The Problem

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
Huge Demand for Performance & Efficiency

Exponential Growth of Neural Networks

1800x more compute
In just 2 years

Tomorrow, multi-trillion parameter models

https://www.youtube.com/watch?v=x2-qB0J7KHW
Data is Key for Future Workloads

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]
Data Overwhelms Modern Machines

In-memory Databases

Graph/Tree Processing

Data → performance & energy bottleneck

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

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
Data is Key for Future Workloads

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
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

VP9

Video Playback

Google’s video codec

Video Capture

Google’s video codec
Data is Key for Future Workloads

development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

Genome Analysis

1 Sequencing

2 Read Mapping

Data → performance & energy bottleneck

3 Variant Calling

4 Scientific Discovery
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](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
Published: 02 April 2018    Article history ▼

Oxford Nanopore MinION

Data → performance & energy bottleneck
One Problem with (Genome) Analysis Today

Special-Purpose Machine for Data Generation

General-Purpose Machine for Data Analysis

FAST

SLOW

Slow and inefficient processing capability

This picture is similar for many “data generators & analyzers” today
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)]
FPGA-based Near-Memory Analytics

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
Data Overwhelms Modern Machines …

- Storage/memory capability

- Communication capability

- Computation capability

- Greatly impacts robustness, energy, performance, cost
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.

Perils of Processor-Centric Design

Most of the system is dedicated to storing and moving data.

Yet, system is still bottlenecked by memory.
Deeper and Larger Memory Hierarchies

AMD Ryzen 5000, 2020

**Core Count:**
8 cores/16 threads

**L1 Caches:**
32 KB per core

**L2 Caches:**
512 KB per core

**L3 Cache:**
32 MB shared
AMD’s 3D Last Level Cache (2021)

AMD increases the L3 size of their 8-core Zen 3 processors from 32 MB to 96 MB

Additional 64 MB L3 cache die stacked on top of the processor die
- Connected using Through Silicon Vias (TSVs)
- Total of 96 MB L3 cache
Deeper and Larger Memory Hierarchies

IBM POWER10, 2020

Cores:
15-16 cores, 8 threads/core

L2 Caches:
2 MB per core

L3 Cache:
120 MB shared

Deeper and Larger Memory Hierarchies

Apple M1 Ultra System (2022)

https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

VP9

Video Playback

Google’s video codec

VP9

Video Capture

Google’s video codec
Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

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

Data Movement Overwhelms Modern Machines

An Intelligent Architecture
Handles Data Well
How to Handle Data Well

- **Ensure data does not overwhelm** the components
  - via intelligent algorithms
  - via intelligent architectures
  - via whole system designs: algorithm-architecture-devices

- **Take advantage of** vast amounts of data and metadata
  - to improve architectural & system-level decisions

- **Understand and exploit** properties of (different) data
  - to improve algorithms & architectures in various metrics
Corollaries: Computing Systems Today …

- Are processor-centric vs. data-centric

- Make designer-dictated decisions vs. data-driven

- Make component-based myopic decisions vs. data-aware
Architectures for Intelligent Machines

Data-centric
Data-driven
Data-aware
A Blueprint for Fundamentally Better Architectures

Onur Mutlu,
"Intelligent Architectures for Intelligent Computing Systems"
[Slides (pptx) (pdf)]
[IEDM Tutorial Slides (pptx) (pdf)]
[Short DATE Talk Video (11 minutes)]
[Longer IEDM Tutorial Video (1 hr 51 minutes)]
We Need to Revisit the Entire Stack

We can get there step by step
Data-Centric (Memory-Centric) Architectures
Data-Centric Architectures: Properties

- **Process data where it resides** *(where it makes sense)*
  - Processing in and near memory structures

- **Low-latency and low-energy data access**
  - Low latency memory
  - Low energy memory

- **Low-cost data storage and processing**
  - High capacity memory at low cost: hybrid memory, compression

- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost, perf.
Processing Data
Where It Makes Sense
Processing in/near Memory: An Old Idea


IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 8, AUGUST 1969

Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be “programmed” to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays
logic in memory, push-down memory, sorting, switching functions.

CELL EQUATIONS: 
\[
\hat{x} = \overline{x}y + wx \\
y = wx \\
\hat{z} = M(x, \overline{y}, z) = x\overline{y} + z(x + \overline{y})
\]

Fig. 1. Cellular sorting array I.

https://doi.org/10.1109/T-C.1969.222754
Processing in/near Memory: An Old Idea


A Logic-in-Memory Computer

HAROLD S. STONE

Abstract—If, as presently projected, the cost of microelectronic arrays in the future will tend to reflect the number of pins on the array rather than the number of gates, the logic-in-memory array is an extremely attractive computer component. Such an array is essentially a microelectronic memory with some combinational logic associated with each storage element.
Why In-Memory Computation Today?

