Future Computing Platforms
Challenges and Opportunities

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
10 November 2021
EdukCircle Intl. Conv. on Engineering & Computer Technology
Current Research Mission

Computer architecture, HW/SW, systems, bioinformatics, security

Build fundamentally better architectures
Four Key Current Directions

- Fundamentally Secure/Reliable/Safe Architectures
- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures
- Fundamentally Low-Latency and Predictable Architectures
- Architectures for AI/ML, Genomics, Medicine, Health
The Transformation Hierarchy

- Problem
- Algorithm
- Program/Language
- System Software
- SW/HW Interface
- Micro-architecture
- Logic
- Devices
- Electrons

Computer Architecture (expanded view)

Computer Architecture (narrow view)
Axiom

To achieve the highest energy efficiency and performance:

we must take the expanded view
of computer architecture

Co-design across the hierarchy:
Algorithms to devices
Specialize as much as possible
within the design goals
Current Research Mission & Major Topics

Build fundamentally better architectures

- Data-centric arch. for low energy & high perf.
  - Proc. in Mem/DRAM, NVM, unified mem/storage

- Low-latency & predictable architectures
  - Low-latency, low-energy yet low-cost memory
  - QoS-aware and predictable memory systems

- Fundamentally secure/reliable/safe arch.
  - Tolerating all bit flips; patchable HW; secure mem

- Architectures for ML/AI/Genomics/Health/Med
  - Algorithm/arch./logic co-design; full heterogeneity

- Data-driven and data-aware architectures
  - ML/AI-driven architectural controllers and design
  - Expressive memory and expressive systems

Broad research spanning apps, systems, logic with architecture at the center

SAFARI
Onur Mutlu’s SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

https://safari.ethz.ch/safari-newsletter-april-2020/

Think BIG, Aim HIGH!

SAFARI

https://safari.ethz.ch
Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 here). 2020 has
Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZIISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1

- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=njX_14584Jw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=16

- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=r7sn41lh-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41

- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3

- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LqzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25

- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39
A Talk on Impactful Research & Teaching

Applying to Grad School & Doing Impactful Research

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
13 June 2020
Undergraduate Architecture Mentoring Workshop @ ISCA 2021

SAFARI ETH Zürich Carnegie Mellon

Arch. Mentoring Workshop @ISCA’21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu
1,563 views • Premiered Jun 16, 2021

Onur Mutlu Lectures
17.2K subscribers

Panel talk at Undergraduate Architecture Mentoring Workshop at ISCA 2021
(https://sites.google.com/wisc.edu/uar...)

https://www.youtube.com/watch?v=83tlorht7Mc&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=54
An Interview on Computing Futures

https://www.youtube.com/watch?v=8ffSEKZhmvo
Referenced Papers, Talks, Artifacts

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/
Future Computing Platforms

Challenges and Opportunities
Why Do We Do Computing?
To Solve Problems
Answer Reworded

To Gain Insight
Answer Extended

To Enable a Better Life & Future
How Does a Computer Solve Problems?
Orchestrating Electrons

In today’s dominant technologies
How Do Problems Get Solved by Electrons?
The Transformation Hierarchy

Computer Architecture (expanded view)

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Computer Architecture (narrow view)
Computer Architecture

- is the **science** and **art** of designing **computing platforms** (hardware, interface, system SW, and programming model)

- to achieve a set of **design goals**
  - E.g., highest performance on earth on workloads X, Y, Z
  - E.g., longest battery life at a form factor that fits in your pocket with cost < $$$ CHF
  - E.g., best average performance across all known workloads at the best performance/cost ratio
  - ...

- Designing a supercomputer is different from designing a smartphone → But, many fundamental principles are similar
Different Platforms, Different Goals

Source: http://www.sia-online.org (semiconductor industry association)
Different Platforms, Different Goals

Source: https://iq.intel.com/5-awesome-uses-for-drone-technology/
Different Platforms, Different Goals

Source: https://taxistartup.com/wp-content/uploads/2015/03/UK-Self-Driving-Cars.jpg
Different Platforms, Different Goals

Source: http://sm.pcmag.com/pcmag_uk/photo/g/google-self-driving-car-the-guts/google-self-driving-car-the-guts_dwx8.jpg
Different Platforms, Different Goals

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu


Accelerating Genome Analysis: A Primer on an Ongoing Journey
DOI Bookmark: 10.1109/MM.2020.3013728

FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications
DOI Bookmark: 10.1109/MM.2021.3088396

MinION from ONT

SmidgION from ONT
Different Platforms, Different Goals

Different Platforms, Different Goals

Source: https://fossbytes.com/wp-content/uploads/2015/06/Supercomputer-TIANHE2-china.jpg
Different Platforms, Different Goals

**Figure 3.** TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

**Figure 4.** Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Different Platforms, Different Goals

New ML applications (vs. TPU3):
- Computer vision
- Natural Language Processing (NLP)
- Recommender system
- Reinforcement learning that plays Go

250 TFLOPS per chip in 2021 vs 90 TFLOPS in TPU3
1 ExaFLOPS per board

https://spectrum.ieee.org/tech-talk/computing/hardware/heres-how-googles-tpu-v4-ai-chip-stacked-up-in-training-tests
Different Platforms, Different Goals

- ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
- Two redundant chips for better safety.

https://youtu.be/Ucp0TTmvqOE?t=4236
Cerebras’s Wafer Scale Engine (2019)

- The largest ML accelerator chip
- 400,000 cores

Cerebras WSE
1.2 Trillion transistors
46,225 mm²

Largest GPU
21.1 Billion transistors
815 mm²

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning
https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/
Cerebras’s Wafer Scale Engine-2 (2021)

- The largest ML accelerator chip (2021)
- 850,000 cores

Cerebras WSE-2
2.6 Trillion transistors
46,225 mm²

Largest GPU
54.2 Billion transistors
826 mm²

NVIDIA Ampere GA100

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning
https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/
Many (Other) AI/ML Chips

- Alibaba
- Amazon
- Facebook
- Google
- Huawei
- Intel
- Microsoft
- NVIDIA
- Tesla
- Many Others and Many Startups are Building Their Own Chips...

- Many More to Come...
Many (Other) AI/ML Chips (2019)

### AI Chip Landscape

<table>
<thead>
<tr>
<th>Tech Giants/System</th>
<th>IC Vender/Fabless</th>
<th>Startup in China</th>
<th>Startup Worldwide</th>
<th>IP/Design Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>intel</td>
<td>Cambricon</td>
<td>Achronix</td>
<td>arm</td>
</tr>
<tr>
<td>Microsoft</td>
<td>SAMSUNG</td>
<td>Bittmain</td>
<td>Achronix</td>
<td>synopsys</td>
</tr>
<tr>
<td>Facebook</td>
<td>NVIDIA</td>
<td>Intel Fusion</td>
<td>Graphcore</td>
<td>IMAGINATION</td>
</tr>
<tr>
<td>Amazon</td>
<td>Qualcomm</td>
<td>ThinkForce</td>
<td>Graphcore</td>
<td>CEVA</td>
</tr>
<tr>
<td>Facebook</td>
<td>AMD</td>
<td>Xilinx</td>
<td>Wave Computing</td>
<td>CADENCE</td>
</tr>
<tr>
<td>Alibaba Group</td>
<td>XILINX</td>
<td>Habana</td>
<td>Mythic</td>
<td>SiFive</td>
</tr>
<tr>
<td>Newzwan</td>
<td>MEDIATRIX</td>
<td>SambaNova</td>
<td>Syntiant</td>
<td>Arteris</td>
</tr>
<tr>
<td>Huawei</td>
<td>UHISOC</td>
<td>Hailo</td>
<td>Opteksys</td>
<td>CORTecs</td>
</tr>
<tr>
<td>Microsoft</td>
<td>AMD</td>
<td>Enflame</td>
<td>Lightmatter</td>
<td>Bitnet</td>
</tr>
<tr>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>Broadcom</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>GUC</td>
</tr>
<tr>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Tesla</td>
<td>NVIDIA</td>
<td>ThinkForce</td>
<td>Lightmatter</td>
<td>NVIDIA</td>
</tr>
</tbody>
</table>

### More on: https://basicmi.github.io/AI-Chip/

All information contained within this infographic is gathered from the internet and periodically updated, no guarantee is given that the information provided is correct, complete, and up-to-date.
Many (Other) AI/ML Chips (2021)

AI Chip Landscape

Chip Vendor/Fabless

- **Google**
  - TPUs
  - Edge TPU
- **Intel**
  - NNP-T
- **Samsung**
  - Exynos 9825
- **NVIDIA**
  - Volta/Turing
  - GeForce RTX
- **Qualcomm**
  - Snapdragon 855
  - Cloud AI 300
- **AMD**
  - EPYC
  - EPYC
- **Xilinx**
  - Zynq
  - Artix
- **MediaTek**
  - Dimensity
  - Tiger T710
- **UNISOC**
  - RK3399pro
  - CV225/255
- **Rockchip**
  - Ambarella
  - Smart voice
- **Samsung**
  - G9080D

Startups in China

- **Cambricon**
  - MLU100/270/200
  - Journey
- **BITMAIN**
  - BM1680/1880
  - DeepEye1000
- **Yitu**
  - QuestCore
  - NTT1
- **Xiaomi**
  - I210
- **AutoNoms**
  - Ar9000
- **Qanqiang**
  - Voilin611
  - TX200/510
- **Tadao**
  - TDA010
  - Houshan
- **Tachyum**
  - MemCore
  - DU

Automated Driving

- **Tesla**
- **NVIDIA**
- **Baidu**
- **Western Digital**
- **Fujitsu**
- **Dell**
- **Nokia**
- **LG**
- **inspur***
- **Hewlett Packard Enterprise**
- **Huawei**
- **Samsung**
- **Infineon**
- **ARM**
- **AMD**
- **MediaTek**
- **Nvidia**
- **Qualcomm**
- **Rockchip**
- **Samsung**
- **Texas Instruments**
- **RENESAS**
- **TOYOTA**
- **Chi Mei**
- **CROX/STOX**

Smart Voice

- **TensorFlow**
- **MLIR**
- **Glow**
- **MT**
- **OctoML**
- **CUDA**
- **TensorRT**
- **SpakML**
- **nGraph**
- **ONNC**
- **Tiramisu Compiler**
- **The Tensor Algebra Compiler (taco)**

Compilers

- **MLPerf**
- **AI - Benchmark**
- **AI Matrix**
- **DAWN Bench**
- **MLMark**

Startups Worldwide

- **FPGA/eFPGA**
- **Achronix**
- **Finisar**
- **Elixir**
- **Graphcore**
- **GC2**
- **SambaNova**
- **Areanna**
- **MYTHIC**
- **Syntiant**
- **Graphcore Technology**
- **GAINBOARD/Lightspur**
- **Shift/Wave**
- **Nexi**
- **Hailo**
- **Hailo**
- **Kalray**
- **MPAA:25B**
- **Grok**
- **Tachyum**
- **Brainchip**
- **Frontier Networks**
- **MNP-Core**
- **Innogrit**
- **Neuralized**
- **BTA Compute**
- **ECAI33**
- **Greenwaves**
- **GAP8**
- **Preferred Networks**
- **MN-20**
- **SIEMENS**
- **Multimat**
- **Neuroblade**
- **Novumind**
- **TeraSystem**
- **Amotive**
- **eSilicon**

More at https://basicmi.github.io/AI-Chip/

SAFARI

https://basicmi.github.io/AI-Chip/
UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.

- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth

---

Experimental Analysis of the UPMEM PIM Engine

Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
IZZAT EL HAJJ, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this data movement bottleneck requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as processing-in-memory (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PrIM (Processing-In-Memory benchmarks), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

UPMEM Memory Modules

- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz
Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
IZZAT EL HAJJI, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
CHRISTINA GIANNOLA, ETH Zürich, Switzerland and NTUA, Greece
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is manifestly to the cost of main memory access. Fundamentally addressing this data movement bottleneck requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as processing-in-memory (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPNEM company has designed and manufactured the first publicly available real-world PIM architecture. The UPNEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Unit (DPU), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPNEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PIM (processing in-memory) benchmark, a benchmark suite of 15 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PIM benchmarks on the UPNEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPNEM-based PIM systems with 160 and 2,560 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

Understanding a Modern PIM Architecture

Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

Juan Gómez Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, Onur Mutlu

https://github.com/CMU-SAFARI/prim-benchmarks

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

2,579 views • Streamed live on Jul 12, 2021

https://www.youtube.com/watch?v=D8Hjy2iU9I4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
More on Analysis of the UPMEM PIM Engine

Inter-DPU Communication

- There is **no direct communication channel between DPUs**

Inter-DPU communication takes places via the host CPU using CPU-DPU and DPU-CPU transfers.

- Example communication patterns:
  - Merging of partial results to obtain the final result
  - Only DPU-CPU transfers
  - Redistribution of intermediate results for further computation
  - DPU-CPU transfers and CPU-DPU transfers

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

1,868 views • Streamed live on Jul 12, 2021

Onur Mutlu Lectures
17.6K subscribers

Talk Title: Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization
Dr. Juan Gómez-Luna, SAFARI Research Group, D-ITET, ETH Zurich

https://www.youtube.com/watch?v=D8Hjy2iU9I4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
Data Movement in Computing Systems

- **Data movement** dominates **performance** and is a major system **energy bottleneck**
- **Total system energy**: data movement accounts for
  - 62% in consumer applications
  - 40% in scientific applications
  - 35% in mobile applications

---

Understanding a Modern Processing-in-Memory Arch: Benchmarking & Experimental Characterization; 21m

3,482 views • Premiered Jul 25, 2021

https://www.youtube.com/watch?v=Pp9jSU2b9oM&list=PL5Q2soXY2Zi8_VVChACnON4sfh2bJ5IrD&index=159
FPGA-based Processing Near Memory


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh♦ Mohammed Alser♦ Damla Senol Cali♦
Dionysios Diamantopoulos∇ Juan Gómez-Luna♦
Henk Corporaal* Onur Mutlu♦∇

♦ETH Zürich ♣Carnegie Mellon University
*Eindhoven University of Technology ∇IBM Research Europe
Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry’s first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power — the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, “Our groundbreaking HBM-PIM is the industry’s first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications.”

Samsung Function-in-Memory DRAM (2021)

- FIMDRAM based on HBM2

[3D Chip Structure of HBM with FIMDRAM]

<table>
<thead>
<tr>
<th>Chip Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>128DQ / 8CH / 16 banks / BL4</td>
</tr>
<tr>
<td>32 PCU blocks (1 FIM block/2 banks)</td>
</tr>
<tr>
<td>1.2 TFLOPS (4H)</td>
</tr>
<tr>
<td>FP16 ADD / Multiply (MUL) / Multiply-Accumulate (MAC) / Multiply-and-Add (MAD)</td>
</tr>
</tbody>
</table>

ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Hassuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyoon Choi¹, Hyun-Sung Shin¹, Jin Kim¹, BengSeng Phua¹, HyoYoungKim¹, Myeong Jun Song¹, Ahn Choi¹, Daeho Kim¹, SooYoungKim¹, Eun-Bong Kim¹, David Wang¹, Shinthaeng Kang¹, Yuwan Ro¹, Seungwoo Seo¹, JoonHo Song¹, Jaryoun Youn¹, Kyomin Sohn¹, Nam Sung Kim¹

¹Samsung Electronics, Hwasung, Korea
²Samsung Electronics, San Jose, CA
³Samsung Electronics, Suwon, Korea
Future of Genome Sequencing & Analysis

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu


Accelerating Genome Analysis: A Primer on an Ongoing Journey
DOI Bookmark: 10.1109/MM.2020.3013728

FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications
DOI Bookmark: 10.1109/MM.2021.3088396

MinION from ONT

SmidgION from ONT
To achieve the highest energy efficiency and performance:

**we must take the expanded view**
of computer architecture

Co-design across the hierarchy:
Algorithms to devices

Specialize as much as possible within the design goals
What Kind of a Future
Do We Want?
How Reliable/Secure/Safe is This Bridge?
Collapse of the “Galloping Gertie”

Source: AP
http://www.wsdot.wa.gov/tnbhistory/connections/connections3.htm
Another View
How Secure Are These People?

Security is about preventing unforeseen consequences
How Safe & Secure Is This Platform?

Source: https://taxistartup.com/wp-content/uploads/2015/03/UK-Self-Driving-Cars.jpg
Reliable, Secure, Safe
Do We Want This?

Source: V. Milutinovic
Or This?

