Memory Systems
and Memory-Centric Computing Systems
Lecture 4b: Simulating Memory

Prof. Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

17 June 2019

TU Wien Fast Course 2019

SAFARI ETH zürich Carnegie Mellon

Simulating Memory

Evaluating New Ideas for New (Memory) Architectures

Potential Evaluation Methods

- How do we assess an idea will improve a target metric X?
- A variety of evaluation methods are available:
  - Theoretical proof
  - Analytical modeling/estimation
  - Simulation (at varying degrees of abstraction and accuracy)
  - Prototyping with a real system (e.g., FPGAs)
  - Real implementation

The Difficulty in Architectural Evaluation

- The answer is usually workload dependent
  - E.g., think caching
  - E.g., think pipelining
  - E.g., think any idea we talked about (RAIDR, Mem. Sched., …)
- Workloads change
- System has many design choices and parameters
  - Architect needs to decide many ideas and many parameters for a design
  - Not easy to evaluate all possible combinations!
- System parameters may change

Simulation: The Field of Dreams

Dreaming and Reality

- An architect is in part a dreamer, a creator
- Simulation is a key tool of the architect
- Simulation enables
  - The exploration of many dreams
  - A reality check of the dreams
  - Deciding which dream is better
- Simulation also enables
  - The ability to fool yourself with false dreams

Why High-Level Simulation?

- Problem: RTL simulation is intractable for design space exploration → too time consuming to design and evaluate
  - Especially over a large number of workloads
  - Especially if you want to predict the performance of a good chunk of a workload on a particular design
  - Especially if you want to consider many design choices
    - Cache size, associativity, block size, algorithms
    - Memory control and scheduling algorithms
    - In-order vs. out-of-order execution
    - Reservation station sizes, ld/st queue size, register file size, …
    - …
- Goal: Explore design choices quickly to see their impact on the workloads we are designing the platform for

Different Goals in Simulation

- Explore the design space quickly and see what you want to
  - potentially implement in a next-generation platform
  - propose as the next big idea to advance the state of the art
  - the goal is mainly to see relative effects of design decisions
- Match the behavior of an existing system so that you can
  - debug and verify it at cycle-level accuracy
  - propose small tweaks to the design that can make a difference in performance or energy
  - the goal is very high accuracy
- Other goals in-between:
  - Refine the explored design space without going into a full detailed, cycle-accurate design
  - Gain confidence in your design decisions made by higher-level design space exploration

Tradeoffs in Simulation

- Three metrics to evaluate a simulator
  - Speed
  - Flexibility
  - Accuracy

- Speed: How fast the simulator runs (xIPS, xCPS, slowdown)
- Flexibility: How quickly one can modify the simulator to evaluate different algorithms and design choices?
- Accuracy: How accurate the performance (energy) numbers the simulator generates are vs. a real design (Simulation error)

- The relative importance of these metrics varies depending on where you are in the design process (what your goal is)

Trading Off Speed, Flexibility, Accuracy

- Speed & flexibility affect:
  - How quickly you can make design tradeoffs
- Accuracy affects:
  - How good your design tradeoffs may end up being
  - How fast you can build your simulator (simulator design time)
- Flexibility also affects:
  - How much human effort you need to spend modifying the simulator
- You can trade off between the three to achieve design exploration and decision goals

High-Level Simulation

- Key Idea: Raise the abstraction level of modeling to give up some accuracy to enable speed & flexibility (and quick simulator design)

- Advantage

  + Can still make the right tradeoffs, and can do it quickly

    + All you need is modeling the key high-level factors, you can omit corner case conditions

    + All you need is to get the "relative trends" accurately, not exact performance numbers

- Disadvantage

  -- Opens up the possibility of potentially wrong decisions

    -- How do you ensure you get the "relative trends" accurately?

