

Computer Architecture

Lecture 14: Simulation (with a Focus on Memory)

Prof. Onur Mutlu

ETH Zürich

Fall 2020

12 November 2020

Simulating (Memory) Systems

Evaluating New Ideas for New (Memory) Architectures

Potential Evaluation Methods

- How do we assess how an idea will affect 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
 - Allows the evaluation & understanding of non-existent systems
- 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

An Example Simulator

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

<i>Segment</i>	<i>DRAM Standards & Architectures</i>
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

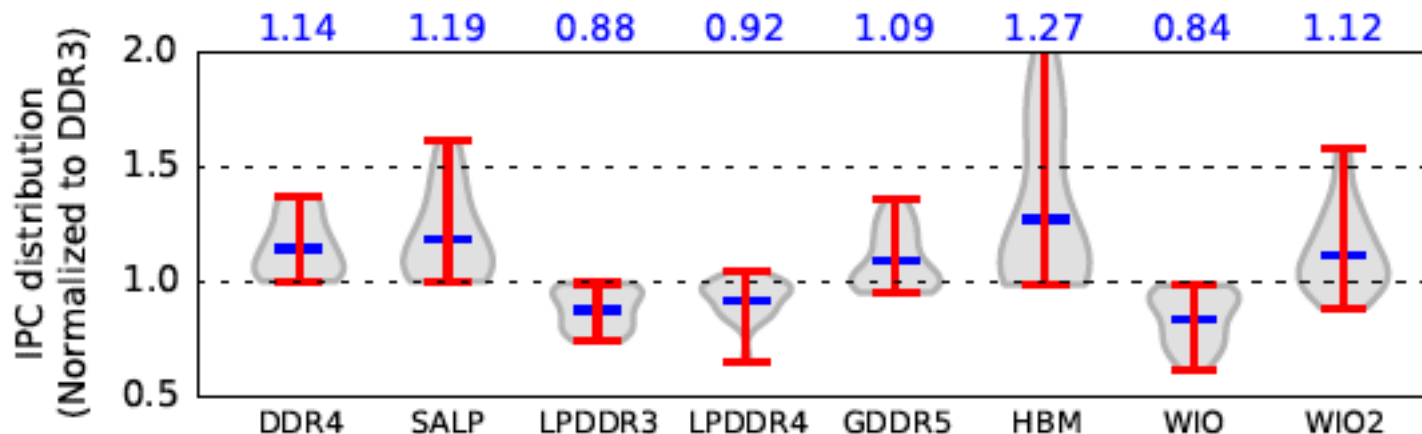
- Provides out-of-the box support for many DRAM standards:
 - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

<i>Simulator</i> (clang -O3)	<i>Cycles (10⁶)</i>		<i>Runtime (sec.)</i>		<i>Req/sec (10³)</i>		<i>Memory</i> (MB)
	<i>Random</i>	<i>Stream</i>	<i>Random</i>	<i>Stream</i>	<i>Random</i>	<i>Stream</i>	
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

<i>Standard</i>	<i>Rate (MT/s)</i>	<i>Timing (CL-RCD-RP)</i>	<i>Data-Bus (Width×Chan.)</i>	<i>Rank-per-Chan</i>	<i>BW (GB/s)</i>
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



Across 22 workloads, simple CPU model

Figure 2. Performance comparison of DRAM standards

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>

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim¹ Weikun Yang^{1,2} Onur Mutlu¹
¹Carnegie Mellon University ²Peking University

Bonus Assignment as Part of HW #4

- Review the Ramulator paper
 - Same points as any other BONUS review in HW #4

An Example Study using Ramulator

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](#)]
[[Abstract](#)]
[[Slides \(pptx\) \(pdf\)](#)]
[[MemBen Benchmark Suite](#)]
[[Source Code for GPGPUSim-Ramulator](#)]

Demystifying Complex Workload–DRAM Interactions: An Experimental Study

Saugata Ghose[†]

Tianshi Li[†]

Nastaran Hajinazar^{‡†}

Damla Senol Cali[†]