Source: V. Milutinovic
Challenge and Opportunity for Future

Sustainable and Energy Efficient
Many Difficult Problems: Climate

Source: https://farm9.staticflickr.com/8571/16376102935_8628150df8_o.jpg
Many Difficult Problems: Congestion

Many Difficult Problems: Intelligence

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg
Many Difficult Problems: Public Health

Source: https://blog.wego.com/7-crowded-places-and-events-that-you-will-love/
Many Difficult Problems: Genome Analysis

devlopment of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

Accelerating Genome Analysis

Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
"Accelerating Genome Analysis: A Primer on an Ongoing Journey"
[Slides (pptx)(pdf)]
[Talk Video (1 hour 2 minutes)]
GenASMS Framework [MICRO 2020]


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali†, Gurpreet S. Kalsi‡, Zülal Bingöl¶, Can Firtina●, Lavanya Subramanian‡, Jeremie S. Kim‡, Rachata Ausavarungnirun©, Mohammed Alser●, Juan Gomez-Luna●, Amirali Boroumand‡, Anant Nori‡, Allison Scibisz†, Sreenivas Subramoney●, Can Alkan, Saugata Ghose†, Onur Mutlu‡,†,‡

†Carnegie Mellon University  ‡Intel Labs  ¶Bilkent University  ●ETH Zürich  ○King Mongkut’s University of Technology North Bangkok  *University of Illinois at Urbana–Champaign
Future of Genome Sequencing & Analysis

Mohammed Alser, Züal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu


Accelerating Genome Analysis: A Primer on an Ongoing Journey
DOI Bookmark: 10.1109/MM.2020.3013728

FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications
DOI Bookmark: 10.1109/MM.2021.3088396

MinION from ONT

SmidgION from ONT
More on Fast & Efficient Genome Analysis

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
Detailed Lectures on Genome Analysis

- **Computer Architecture, Fall 2020, Lecture 3a**
  - Introduction to Genome Sequence Analysis (ETH Zürich, Fall 2020)
  - [Video Link](https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5)

- **Computer Architecture, Fall 2020, Lecture 8**
  - Intelligent Genome Analysis (ETH Zürich, Fall 2020)
  - [Video Link](https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14)

- **Computer Architecture, Fall 2020, Lecture 9a**
  - GenASM: Approx. String Matching Accelerator (ETH Zürich, Fall 2020)
  - [Video Link](https://www.youtube.com/watch?v=XoLpzmNPas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15)

- **Accelerating Genomics Project Course, Fall 2020, Lecture 1**
  - Accelerating Genomics (ETH Zürich, Fall 2020)
  - [Video Link](https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqlgwiDRQDTyId)

[SAFARI](https://www.youtube.com/onurmutlulectures)
Challenge and Opportunity for Future

High Performance

(to solve the toughest & all problems)
Personalized Medicine

Source: Jane Ades, NHGRI
Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali+, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018    Article history ▼

[Preliminary arxiv.org version]
Challenge and Opportunity for Future

Personalized and Private

(in every aspect of life: health, medicine, spaces, devices, robotics, ...)

SAFARI
This Lecture is About …

- **Questioning** what limits us in designing the best computing architectures for the future

- **Providing directions** for fundamentally better designs

- **Advocating** principled approaches
Increasingly Demanding Applications

Dream...

and, they will come

As applications push boundaries, computing platforms will become increasingly strained.
Key Realization
Modern Systems are Bottlenecked by Data Storage and Movement
Focus is on Data Storage Systems (Memory)

- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor

- Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits
Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor.

Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits.
Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor.

Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits.
Computing is Bottlenecked by Data
Data is Key for AI, ML, Genomics, ...

- Important workloads are all data intensive

- They require rapid and efficient processing of large amounts of data

- Data is increasing
  - We can generate more than we can process
Memory Is Critical for Performance (I)

In-memory Databases
[Mao+, EuroSys’12; Clapp+ (Intel), IISWC’15]

Graph/Tree Processing
[Xu+, IISWC’12; Umuroglu+, FPL’15]

In-Memory Data Analytics
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
Memory Is Critical for Performance (I)

In-memory Databases

Graph/Tree Processing

Memory → bottleneck

In-Memory Data Analytics
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
Memory Is Critical for Performance (II)

Chrome
Google’s web browser

TensorFlow Mobile
Google’s machine learning framework

VP9
YouTube
Video Playback
Google’s video codec

VP9
YouTube
Video Capture
Google’s video codec
Memory Is Critical for Performance (II)

Chrome

TensorFlow Mobile

Memory → bottleneck

VP9

Video Playback
Google’s video codec

Video Capture
Google’s video codec
1 Sequencing

reference: TTTATCGCTTCCATGACGCAG
read1: ATCGCATCC
read2: TATCGCATC
read3: CATCCATGA
read4: CGCTTCCAT
read5: CCATGACGC
read6: TTCCATGAC

2 Genome Analysis

3 Variant Calling

4 Scientific Discovery
Memory → bottleneck

Reference: TTTATCGCTTCCATGACGCACG
read1: ATCGCATCC
read2: TATCGCATC
read3: CATCCATGA
read4: CGCTTCCAT
read5: CCATGACGC
read6: TTCCATGAC
New Genome Sequencing Technologies

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali, Jeremie S. Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018  Article history ▼

[Open arxiv.org version]
Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

*Briefings in Bioinformatics*, bby017, [https://doi.org/10.1093/bib/bby017](https://doi.org/10.1093/bib/bby017)

Published: 02 April 2018  Article history ▼
Memory is Critical for Energy


62.7% of the total system energy is spent on data movement

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand\(^1\)  Rachata Ausavarungnirun\(^1\)  Aki Kuusela\(^3\)  Allan Knies\(^3\)
Saugata Ghose\(^1\)  Eric Shiu\(^3\)  Rahul Thakur\(^3\)  Parthasarathy Ranganathan\(^3\)
Youngsok Kim\(^2\)  Daehyun Kim\(^4,3\)  Onur Mutlu\(^5,1\)

SAFARI
Memory is Critical for Reliability

- Data from all of Facebook’s servers worldwide
- Meza+, “Revisiting Memory Errors in Large-Scale Production Data Centers,” DSN’15.

As memory capacity increases, system reliability reduces.
Large-Scale Failure Analysis of DRAM Chips

- Analysis and modeling of memory errors found in all of Facebook’s server fleet

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"

Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.

[Slides (pptx) (pdf)] [DRAM Error Model]
Modern Systems are Bottlenecked by Memory
An “Early” Overview Paper…

- Onur Mutlu,

"Memory Scaling: A Systems Architecture Perspective"
Proceedings of the *5th International Memory Workshop (IMW)*, Monterey, CA, May 2013. Slides (pptx) (pdf)
EETimes Reprint

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

Four Key Issues in Future Platforms

- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
  - Memory-centric (Data-centric) Architectures
- Fundamentally **Low-Latency and Predictable** Architectures
- Architectures for **AI/ML, Genomics, Medicine, Health**
Maslow’s (Human) Hierarchy of Needs


- We need to start with reliability, security, safety...
How Reliable/Secure/Safe is This Bridge?

Source: http://www.technologystudent.com/struct1/tacom1.png
Collapse of the “Galloping Gertie”

Source: AP
http://www.wsdot.wa.gov/tnbhistory/connections/connections3.htm
How Secure Are These People?

Security is about preventing unforeseen consequences
The Problem

We do not seem to have design principles for guaranteeing reliability and security.
As Memory Scales, It Becomes Unreliable

- Data from all of Facebook’s servers worldwide
- Meza+, “Revisiting Memory Errors in Large-Scale Production Data Centers,” DSN’15.

Intuition: quadratic increase in capacity
The DRAM Scaling Problem

- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor must be large enough for long data retention time

- As DRAM cell becomes smaller, it becomes more vulnerable
Infrastructures to Understand Such Issues

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study (Khan et al., SIGMETRICS 2014)

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems (Qureshi et al., DSN 2015)
Infrastructures to Understand Such Issues

SoftMC: Open Source DRAM Infrastructure


- Flexible
- Easy to Use (C++ API)
- Open-source

github.com/CMU-SAFARI/SoftMC
SoftMC

- https://github.com/CMU-SAFARI/SoftMC

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan\textsuperscript{1,2,3}  Nandita Vijaykumar\textsuperscript{3}  Samira Khan\textsuperscript{4,3}  Saugata Ghose\textsuperscript{3}  Kevin Chang\textsuperscript{3}  Gennady Pekhimenko\textsuperscript{5,3}  Donghyuk Lee\textsuperscript{6,3}  Oguz Ergin\textsuperscript{2}  Onur Mutlu\textsuperscript{1,3}

\textsuperscript{1}ETH Zürich  \textsuperscript{2}TOBB University of Economics & Technology  \textsuperscript{3}Carnegie Mellon University  \textsuperscript{4}University of Virginia  \textsuperscript{5}Microsoft Research  \textsuperscript{6}NVIDIA Research
One can predictably induce errors in most DRAM memory chips
A simple hardware failure mechanism can create a widespread system security vulnerability.
Repeatedly reading a row enough times (before memory gets refreshed) induces disturbance errors in adjacent rows in most real DRAM chips you can buy today.

*Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors*, (Kim et al., ISCA 2014)
Most DRAM Modules Are Vulnerable

A company

86%
(37/43)

Up to
1.0×10^7
errors

B company

83%
(45/54)

Up to
2.7×10^6
errors

C company

88%
(28/32)

Up to
3.3×10^5
errors

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)
Recent DRAM Is More Vulnerable

- A Modules
- B Modules
- C Modules

Errors per $10^9$ Cells

Module Vintage

2008 2009 2010 2011 2012 2013 2014
Recent DRAM Is More Vulnerable
Recent DRAM Is More Vulnerable

All modules from 2012–2013 are vulnerable
A Simple Program Can Induce Many Errors

```
loop:
    mov (X), %eax
    mov (Y), %ebx
    clflush (X)
    clflush (Y)
    mfence
    jmp loop
```

Download from: https://github.com/CMU-SAFARI/rowhammer
A Simple Program Can Induce Many Errors

```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```

Download from: https://github.com/CMU-SAFARI/rowhammer
A Simple Program Can Induce Many Errors

```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```

Download from: [https://github.com/CMU-SAFARI/rowhammer](https://github.com/CMU-SAFARI/rowhammer)
A Simple Program Can Induce Many Errors

```assembly
loop:
mov (X), %eax
mov (Y), %ebx
clflush (X)
clflush (Y)
mfence
jmp loop
```

Download from: [https://github.com/CMU-SAFARI/rowhammer](https://github.com/CMU-SAFARI/rowhammer)
## Observed Errors in Real Systems

<table>
<thead>
<tr>
<th>CPU Architecture</th>
<th>Errors</th>
<th>Access-Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Haswell (2013)</td>
<td>22.9K</td>
<td>12.3M/sec</td>
</tr>
<tr>
<td>Intel Ivy Bridge (2012)</td>
<td>20.7K</td>
<td>11.7M/sec</td>
</tr>
<tr>
<td>Intel Sandy Bridge (2011)</td>
<td>16.1K</td>
<td>11.6M/sec</td>
</tr>
<tr>
<td>AMD Piledriver (2012)</td>
<td>59</td>
<td>6.1M/sec</td>
</tr>
</tbody>
</table>

A real reliability & security issue

One Can Take Over an Otherwise-Secure System

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology

Project Zero

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

News and updates from the Project Zero team at Google

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)
RowHammer Security Attack Example

“Rowhammer” is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows (Kim et al., ISCA 2014).

- Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

- We tested a selection of laptops and found that a subset of them exhibited the problem.
- We built two working privilege escalation exploits that use this effect.
  - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)

- One exploit uses rowhammer-induced bit flips to gain kernel privileges on x86-64 Linux when run as an unprivileged userland process.
- When run on a machine vulnerable to the rowhammer problem, the process was able to induce bit flips in page table entries (PTEs).
- It was able to use this to gain write access to its own page table, and hence gain read-write access to all of physical memory.

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn & Dullien, 2015)
Security Implications

Rowhammer
Security Implications

It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after
More Security Implications (I)

“We can gain unrestricted access to systems of website visitors.”

Not there yet, but...

ROOT privileges for web apps!

Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript (DIMVA'16)

Source: https://lab.dsst.io/32c3-slides/7197.html
More Security Implications (II)

“Can gain control of a smart phone deterministically”

Hammer And Root

Millions of Androids

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/

Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS’16
More Security Implications (III)

- Using an integrated GPU in a mobile system to remotely escalate privilege via the WebGL interface

"GRAND PWNING UNIT" — Drive-by Rowhammer attack uses GPU to compromise an Android phone

JavaScript based GLitch pwns browsers by flipping bits inside memory chips.

DAN GOODIN - 5/3/2018, 12:00 PM

Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU

Pietro Frigo
Vrije Universiteit
Amsterdam
p.frigo@vu.nl

Cristiano Giuffrida
Vrije Universiteit
Amsterdam
giuffrida@cs.vu.nl

Herbert Bos
Vrije Universiteit
Amsterdam
herbertb@cs.vu.nl

Kaveh Razavi
Vrije Universiteit
Amsterdam
kaveh@cs.vu.nl
Packets over a LAN are all it takes to trigger serious Rowhammer bit flips.

The bar for exploiting potentially serious DDR weakness keeps getting lower.

DAN GOODIN - 5/10/2018, 5:26 PM

Throwhammer: Rowhammer Attacks over the Network and Defenses

Andrei Tatar
VU Amsterdam

Radhesh Krishnan
VU Amsterdam

Elias Athanasopoulos
University of Cyprus

Herbert Bos
VU Amsterdam

Kaveh Razavi
VU Amsterdam

Cristiano Giuffrida
VU Amsterdam
More Security Implications (V)

- Rowhammer over RDMA (II)

Nethammer—Exploiting DRAM Rowhammer Bug Through Network Requests

**Nethammer:**
*Inducing Rowhammer Faults through Network Requests*

- Moritz Lipp
  Graz University of Technology

- Misiker Tadesse Aga
  University of Michigan

- Michael Schwarz
  Graz University of Technology

- Daniel Gruss
  Graz University of Technology

- Clémentine Maurice
  Univ Rennes, CNRS, IRISA

- Lukas Raab
  Graz University of Technology

- Lukas Lamster
  Graz University of Technology
RAMBleed: Reading Bits in Memory Without Accessing Them

Andrew Kwong  
*University of Michigan*  
ankwong@umich.edu

Daniel Genkin  
*University of Michigan*  
genkin@umich.edu

Daniel Gruss  
*Graz University of Technology*  
daniel.gruss@iaik.tugraz.at

Yuval Yarom  
*University of Adelaide and Data61*  
yval@cs.adelaide.edu.au
More Security Implications (VII)

- USENIX Security 2019

Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks

Sanghyun Hong, Pietro Frigo†, Yiğitcan Kaya, Cristiano Giuffrida‡, Tudor Dumitraş

University of Maryland, College Park
†Vrije Universiteit Amsterdam

A Single Bit-flip Can Cause Terminal Brain Damage to DNNs

One specific bit-flip in a DNN's representation leads to accuracy drop over 90%

Our research found that a specific bit-flip in a DNN's bitwise representation can cause the accuracy loss up to 90%, and the DNN has 40-50% parameters, on average, that can lead to the accuracy drop over 10% when individually subjected to such single bitwise corruptions...

Read More
More Security Implications (VIII)

USENIX Security 2020

DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips

Fan Yao  
University of Central Florida  
fan.yao@ucf.edu

Adnan Siraj Rakin  
Arizona State University  
asrakin@asu.edu

Deliang Fan  

Degrade the inference accuracy to the level of Random Guess

Example: ResNet-20 for CIFAR-10, 10 output classes
Before attack, Accuracy: 90.2%  After attack, Accuracy: ~10% (1/10)
More Security Implications (IX)

- Rowhammer on MLC NAND Flash (based on [Cai+, HPCA 2017])

Security

Rowhammer RAM attack adapted to hit flash storage

Project Zero's two-year-old dog learns a new trick

By Richard Chirgwin 17 Aug 2017 at 04:27

From random block corruption to privilege escalation:
A filesystem attack vector for rowhammer-like attacks

Anil Kurmus  Nikolas Ioannou  Matthias Neugschwandtner  Nikolaos Papandreou  Thomas Parnell
IBM Research – Zurich
More Security Implications?
Apple’s Patch for RowHammer


Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application may induce memory corruption to escalate privileges

Description: A disturbance error, also known as Rowhammer, exists with some DDR3 RAM that could have led to memory corruption. **This issue was mitigated by increasing memory refresh rates.**

CVE-ID

CVE-2015-3693: Mark Seaborn and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

HP, Lenovo, and other vendors released similar patches
Solution Direction: Principled Designs

Design fundamentally secure computing architectures

Predict and prevent such safety issues
Our Solution to RowHammer

- **PARA:** Probabilistic Adjacent Row Activation

- **Key Idea**
  - After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability: $p = 0.005$

- **Reliability Guarantee**
  - When $p = 0.005$, errors in one year: $9.4 \times 10^{-14}$
  - By adjusting the value of $p$, we can vary the strength of protection against errors
Advantages of PARA

• *PARA refreshes rows infrequently*
  – Low power
  – Low performance-overhead
    • Average slowdown: *0.20%* (for 29 benchmarks)
    • Maximum slowdown: *0.75%*

• *PARA is stateless*
  – Low cost
  – Low complexity

• *PARA is an effective and low-overhead solution to prevent disturbance errors*
Requirements for PARA

• If implemented in **DRAM chip** (done today)
  – Enough slack in timing and refresh parameters
  – Plenty of slack today:
    • Chang et al., “Understanding Latency Variation in Modern DRAM Chips,” SIGMETRICS 2016.
    • Lee et al., “Design-Induced Latency Variation in Modern DRAM Chips,” SIGMETRICS 2017.
    • Chang et al., “Understanding Reduced-Voltage Operation in Modern DRAM Devices,” SIGMETRICS 2017.