Simulation as Progressive Refinement

- High-level models (Abstract, C)
- …
- Medium-level models (Less abstract)
- …
- Low-level models (RTL with everything modeled)
- …
- Real design

- As you refine (go down the above list)
  - Abstraction level reduces
  - Accuracy (hopefully) increases (not necessarily, if not careful)
  - Flexibility reduces; Speed likely reduces except for real design
  - You can loop back and fix higher-level models

Making The Best of Architecture

- A good architect is comfortable at all levels of refinement
  - Including the extremes
- A good architect knows when to use what type of simulation
  - And, more generally, what type of evaluation method
- Recall: A variety of evaluation methods are available:
  - Theoretical proof
  - Analytical modeling
  - Simulation (at varying degrees of abstraction and accuracy)
  - Prototyping with a real system (e.g., FPGAs)
  - Real implementation

Ramulator: A Fast and Extensible DRAM Simulator

[IEEE Comp Arch Letters'15]

Ramulator Motivation

- DRAM and Memory Controller landscape is changing
- Many new and upcoming standards
- Many new controller designs
- A fast and easy-to-extend simulator is very much needed

Segment	*DRAM Standards & Architectures*
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Ramulator

- Provides out-of-the box support for many DRAM standards:
  - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

Simulator (clang -O3)	*Cycles (10^6)*		*Runtime (sec.)*		*Req/sec (10^3)*		*Memory (MB)*
	Random	*Stream*	*Random*	*Stream*	*Random*	*Stream*	
Ramulator	652	411	752	249	133	402	2.1
DRAMSim2	645	413	2,030	876	49	114	1.2
USIMM	661	409	1,880	750	53	133	4.5
DrSim	647	406	18,109	12,984	6	8	1.6
NVMain	666	413	6,881	5,023	15	20	4,230.0

Table 3. Comparison of five simulators using two traces

Case Study: Comparison of DRAM Standards

Standard	*Rate (MT/s)*	*Timing (CL-RCD-RP)*	*Data-Bus (Width×Chan.)*	*Rank-per-Chan*	*BW (GB/s)*
DDR3	1,600	11-11-11	64-bit × 1	1	11.9
DDR4	2,400	16-16-16	64-bit × 1	1	17.9
SALP†	1,600	11-11-11	64-bit × 1	1	11.9
LPDDR3	1,600	12-15-15	64-bit × 1	1	11.9
LPDDR4	2,400	22-22-22	32-bit × 2*	1	17.9
GDDR5 [12]	6,000	18-18-18	64-bit × 1	1	44.7
HBM	1,000	7-7-7	128-bit × 8*	1	119.2
WIO	266	7-7-7	128-bit × 4*	1	15.9
WIO2	1,066	9-10-10	128-bit × 8*	1	127.2

