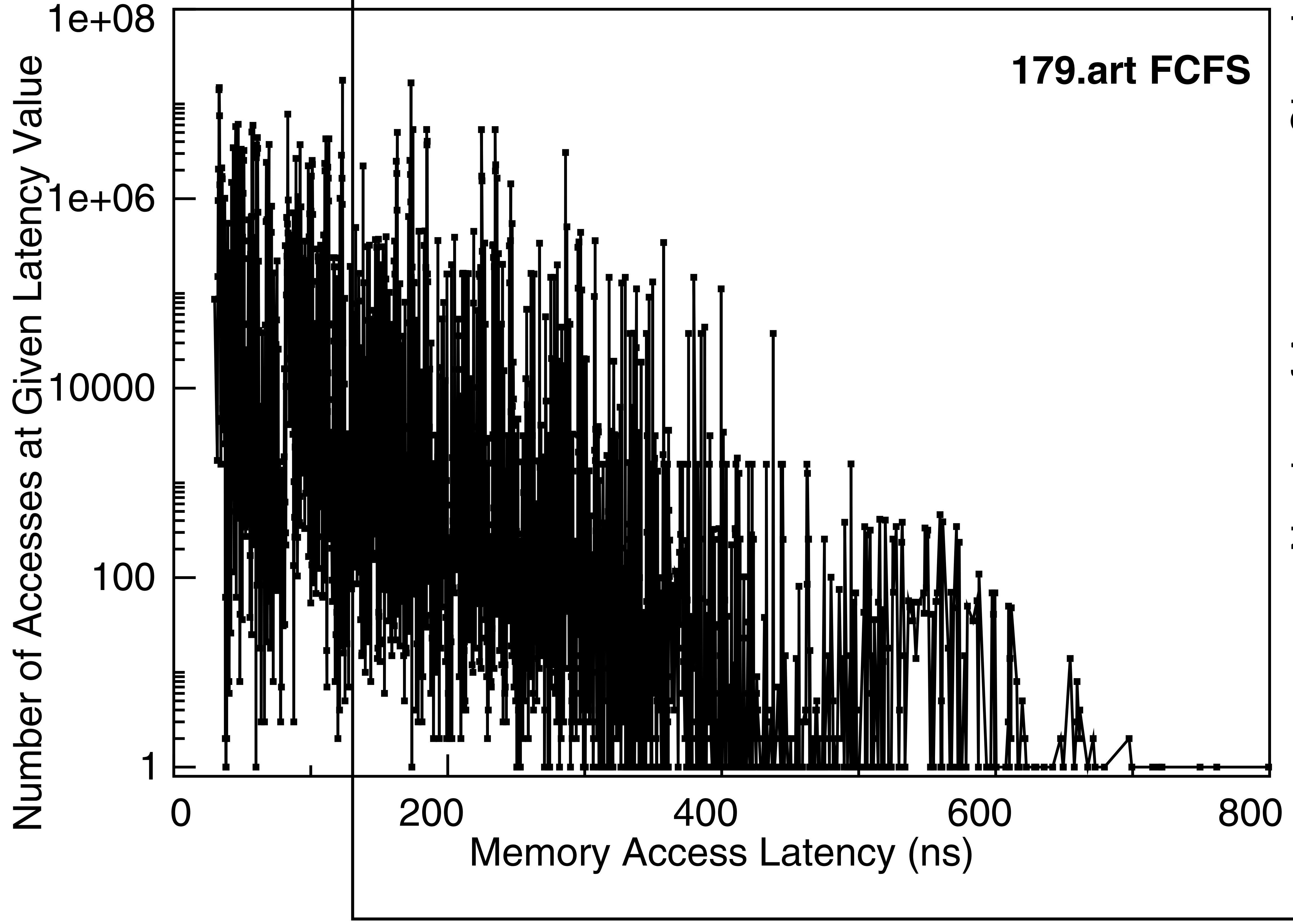


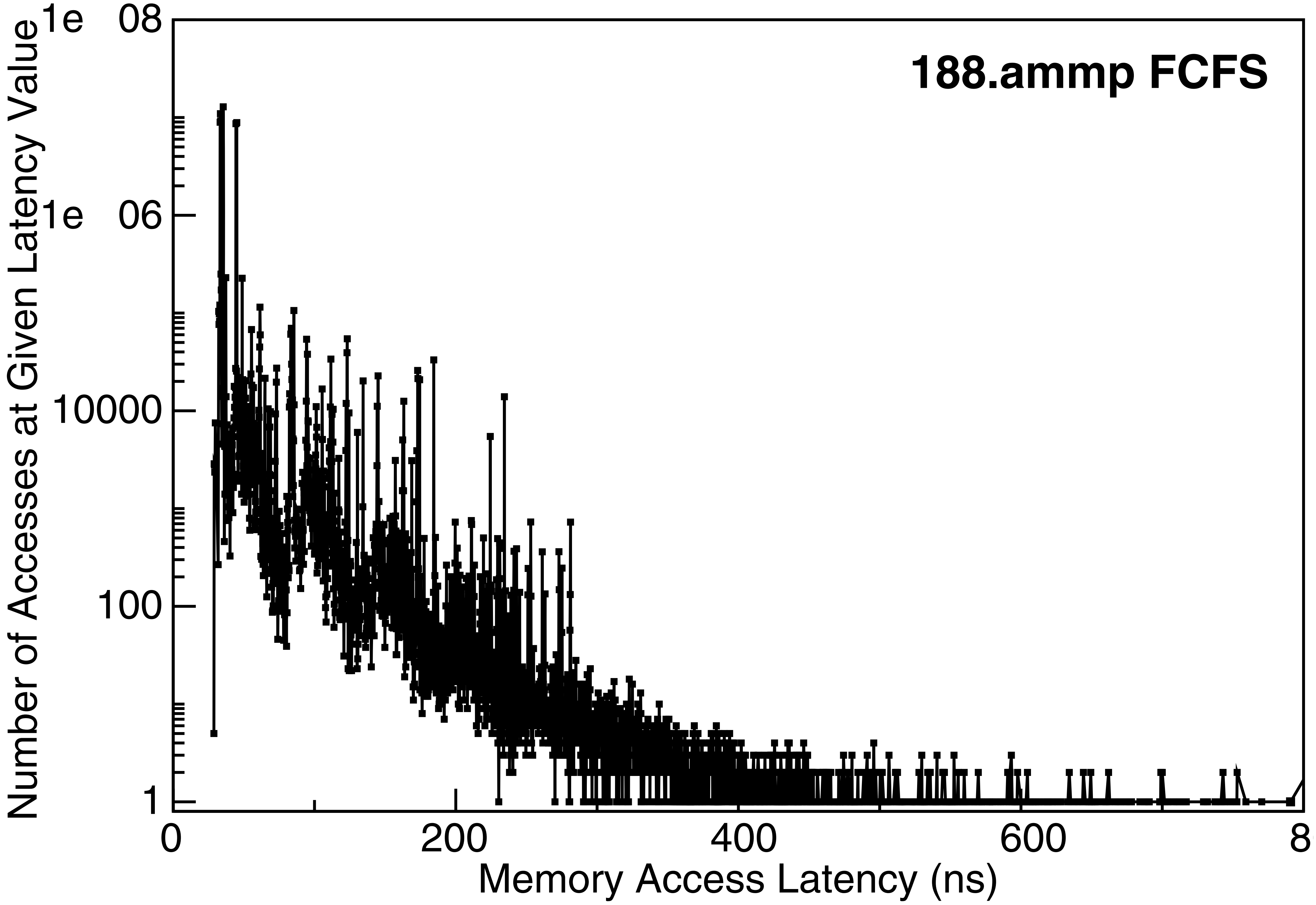
One **DRAM device** with eight internal **BANKS**, each of which connects to the shared I/O bus.

One **DIMM** can have one **RANK**, two **RANKs**, or even more depending on its configuration.