• If implemented in **memory controller** (done today)
  – Better coordination between memory controller and DRAM
  – Memory controller should know which rows are physically adjacent
Probabilistic Activation in Real Life (I)

https://twitter.com/isislovecruft/status/1021939922754723841
Probabilistic Activation in Real Life (II)

https://twitter.com/isislovecruft/status/1021939922754723841
First RowHammer Analysis

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,

"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"


[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data] [Lecture Video (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).
Retrospective on RowHammer & Future

- Onur Mutlu,
  "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
  [Slides (pptx) (pdf)]

The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu
[Slides from COSADE 2019 (pptx)]
[Slides from VLSI-SOC 2020 (pptx) (pdf)]
[Talk Video (1 hr 15 minutes, with Q&A)]
RowHammer in 2020
RowHammer in 2020 (I)

- Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,

"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"


[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim$†, Minesh Patel$, A. Giray Yağlıkçı$, Hasan Hassan$, Roknoddin Azizi$, Lois Orosa$, Onur Mutlu$†

$ETH Zürich †Carnegie Mellon University
Key Takeaways from 1580 Chips

• Newer DRAM chips are more vulnerable to RowHammer

• There are chips today whose weakest cells fail after only 4800 hammers

• Chips of newer DRAM technology nodes can exhibit RowHammer bit flips 1) in more rows and 2) farther away from the victim row.

• Existing mitigation mechanisms are NOT effective
RowHammer in 2020 (II)

- Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi,

"TRRespass: Exploiting the Many Sides of Target Row Refresh"


[Slides (pptx) (pdf)]
[ Lecture Slides (pptx) (pdf)]
[ Talk Video (17 minutes)]
[ Lecture Video (59 minutes)]
[ Source Code]
[ Web Article]

Best paper award.
Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020

TRRespass: Exploiting the Many Sides of Target Row Refresh

Pietro Frigo*†   Emanuele Vannacci*†   Hasan Hassan§   Victor van der Veen¶
Onur Mutlu§   Cristiano Giuffrida*   Herbert Bos*   Kaveh Razavi*

*Vrije Universiteit Amsterdam
§ETH Zürich
¶Qualcomm Technologies Inc.
RowHammer is still an open problem

Security by obscurity is likely not a good solution
RowHammer in 2020 (III)

- Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu,

"Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers"


[Slides (pptx) (pdf)]
[Talk Video (17 minutes)]

Are We Susceptible to Rowhammer?
An End-to-End Methodology for Cloud Providers

Lucian Cojocar, Jeremie Kim§†, Minesh Patel§, Lillian Tsai‡,
Stefan Saroiu, Alec Wolman, and Onur Mutlu§†
Microsoft Research, §ETH Zürich, †CMU, ‡MIT
RowHammer in 2020 (IV)

**Session 1A: Security & Privacy I**

5:00 PM CEST – 5:15 PM CEST
**Graphene: Strong yet Lightweight Row Hammer Protection**
Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, Jae W. Lee (Seoul National University)

5:15 PM CEST – 5:30 PM CEST
**Persist Level Parallelism: Streamlining Integrity Tree Updates for Secure Persistent Memory**
Alexander Freij, Shougang Yuan, Huiyang Zhou (NC State University); Yan Solihin (University of Central Florida)

5:30 PM CEST – 5:45 PM CEST
**PThammer: Cross-User-Kernel-Boundary Rowhammer through Implicit Accesses**
Zhi Zhang (University of New South Wales and Data61, CSIRO, Australia); Yueqiang Cheng (Baidu Security); Dongxi Liu, Surya Nepal (Data61, CSIRO, Australia); Zhi Wang (Florida State University); Yuval Yarom (University of Adelaide and Data61, CSIRO, Australia)
RowHammer in 2020 (V)

Session #5: Rowhammer

Session chair: Michael Franz (UC Irvine)

RAMBleed: Reading Bits in Memory Without Accessing Them
Andrew Kwong (University of Michigan), Daniel Genkin (University of Michigan), Daniel Gruss (Data61)

Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers
Lucian Cojocar (Microsoft Research), Jeremie Kim (ETH Zurich, CMU), Minesh Patel (ETH Zurich, Microsoft Research), Onur Mutlu (ETH Zurich, CMU)

Leveraging EM Side-Channel Information to Detect Rowhammer Attacks
Zhenkai Zhang (Texas Tech University), Zihao Zhan (Vanderbilt University), Daniel Balasubramanian, Peter Volgyesi (Vanderbilt University), Xenofon Koutsoukos (Vanderbilt University)

TRRespase: Exploiting the Many Sides of Target Row Refresh
Pietro Frigo (Vrije Universiteit Amsterdam, The Netherlands), Emanuele Vannacci (Vrije Universiteit, Amsterdam), Cristiano Giuffrida (Vrije Universiteit, Amsterdam, The Netherlands), Kaveh Razavi (Vrije Universiteit Amsterdam, The Netherlands)
DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips

Fan Yao, University of Central Florida; Adnan Siraj Rakin and Deliang Fan, Arizona State University
A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]
Two Key RowHammer Papers at MICRO 2021

[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[arXiv version]

A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa*  
ETH Zürich

A. Giray Yağlıkçı*  
ETH Zürich

Haocong Luo  
ETH Zürich

Ataberk Olgun  
ETH Zürich, TOBB ETÜ

Jisung Park  
ETH Zürich

Hasan Hassan  
ETH Zürich

Minesh Patel  
ETH Zürich

Jeremie S. Kim  
ETH Zürich

Onur Mutlu  
ETH Zürich
Two Key RowHammer Papers at MICRO 2021

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu,

"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"

Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.

[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[arXiv version]

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan†  Yahya Can Tuğrul†‡  Jeremie S. Kim†  Victor van der Veenσ
Kaveh Razavi†  Onur Mutlu†

†ETH Zürich  ‡TOBB University of Economics & Technology  σQualcomm Technologies Inc.
More to Come...
Detailed Lectures on RowHammer

- **Computer Architecture, Fall 2020, Lecture 4b**
  - RowHammer (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=KDr632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8](https://www.youtube.com/watch?v=KDr632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8)

- **Computer Architecture, Fall 2020, Lecture 5a**
  - RowHammer in 2020: TRRespas (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9](https://www.youtube.com/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9)

- **Computer Architecture, Fall 2020, Lecture 5b**
  - RowHammer in 2020: Revisiting RowHammer (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=qR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10](https://www.youtube.com/watch?v=qR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10)

- **Computer Architecture, Fall 2020, Lecture 5c**
  - Secure and Reliable Memory (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11](https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11)

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
The Story of RowHammer Lecture ...

- Onur Mutlu, "The Story of RowHammer"
  Keynote Talk at Secure Hardware, Architectures, and Operating Systems Workshop (SeHAS), held with HiPEAC 2021 Conference, Virtual, 19 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]
Mitigation Mechanism Evaluation

Ideal mechanism is significantly better than any existing mechanism for $HC_{first} < 1024$

Significant opportunity for developing a RowHammer solution with low performance overhead that supports low $HC_{first}$
Future of Main Memory Reliability

- DRAM is becoming less reliable → more vulnerable

- Due to difficulties in DRAM scaling, other problems may also appear (or they may be going unnoticed)

  - Some errors may already be slipping into the field
    - Read disturb errors (Rowhammer)
    - Retention errors
    - Read errors, write errors
    - ...

- These errors can also pose security vulnerabilities
All Memory Technologies are Vulnerable

- DRAM
- Flash memory
- Emerging Technologies
  - Phase Change Memory
  - STT-MRAM
  - RRAM, memristors
  - ...
How Do We Keep Memory Secure?

- **Understand:** Methodologies for failure modeling and discovery
  - Modeling and prediction based on real (device) data

- **Architect:** Principled co-architecting of system and memory
  - Good partitioning of duties across the stack

- **Design & Test:** Principled design, automation, testing
  - High coverage and good interaction with system reliability methods
Understand and Model with Experiments (DRAM)

Understand and Model with Experiments (Flash)

HAPS-52 Mother Board

Virtex-V FPGA
(NAND Controller)

Virtex-II Pro
(USB controller)

USB Daughter Board

NAND Daughter Board


Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD’s reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

https://arxiv.org/pdf/1706.08642
Planar vs. 3D NAND Flash Memory

Planar NAND Flash Memory
- Scaling: Reduce flash cell size, Reduce distance b/w cells
- Reliability: Scaling hurts reliability
  - Not well studied!

3D NAND Flash Memory
- Scaling: Increase # of layers
Fundamentally Secure, Reliable, Safe Computing Architectures
One Important Takeaway

Main Memory Needs
Intelligent Controllers
In-Field Patch-ability (Intelligent Memory) Can Avoid Many Failures
Four Key Issues in Future Platforms

- Fundamentally Secure/Reliable/Safe Architectures

- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally Low-Latency and Predictable Architectures

- Architectures for AI/ML, Genomics, Medicine, Health
Maslow’s (Human) Hierarchy of Needs, Revisited


Everlasting energy

Source: https://www.simplypsychology.org/maslow.html
Do We Want This?

Source: V. Milutinovic
Or This?
Challenge and Opportunity for Future

High Performance,
Energy Efficient,
Sustainable
The Problem

Data access is the major performance and energy bottleneck

Our current design principles cause great energy waste
(and great performance loss)
The Problem

Processing of data is performed far away from the data.
A Computing System

- Three key components
- Computation
- Communication
- Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.
A Computing System

- Three key components
- Computation
- Communication
- Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.

Today’s Computing Systems

- Are overwhelmingly processor centric
- All data processed in the processor → at great system cost
- Processor is heavily optimized and is considered the master
- Data storage units are dumb and are largely unoptimized (except for some that are on the processor die)
Yet …

- “It’s the Memory, Stupid!” (Richard Sites, MPR, 1996)

I expect that over the coming decade memory subsystem design will be the only important design issue for microprocessors.

Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors

Onur Mutlu §  Jared Stark †  Chris Wilkerson ‡  Yale N. Patt §

§ECE Department  †Microprocessor Research  ‡Desktop Platforms Group
The University of Texas at Austin  Intel Labs  Intel Corporation
{onur,patt}@ece.utexas.edu  jared.w.stark@intel.com  chris.wilkerson@intel.com
The Performance Perspective (2015)

- All of Google’s Data Center Workloads (2015):

The Performance Perspective (2015)

- All of Google’s Data Center Workloads (2015):

**Figure 11: Half of cycles are spent stalled on caches.**

Perils of Processor-Centric Design

- Grossly-imbalanced systems
  - Processing done only in **one place**
  - Everything else just stores and moves data: **data moves a lot**
    - Energy inefficient
    - Low performance
    - Complex

- Overly complex and bloated processor (and accelerators)
  - To tolerate data access from memory
  - Complex hierarchies and mechanisms
    - Energy inefficient
    - Low performance
    - Complex
Most of the system is dedicated to storing and moving data yet, system is still bottlenecked by memory.
Three Key Systems Trends

1. Data access is a major bottleneck
   - Applications are increasingly data hungry

2. Energy consumption is a key limiter

3. Data movement energy dominates compute
   - Especially true for off-chip to on-chip movement
Data Movement vs. Computation Energy

Communication Dominates Arithmetic

Dally, HiPEAC 2015

- 64-bit DP: 20 pJ
- 256-bit buses
- 256-bit access 8 kB SRAM
- 26 pJ
- 256 pJ
- 16 nJ
- 500 pJ
- 1 nJ

Efficient off-chip link

DRAM Rd/Wr
A memory access consumes \(~100-1000\times\) the energy of a complex addition.
Data Movement vs. Computation Energy

- **Data movement** is a major system energy bottleneck
  - Comprises 41% of mobile system energy during web browsing [2]
  - Costs $\sim 115$ times as much energy as an ADD operation [1, 2]

---

[1]: Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO’16)
[2]: Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC’14)

SAFARI
62.7% of the total system energy is spent on data movement.
We Do Not Want to Move Data!

Communication Dominates Arithmetic

Dally, HiPEAC 2015

A memory access consumes $\sim 1000X$ the energy of a complex addition
We Need A Paradigm Shift To …

- Enable computation with **minimal data movement**

- **Compute where it makes sense** *(where data resides)*

- Make computing architectures more **data-centric**
Goal: Processing Inside Memory

Many questions ... How do we design the:
- compute-capable memory & controllers?
- processor chip and in-memory units?
- software and hardware interfaces?
- system software, compilers, languages?
- algorithms and theoretical foundations?
PIM Review and Open Problems

A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut’s University of Technology North Bangkok


\url{https://arxiv.org/pdf/1903.03988.pdf}
A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut's University of Technology North Bangkok

Abstract

Modern computing systems are overwhelmingly designed to move data to computation. This design choice goes directly against at least three key trends in computing that cause performance, scalability and energy bottlenecks: (1) data access is a key bottleneck as many important applications are increasingly data-intensive, and memory bandwidth and energy do not scale well, (2) energy consumption is a key limiter in almost all computing platforms, especially server and mobile systems, (3) data movement, especially off-chip to on-chip, is very expensive in terms of bandwidth, energy and latency, much more so than computation. These trends are especially severely-felt in the data-intensive server and energy-constrained mobile systems of today.

At the same time, conventional memory technology is facing many technology scaling challenges in terms of reliability, energy, and performance. As a result, memory system architects are open to organizing memory in different ways and making it more intelligent, at the expense of higher cost. The emergence of 3D-stacked memory plus logic, the adoption of error correcting codes inside the latest DRAM chips, proliferation of different main memory standards and chips, specialized for different purposes (e.g., graphics, low-power, high bandwidth, low latency), and the necessity of designing new solutions to serious reliability and security issues, such as the RowHammer phenomenon, are an evidence of this trend.

This chapter discusses recent research that aims to practically enable computation close to data, an approach we call processing-in-memory (PIM). PIM places computation mechanisms in or near where the data is stored (i.e., inside the memory chips, in the logic layer of 3D-stacked memory, or in the memory controllers), so that data movement between the computation units and memory is reduced or eliminated. While the general idea of PIM is not new, we discuss motivating trends in applications as well as memory circuits/technology that greatly exacerbate the need for enabling it in modern computing systems. We examine at least two promising new approaches to designing PIM systems to accelerate important data-intensive applications: (1) processing using memory by exploiting analog operational properties of DRAM chips to perform massively-parallel operations in memory, with low-cost changes, (2) processing near memory by exploiting 3D-stacked memory technology design to provide high memory bandwidth and low memory latency in-in-memory logic. In both approaches, we describe and tackle relevant cross-layer research, design, and adoption challenges in devices, architecture, systems, and programming models. Our focus is on the development of in-memory processing designs that can be adopted in real computing platforms at low cost. We conclude by discussing work on solving key challenges to the practical adoption of PIM.

Keywords: memory systems, data movement, main memory, processing-in-memory, near-data processing, computation-in-memory, processing using memory, processing near memory, 3D-stacked memory, non-volatile memory, energy efficiency, high-performance computing, computer architecture, computing paradigm, emerging technologies, memory scaling, technology scaling, dependable systems, robust systems, hardware security, system security, latency, low-latency computing
1. Introduction

Main memory, built using the Dynamic Random Access Memory (DRAM) technology, is a major component in nearly all computing systems, including servers, cloud platforms, mobile/embedded devices, and sensor systems. Across all of these systems, the data working set sizes of modern applications are rapidly growing, while the need for fast analysis of such data is increasing. Thus, main memory is becoming an increasingly significant bottleneck across a wide variety of computing systems and applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Allowing the main memory bottleneck requires the memory capacity, energy, cost, and performance to all scale in an efficient manner across technology generations. Unfortunately, it has become increasingly difficult in recent years, especially the past decade, to scale all of these dimensions [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].

A major reason for the main memory bottleneck is the high energy and latency cost associated with data movement. In modern computers, to perform any operation on data that resides in main memory, the processor must retrieve the data from main memory. This requires the memory controller to issue commands to a DRAM module across a relatively slow and power-hungry off-chip bus (known as the memory channel). DRAM module sends the requested data across the memory channel, after which the data is placed in the caches and registers. The CPU can perform computation on the data once the data is in its registers. Data movement from the DRAM to the CPU incurs long latency and consumes a significant amount of energy [7, 50, 51, 52, 53, 54]. These costs are often exacerbated by the fact that much of the data brought into the caches is not reused by the CPU [52, 53, 55, 56], providing little benefit in return for the high latency and energy cost.

The cost of data movement is a fundamental issue with the processor-centric nature of contemporary computer systems. The CPU is considered to be the master in the system, and computation is performed only in the processor (and accelerators). In contrast, data storage and communication units, including the main memory, are treated as unintelligent workers that are incapable of computation. As a result of this processor-centric design paradigm, data moves a lot in the system between the computation units and communication/ storage units so that computation can be done on it. With the increasingly data-centric nature of contemporary and emerging appli-
We Need to Think Differently from the Past Approaches
Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory
5.2. Two Approaches: Processing Using Memory (PUM) vs. Processing Near Memory (PNM)

Many recent works take advantage of the memory technology innovations that we discuss in Section 5.1 to enable and implement PIM. We find that these works generally take one of two approaches, which are categorized in Table 1: (1) processing using memory or (2) processing near memory. We briefly describe each approach here. Sections 6 and 7 will provide example approaches and more detail for both.

Table 1: Summary of enabling technologies for the two approaches to PIM used by recent works. Adapted from [309].