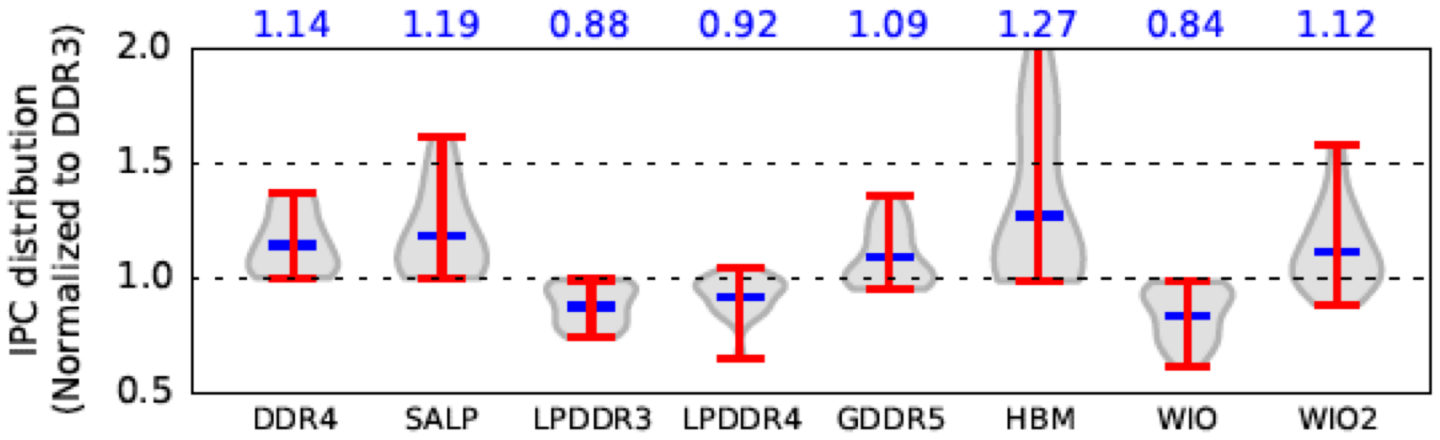


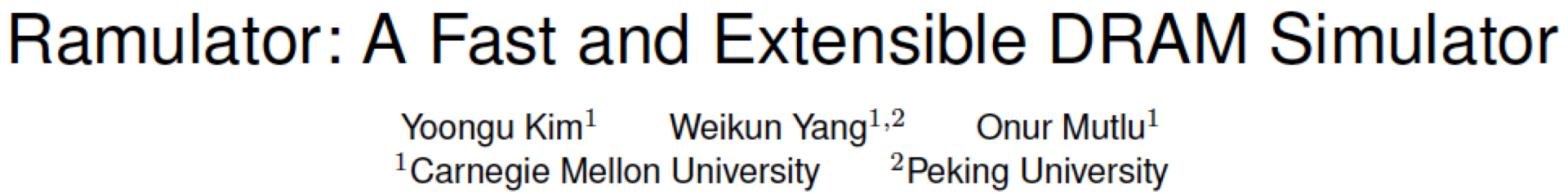
Figure 2. Performance comparison of DRAM standards

Across 22 workloads, simple CPU model

Ramulator Paper and Source Code

- Yoongu Kim, Weikun Yang, and Onur Mutlu,
  "Ramulator: A Fast and Extensible DRAM Simulator"
  IEEE Computer Architecture Letters (***CAL***), March 2015.
  [Source Code]

- Source code is released under the liberal MIT License
  - https://github.com/CMU-SAFARI/ramulator



Optional Assignment

- Review the Ramulator paper
  - Email me your review (omutlu@gmail.com)
- Download and run Ramulator
  - Compare DDR3, DDR4, SALP, HBM for the libquantum benchmark (provided in Ramulator repository)
  - Email me your report (omutlu@gmail.com)
- This **will** help you get into **memory systems research**

An Example Study with Ramulator (I)

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu,
  "Demystifying Workload–DRAM Interactions: An Experimental Study"
  *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Phoenix, AZ, USA, June 2019.
  [Preliminary arXiv Version]

Understanding the Interactions of Workloads and DRAM Types: A Comprehensive Experimental Study

Saugata Ghose[†] Tianshi Li[†] Nastaran Hajinazar[⋆] Damla Senol Cali[†] Onur Mutlu[‡†]

[†]Carnegie Mellon University [⋆]Simon Fraser University [‡]ETH Zürich

An Example Study with Ramulator (II)

- We identify important families of workloads, as well as prevalent types of DRAM chips, and rigorously analyze the combined DRAM–workload behavior.
- We perform a comprehensive experimental study of the interaction between nine different DRAM types and 115 modern applications and multiprogrammed workloads.
- We draw 12 key observations from our characterization, enabled in part by our development of new metrics that take into account contention between memory requests due to hardware design.

Understanding the Interactions of Workloads and DRAM Types: A Comprehensive Experimental Study

Saugata Ghose[†] Tianshi Li[†] Nastaran Hajinazar[⋆] Damla Senol Cali[†] Onur Mutlu[‡†]

[†]Carnegie Mellon University [⋆]Simon Fraser University [‡]ETH Zürich

An Example Study with Ramulator (II)

- Notably, we find that (1) newer DRAM technologies such as DDR4 and HMC often do not outperform older technologies such as DDR3, due to higher access latencies and, in the case of HMC, poor exploitation of locality;
- (2) there is no single memory type that can cater to all of the SoC accelerators (e.g., GDDR5 significantly outperforms other memories for multimedia, while HMC significantly outperforms other memories for networking);
- (3) there is still a strong need to lower DRAM latency, but unfortunately the current design trend of commodity DRAM is toward higher latencies to obtain other benefits

Understanding the Interactions of Workloads and DRAM Types: A Comprehensive Experimental Study