Onur Mutlu^{§†}

[†]Carnegie Mellon University

[‡]Simon Fraser University

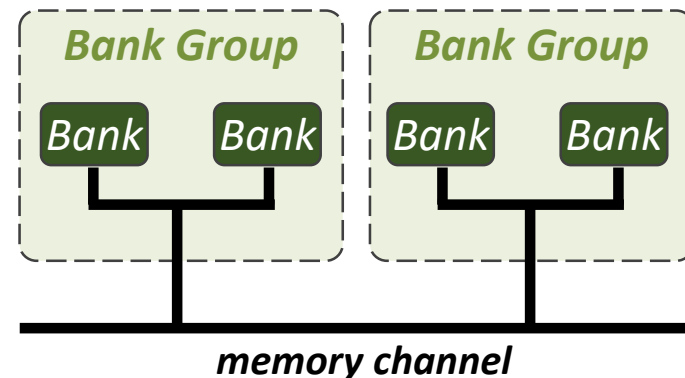
[§]ETH Zürich

- Manufacturers are developing many new types of DRAM
 - **DRAM limits performance, energy improvements:**
new types may overcome some limitations
 - Memory systems now serve a **very diverse set of applications:**
can no longer take a one-size-fits-all approach
- **So which DRAM type works best with which application?**
 - Difficult to understand intuitively due to the complexity of the interaction
 - Can't be tested methodically on real systems: new type needs a new CPU
- We perform a **wide-ranging experimental study to uncover the combined behavior** of workloads and DRAM types
 - **115 prevalent/emerging applications and multiprogrammed workloads**
 - **9 modern DRAM types:** DDR3, DDR4, GDDR5, HBM, HMC, LPDDR3, LPDDR4, Wide I/O, Wide I/O 2

Modern DRAM Types: Comparison to DDR3

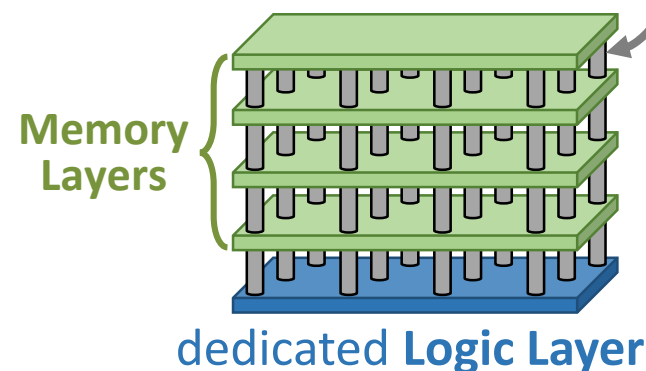
DRAM Type	Banks per Rank	Bank Groups	3D-Stacked	Low-Power
DDR3	8			
DDR4	16	✓	increased latency	
GDDR5	16	✓	increased area/power	
HBM High-Bandwidth Memory	16		✓	
HMC Hybrid Memory Cube	256	narrower rows, higher latency	✓	
Wide I/O	4		✓	✓
Wide I/O 2	8		✓	✓
LPDDR3	8			✓
LPDDR4	16			✓