<table>
<thead>
<tr>
<th>Approach</th>
<th>Enabling Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Using Memory</td>
<td>SRAM DRAM Phase-change memory (PCM) Magnetic RAM (MRAM) Resistive RAM (RRAM)/memristors</td>
</tr>
<tr>
<td>Processing Near Memory</td>
<td>Logic layers in 3D-stacked memory Silicon interposers Logic in memory controllers</td>
</tr>
</tbody>
</table>

Processing using memory (PUM) exploits the existing memory architecture and the operational principles of the memory circuitry to enable operations within main memory with minimal changes. PUM makes use

Starting Simple: Data Copy and Initialization

*memmove & memcpy: 5% cycles in Google’s datacenter* [Kanev+ ISCA’15]

Forking

Zero initialization (e.g., security)

Checkpointing

VM Cloning

Deduplication

Page Migration

Many more
Today’s Systems: Bulk Data Copy

1) High latency
2) High bandwidth utilization
3) Cache pollution
4) Unwanted data movement

1046ns, 3.6μJ  (for 4KB page copy via DMA)
Future Systems: In-Memory Copy

1) Low latency
2) Low bandwidth utilization
3) No cache pollution
4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates
Negligible HW cost

Step 1: Activate row A
Step 2: Activate row B

DRAM subarray

Row Buffer (4 Kbytes)

8 bits

Data Bus
RowClone: Latency and Energy Savings

More on RowClone

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,

"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"

Proceedings of the 46th International Symposium on Microarchitecture (MICRO), Davis, CA, December 2013. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu

Carnegie Mellon University †Intel Pittsburgh
Can this be improved to do faster inter-subarray copy?
- Yes, see LISA [Chang et al., HPCA 2016]

Can we enable data movement at smaller granularities within a bank?
- Yes, see FIGARO [Wang et al., MICRO 2020]

Can this be improved to do better inter-bank copy?
- Yes, see Network-on-Memory [CAL 2020]

Can similar ideas and DRAM properties be used to perform computation on data?
- Yes, see Ambit [Seshadri et al., CAL 2015, MICRO 2017]
LIISA: Increasing Connectivity in DRAM

Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
[Slides (pptx) (pdf)]
[Source Code]

Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang†, Prashant J. Nair*, Donghyuk Lee†, Saugata Ghose†, Moinuddin K. Qureshi*, and Onur Mutlu†
†Carnegie Mellon University *Georgia Institute of Technology
FIGARO: Fine-Grained In-DRAM Copy

Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu,
"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"

FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching
Yaohua Wang† Lois Orosa† Xiangjun Peng‡* Yang Guo* Saugata Ghose‡† Minesh Patel† Jeremie S. Kim† Juan Gómez Luna† Mohammad Sadrosadati§ Nika Mansouri Ghiasi† Onur Mutlu†‡

*National University of Defense Technology †ETH Zürich ‡Chinese University of Hong Kong
§University of Illinois at Urbana–Champaign ‡‡Carnegie Mellon University §Institute of Research in Fundamental Sciences
Network-On-Memory: Fast Inter-Bank Copy

- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungrunirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,

"NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"

Mindset: Memory as an Accelerator

Memory similar to a “conventional” accelerator
In-Memory Bulk Bitwise Operations

- We can also support in-DRAM AND, OR, NOT, MAJ
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating multiple rows performs computation
- 30-60X performance and energy improvement

- New memory technologies enable even more opportunities
  - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
  - Can operate on data with minimal movement
In-DRAM AND/OR: Triple Row Activation

Final State

\[ AB + BC + AC \]

\[ C(A + B) + \sim C(AB) \]

In-DRAM Acceleration of Database Queries

Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,

"Fast Bulk Bitwise AND and OR in DRAM"

More on Ambit


Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri$^{1,5}$ Donghyuk Lee$^{2,5}$ Thomas Mullins$^{3,5}$ Hasan Hassan$^4$ Amirali Boroumand$^5$
Jeremie Kim$^{4,5}$ Michael A. Kozuch$^3$ Onur Mutlu$^{4,5}$ Phillip B. Gibbons$^5$ Todd C. Mowry$^5$

$^1$Microsoft Research India  $^2$NVIDIA Research  $^3$Intel  $^4$ETH Zürich  $^5$Carnegie Mellon University
In-DRAM Bulk Bitwise Execution


[Preliminary arXiv version]

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"

- [2-page Extended Abstract]
- [Short Talk Slides (pptx) (pdf)]
- [Talk Slides (pptx) (pdf)]
- [Short Talk Video (5 mins)]
- [Full Talk Video (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar¹,²  
Nika Mansouri Ghiasi¹

*Geraldo F. Oliveira¹  
Minesh Patel¹

Sven Gregorio¹  
Mohammed Alser¹

João Dinis Ferreira¹  
Saugata Ghose³

¹ETH Zürich  
²Simon Fraser University  
³University of Illinois at Urbana-Champaign
SIMDRAM Key Idea

• **SIMDRAM**: An end-to-end processing-using-DRAM framework that provides the programming interface, the ISA, and the hardware support for:

  - **Efficiently** computing *complex* operations in DRAM
  - Providing the ability to implement *arbitrary* operations as required
  - Using an *in-DRAM* massively-parallel SIMD substrate that requires *minimal* changes to DRAM architecture
SIMDRAM Framework: Overview

**User Input**

**Desired operation**

AND/OR/NOT logic

**SIMDRAM-enabled application**

```c
foo () {
    bbop_new
}
```

**Step 1: Generate MAJ logic**

MAJ

**Step 2: Generate sequence of DRAM commands**

- ACT/PRE
- ACT/PRE
- ACT/PRE
- ACT/ACT/PRE
- done

**SIMDRAM Output**

New SIMDRAM µProgram

µProgram

Main memory

- bbop_new

**New SIMDRAM instruction**

**Control Unit**

**µProgram**

**Memory Controller**

**Step 3: Execution according to µProgram**

- ACT/PRE
- ACT/PRE
- ACT/PRE
- ACT/PRE/PRE
- done

**Instruction result in memory**

ACT/PRE

SAFARI
More on the SIMDGRAM Framework


[2-page Extended Abstract]
[Short Talk Slides (pptx) (pdf)]
[Talk Slides (pptx) (pdf)]
[Short Talk Video (5 mins)]
[Full Talk Video (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar\(^1,2\)  \text{Nika Mansouri Ghiasi}\(^1\)

Geraldo F. Oliveira\(^1\)  Minesh Patel\(^1\)  Juan Gómez-Luna\(^1\)

Sven Gregorio\(^1\)  Mohammed Alser\(^1\)  Onur Mutlu\(^1\)  João Dinis Ferreira\(^1\)  Saugata Ghose\(^3\)

\(^1\)ETH Zürich  \(^2\)Simon Fraser University  \(^3\)University of Illinois at Urbana–Champaign
In-DRAM Physical Unclonable Functions

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu,
  "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"


[Lightning Talk Video]
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
[Full Talk Lecture Video (28 minutes)]
In-DRAM True Random Number Generation

Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"


[Slides (pptx) (pdf)]
[Full Talk Video (21 minutes)]
[Full Talk Lecture Video (27 minutes)]

Top Picks Honorable Mention by IEEE Micro.

D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim‡§ Minesh Patel§ Hasan Hassan§ Lois Orosa§ Onur Mutlu‡§
‡Carnegie Mellon University §ETH Zürich

SAFARI
Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory
Another Example: In-Memory Graph Processing

- Large graphs are everywhere (circa 2015)
  - 36 Million Wikipedia Pages
  - 1.4 Billion Facebook Users
  - 300 Million Twitter Users
  - 30 Billion Instagram Photos

- Scalable large-scale graph processing is challenging
  - 32 Cores
  - 128... +42%
Key Bottlenecks in Graph Processing

```java
for (v: graph.vertices) {
    for (w: v.successors) {
        w.next_rank += weight * v.rank;
    }
}
```

1. Frequent random memory accesses
2. Little amount of computation
Opportunity: 3D-Stacked Logic+Memory

Other “True 3D” technologies under development
Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores

Host Processor
Memory-Mapped Accelerator Interface (Noncacheable, Physically Addressed)

Crossbar Network

In-Order Core

DRAM Controller
NI

LP
PF Buffer
MTP
Message Queue

SAFARI Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015.
Tesseract System for Graph Processing

Host Processor
Memory-Mapped Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

In-Order Core

DRAM Controller

Communications via Remote Function Calls

Message Queue
Tesseract System for Graph Processing

Host Processor
Memory-Mapped Accelerator Interface
Noncacheable, Physically Addressed)

Memory
Logic

Prefetching

LP
PF Buffer
MTP
Message Queue

Crossbar Network

DRAM Controller
NI
# Evaluated Systems

<table>
<thead>
<tr>
<th>DDR3-OoO</th>
<th>HMC-OoO</th>
<th>HMC-MC</th>
<th>Tesseract</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
</tr>
<tr>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
</tr>
<tr>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
</tr>
<tr>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
</tr>
<tr>
<td><img src="image5.png" alt="Data rates" /></td>
<td><img src="image6.png" alt="Data rates" /></td>
<td><img src="image7.png" alt="Data rates" /></td>
<td><img src="image8.png" alt="Data rates" /></td>
</tr>
<tr>
<td>102.4GB/s</td>
<td>640GB/s</td>
<td>640GB/s</td>
<td>8TB/s</td>
</tr>
</tbody>
</table>

*SAFARI* Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015.
Tesseract Graph Processing Performance

>13X Performance Improvement

On five graph processing algorithms

- DDR3-OoO
- HMC-OoO
- HMC-MC
- Tesseract
- Tesseract-LP
- Tesseract-LP-MTP

SAFARI Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015.
Tesseract Graph Processing Performance

Memory Bandwidth Consumption

- DDR3-OoO: 80GB/s
- HMC-OoO: 190GB/s
- HMC-MC: 243GB/s
- Tesseract: 1.3TB/s
- Tesseract-LP: 2.2TB/s
- Tesseract-LP-MTP: 2.9TB/s
Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015.
More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,

"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"

PIM on Mobile Devices


Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand\(^1\), Rachata Ausavarunngirun\(^1\), Aki Kuusela\(^3\), Saugata Ghose\(^1\), Eric Shiu\(^3\), Rahul Thakur\(^3\), Youngsok Kim\(^2\), Parthasarathy Ranganathan\(^3\), Daehyun Kim\(^4,3\), Onur Mutlu\(^5,1\)
Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, Onur Mutlu
Consumer Devices

Consumer devices are everywhere!

Energy consumption is a first-class concern in consumer devices
Four Important Workloads

Chrome
Google’s web browser

TensorFlow Mobile
Google’s machine learning framework

VP9
Google’s video codec

Video Playback

Video Capture
Google’s video codec
Energy Cost of Data Movement

1st key observation: **62.7%** of the total system energy is spent on data movement

**Potential solution:** move computation close to data

**Challenge:** limited area and energy budget
Using PIM to Reduce Data Movement

2nd key observation: a significant fraction of the data movement often comes from simple functions.

We can design lightweight logic to implement these simple functions in memory.

Offloading to PIM logic reduces energy and improves performance, on average, by 2.3X and 2.2X.
Workload Analysis

Chrome
Google’s web browser

TensorFlow Mobile
Google’s machine learning framework

VP9
Video Playback
Google’s video codec

VP9
Video Capture
Google’s video codec
TensorFlow Mobile

57.3% of the inference energy is spent on data movement

54.4% of the data movement energy comes from packing/unpacking and quantization
More on PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungrunrunt, Eric Shiu, Rahul Thakur, Daeyhun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,

"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"


[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
[Lightning Talk Video (2 minutes)]
[Full Talk Video (21 minutes)]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand\textsuperscript{1} \quad Saugata Ghose\textsuperscript{1} \quad Youngsok Kim\textsuperscript{2}
Rachata Ausavarungrunrunt\textsuperscript{1} \quad Eric Shiu\textsuperscript{3} \quad Rahul Thakur\textsuperscript{3}
Aki Kuusela\textsuperscript{3} \quad Allan Knies\textsuperscript{3} \quad Parthasarathy Ranganathan\textsuperscript{3}
Daeyhun Kim\textsuperscript{4,3} \quad Onur Mutlu\textsuperscript{5,1}
Truly Distributed GPU Processing with PIM

```c
__global__
void applyScaleFactorsKernel( uint8_T * const out,
                           uint8_T const * const in,
                           const double *factor,
                           size_t const numRows, size_t const numCols )
{
  // Work out which pixel we are working on.
  const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
  const int colIdx = blockIdx.y;
  const int sliceIdx = threadIdx.z;

  // Check this thread isn't off the image
  if ( rowIdx >= numRows ) return;

  // Compute the index of my element
  size_t linearIdx = rowIdx + colIdx*numRows +
                     sliceIdx*numRows*numCols;
```
Accelerating GPU Execution with PIM (I)

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler,

"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"
[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]
Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik$^1$ Xulong Tang$^1$ Adwait Jog$^2$ Onur Kayiran$^3$
Asit K. Mishra$^4$ Mahmut T. Kandemir$^1$ Onur Mutlu$^5,6$ Chita R. Das$^1$

$^1$Pennsylvania State University  $^2$College of William and Mary
$^3$Advanced Micro Devices, Inc.  $^4$Intel Labs  $^5$ETH Zürich  $^6$Carnegie Mellon University
Accelerating Linked Data Structures

Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh† Samira Khan‡ Nandita Vijaykumar†
Kevin K. Chang† Amirali Boroumand† Saugata Ghose† Onur Mutlu§†
†Carnegie Mellon University ‡University of Virginia §ETH Zürich
Accelerating Dependent Cache Misses


[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]

Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi*, Khubaib†, Eiman Ebrahimi‡, Onur Mutlu§, Yale N. Patt*

*The University of Texas at Austin †Apple ‡NVIDIA §ETH Zürich & Carnegie Mellon University
Accelerating Runahead Execution

- Milad Hashemi, Onur Mutlu, and Yale N. Patt, "Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads"

Proceedings of the 49th International Symposium on Microarchitecture (MICRO), Taipei, Taiwan, October 2016.
[Slides (pptx) (pdf)] [Lightning Session Slides (pdf)] [Poster (pptx) (pdf)]
Accelerating Climate Modeling

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,

"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, September 2020.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (23 minutes)]
Nominated for the Stamatis Vassiliadis Memorial Award.

NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh\(^a,b,c\)  Dionysios Diamantopoulos\(^c\)  Christoph Hagleitner\(^c\)  Juan Gómez-Luna\(^b\)  Sander Stuijk\(^a\)  Onur Mutlu\(^b\)  Henk Corporaal\(^a\)  
\(^a\)Eindhoven University of Technology  \(^b\)ETH Zürich  \(^c\)IBM Research Europe, Zurich
Accelerating Approximate String Matching


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (ppts) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]
Accelerating Time Series Analysis

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu, "NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
  [Slides (pptx) (pdf)]
  [Talk Video (10 minutes)]

**NATSA: A Near-Data Processing Accelerator for Time Series Analysis**

Ivan Fernandez§ Ricardo Quislant§ Christina Giannoula† Mohammed Alser†
Juan Gómez-Luna‡ Eladio Gutiérrez§ Oscar Plata§ Onur Mutlu‡

§University of Malaga †National Technical University of Athens ‡ETH Zürich
DAMOV Methodology & Workloads

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Data movement between the CPU and main memory is a first-order obstacle against improving performance, scalability, and energy efficiency in modern systems. Computer systems employ a range of techniques to reduce overheads tied to data movement, spanning from traditional mechanisms (e.g., deep multi-level cache hierarchies, aggressive hardware prefetchers) to emerging techniques such as Near-Data Processing (NDP), where some computation is moved close to memory. Prior NDP works investigate the root causes of data movement bottlenecks using different profiling methodologies and tools. However, there is still a lack of understanding about the key metrics that can identify different data movement bottlenecks and their relation to traditional and emerging data movement mitigation mechanisms. Our goal is to methodically identify potential sources of data movement over a broad set of applications and to comprehensively compare traditional compute-centric data movement mitigation techniques (e.g., caching and prefetching) to more memory-centric techniques (e.g., NDP), thereby developing a rigorous understanding of the best techniques to mitigate each source of data movement.

With this goal in mind, we perform the first large-scale characterization of a wide variety of applications, across a wide range of application domains, to identify fundamental program properties that lead to data movement to/from main memory. We develop the first systematic methodology to classify applications based on the sources contributing to data movement bottlenecks. From our large-scale characterization of 77K functions across 345 applications, we select 144 functions to form the first open-source benchmark suite (DAMOV) for main memory data movement studies. We select a diverse range of functions that (1) represent different types of data movement bottlenecks, and (2) come from a wide range of application domains. Using NDP as a case study, we identify new insights about the different data movement bottlenecks and use these insights to determine the most suitable data movement mitigation mechanism for a particular application. We open-source DAMOV and the complete source code for our new characterization methodology at https://github.com/CMU-SAFARI/DAMOV.
We Need to Revisit the Entire Stack

We can get there step by step
A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d} 

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut’s University of Technology North Bangkok


SAFARI

A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungrun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut's University of Technology North Bangkok

Abstract

Modern computing systems are overwhelmingly designed to move data to computation. This design choice goes directly against at least three key trends in computing that cause performance, scalability and energy bottlenecks: (1) data access is a key bottleneck as many important applications are increasingly data-intensive, and memory bandwidth and energy do not scale well, (2) energy consumption is a key limiter in most computing platforms, especially server and mobile systems, (3) data movement, especially off-chip to on-chip, is very expensive in terms of bandwidth, energy and latency, much more so than computation. These trends are especially severely-felt in the data-intensive server and energy-constrained mobile systems of today.

At the same time, conventional memory technology is facing many technology scaling challenges in terms of reliability, energy, and performance. As a result, memory system architects are open to organizing memory in different ways and making it more intelligent, at the expense of higher cost. The emergence of 3D-stacked memory plus logic, the adoption of error correcting codes inside the latest DRAM chips, proliferation of different main memory standards and chips, specialized for different purposes (e.g., graphics, low-power, high bandwidth, low latency), and the necessity of designing new solutions to serious reliability and security issues, such as the RowHammer phenomenon, are an evidence of this trend.