Saugata Ghose[†] Tianshi Li[†] Nastaran Hajinazar[⋆] Damla Senol Cali[†] Onur Mutlu[‡†]

Ramulator for PIM

- Gagandeep Singh, Stefano Corda, Geraldo Francisco de Oliveira, Juan Gomez-Luna, Giovanni Mariani, Sander Stujik, Onur Mutlu, and Henk Corporaal,
  "NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning"
  *Proceedings of the 56th Design Automation Conference (**DAC**)*, Las Vegas, NV, USA, June 2019.
  [Source Code for Ramulator-PIM]

NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning

Gagandeep Singh[a,c] Juan Gómez-Luna[b] Giovanni Mariani[c] Geraldo F. Oliveira[b]
Stefano Corda[a,c] Sander Stuijk[a] Onur Mutlu[b] Henk Corporaal[a]
[a]Eindhoven University of Technology [b]ETH Zürich [c]IBM Research - Zurich

Some More Suggested Readings

Some Key Readings on DRAM (I)

- DRAM Organization and Operation
  - Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.
    https://people.inf.ethz.ch/omutlu/pub/tldram_hpca13.pdf
  - Kim et al., "A Case for Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.
    https://people.inf.ethz.ch/omutlu/pub/salp-dram_isca12.pdf
  - Lee et al., "Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost," ACM TACO 2016.
    https://people.inf.ethz.ch/omutlu/pub/smla_high-bandwidth-3d-stacked-memory_taco16.pdf

Some Key Readings on DRAM (II)

- DRAM Refresh
  - Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
    https://people.inf.ethz.ch/omutlu/pub/raidr-dram-refresh_isca12.pdf
  - Chang et al., "Improving DRAM Performance by Parallelizing Refreshes with Accesses," HPCA 2014.
    https://people.inf.ethz.ch/omutlu/pub/dram-access-refresh-parallelization_hpca14.pdf
  - Patel et al., "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," ISCA 2017.
    https://people.inf.ethz.ch/omutlu/pub/reaper-dram-retention-profiling-lpddr4_isca17.pdf

Reading on Simulating Main Memory

- How to evaluate future main memory systems?
- An open-source simulator and its brief description

- Yoongu Kim, Weikun Yang, and Onur Mutlu,
  "Ramulator: A Fast and Extensible DRAM Simulator"
  IEEE Computer Architecture Letters (***CAL***), March 2015.
  [Source Code]

Some Key Readings on Memory Control 1

- Mutlu+, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," ISCA 2008.
  https://people.inf.ethz.ch/omutlu/pub/parbs_isca08.pdf

- Kim et al., "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," MICRO 2010.
  https://people.inf.ethz.ch/omutlu/pub/tcm_micro10.pdf

- Subramanian et al., "BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling," TPDS 2016.
  https://people.inf.ethz.ch/omutlu/pub/bliss-memory-scheduler_ieee-tpds16.pdf

- Usui et al., "DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators," TACO 2016.
  https://people.inf.ethz.ch/omutlu/pub/dash_deadline-aware-heterogeneous-memory-scheduler_taco16.pdf

Some Key Readings on Memory Control 2

- Ipek+, "Self Optimizing Memory Controllers: A Reinforcement Learning Approach," ISCA 2008.
  https://people.inf.ethz.ch/omutlu/pub/rlmc_isca08.pdf

- Ebrahimi et al., "Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems," ASPLOS 2010.
  https://people.inf.ethz.ch/omutlu/pub/fst_asplos10.pdf

- Subramanian et al., "The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory," MICRO 2015.
  https://people.inf.ethz.ch/omutlu/pub/application-slowdown-model_micro15.pdf

- Lee et al., "Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM," PACT 2015.
  https://people.inf.ethz.ch/omutlu/pub/decoupled-dma_pact15.pdf

More Readings

- To come as we cover the future topics

- Search for "DRAM" or "Memory" in:
  - https://people.inf.ethz.ch/omutlu/projects.htm

Memory Systems
and Memory-Centric Computing Systems
Lecture 4b: Simulating Memory

Prof. Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

17 June 2019

TU Wien Fast Course 2019

SAFARI ETH zürich Carnegie Mellon