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

<table>
<thead>
<tr>
<th>Segment</th>
<th>DRAM Standards &amp; Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commodity</td>
<td>DDR3 (2007) [14]; DDR4 (2012) [18]</td>
</tr>
<tr>
<td>Performance</td>
<td>eDRAM [28], [32]; RLDRAM3 (2011) [29]</td>
</tr>
<tr>
<td>3D-Stacked</td>
<td>WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10];</td>
</tr>
<tr>
<td></td>
<td>(2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33];</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Simulator (clang -O3)</th>
<th>Cycles ($10^6$)</th>
<th>Runtime (sec.)</th>
<th>Req/sec ($10^3$)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random</td>
<td>Stream</td>
<td>Random</td>
<td>Stream</td>
</tr>
<tr>
<td>Ramulator</td>
<td>652</td>
<td>411</td>
<td>752</td>
<td>249</td>
</tr>
<tr>
<td>DRAMSim2</td>
<td>645</td>
<td>413</td>
<td>2,030</td>
<td>876</td>
</tr>
<tr>
<td>USIMM</td>
<td>661</td>
<td>409</td>
<td>1,880</td>
<td>750</td>
</tr>
<tr>
<td>DrSim</td>
<td>647</td>
<td>406</td>
<td>18,109</td>
<td>12,984</td>
</tr>
<tr>
<td>NVMaint</td>
<td>666</td>
<td>413</td>
<td>6,881</td>
<td>5,023</td>
</tr>
</tbody>
</table>

Table 3. Comparison of five simulators using two traces
Case Study: Comparison of DRAM Standards

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

Across 22 workloads, simple CPU model

Figure 2. Performance comparison of DRAM standards
Ramulator Paper and Source Code


[Source Code]

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

---

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim\(^1\)  Weikun Yang\(^1,2\)  Onur Mutlu\(^1\)
\(^1\)Carnegie Mellon University  \(^2\)Peking University
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)


- [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
Why Study Workload–DRAM Interactions?

- 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

<table>
<thead>
<tr>
<th>DRAM Type</th>
<th>Banks per Rank</th>
<th>Bank Groups</th>
<th>3D-Stacked</th>
<th>Low-Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDR4</td>
<td>16</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDDR5</td>
<td>16</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HBM High-Bandwidth Memory</td>
<td>16</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMC Hybrid Memory Cube</td>
<td>256</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wide I/O</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wide I/O 2</td>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LPDDR3</td>
<td>8</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>LPDDR4</td>
<td>16</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Bank groups**
- **Bank Group**
  - Memory channel
  - increased latency
  - increased area/power

**3D-stacked DRAM**
- high bandwidth with Through-Silicon Vias (TSVs)
  - narrower rows, higher latency
  - dedicated Logic Layer

---

Page 25 of 25
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
Key Takeaways

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
Conclusion

- 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

Abstract

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.
  - [Source Code for Ramulator-PIM](https://github.com/CMU-SAFARI/ramulator-pim)
  - [Source Code for Ramulator](https://github.com/CMU-SAFARI/ramulator)
Ramulator for PIM

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