This chapter discusses recent research that aims to practically enable computation close to data, an approach we call processing-in-memory (PIM). PIM places computation mechanisms in or near where the data is stored (i.e., inside the memory chips, in the logic layer of 3D-stacked memory, or in the memory controller), so that data movement between the computation units and memory is reduced or eliminated. While the general idea of PIM is not new, we discuss motivating trends in applications as well as memory circuits/technology that greatly exacerbate the need for enabling it in modern computing systems. We examine at least two promising new approaches to designing PIM systems to accelerate important data-intensive applications: (1) processing using memory by exploiting analog operational properties of DRAM chips to perform massively-parallel operations in memory, with low-cost changes, (2) processing near memory by exploiting 3D-stacked memory technology design to provide high memory bandwidth and low memory latency to in-memory logic. In both approaches, we describe and tackle relevant cross-layer research, design, and adoption challenges in devices, architecture, systems, and programming models. Our focus is on the development of in-memory processing designs that can be adopted in real computing platforms at low cost. We conclude by discussing work on solving key challenges to the practical adoption of PIM.

Keywords: memory systems, data movement, main memory, processing-in-memory, near-data processing, computation-in-memory, processing using memory, processing near memory, 3D-stacked memory, non-volatile memory, energy efficiency, high-performance computing, computer architecture, computing paradigm, emerging technologies, memory scaling, technology scaling, dependable systems, robust systems, hardware security, system security, latency, low-latency computing
1. Introduction

Main memory, built using the Dynamic Random Access Memory (DRAM) technology, is a major component in nearly all computing systems, including servers, cloud platforms, mobile/embedded devices, and sensor systems. Across all of these systems, the data working set sizes of modern applications are rapidly growing, while the need for fast analysis of such data is increasing. Thus, main memory is becoming an increasingly significant bottleneck across a wide variety of computing systems and applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Allowing the main memory bottleneck requires the memory capacity, energy, cost, and performance to all scale in an efficient manner across technology generations. Unfortunately, it has become increasingly difficult in recent years, especially the past decade, to scale all of these dimensions [1, 2, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49].

A major reason for the main memory bottleneck is the high energy and latency cost associated with data movement. In modern computers, to perform any operation on data that resides in main memory, the processor must retrieve the data from main memory. This requires the memory controller to issue commands to a DRAM module across a relatively slow and power-hungry off-chip bus (known as the memory channel). The DRAM module sends the requested data across the memory channel, after which the data is placed in the caches and registers. The CPU can perform computation on the data once the data is in its registers. Data movement from the DRAM to the CPU incurs long latency and consumes a significant amount of energy [7, 50, 51, 52, 53, 54]. These costs are often exacerbated by the fact that much of the data brought into the caches is not reused by the CPU [52, 53, 55, 56], providing little benefit in return for the high latency and energy cost.

The cost of data movement is a fundamental issue with the processor-centric nature of contemporary computer systems. The CPU is considered to be the master in the system, and computation is performed only in the processor (and accelerators). In contrast, data storage and communication units, including the main memory, are treated as unintelligent workers that are incapable of computation. As a result of this processor-centric design paradigm, data moves a lot in the system between the computation units and communication/storage units so that computation can be done on it. With the increasingly data-centric nature of contemporary and emerging appli-
PIM Review and Open Problems (II)

A Workload and Programming Ease Driven Perspective of Processing-in-Memory
Saugata Ghose† Amirali Boroumand† Jeremie S. Kim†§ Juan Gómez-Luna§ Onur Mutlu§†
†Carnegie Mellon University §ETH Zürich

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,
"Processing-in-Memory: A Workload-Driven Perspective"
[Preliminary arXiv version]

SAFARI
UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.

- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth

UPMEM Memory Modules

- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz
Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
IZZAT EL HAJJI, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
CHRISTINA CIANNOULA, ETH Zürich, Switzerland and NTUA, Greece
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is manifested in a high cost of main memory access. Fundamentally addressing this data movement bottleneck requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as processing-in-memory (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer whose processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPNEM company has designed and manufactured the first publicly available real-world PIM architecture. The UPNEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (PUs), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPNEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PIM (processing-in-memory benchmarks), a benchmark suite of 16 workloads from different application domains (e.g., dense-space linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PIM benchmarks on the UPNEM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPNEM-based PIM systems with 180 and 2,560 CPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

Experimental Analysis of the UPMEM PIM Engine

Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
IZZAT EL HAJJ, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
CHRISTINA GIANNOUTA, ETH Zürich, Switzerland and NTUA, Greece
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this data movement bottleneck requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as processing-in-memory (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PrIM (Processing-In-Memory benchmarks), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Benchmark</th>
<th>Short name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense linear algebra</td>
<td>Vector Addition</td>
<td>VA</td>
</tr>
<tr>
<td></td>
<td>Matrix-Vector Multiply</td>
<td>GEMV</td>
</tr>
<tr>
<td>Sparse linear algebra</td>
<td>Sparse Matrix-Vector Multiply</td>
<td>SpMV</td>
</tr>
<tr>
<td>Databases</td>
<td>Select</td>
<td>SEL</td>
</tr>
<tr>
<td></td>
<td>Unique</td>
<td>UNI</td>
</tr>
<tr>
<td>Data analytics</td>
<td>Binary Search</td>
<td>BS</td>
</tr>
<tr>
<td></td>
<td>Time Series Analysis</td>
<td>TS</td>
</tr>
<tr>
<td>Graph processing</td>
<td>Breadth-First Search</td>
<td>BFS</td>
</tr>
<tr>
<td>Neural networks</td>
<td>Multilayer Perceptron</td>
<td>MLP</td>
</tr>
<tr>
<td>Bioinformatics</td>
<td>Needleman-Wunsch</td>
<td>NW</td>
</tr>
<tr>
<td>Image processing</td>
<td>Image histogram (short)</td>
<td>HST-S</td>
</tr>
<tr>
<td></td>
<td>Image histogram (large)</td>
<td>HST-L</td>
</tr>
<tr>
<td>Parallel primitives</td>
<td>Reduction</td>
<td>RED</td>
</tr>
<tr>
<td></td>
<td>Prefix sum (scan-scan-add)</td>
<td>SCAN-SSA</td>
</tr>
<tr>
<td></td>
<td>Prefix sum (reduce-scan-scan)</td>
<td>SCAN-RSS</td>
</tr>
<tr>
<td></td>
<td>Matrix transposition</td>
<td>TRNS</td>
</tr>
</tbody>
</table>
PrIM Benchmarks are Open Source

- All microbenchmarks, benchmarks, and scripts
- https://github.com/CMU-SAFARI/prim-benchmarks

PrIM (Processing-In-Memory Benchmarks)

PrIM is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publicly-available real-world processing-in-memory (PIM) architecture, the UPMEM PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

PrIM provides a common set of workloads to evaluate the UPMEM PIM architecture with and can be useful for programming, architecture and system researchers alike to improve multiple aspects of future PIM hardware and software. The workloads have different characteristics, exhibiting heterogeneity in their memory access patterns, operations and data types, and communication patterns. This repository also contains baseline CPU and GPU implementations of PrIM benchmarks for comparison purposes.

PrIM also includes a set of microbenchmarks can be used to assess various architecture limits such as compute throughput and memory bandwidth.
Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

Juan Gómez-Luna\textsuperscript{1}  Izzat El Hajj\textsuperscript{2}  Ivan Fernandez\textsuperscript{1,3}  Christina Giannoula\textsuperscript{1,4}  Geraldo F. Oliveira\textsuperscript{1}  Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich  \textsuperscript{2}American University of Beirut  \textsuperscript{3}University of Malaga  \textsuperscript{4}National Technical University of Athens

https://github.com/CMU-SAFARI/prim-benchmarks
Understanding a Modern PIM Architecture

Juan Gómez Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, Onur Mutlu

https://github.com/CMU-SAFARI/prim-benchmarks

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture
2,579 views • Streamed live on Jul 12, 2021

https://www.youtube.com/watch?v=D8Hjy2iU9I4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
More on Analysis of the UPMEM PIM Engine

Inter-DPU Communication

- There is no direct communication channel between DPUs

Inter-DPU communication takes places via the host CPU using CPU-DPU and DPU-CPU transfers

Example communication patterns:
- Merging of partial results to obtain the final result
- Only DPU-CPU transfers
- Redistribution of intermediate results for further computation
- DPU-CPU transfers and CPU-DPU transfers

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

Talk Title: Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization
Dr. Juan Gómez-Luna, SAFARI Research Group, D-ITET, ETH Zurich

https://www.youtube.com/watch?v=D8Hjy2iU9j4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
More on Analysis of the UPMEM PIM Engine

Data Movement in Computing Systems

- Data movement dominates performance and is a major system energy bottleneck
- Total system energy: data movement accounts for
  - 62% in consumer applications
  - 40% in scientific applications
  - 35% in mobile applications

Understanding a Modern Processing-in-Memory Arch: Benchmarking & Experimental Characterization; 21m

https://www.youtube.com/watch?v=Pp9jSU2b9oM&list=PL5Q2soXY2Zi8_VVChACnON4sfh2bJ5IrD&index=159
FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu,

"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh○ Mohammed Alser○ Damla Senol Cali×
Dionysios Diamantopoulos▽ Juan Gómez-Luna○
Henk Corporaal* Onur Mutlu○×

○ETH Zürich  ×Carnegie Mellon University
*Eindhoven University of Technology  ▽IBM Research Europe

SAFARI
DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland  
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland  
LOIS OROSA, ETH Zürich, Switzerland  
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA  
NANDITA VIJAYKUMAR, University of Toronto, Canada  
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland  
MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland  
ONUR MUTLU, ETH Zürich, Switzerland

Data movement between the CPU and main memory is a first-order bottleneck against improving performance, scalability, and energy efficiency in modern systems. Computer systems employ a range of techniques to reduce overheads tied to data movement, spanning from traditional mechanisms (e.g., deep multi-level cache hierarchies, aggressive hardware prefetchers) to emerging techniques such as Near-Data Processing (NDP), where some computation is moved close to memory. Prior NDP works investigate the root causes of data movement bottlenecks using different profiling methodologies and tools. However, there is still a lack of understanding about the key metrics that can identify different data movement bottlenecks and their relation to traditional and emerging data movement mitigation mechanisms. Our goal is to methodically identify potential sources of data movement over a broad set of applications and to comprehensively compare traditional compute-centric data movement mitigation techniques (e.g., caching and prefetching) to more memory-centric techniques (e.g., NDP), thereby developing a rigorous understanding of the best techniques to mitigate each source of data movement.

With this goal in mind, we perform the first large-scale characterization of a wide variety of applications, across a wide range of application domains, to identify fundamental program properties that lead to data movement to/from main memory. We develop the first systematic methodology to classify applications based on the sources contributing to data movement bottlenecks. From our large-scale characterization of 77K functions across 345 applications, we select 144 functions to form the first open-source benchmark suite (DAMOV) for main memory data movement studies. We select a diverse range of functions that (1) represent different types of data movement bottlenecks, and (2) come from a wide range of application domains. Using NDP as a case study, we identify new insights about the different data movement bottlenecks and use these insights to determine the most suitable data movement mitigation mechanism for a particular application. We open-source DAMOV and the complete source code for our new characterization methodology at https://github.com/CMU-SAFARI/DAMOV.

Step 1: Application Profiling

• We analyze 345 applications from distinct domains:

- Graph Processing
- Deep Neural Networks
- Physics
- High-Performance Computing
- Genomics
- Machine Learning
- Databases
- Data Reorganization
- Image Processing
- Map-Reduce
- Benchmarking
- Linear Algebra

...
DAMOV is Open Source

• We open-source our benchmark suite and our toolchain

DAMOV-SIM

DAMOV Benchmarks

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including BWA, Chai, Darknet, GASE, Hardware Effects, Hashjoin, HPCC, HPCG, Ligra, PARSEC, Parboil, PolyBench, Phoenix, Rodinia, SPLASH-2, STREAM.
DAMOV is Open Source

- We open-source our benchmark suite and our toolchain

Get DAMOV at: https://github.com/CMU-SAFARI/DAMOV

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including BWA, Chai, Darknet, GASE, Hardware Effects, Hashjoin, HPCC, HPCG, Ligra, PARSEC, Parboil, PolyBench, Phoenix, Rodinia, SPLASH-2, STREAM.
More on DAMOV Analysis Methodology & Workloads

Step 3: Memory Bottleneck Classification (2/2)

- **Goal:** identify the specific sources of data movement bottlenecks

**Scalability Analysis:**
- 1, 4, 16, 64, and 256 out-of-order/in-order host and NDP CPU cores
- 3D-stacked memory as main memory

SAFARI Live Seminar: DAMOV: A New Methodology & Benchmark Suite for Data Movement Bottlenecks

https://www.youtube.com/watch?v=GWideVyo0nM&list=PL5Q2soXY2ZJ_OTAYm--dYByNPL7JhwR9&index=3
More on DAMOV

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan Fernandez, Mohammad Sadrosadati, and Onur Mutlu, "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
  Preprint in arXiv, 8 May 2021.
- [arXiv preprint]
- [DAMOV Suite and Simulator Source Code]
- [SAFARI Live Seminar Video (2 hrs 40 mins)]
- [Short Talk Video (21 minutes)]

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland
Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry’s first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power – the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, “Our groundbreaking HBM-PIM is the industry’s first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications.”

Samsung Function-in-Memory DRAM (2021)

- FIMDRAM based on HBM2

[3D Chip Structure of HBM with FIMDRAM]

**Chip Specification**

- 128DQ / 8CH / 16 banks / BL4
- 32 PCU blocks (1 FIM block/2 banks)
- 1.2 TFLOPS (4H)
- FP16 ADD / Multiply (MUL) / Multiply-Accumulate (MAC) / Multiply-and-Add (MAD)

---

**ISSCC 2021 / SESSION 25 / DRAM / 25.4**

25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Soon Kwon¹, Suk Han Lee², Jea-Hoon Lee³, Sang-Hyuk Kwon⁴, Je Min Ryu⁵, Jong-Pil Son⁶, Seongil O⁷, Hak-Soo Yu⁸, Hassan Lee⁰, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim⁹, Jongyoo Choi¹⁰, Hyun-Sung Shin¹¹, Jin Kim¹², Beng-Seng Phua¹³, HyoYoung Min Kim¹⁴, Myeong Jun Song¹⁵, Ahn Choi¹⁶, Daeho Kim¹⁷, SooYoung Kim¹⁸, Eun-Bong Kim¹⁹, David Wang²⁰, Shinhong Kang²¹, Yuhwan Ro²², Seungwoo Seo²³, JoonHo Song²⁴, Jaryoung Youn²¹, Kyomin Sohn²², Nam Sung Kim²³

¹Samsung Electronics, Hwasung, Korea
²Samsung Electronics, San Jose, CA
³Samsung Electronics, Suwon, Korea
Chip Implementation

- Mixed design methodology to implement FIMDRAM
  - Full-custom + Digital RTL

---

ISSCC 2021 / SESSION 25 / DRAM / 25.4

A 7nm 5GB Function-in-Memory DRAM, Based on HBM2 with a 1.3TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Choon Kee, Suk Han Lee, Jinhoon Lee, Sang-Hyuk Kwon, Ja Min Ryu, Jong Pyo Soo, Seong C. Kim, Jeong Su Kim, Han Suk Lee, Seo Young Kim, Yeonjeong Cho, Jeonggon Choi, Hyun-Jin Kang, Jin Kim, SeokYoung Park, HyungMin Kim, Myeong Jun Song, Ahn Choi, DaeHo Kim, SoYoung Kim, EunBong Kim, Carol Wang, DinhKang Yang, Woonseok Seo, Jeohoon Song, Jeongyeon You, Kyomin Sohn, Nam Soon Kim

*Samsung Electronics, Hweepong, Korea
*Samsung Electronics, San Jose, CA
*Samsung Electronics, Suwon, Korea
Detailed Lectures on PIM (I)

- Computer Architecture, Fall 2020, Lecture 6
  - Computation in Memory (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=oGcZAGwfEUE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=12

- Computer Architecture, Fall 2020, Lecture 7
  - Near-Data Processing (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=j2GIigqn1Qw&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=13

- Computer Architecture, Fall 2020, Lecture 11a
  - Memory Controllers (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=TeG773OgiMQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=20

- Computer Architecture, Fall 2020, Lecture 12d
  - Real Processing-in-DRAM with UPMEM (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=25

SAFARI

https://www.youtube.com/onurmutlulectures
Detailed Lectures on PIM (II)

- Computer Architecture, Fall 2020, Lecture 15
  - **Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=AIE1rD9G_YU&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=28](https://www.youtube.com/watch?v=AIE1rD9G_YU&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=28)

- Computer Architecture, Fall 2020, Lecture 16a
  - **Opportunities & Challenges of Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29](https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29)

- Computer Architecture, Fall 2020, Guest Lecture
  - **In-Memory Computing: Memory Devices & Applications** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=wNmqQHiEZNk&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=41](https://www.youtube.com/watch?v=wNmqQHiEZNk&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=41)

---

[SAFARI](https://www.youtube.com/onurmutlulectures)
A Tutorial on PIM

Onur Mutlu,
"Memory-Centric Computing Systems"
[Slides (pptx) (pdf)]
[Executive Summary Slides (pptx) (pdf)]
[Tutorial Video (1 hour 51 minutes)]
[Executive Summary Video (2 minutes)]
[Abstract and Bio]
[Related Keynote Paper from VLSI-DAT 2020]
[Related Review Paper on Processing in Memory]

https://www.youtube.com/watch?v=H3sEaINPBOE

https://www.youtube.com/onurmutlulectures
Memory-Centric Computing Systems

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
12 December 2020
IEDM Tutorial

https://www.youtube.com/watch?v=H3sEaINPBOE
https://www.youtube.com/onurmutlulectures

https://www.youtube.com/onurmutlulectures
Challenge and Opportunity for Future Computing Architectures with Minimal Data Movement
Challenge and Opportunity for Future

Fundamentally Energy-Efficient (Data-Centric) Computing Architectures
Challenge and Opportunity for Future

Fundamentally High-Performance (Data-Centric) Computing Architectures
Four Key Issues in Future Platforms

- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
  - **Memory-centric** (Data-centric) Architectures
- Fundamentally **Low-Latency and Predictable** Architectures
- Architectures for **AI/ML, Genomics, Medicine, Health**
Concluding Remarks
A Quote from A Famous Architect

- “architecture [...] based upon principle, and not upon precedent”
Precedent-Based Design

- “architecture [...] based upon principle, and not upon precedent”
Principled Design

- “architecture [...] based upon principle, and not upon precedent”
Another Example: Precedent-Based Design

Source: http://cookiemagik.deviantart.com/art/Train-station-207266944
Principled Design
Another Principled Design
Another Principled Design
Principle Applied to Another Structure

Overarching Principles for Computing?