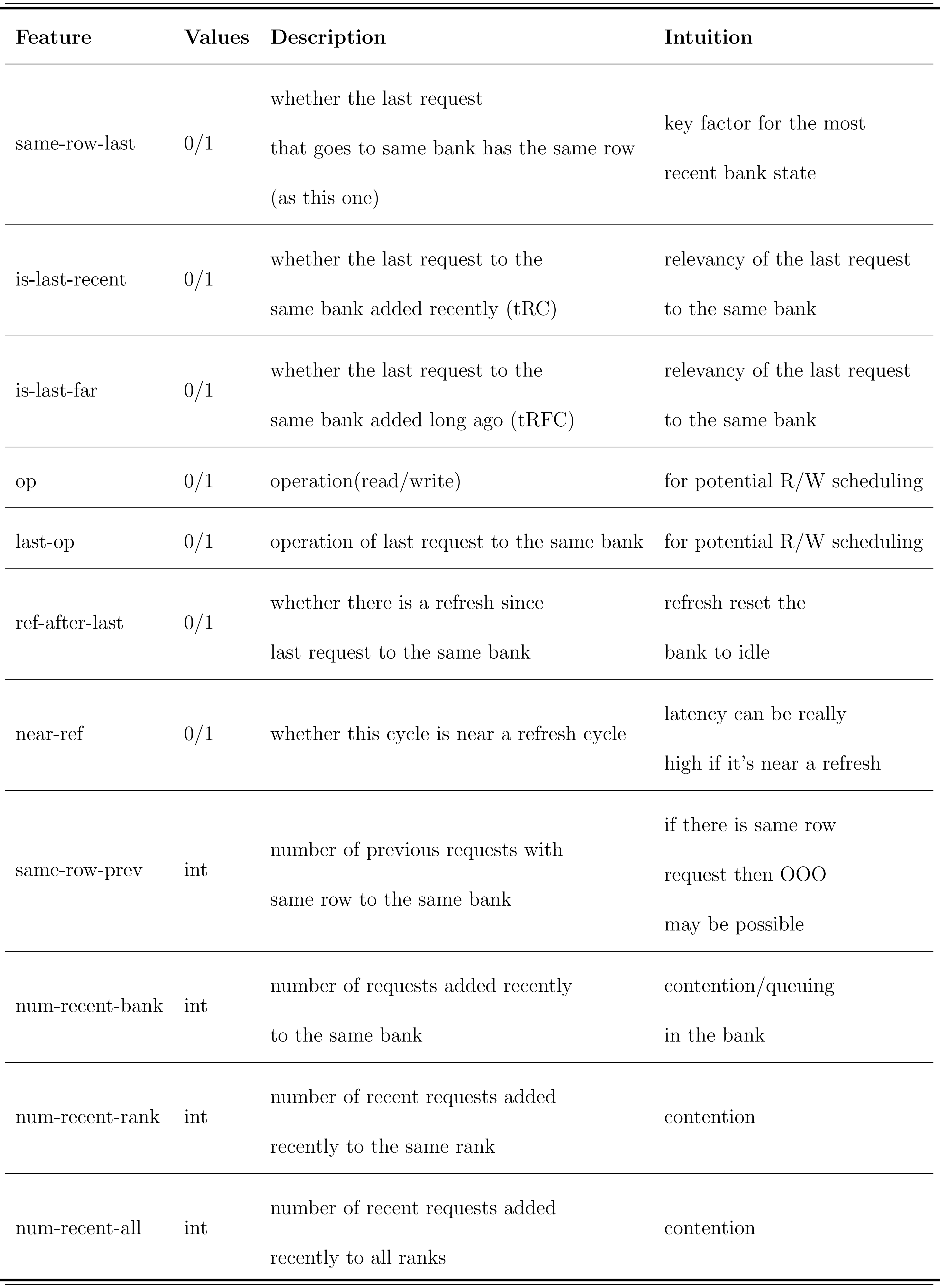
One **DRAM bank** is comprised of many **DRAM ARRAYS**, depending on the part's configuration. This example shows four arrays, indicating a x4 part (4 data pins).

Background


179.art FCFS
Number of Accesses at Given Latency Value
1e+08
1e+06
10000
100
1
0
200
400
600
800
Memory Access Latency (ns)


188.ammp FCFS
Number of Accesses at Given Latency Value
1e 08
1e 06
10000
100
1
0
200
400
600
8
Memory Access Latency (ns)

Features Extracted



Feature	Values	Description	Intuition
same-row-last	0/1	whether the last request that goes to same bank has the same row (as this one)	key factor for the most recent bank state
is-last-recent	0/1	whether the last request to the same bank added recently (tRC)	relevancy of the last request to the same bank
is-last-far	0/1	whether the last request to the same bank added long ago (tRFC)	relevancy of the last request to the same bank
op	0/1	operation(read/write)	for potential R/W scheduling
last-op	0/1	operation of last request to the same bank	for potential R/W scheduling
ref-after-last	0/1	whether there is a refresh since last request to the same bank	refresh reset the bank to idle
near-ref	0/1	whether this cycle is near a refresh cycle	latency can be really high if it's near a refresh
same-row-prev	int	number of previous requests with same row to the same bank	if there is same row request then OOO may be possible
num-recent-bank	int	number of requests added recently to the same bank	contention/queuing in the bank
num-recent-rank	int	number of recent requests added recently to the same rank	contention
num-recent-all	int	number of recent requests added recently to all ranks	contention