Bank groups



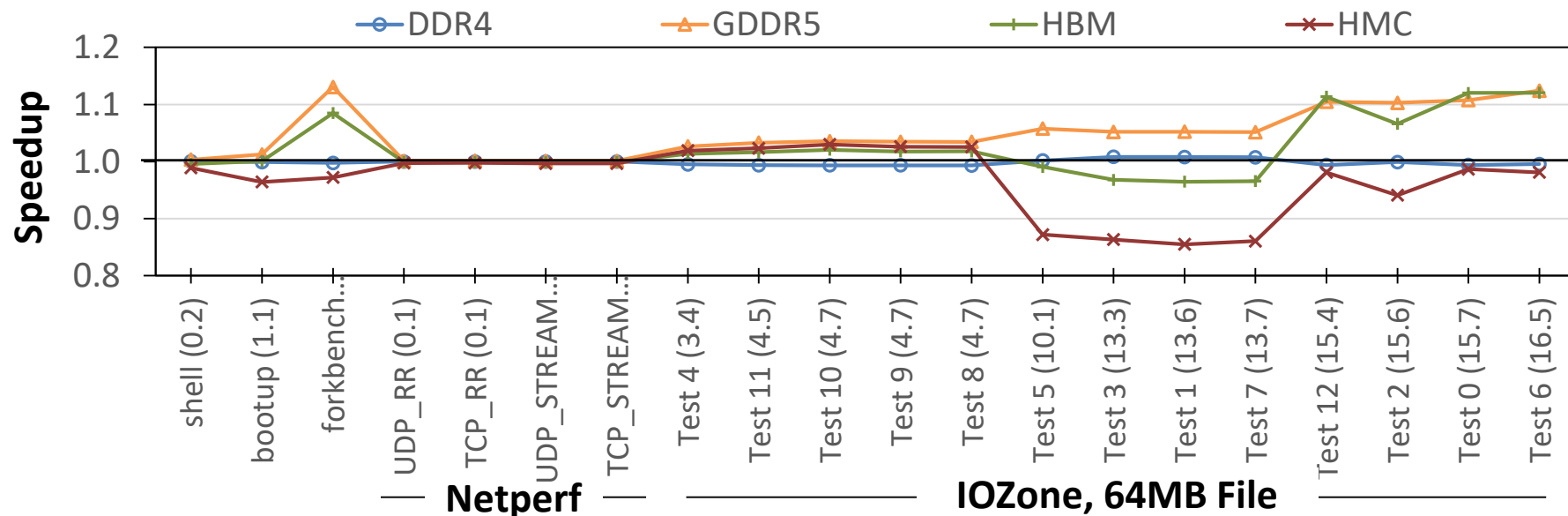
3D-stacked DRAM

high bandwidth with
Through-Silicon
Vias (TSVs)



4. Need for Lower Access Latency: Performance

- New DRAM types often increase access latency in order to provide more banks, higher throughput
- Many applications can't make up for the increased latency
 - Especially true of common OS routines (e.g., file I/O, process forking)



- A variety of desktop/scientific, server/cloud, GPGPU applications

Several applications don't benefit from more parallelism

1. DRAM latency remains a critical bottleneck for many applications
2. Bank parallelism is not fully utilized by a wide variety of our applications
3. Spatial locality continues to provide significant performance benefits if it is exploited by the memory subsystem
4. For some classes of applications, low-power memory can provide energy savings without sacrificing significant performance

- Manufacturers are developing many new types of DRAM
 - **DRAM limits performance, energy improvements:**
new types may overcome some limitations
 - Memory systems now serve a **very diverse set of applications:**
can no longer take a one-size-fits-all approach
 - Difficult to intuitively determine which DRAM–workload pair works best
- We perform a **wide-ranging experimental study to uncover the combined behavior** of workloads, DRAM types
 - 115 prevalent/emerging applications and multiprogrammed workloads
 - 9 modern DRAM types
- 12 key observations on DRAM–workload behavior

Open-source tools: <https://github.com/CMU-SAFARI/ramulator>

Full paper: <https://arxiv.org/pdf/1902.07609>

For More Information...

- 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](#)]
[[Abstract](#)]
[[Slides \(pptx\) \(pdf\)](#)]
[[MemBen Benchmark Suite](#)]
[[Source Code for GPGPUSim-Ramulator](#)]

Demystifying Complex Workload–DRAM Interactions: An Experimental Study

Saugata Ghose[†]

Tianshi Li[†]

Nastaran Hajinazar^{‡†}

Damla Senol Cali[†]

Onur Mutlu^{§†}

[†]Carnegie Mellon University

[‡]Simon Fraser University

[§]ETH Zürich

Ramulator for Processing in Memory

Simulation Infrastructures for PIM

- **Ramulator** extended for PIM
 - ❑ Flexible and extensible DRAM simulator
 - ❑ Can model many different memory standards and proposals
 - ❑ Kim+, “**Ramulator: A Flexible and Extensible DRAM Simulator**”, IEEE CAL 2015.
 - ❑ <https://github.com/CMU-SAFARI/ramulator-pim>
 - ❑ <https://github.com/CMU-SAFARI/ramulator>
 - ❑ [[Source Code for Ramulator-PIM](#)]

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim¹ Weikun Yang^{1,2} Onur Mutlu¹
¹Carnegie Mellon University ²Peking University

Ramulator for PIM

- Gagandeep Singh, Juan Gomez-Luna, Giovanni Mariani, Geraldo F. Oliveira, Stefano Corda, 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.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Poster \(pptx\)](#)] [[pdf](#)]
[[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

What We Discussed Is Applicable to
Other Types of Simulation

Case Study: COVID-19 Spread Modeling and Prediction

COVID-19 Measures: Evaluation Methods

- How do we assess how an idea will affect 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

Simulating COVID-19 Spread

- An architect is in part a dreamer, a creator
- Simulation is a key tool of the architect
 - Allows the evaluation & understanding of non-existent systems
- 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

Goals in Simulating COVID-19 Spread

- 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

Computer Architecture

Lecture 14: Simulation (with a Focus on Memory)

Prof. Onur Mutlu

ETH Zürich

Fall 2020

12 November 2020

OLD: Bonus Assignment as Part of HW #3

- Review the Ramulator paper
 - Same points as any other BONUS review in HW #3
- Download and run Ramulator
 - Compare DDR3, DDR4, SALP, HBM for the libquantum benchmark (provided in Ramulator repository)
 - Turn in your short report with HW #3
 - 750 BONUS points
- This **will** help you get into **memory systems research**