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com
We Need to Exploit Good Principles

- Data-centric design
- All components intelligent
- Good cross-layer communication, expressive interfaces
- Better-than-worst-case design
- Heterogeneity
- Flexibility, adaptability

Open minds
Concluding Remarks

- It is time to design principled computing architectures to achieve the highest security, performance, and efficiency

- Discover design principles for fundamentally secure and reliable computer architectures

- Design complete systems to be balanced and energy-efficient, i.e., data-centric (or memory-centric) and low-latency

- Enable new platforms for genomics, medicine, health, AI/ML

- This can
  - Lead to orders-of-magnitude improvements
  - Enable new applications & computing platforms
  - Enable better understanding of nature

- ...
The Future is Very Bright

- Regardless of challenges
  - in underlying technology and overlying problems/requirements
We Need to Think and Act Across the Stack

We can get there step by step
A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut’s University of Technology North Bangkok

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "A Modern Primer on Processing in Memory"

A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut’s University of Technology North Bangkok

Abstract

Modern computing systems are overwhelmingly designed to move data to computation. This design choice goes directly against at least three key trends in computing that cause performance, scalability and energy bottlenecks: (1) data access is a key bottleneck as many important applications are increasingly data-intensive, and memory bandwidth and energy do not scale well, (2) energy consumption is a key limiter in almost all computing platforms, especially server and mobile systems, (3) data movement, especially off-chip to on-chip, is very expensive in terms of bandwidth, energy and latency, much more so than computation. These trends are especially severely-felt in the data-intensive server and energy-constrained mobile systems of today.

At the same time, conventional memory technology is facing many technology scaling challenges in terms of reliability, energy, and performance. As a result, memory system architects are open to organizing memory in different ways and making it more intelligent, at the expense of higher cost. The emergence of 3D-stacked memory plus logic, the adoption of error correcting codes inside the latest DRAM chips, proliferation of different main memory standards and chips, specialized for different purposes (e.g., graphics, low-power, high bandwidth, low latency), and the necessity of designing new solutions to serious reliability and security issues, such as the RowHammer phenomenon, are an evidence of this trend.

This chapter discusses recent research that aims to practically enable computation close to data, an approach we call \textit{processing-in-memory} (PIM). PIM places computation mechanisms in or near where the data is stored (i.e., inside the memory chips, in the logic layer of 3D-stacked memory, or in the memory controllers), so that data movement between the computation units and memory is reduced or eliminated. While the general idea of PIM is not new, we discuss motivating trends in applications as well as memory circuits/technology that greatly exacerbate the need for enabling it in modern computing systems. We examine at least two promising new approaches to designing PIM systems to accelerate important data-intensive applications: (1) \textit{processing using memory} by exploiting analog operational properties of DRAM chips to perform massively-parallel operations in memory, with low-cost changes, (2) \textit{processing near memory} by exploiting 3D-stacked memory technology design to provide high memory bandwidth and low memory latency to in-memory logic. In both approaches, we describe and tackle relevant cross-layer research, design, and adoption challenges in devices, architecture, systems, and programming models. Our focus is on the development of in-memory processing designs that can be adopted in real computing platforms at low cost. We conclude by discussing work on solving key challenges to the practical adoption of PIM.

\textit{Keywords:} memory systems, data movement, main memory, processing-in-memory, near-data processing, computation-in-memory, processing using memory, processing near memory, 3D-stacked memory, non-volatile memory, energy efficiency, high-performance computing, computer architecture, computing paradigm, emerging technologies, memory scaling, technology scaling, dependable systems, robust systems, hardware security, system security, latency, low-latency computing
1. Introduction

Main memory, built using the Dynamic Random Access Memory (DRAM) technology, is a major component in nearly all computing systems, including servers, cloud platforms, mobile/embedded devices, and sensor systems. Across all of these systems, the data working set sizes of modern applications are rapidly growing, while the need for fast access of such data is increasing. Thus, main memory is becoming an increasingly significant bottleneck across a wide variety of computing systems and applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Alleviating the main memory bottleneck requires the memory capacity, energy, cost, and performance to all scale in an efficient manner across technology generations. Unfortunately, it has become increasingly difficult in recent years, especially in the past decade, to scale all of these dimensions [1, 2, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49], and the main memory bottleneck has been worsening.

A major reason for the main memory bottleneck is the high energy and latency cost associated with data movement. In modern computers, to perform any operation on data that resides in main memory, the processor must retrieve the data from main memory. This requires the memory controller to issue commands to a DRAM module across a relatively slow and power-hungry off-chip bus (known as the memory channel). The DRAM module sends the requested data across the memory channel, after which the data is placed in the caches and registers. The CPU can perform computation on the data once the data is in its registers. Data movement from the DRAM to the CPU incurs long latency and consumes a significant amount of energy [7, 50, 51, 52, 53, 54]. These costs are often exacerbated by the fact that much of the data brought into the caches is not reused by the CPU [52, 53, 55, 56], providing little benefit in return for the high latency and energy cost.

The cost of data movement is a fundamental issue with the processor-centric nature of contemporary computer systems. The CPU is considered to be the master in the system, and computation is performed only in the processor (and accelerators). In contrast, data storage and communication units, including the main memory, are treated as unintelligent workers that are incapable of computation. As a result of this processor-centric design paradigm, data moves a lot in the system between the computation units and communication/storage units so that computation can be done on it. With the increasingly data-centric nature of contemporary and emerging appli-
A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose†  Amirali Boroumand†  Jeremie S. Kim†§  Juan Gómez-Luna§  Onur Mutlu§†

†Carnegie Mellon University  §ETH Zürich


A Tutorial on Memory-Centric Systems

- Onur Mutlu,
  "Memory-Centric Computing Systems"
  [Slides (pptx) (pdf)]
  [Executive Summary Slides (pptx) (pdf)]
  [Tutorial Video (1 hour 51 minutes)]
  [Executive Summary Video (2 minutes)]
  [Abstract and Bio]
  [Related Keynote Paper from VLSI-DAT 2020]
  [Related Review Paper on Processing in Memory]

https://www.youtube.com/watch?v=H3sEaINPBOE
Memory-Centric Computing Systems

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
12 December 2020
IEDM Tutorial

SAFARI
ETH Zürich
Carnegie Mellon
Funding Acknowledgments

- Alibaba, AMD, ASML, Google, Facebook, Hi-Silicon, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware
- NSF
- NIH
- GSRC
- SRC
- CyLab
Acknowledgments

- **My current and past students and postdocs**
  - Rachata Ausavarungnirun, Abhishek Bhowmick, Amirali Boroumand, Rui Cai, Yu Cai, Kevin Chang, Saugata Ghose, Kevin Hsieh, Tyler Huberty, Ben Jaiyen, Samira Khan, Jeremie Kim, Yoongu Kim, Yang Li, Jamie Liu, Lavanya Subramanian, Donghyuk Lee, Yixin Luo, Justin Meza, Gennady Pekhimenko, Vivek Seshadri, Lavanya Subramanian, Nandita Vijaykumar, HanBin Yoon, Jishen Zhao, ...

- **My collaborators**
  - Can Alkan, Chita Das, Phil Gibbons, Sriram Govindan, Norm Jouppi, Mahmut Kandemir, Mike Kozuch, Konrad Lai, Ken Mai, Todd Mowry, Yale Patt, Moinuddin Qureshi, Partha Ranganathan, Bikash Sharma, Kushagra Vaid, Chris Wilkerson, ...
Acknowledgments

SAFARI
SAFARI Research Group
safari.ethz.ch

Think BIG, Aim HIGH!

https://safari.ethz.ch
Onur Mutlu’s SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

https://safari.ethz.ch/safari-newsletter-january-2021/

Think BIG, Aim HIGH!

https://safari.ethz.ch
Dear SAFARI friends,

2019 and the first three months of 2020 have been very positive and eventful times for SAFARI.
Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 here). 2020 has
Future Computing Platforms
Challenges and Opportunities

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
10 November 2021
EdukCircle Intl. Conv. on Engineering & Computer Technology
Four Key Issues in Future Platforms

- Fundamentally **Secure/Reliable/Safe** Architectures

- Fundamentally **Energy-Efficient** Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally **Low-Latency and Predictable** Architectures

- Architectures for **AI/ML, Genomics, Medicine, Health**
Low Latency Communication is Critical
Maslow’s Hierarchy of Needs, A Third Time


Source: https://www.simplypsychology.org/maslow.html
Challenge and Opportunity for Future

Fundamentally Low-Latency Computing Architectures
Main Memory Latency Lags Behind

Memory latency remains almost constant
The Memory Latency Problem

- High memory latency is a significant **limiter of system performance and energy-efficiency**

- It is becoming increasingly so with **higher memory contention** in multi-core and heterogeneous architectures
  - Exacerbating the bandwidth need
  - Exacerbating the QoS problem

- It increases **processor design complexity** due to the mechanisms incorporated to tolerate memory latency
Retrospective: Conventional Latency Tolerance Techniques

- Caching [initially by Wilkes, 1965]
  - Widely used, simple, effective, but inefficient, passive
  - Not all applications/phases exhibit temporal or spatial locality

- Prefetching [initially in IBM 360/91, 1967]
  - Works well for regular memory access patterns
  - Prefetching irregular access patterns is difficult, inaccurate, and hardware-intensive

- Multithreading [initially in CDC 6600, 1964]
  - Works well if there are multiple threads
  - Improving single thread performance using multithreading hardware is an ongoing research effort

- Out-of-order execution [initially by Tomasulo, 1967]
  - Tolerates cache misses that cannot be prefetched
  - Requires extensive hardware resources for tolerating long latencies

None of These
Fundamentally Reduce Memory Latency
Truly Reducing Memory Latency
Why the Long Memory Latency?

- **Reason 1: Design of DRAM Micro-architecture**
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2: “One size fits all” approach to latency specification**
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips
  - Same latency parameters for all parts of a DRAM chip
  - Same latency parameters for all supply voltage levels
  - Same latency parameters for all application data
  - ...
Tackling the Fixed Latency Mindset

- Reliable operation latency is actually very heterogeneous
  - Across temperatures, chips, parts of a chip, voltage levels, ...

- Idea: **Dynamically find out and use the lowest latency one can reliably access a memory location with**
  - Adaptive-Latency DRAM [HPCA 2015]
  - Flexible-Latency DRAM [SIGMETRICS 2016]
  - Design-Induced Variation-Aware DRAM [SIGMETRICS 2017]
  - Voltron [SIGMETRICS 2017]
  - DRAM Latency PUF [HPCA 2018]
  - DRAM Latency True Random Number Generator [HPCA 2019]
  - ...

- We would like to find sources of latency heterogeneity and exploit them to minimize latency
Latency Variation in Memory Chips

Heterogeneous manufacturing & operating conditions → latency variation in timing parameters
Why is Latency High?

- **DRAM latency:** Delay as specified in DRAM standards
  - Doesn’t reflect true DRAM device latency
- Imperfect manufacturing process $\rightarrow$ latency variation
- **High standard latency** chosen to increase yield

![Diagram showing DRAM A, B, and C with high standard latency and manufacturing variation](image)
What Causes the Long Memory Latency?

- **Conservative timing margins!**

- **DRAM timing parameters are set to cover the worst case**

- **Worst-case temperatures**
  - 85 degrees vs. common-case
  - to enable a wide range of operating conditions

- **Worst-case devices**
  - DRAM cell with smallest charge across any acceptable device
  - to tolerate process variation at acceptable yield

- **This leads to large timing margins for the common case**
Understanding and Exploiting Variation in DRAM Latency
Adaptive-Latency DRAM

• **Key idea**
  – Optimize DRAM timing parameters online

• **Two components**
  – DRAM manufacturer provides multiple sets of reliable DRAM timing parameters at different temperatures for each DIMM
  – System monitors DRAM temperature & uses appropriate DRAM timing parameters

Latency Reduction Summary of 115 DIMMs

- **Latency reduction for read & write (55°C)**
  - Read Latency: **32.7%**
  - Write Latency: **55.1%**

- **Latency reduction for each timing parameter (55°C)**
  - Sensing: **17.3%**
  - Restore: **37.3%** (read), **54.8%** (write)
  - Precharge: **35.2%**
AL-DRAM: Real System Evaluation

- **System**
  - **CPU**: AMD 4386 (8 Cores, 3.1GHz, 8MB LLC)
  - **DRAM**: 4GB DDR3-1600 (800MHz Clock)
  - **OS**: Linux
  - **Storage**: 128GB SSD

---

**Table: DDR3 DRAM Timing**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved.</td>
</tr>
<tr>
<td>29:24</td>
<td><strong>Tras</strong>: row active strobe. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank.</td>
</tr>
<tr>
<td>23:21</td>
<td>Reserved.</td>
</tr>
<tr>
<td>20:16</td>
<td><strong>Trp</strong>: row precharge time. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from a precharge command to an activate command or auto refresh command, both to the same bank.</td>
</tr>
</tbody>
</table>
AL-DRAM: Single-Core Evaluation

AL-DRAM improves single-core performance on a real system
AL-DRAM: Multi-Core Evaluation

AL-DRAM provides higher performance on multi-programmed & multi-threaded workloads
Reducing Latency Also Reduces Energy

- AL-DRAM reduces DRAM power consumption by 5.8%
- Major reason: reduction in row activation time
More on Adaptive-Latency DRAM

- Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case"

[Slides (pptx) (pdf)] [Full data sets]

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case

Donghyuk Lee    Yoongu Kim    Gennady Pekhimenko
Samira Khan    Vivek Seshadri    Kevin Chang    Onur Mutlu
Carnegie Mellon University
CLR-DRAM: Capacity-Latency Reconfigurability

- Haocong Luo, Taha Shahroodi, Hasan Hassan, Minesh Patel, A. Giray Yaglikci, Lois Orosa, Jisung Park, and Onur Mutlu,

"CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off"


[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off

Haocong Luo§† Taha Shahroodi§ Hasan Hassan§ Minesh Patel§
A. Giray Yaglikci§ Lois Orosa§ Jisung Park§ Onur Mutlu§

§ETH Zürich †ShanghaiTech University
Analysis of Latency Variation in DRAM Chips

Kevin Chang, Abhijith Kashyap, Hasan Hassan, Samira Khan, Kevin Hsieh, Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Tianshi Li, and Onur Mutlu,

"Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"

[Slides (pptx) (pdf)]
[Source Code]

Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization

Kevin K. Chang¹ Abhijith Kashyap¹ Hasan Hassan¹,²
Saugata Ghose¹ Kevin Hsieh¹ Donghyuk Lee¹ Tianshi Li¹,³
Gennady Pekhimenko¹ Samira Khan⁴ Onur Mutlu⁵,¹

¹Carnegie Mellon University  ²TOBB ETÜ  ³Peking University  ⁴University of Virginia  ⁵ETH Zürich

SAFARI
Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee, NVIDIA and Carnegie Mellon University
Samira Khan, University of Virginia
Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Carnegie Mellon University
Gennady Pekhimenko, Vivek Seshadri, Microsoft Research
Onur Mutlu, ETH Zürich and Carnegie Mellon University
Solar-DRAM: Putting It Together

Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines"
[Slides (pptx) (pdf)]
[Talk Video (16 minutes)]

Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines

Jeremie S. Kim‡§ Minesh Patel§ Hasan Hassan§ Onur Mutlu§‡
‡Carnegie Mellon University §ETH Zürich
Challenge and Opportunity for Future

Fundamentally Low-Latency Computing Architectures
D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim  Minesh Patel
Hasan Hassan  Lois Orosa  Onur Mutlu

SAFARI

ETH Zürich  Carnegie Mellon
DRAM Latency Characterization of 282 LPDDR4 DRAM Devices

- Latency failures come from accessing DRAM with reduced timing parameters.

- Key Observations:
  1. A cell’s latency failure probability is determined by random process variation
  2. Some cells fail randomly
D-RaNGe Key Idea

High % chance to fail with reduced $t_{RCD}$

Low % chance to fail with reduced $t_{RCD}$

Fails randomly with reduced $t_{RCD}$
D-RaNGe Key Idea

We refer to cells that fail randomly when accessed with a reduced $t_{RCD}$ as RNG cells.
Our D-RaNGe Evaluation

• We generate random values by repeatedly accessing RNG cells and aggregating the data read.

• The random data satisfies the NIST statistical test suite for randomness.

• The D-RaNGe generates random numbers:
  - Throughput: 717.4 Mb/s
  - Latency: 64 bits in <1us
  - Power: 4.4 nJ/bit
DRAM Latency True Random Number Generator

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"


[Slides (pptx) (pdf)]
[Full Talk Video (21 minutes)]
[Full Talk Lecture Video (27 minutes)]

Top Picks Honorable Mention by IEEE Micro.

---

D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim‡$ Minesh Patel$ Hasan Hassan$ Lois Orosa$ Onur Mutlu$‡

‡Carnegie Mellon University $ETH Zürich
The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim†§, Minesh Patel§, Hasan Hassan§, Onur Mutlu§†

†Carnegie Mellon University §ETH Zürich
Lectures on Low-Latency Memory

- Computer Architecture, Fall 2020, Lecture 10
  - **Low-Latency Memory** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=vQd1YgOH1Mw&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=19](https://www.youtube.com/watch?v=vQd1YgOH1Mw&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=19)

- Computer Architecture, Fall 2020, Lecture 12b
  - **Capacity-Latency Reconfigurable DRAM** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=DUtPFW3jxq4&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=23](https://www.youtube.com/watch?v=DUtPFW3jxq4&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=23)

- Computer Architecture, Fall 2019, Lecture 11a
  - **DRAM Latency PUF** (ETH Zürich, Fall 2019)
  - [https://www.youtube.com/watch?v=7gqn8TZpjxE&list=PL5Q2soXY2Zi-DyoI3HbqcdtUm9YWRR_z-&index=15](https://www.youtube.com/watch?v=7gqn8TZpjxE&list=PL5Q2soXY2Zi-DyoI3HbqcdtUm9YWRR_z-&index=15)

- Computer Architecture, Fall 2019, Lecture 11b
  - **DRAM True Random Number Generator** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=Y3hPv1I5f8Y&list=PL5Q2soXY2Zi-DyoI3HbqcdtUm9YWRR_z-&index=16](https://www.youtube.com/watch?v=Y3hPv1I5f8Y&list=PL5Q2soXY2Zi-DyoI3HbqcdtUm9YWRR_z-&index=16)

SAFARI [https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
A Tutorial on Low-Latency Memory

Computer Architecture
Lecture 10: Low-Latency Memory

Prof. Onur Mutlu
ETH Zürich
Fall 2020
22 October 2020

https://www.youtube.com/onurmutlulectures
Fundamentally Low-Latency Computing Architectures
Four Key Issues in Future Platforms

- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
  - **Memory-centric** (Data-centric) Architectures
- Fundamentally **Low-Latency and Predictable** Architectures
- Architectures for **AI/ML, Genomics, Medicine, Health**
Intel Optane Persistent Memory (2019)

- Non-volatile main memory
- Based on 3D-XPoint Technology
PCM as Main Memory: Idea in 2009


Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee† Engin Ipek† Onur Mutlu‡ Doug Burger†

†Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

‡Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu
PCM as Main Memory: Idea in 2009

Cerebras’s Wafer Scale Engine (2019)

- The largest ML accelerator chip
- 400,000 cores

Cerebras WSE
1.2 Trillion transistors
46,225 mm²

Largest GPU
21.1 Billion transistors
815 mm²

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning
https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/
Cerebras’s Wafer Scale Engine-2 (2021)

- The largest ML accelerator chip (2021)
- 850,000 cores

Cerebras WSE-2
2.6 Trillion transistors
46,225 mm²

Largest GPU
54.2 Billion transistors
826 mm²

NVIDIA Ampere GA100

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning
https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/
UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
  - Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.

- **Replaces standard DIMMs**
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth

[Links to related content]
Experimental Analysis of the UPMEM PIM Engine

Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
IZZAT EL HAJJ, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this data movement bottleneck requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as processing-in-memory (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPU), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PrIM (Processing-In-Memory benchmarks), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry’s first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power – the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, “Our groundbreaking HBM-PIM is the industry’s first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications.”

Samsung Function-in-Memory DRAM (2021)

- FIMDRAM based on HBM2

Chip Specification

- 128DQ / 8CH / 16 banks / BL4
- 32 PCU blocks (1 FIM block/2 banks)
- 1.2 TFLOPS (4H)
- FP16 ADD / Multiply (MUL) / Multiply-Accumulate (MAC) / Multiply-and-Add (MAD)

3D Chip Structure of HBM with FIMDRAM
Programmable Computing Unit

- Configuration of PCU block
  - Interface unit to control data flow
  - Execution unit to perform operations
  - Register group
    - 32 entries of CRF for instruction memory
    - 16 GRF for weight and accumulation
    - 16 SRF to store constants for MAC operations

[Block diagram of PCU in FIMDRAM]
### Available instruction list for FIM operation

<table>
<thead>
<tr>
<th>Type</th>
<th>CMD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating Point</td>
<td>ADD</td>
<td>FP16 addition</td>
</tr>
<tr>
<td></td>
<td>MUL</td>
<td>FP16 multiplication</td>
</tr>
<tr>
<td></td>
<td>MAC</td>
<td>FP16 multiply-accumulate</td>
</tr>
<tr>
<td></td>
<td>MAD</td>
<td>FP16 multiply and add</td>
</tr>
<tr>
<td>Data Path</td>
<td>MOVE</td>
<td>Load or store data</td>
</tr>
<tr>
<td></td>
<td>FILL</td>
<td>Copy data from bank to GRFs</td>
</tr>
<tr>
<td>Control Path</td>
<td>NOP</td>
<td>Do nothing</td>
</tr>
<tr>
<td></td>
<td>JUMP</td>
<td>Jump instruction</td>
</tr>
<tr>
<td></td>
<td>EXIT</td>
<td>Exit instruction</td>
</tr>
</tbody>
</table>
**Chip Implementation**

- Mixed design methodology to implement FIMDRAM
  - Full-custom + Digital RTL
More on Processing in Memory (I)

More on Processing in Memory (II)


---

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
More on Processing in Memory (III)

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"


- [2-page Extended Abstract]
- [Short Talk Slides (pptx) (pdf)]
- [Talk Slides (pptx) (pdf)]
- [Short Talk Video (5 mins)]
- [Full Talk Video (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar\textsuperscript{1,2}    *Geraldo F. Oliveira\textsuperscript{1}
Nika Mansouri Ghiasi\textsuperscript{1}    Minesh Patel\textsuperscript{1}
      Juan Gómez-Luna\textsuperscript{1}    Mohammed Alser\textsuperscript{1}
      Onur Mutlu\textsuperscript{1}    João Dinis Ferreira\textsuperscript{1}

\textsuperscript{1}ETH Zürich    \textsuperscript{2}Simon Fraser University    \textsuperscript{3}University of Illinois at Urbana–Champaign
More on Processing in Memory (IV)

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,
  "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
  [Slides (pdf)] [Lightning Session Slides (pdf)]

A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn  Sungpack Hong§  Sungjoo Yoo  Onur Mutlu†  Kiyoung Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University  §Oracle Labs  †Carnegie Mellon University
More on Processing in Memory (V)

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"

Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi, "PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"
TESLA Full Self-Driving Computer (2019)

- ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
- Two redundant chips for better safety.

https://youtu.be/Ucp0TTmvqOE?t=4236
Google TPU Generation I (~2016)

Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Google TPU Generation II (2017)

4 TPU chips vs 1 chip in TPU1

High Bandwidth Memory vs DDR3

Floating point operations vs FP16

45 TFLOPS per chip vs 23 TOPS

Designed for training and inference vs only inference

Google TPU Generation III

32GB HBM per chip vs 16GB HBM in TPU2
4 Matrix Units per chip vs 2 Matrix Units in TPU2
90 TFLOPS per chip vs 45 TFLOPS in TPU2

https://cloud.google.com/tpu/docs/system-architecture
An Example Modern Systolic Array: TPU (II)

As reading a large SRAM uses much more power than arithmetic, the matrix unit uses systolic execution to save energy by reducing reads and writes of the Unified Buffer [Kun80][Ram91][Ovt15b]. Figure 4 shows that data flows in from the left, and the weights are loaded from the top. A given 256-element multiply-accumulate operation moves through the matrix as a diagonal wavefront. The weights are preloaded, and take effect with the advancing wave alongside the first data of a new block. Control and data are pipelined to give the illusion that the 256 inputs are read at once, and that they instantly update one location of each of 256 accumulators. From a correctness perspective, software is unaware of the systolic nature of the matrix unit, but for performance, it does worry about the latency of the unit.

Figure 1. TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB.
Many (Other) AI/ML Chips

- Alibaba
- Amazon
- Facebook
- Google
- Huawei
- Intel
- Microsoft
- NVIDIA
- Tesla
- Many Others and Many Startups...

- Many More to Come...
Many (Other) AI/ML Chips

Tech Giants/System
- Google
- Microsoft
- Amazon
- Facebook
- Alibaba
- NVIDIA
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Qualcomm
- AMD
- Xilinx
- Intel
- Samsung
- NVIDIA
- Intel
- Quali
Lectures on Systolic Arrays & ML/AI Acceleration

- Digital Design and Computer Architecture, Spring 2021, Lecture 19
  - VLIW, Systolic Arrays, DAE (ETH Zürich, Spring 2021)
    - https://www.youtube.com/watch?v=UtLy4Yagdys&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LIN&index=21

- Computer Architecture, Fall 2020, Lecture 9b
  - EDEN: Efficient DNN Inference (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=HmB32OXMKMY&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=16

- Computer Architecture, Fall 2020, Lecture 9c
  - SMASH: Accelerating Sparse Matrix Operations (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=78aikMbzkGc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=17

https://www.youtube.com/onurmutlulectures
Lecture on Systolic Arrays & ML Acceleration

An Example Modern Systolic Array: TPU (II)

Figure 1. TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB.
Accelerating Genome Analysis
Our Dream (circa 2007)

- An embedded device that can perform comprehensive genome analysis in real time (within a minute)
  - Which of these DNAs does this DNA segment match with?
  - What is the likely genetic disposition of this patient to this drug?
  - What disease/condition might this particular DNA/RNA piece associated with?
  - . . .
What Is a Genome Made Of?

The chromosome is made up of genes

The genes consist of DNA

Chromosome - 23 pairs

Cell

Nucleus

The discovery of DNA's double-helical structure (Watson+, 1953)
DNA Under Electron Microscope

human chromosome #12 from HeLa’s cell
DNA Sequencing

- **Goal:**
  - Find the complete sequence of A, C, G, T’s in DNA.

- **Challenge:**
  - There is no machine that takes long DNA as an input, and gives the complete sequence as output.
  - All sequencing machines chop DNA into pieces and identify relatively small pieces (but not how they fit together).
Untangling Yarn Balls & DNA Sequencing
Genome Sequencers

... and more! All produce data with different properties.

Roche/454
AB SOLiD
Illumina HiSeq2000
Pacific Biosciences RS
Ion Torrent PGM
Ion Torrent Proton
Illumina MiSeq
Complete Genomics
Oxford Nanopore MinION
Illumina NovaSeq 6000
Oxford Nanopore GridION
1 Sequencing

2 Read Mapping

3 Variant Calling

4 Scientific Discovery

Genome Analysis

Reference Genome

Read Alignment

Billions of Short Reads

Short Read

Prescription

Variant Calling

Scientific Discovery
Genome Sequence Alignment: Example

Source: By Aaron E. Darling, István Miklós, Mark A. Ragan - Figure 1 from Darling AE, Miklós I, Ragan MA (2008). "Dynamics of Genome Rearrangement in Bacterial Populations". PLOS Genetics. DOI:10.1371/journal.pgen.1000128., CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=30550950
Bottlenecks in Mapping!!

1. Sequencing
2. Read Mapping

Illumina HiSeq4000
300 M bases/min

on average
2 M bases/min (0.6%)
Hash Table Based Read Mappers

- Guaranteed to find all mappings → sensitive
- Can tolerate up to \( e \) errors

http://mrfast.sourceforge.net/

Personalized copy number and segmental duplication maps using next-generation sequencing

Can Alkan\(^1,2\), Jeffrey M Kidd\(^1\), Tomas Marques-Bonet\(^1,3\), Gozde Aksay\(^1\), Francesca Antonacci\(^1\), Fereydoun Hormozdiari\(^4\), Jacob O Kitzman\(^1\), Carl Baker\(^1\), Maika Malig\(^1\), Onur Mutlu\(^5\), Cenk Sahinalp\(^4\), Richard A Gibbs\(^6\) & Evan E Eichler\(^1,2\)

Read Mapping Execution Time Breakdown

- Read Alignment (Edit-distance comp): 93%
- Candidate alignment locations (CAL): 4%
- SAM printing: 3%

SAFARI
Filter fast before you align

Minimize costly “approximate string comparisons”
Our First Filter: Pure Software Approach

- Download the source code and try for yourself
  - Download link to FastHASH

Xin et al. BMC Genomics 2013, 14(Suppl 1):S13
http://www.biomedcentral.com/1471-2164/14/S1/S13

Accelerating read mapping with FastHASH

Hongyi Xin¹, Donghyuk Lee¹, Farhad Hormozdiari², Samihan Yedkar¹, Onur Mutlu¹*, Can Alkan³*

From The Eleventh Asia Pacific Bioinformatics Conference (APBC 2013)
Vancouver, Canada. 21-24 January 2013
Shifted Hamming distance: a fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping

Hongyi Xin¹⋅*, John Greth², John Emmons², Gennady Pekhimenko¹, Carl Kingsford³, Can Alkan⁴⋅*, and Onur Mutlu²⋅*

GateKeeper: FPGA-Based Alignment Filtering

1st
FPGA-based Alignment Filter.

$x10^{12}$
mappings

Low Speed & High Accuracy
Medium Speed, Medium Accuracy
High Speed, Low Accuracy

$x10^3$
mappings

1 High throughput DNA sequencing (HTS) technologies
2 Read Pre-Alignment Filtering Fast & Low False Positive Rate
3 Read Alignment Slow & Zero False Positives

GateKeeper: FPGA-Based Alignment Filtering

Alignment Filter

1st
FPGA-based Alignment Filter.
GateKeeper: FPGA-Based Alignment Filtering

Mohammed Alser, Hasan Hassan, Hongyi Xin, Oguz Ergin, Onur Mutlu, and Can Alkan

"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"

*Bioinformatics*, [published online, May 31], 2017.

[Source Code]
[Online link at Bioinformatics Journal]

GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping

Mohammed Alser ✉, Hasan Hassan, Hongyi Xin, Oğuz Ergin, Onur Mutlu ✉, Can Alkan ✉

*Bioinformatics*, Volume 33, Issue 21, 1 November 2017, Pages 3355–3363,
https://doi.org/10.1093/bioinformatics/btx342

Published: 31 May 2017    Article history ▼
DNA Read Mapping & Filtering

- **Problem:** Heavily bottlenecked by Data Movement

- GateKeeper FPGA performance limited by DRAM bandwidth [Alser+, Bioinformatics 2017]

- Ditto for SHD on SIMD [Xin+, Bioinformatics 2015]

- **Solution:** Processing-in-memory can alleviate the bottleneck

- However, we need to design mapping & filtering algorithms to fit processing-in-memory
In-Memory DNA Sequence Analysis

- Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu,

"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"

*BMC Genomics*, 2018.

Proceedings of the *16th Asia Pacific Bioinformatics Conference (APBC)*, Yokohama, Japan, January 2018.

[Slides (pptx) (pdf)]
[Source Code]
[arxiv.org Version (pdf)]
[Talk Video at AACBB 2019]

GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim$^{1,6,*}$, Damla Senol Cali$^1$, Hongyi Xin$^2$, Donghyuk Lee$^3$, Saugata Ghose$^1$, Mohammed Alser$^4$, Hasan Hassan$^6$, Oguz Ergin$^5$, Can Alkan$^{4,*}$ and Onur Mutlu$^{6,1,*}$

*From The Sixteenth Asia Pacific Bioinformatics Conference 2018
Yokohama, Japan. 15-17 January 2018*
Mohammed Alser, Hasan Hassan, Akash Kumar, Onur Mutlu, and Can Alkan, "Shouji: A Fast and Efficient Pre-Alignment Filter for Sequence Alignment" *Bioinformatics*, [published online, March 28], 2019. [Source Code] [Online link at Bioinformatics Journal]

Sequence alignment

**Shouji: a fast and efficient pre-alignment filter for sequence alignment**

Mohammed Alser¹.².³.*, Hasan Hassan¹, Akash Kumar², Onur Mutlu¹.³.*, and Can Alkan³.*

¹Computer Science Department, ETH Zürich, Zürich 8092, Switzerland, ²Chair for Processor Design, Center For Advancing Electronics Dresden, Institute of Computer Engineering, Technische Universität Dresden, 01062 Dresden, Germany and ³Computer Engineering Department, Bilkent University, 06800 Ankara, Turkey

*To whom correspondence should be addressed.

Received on September 13, 2018; revised on February 27, 2019; editorial decision on March 7, 2019; accepted on March 27, 2019
SneakySnake [Alser+, Bioinformatics 2020]

Mohammed Alser, Taha Shahroodi, Juan-Gomez Luna, Can Alkan, and Onur Mutlu,
"SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs"


[Source Code]
[Online link at Bioinformatics Journal]

Subject Section

SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs

Mohammed Alser\(^1,2,*\), Taha Shahroodi\(^1\), Juan Gómez-Luna\(^1,2\),
Can Alkan\(^4,*\), and Onur Mutlu\(^1,2,3,4,*\)

\(^1\) Department of Computer Science, ETH Zurich, Zurich 8006, Switzerland
\(^2\) Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich 8006, Switzerland
\(^3\) Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh 15213, PA, USA
\(^4\) Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey

[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]
Quick Note: Key Principles and Results

- Two key principles:
  - Exploit the structure of the genome to minimize computation
  - Morph and exploit the structure of the underlying hardware to maximize performance and efficiency

- Algorithm-architecture co-design for DNA read mapping
  - **Speeds up** read mapping by \( \sim 100-1000X \)
  - **Improves accuracy** of read mapping in the presence of errors
Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018 Article history

[Preliminary arxiv.org version]
Figure 1. The analyzed genome assembly pipeline using nanopore sequence data, with its five steps and the associated tools for each step.

Recall Our Dream

- An embedded device that can perform comprehensive genome analysis in real time (within a minute)

- Still a long ways to go
  - Energy efficiency
  - Performance (latency)
  - Security
  - **Huge memory bottleneck**
Future of Genome Sequencing & Analysis

MinION from ONT

SmidgION from ONT
Why Do We Care? An Example from 2020

200 Oxford Nanopore sequencers have left UK for China, to support rapid, near-sample coronavirus sequencing for outbreak surveillance

Fri 31st January 2020

Following extensive support of, and collaboration with, public health professionals in China, Oxford Nanopore has shipped an additional 200 MinION sequencers and related consumables to China. These will be used to support the ongoing surveillance of the current coronavirus outbreak, adding to a large number of the devices already installed in the country.

Each MinION sequencer is approximately the size of a stapler, and can provide rapid sequence information about the coronavirus.

700Kg of Oxford Nanopore sequencers and consumables are on their way for use by Chinese scientists in understanding the current coronavirus outbreak.

Source: https://nanoporetech.com/about-us/news/200-oxford-nanopore-sequencers-have-left-uk-china-support-rapid-near-sample
Sequencing of COVID-19

- **Whole genome sequencing (WGS) and sequence data analysis are important**
  - To detect the virus from a human sample such as saliva, Bronchoalveolar fluid etc.
  - To understand the sources and modes of transmission of the virus
  - To discover the genomic characteristics of the virus, and compare with better-known viruses (e.g., 02-03 SARS epidemic)
  - To design and evaluate the diagnostic tests and deep-dive studies

- **Two key areas of COVID-19 genomic research**
  - To sequence the genome of the virus itself, COVID-19, in order to track the mutations in the virus.
  - To explore the genes of infected patients. This analysis can be used to understand why some people get more severe symptoms than others, as well as, help with the development of new treatments in the future.
COVID-19 Nanopore Sequencing (I)

SARS-CoV-2 Whole genome sequencing

- RT Step: ~1 hr
- PCR: ~2.30 hr
- Add Barcodes: ~1 hr
- Add Adapter: ~30 min
- Sequence: ~1 hr
- Analyse: ~1 hr

7 hr
RNA to answer

Of which ~1 hr sequencing time

From ONT (https://nanoporetech.com/covid-19/overview)
COVID-19 Nanopore Sequencing (II)

How are scientists using nanopore sequencing to research COVID-19?

- Samples are collected
- Validated SARS-CoV-2 RT-PCR test performed
- SARS-CoV-2 positive samples
- SARS-CoV-2 negative samples: used as negative controls

How can this be used?
- Genomic epidemiology: analyse variants & mutation rate, track spread of virus, identify clusters of transmission

What are the results?
- From RNA to full SARS-CoV-2 consensus sequence in ~7 hours
- Targeted amplification of SARS-CoV-2 genome + multiplexed, rapid nanopore sequencing

How?
- Metagenomic nanopore sequencing
- SARS-CoV-2 Direct RNA whole genome sequencing: assess viral genome in its native RNA form and the effect of base modifications
- Immune repertoire: assess response of the immune system to SARS-CoV-2 infection by sequencing of full-length immune cell receptor genes and transcripts
- Whole human genome sequencing: investigate what might cause different responses to the virus in different people based on their genome

What's next?

From ONT (https://nanoporetech.com/covid-19/overview)
Future of Genome Sequencing & Analysis

MinION from ONT

SmidgION from ONT
Accelerating Genome Analysis: Overview

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,

"Accelerating Genome Analysis: A Primer on an Ongoing Journey"


[Slides (pptx)(pdf)]
[Talk Video (1 hour 2 minutes)]

Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

Saugata Ghose
University of Illinois at Urbana–Champaign and Carnegie Mellon University

Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and Bilkent University
More on Fast Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Invited Lecture at Technion, Virtual, 26 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
Detailed Lectures on Genome Analysis

- **Computer Architecture, Fall 2020, Lecture 3a**
  - **Introduction to Genome Sequence Analysis** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5](https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5)

- **Computer Architecture, Fall 2020, Lecture 8**
  - **Intelligent Genome Analysis** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14](https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14)

- **Computer Architecture, Fall 2020, Lecture 9a**
  - **GenASM: Approx. String Matching Accelerator** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=XoLpmn-N-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15](https://www.youtube.com/watch?v=XoLpmn-N-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15)

- **Accelerating Genomics Project Course, Fall 2020, Lecture 1**
  - **Accelerating Genomics** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCgLgwIDRQDTyId](https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCgLgwIDRQDTyId)

---

**SAFARI**

[https://www.youtube.com/onurmutlulecutes](https://www.youtube.com/onurmutlulecutes)
Challenge and Opportunity for Future

High Performance

(to solve the toughest & all problems)
Challenge and Opportunity for Future

Personalized and Private

(in every aspect of life: health, medicine, spaces, devices, robotics, ...)
More on My Research & Teaching
Brief Self Introduction

- **Onur Mutlu**
  - Full Professor @ ETH Zurich ITET (INFK), since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-...
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - [https://people.inf.ethz.ch/omutlu/](https://people.inf.ethz.ch/omutlu/)
  - [omutlu@gmail.com](mailto:omutlu@gmail.com) (Best way to reach me)
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)

- **Research and Teaching in:**
  - Computer architecture, computer systems, hardware security, bioinformatics
  - Memory and storage systems
  - Hardware security, safety, predictability
  - Fault tolerance
  - Hardware/software cooperation
  - Architectures for bioinformatics, health, medicine
  - ...
Current Research Mission

Computer architecture, HW/SW, systems, bioinformatics, security

Heterogeneous Processors and Accelerators

Hybrid Main Memory

Persistent Memory/Storage

Graphics and Vision Processing

Build fundamentally better architectures

SAFARI
Four Key Current Directions

- Fundamentally Secure/Reliable/Safe Architectures
- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures
- Fundamentally Low-Latency and Predictable Architectures
- Architectures for AI/ML, Genomics, Medicine, Health
The Transformation Hierarchy

Computer Architecture (expanded view)

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Computer Architecture (narrow view)
**Axiom**

To achieve the highest energy efficiency and performance:

we must take the expanded view of computer architecture

Co-design across the hierarchy: Algorithms to devices

Specialize as much as possible within the design goals
Build fundamentally better architectures

- Data-centric arch. for low energy & high perf.
  - Proc. in Mem/DRAM, NVM, unified mem/storage
- Low-latency & predictable architectures
  - Low-latency, low-energy yet low-cost memory
  - QoS-aware and predictable memory systems
- Fundamentally secure/reliable/safe arch.
  - Tolerating all bit flips; patchable HW; secure mem
- Architectures for ML/AI/Genomics/Health/Med
  - Algorithm/arch./logic co-design; full heterogeneity
- Data-driven and data-aware architectures
  - ML/AI-driven architectural controllers and design
  - Expressive memory and expressive systems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>Program/Language</th>
<th>System Software</th>
<th>SW/HW Interface</th>
<th>Micro-architecture</th>
<th>Logic</th>
<th>Devices</th>
<th>Electrons</th>
</tr>
</thead>
</table>

Broad research spanning apps, systems, logic with architecture at the center
Onur Mutlu’s SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

https://safari.ethz.ch/safari-newsletter-april-2020/

Think BIG, Aim HIGH!

SAFARI

https://safari.ethz.ch
SAFARI Newsletter January 2021 Edition

https://safari.ethz.ch/safari-newsletter-january-2021/

Think Big, Aim High, and Have a Wonderful 2021!

Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 here). 2020 has
Principle: Teaching and Research

Teaching drives Research

Research drives Teaching

...
Principle: Insight and Ideas

Focus on Insight

Encourage New Ideas
Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZISOcGFMrlist=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1

- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=njX_14584Jw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=16

- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=r7sn41Ih-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41

- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3

- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LqzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25

- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39
An Interview on Research and Education

- Computing Research and Education (@ ISCA 2019)
  - https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz

- Maurice Wilkes Award Speech (10 minutes)
  - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15
More Thoughts and Suggestions

- Onur Mutlu,
  "Some Reflections (on DRAM)"
  Award Speech for ACM SIGARCH Maurice Wilkes Award, at the ISCA Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
  "How to Build an Impactful Research Group"
  57th Design Automation Conference Early Career Workshop (DAC), Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]
Referenced Papers, Talks, Artifacts

- All are available at

  https://people.inf.ethz.ch/omutulu/projects.htm

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/
Readings, Videos, Reference Materials
Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1

- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=njX_14584Jw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=16

- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=r7sn41lH-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41

- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3

- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LqzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25

- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39
Accelerated Memory Course (~6.5 hours)

- ACACES 2018
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu July 9-13, 2018
  - ~6.5 hours of lectures

- Website for the Course including Videos, Slides, Papers
  - [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x)

- All Papers are at:
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)
  - Final lecture notes and readings (for all topics)
Longer Memory Course (~18 hours)

- **TU Wien 2019**
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu June 12-19, 2019
  - ~18 hours of lectures

- **Website for the Course including Videos, Slides, Papers**
  - https://safari.ethz.ch/memory_systems/TUWien2019
  - https://www.youtube.com/playlist?list=PL5Q2soXY2 Zi_gntM55VoMIKlw7YrXOhbl

- **All Papers are at:**
  - https://people.inf.ethz.ch/omutlu/projects.htm
  - Final lecture notes and readings (for all topics)
An Interview on Research and Education

- **Computing Research and Education (@ ISCA 2019)**
  - [https://www.youtube.com/watch?v=8ffSEKZhmv0&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz](https://www.youtube.com/watch?v=8ffSEKZhmv0&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz)

- **Maurice Wilkes Award Speech (10 minutes)**
  - [https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15](https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15)
More Thoughts and Suggestions

- Onur Mutlu,
  "Some Reflections (on DRAM)"
  Award Speech for ACM SIGARCH Maurice Wilkes Award, at the ISCA Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
  "How to Build an Impactful Research Group"
  57th Design Automation Conference Early Career Workshop (DAC), Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]
A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{d}King Mongkut’s University of Technology North Bangkok


Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{b,c}

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}King Mongkut’s University of Technology North Bangkok


Reference Overview Paper III

Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND, RACHATA AUSAVARUNGNIRUN
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University

[Preliminary arxiv.org version]

Onur Mutlu and Lavanya Subramanian, "Research Problems and Opportunities in Memory Systems"
Invited Article in Supercomputing Frontiers and Innovations (SUPERFRI), 2014/2015.

Onur Mutlu,

"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"


[Slides (pptx) (pdf)]
Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD’s reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

https://arxiv.org/pdf/1706.08642
Onur Mutlu and Jeremie Kim, "RowHammer: A Retrospective"  
[Preliminary arXiv version]  
[Slides from COSADE 2019 (pptx)]  
[Slides from VLSI-SOC 2020 (pptx) (pdf)]  
[Talk Video (30 minutes)]

**RowHammer: A Retrospective**

Onur Mutlu$^\$\$  
$^\$\$ETH Zürich  
Jeremie S. Kim$^\$\$  
$^\$\$Carnegie Mellon University
Related Videos and Course Materials (I)

- **Parallel Computer Architecture Course Materials (Lecture Videos)**
Related Videos and Course Materials (II)

- Seminar in Computer Architecture Course Lecture Videos (Spring 2020, Fall 2019, Spring 2019, 2018)
- Seminar in Computer Architecture Course Materials (Spring 2020, Fall 2019, Spring 2019, 2018)
- Memory Systems Course Lecture Videos (Sept 2019, July 2019, June 2019, October 2018)
- Memory Systems Short Course Lecture Materials (Sept 2019, July 2019, June 2019, October 2018)
- ACACES Summer School Memory Systems Course Lecture Videos (2018, 2013)
- ACACES Summer School Memory Systems Course Materials (2018, 2013)
Some Open Source Tools (I)

- **Rowhammer** – Program to Induce RowHammer Errors
  - [https://github.com/CMU-SAFARI/rowhammer](https://github.com/CMU-SAFARI/rowhammer)

- **Ramulator** – Fast and Extensible DRAM Simulator
  - [https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)

- **MemSim** – Simple Memory Simulator
  - [https://github.com/CMU-SAFARI/memsim](https://github.com/CMU-SAFARI/memsim)

- **NOCulator** – Flexible Network-on-Chip Simulator
  - [https://github.com/CMU-SAFARI/NOCulator](https://github.com/CMU-SAFARI/NOCulator)

- **SoftMC** – FPGA-Based DRAM Testing Infrastructure
  - [https://github.com/CMU-SAFARI/SoftMC](https://github.com/CMU-SAFARI/SoftMC)

- **Other open-source software from my group**
  - [https://github.com/CMU-SAFARI/](https://github.com/CMU-SAFARI/)
  - [http://www.ece.cmu.edu/~safari/tools.html](http://www.ece.cmu.edu/~safari/tools.html)
Some Open Source Tools (II)

- MQSim – A Fast Modern SSD Simulator
  - https://github.com/CMU-SAFARI/MQSim
- Mosaic – GPU Simulator Supporting Concurrent Applications
  - https://github.com/CMU-SAFARI/Mosaic
- IMPICA – Processing in 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/IMPICA
- SMLA – Detailed 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/SMLA
- HWASim – Simulator for Heterogeneous CPU-HWA Systems
  - https://github.com/CMU-SAFARI/HWASim

- Other open-source software from my group
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html
More Open Source Tools (III)

- A lot more open-source software from my group
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html
ramulator-pim
A fast and flexible simulation infrastructure for exploring general-purpose processing-in-memory (PIM) architectures. Ramulator-PIM combines a widely-used simulator for out-of-order and in-order processors (ZSim) with Ramulator, a DRAM simulator with memory models for DDRx, LPDDRx, GDDRx, WIOx, HBMx, and HMCx. Ramulator is described in the IEEE ...

SMASH
SMASH is a hardware-software cooperative mechanism that enables highly-efficient indexing and storage of sparse matrices. The key idea of SMASH is to compress sparse matrices with a hierarchical bitmap compression format that can be accelerated from hardware. Described by Kanellopoulos et al. (MICRO '19)
https://people.inf.ethz.ch/omutlu/pub/SMA...

MQSim
MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implementations, steady-state SSD conditions, and the full end-to-end latency of requests in modern SSDs. It is described in detail in the FAST 2018 paper by A...

Apollo
Apollo is an assembly polishing algorithm that attempts to correct the errors in an assembly. It can take multiple set of reads in a single run and polish the assemblies of genomes of any size. Described in the Bioinformatics journal paper (2020) by Firtina et al. at https://people.inf.ethz.ch/omutlu/pub/apollo-technology-independent-genome-asse...

ramulator
A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in the IEEE CAL 2015 paper by Kim et al. at http://users.ece.cmu.edu/~omutlu/pb/ramulator_dram_simulator-ieee-cal15.pdf

Shifted-Hamming-Distance

SneakySnake
The first and the only pre-alignment filtering algorithm that works on all modern high-performance computing architectures. It works efficiently and fast on CPU, FPGA, and GPU architectures and that greatly (by more than two orders of magnitude) expedites sequence alignment calculation. Described by Alser et al. (preliminary version at https://a...

AirLift
AirLift is a tool that updates mapped reads from one reference genome to another. Unlike existing tools, it accounts for regions not shared between the two reference genomes and enables remapping across all parts of the references. Described by Kim et al. (preliminary version at http://arxiv.org/abs/1912.08735)

GPGPUSim-Ramulator
The source code for GPGPUSim+Ramulator simulator. In this version, GPGPUSim uses Ramulator to simulate the DRAM. This simulator is used to produce some of the lead-in text...
Referenced Papers, Talks, Artifacts

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm
  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en
  https://www.youtube.com/onurmutlulectures
  https://github.com/CMU-SAFARI/
An Interview on Research and Education

- Computing Research and Education (@ ISCA 2019)
  - [Video Link](https://www.youtube.com/watch?v=8ffSEKZhmv0&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz)

- Maurice Wilkes Award Speech (10 minutes)
  - [Video Link](https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=15)
More Thoughts and Suggestions

- Onur Mutlu, "Some Reflections (on DRAM)"
  Award Speech for ACM SIGARCH Maurice Wilkes Award, at the ISCA Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu, "How to Build an Impactful Research Group"
  57th Design Automation Conference Early Career Workshop (DAC), Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]
End of Backup Slides