- **Push from Technology**
  - Memory technology scaling is not going well (e.g., RowHammer)
  - Many scaling issues demand intelligence in memory

- **Pull from Applications & Systems**
  - Data access is the major bottleneck
  - Systems are energy & power limited
  - Data movement much more energy-hungry than computation
  - Need all at the same time: performance, energy, sustainability
  - We can improve all metrics by minimizing data movement

- **Designs are squeezed in the middle**
Processing-in-Memory Landscape Today

And, many other experimental chips and startups
Memory Scaling Issues Are Real

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
Memory Scaling Issues Are Real

- 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 (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

Onur Mutlu$^{§,‡}$ Jeremie S. Kim$^{‡,§}$

$^§$ETH Zürich  $^‡$Carnegie Mellon University
The Push from Circuits and Devices

Main Memory Needs
Intelligent Controllers
Emerging Memories Also Need Intelligent Controllers


One of the 13 computer architecture papers of 2009 selected as Top Picks by IEEE Micro. Selected as a CACM Research Highlight. 2022 Persistent Impact Prize.

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
Intelligent Memory Controllers Can Avoid Many Failures & Enable Better Scaling
Three Key Systems & Application Trends

1. Data access is the 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
Do We Want This?

Source: V. Milutinovic
Or This?

Source: V. Milutinovic
Challenge and Opportunity for Future

High Performance, Energy Efficient, Sustainable
(All at the Same Time)
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 \(\rightarrow\) at great system cost
- Processor is heavily optimized and is considered the master
- Data storage units are dumb and are largely unoptimized
Yet …

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

Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"


One of the 15 computer arch. papers of 2003 selected as Top Picks by IEEE Micro. HPCA Test of Time Award (awarded in 2021).
All of Google’s Data Center Workloads (2015):

The Energy Perspective

Communication Dominates Arithmetic

Dally, HiPEAC 2015

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

Efficient off-chip link

DRAM Rd/Wr

20mm
A memory access consumes ~100-1000X the energy of a complex addition.
Data Movement vs. Computation Energy

Energy for a 32-bit Operation (log scale)

- ADD (int)
- ADD (float)
- Register File
- MULT (int)
- MULT (float)
- SRAM Cache
- DRAM

Energy (pJ)  ADD (int) Relative Cost

- 0.1
- 0.9
- 1
- 3.1
- 3.7
- 5
- 640

Data Movement vs. Computation Energy

A memory access consumes 6400X the energy of a simple integer addition.
Energy Waste in Mobile Devices


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

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹
Rachata Ausavarungrunr¹
Aki Kuusela³
Allan Knies³

Saugata Ghose¹
Eric Shiu³
Parthasarathy Ranganathan³

Youngsok Kim²
Rahul Thakur³

Daehyun Kim⁴,³
Onur Mutlu⁵,¹

SAFARI
We Do Not Want to Move Data!

Communication Dominates Arithmetic

A memory access consumes $\sim 100-1000X$ 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?
- processors & communication units?
- software & hardware interfaces?
- system software, compilers, languages?
- algorithms & theoretical foundations?
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  
We Need to Think Differently from the Past Approaches
Processing in Memory: Two Approaches

1. Processing near Memory
2. Processing using Memory
Mindset: Memory as an Accelerator

Memory similar to a “conventional” accelerator
Eliminating the Adoption Barriers

Processing-in-Memory in the Real World
This does not include many experimental chips and startups
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 Haji, 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, unlocking 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 architecture, 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 GPU 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 use 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 benchmarks (PIM microbenchmarks), 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 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 140 and 2,560 PUs 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.

More on the UPMEM PIM System

https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2ZI9xidyIgBxUz7xRPS-wisBN&index=26
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 GIANNOLU, 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.

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
Upcoming TECHCON Presentation

- Dr. Juan Gomez-Luna
  - Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware
  - Based on two major works

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-In-Memory Hardware

Year: 2021, Pages: 1-7
DOI Bookmark: 10.1109/IGSC54211.2021.9651614

Authors
Juan Gómez-Luna, ETH Zürich
Izzat El Hajj, American University of Beirut
Ivan Fernandez, University of Malaga
Christina Giannoula, National Technical University of Athens
Gerald F. Oliveira, ETH Zürich
Onur Mutlu, ETH Zürich

https://www.youtube.com/watch?v=nphV36SrysA
The throughput saturation point is as low as \( \frac{1}{4} \) OP/B, i.e., 1 integer addition per every 32-bit element fetched.

**KEY TAKEAWAY 1**

The UPMEM PIM architecture is fundamentally compute bound. As a result, the most suitable workloads are memory-bound.
Key Takeaway 2

The most well-suited workloads for the UPMEM PIM architecture use no arithmetic operations or use only simple operations (e.g., bitwise operations and integer addition/subtraction).

KEY TAKEAWAY 2
Key Takeaway 3

The most well-suited workloads for the UPMEM PIM architecture require little or no communication across DPUs (inter-DPU communication).
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

el1goluj@gmail.com

https://github.com/CMU-SAFARI/prim-benchmarks
UPMEM PIM System Summary & Analysis

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

"Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"


[arXiv version]
[PrIM Benchmarks Source Code]
[Slides (pptx) (pdf)]
[Talk Video (37 minutes)]
[Lightning Talk Video (3 minutes)]

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

Juan Gómez-Luna
ETH Zürich

Izzat El Hajj
American University of Beirut

Ivan Fernandez
University of Malaga

Christina Giannoula
National Technical University of Athens

Geraldo F. Oliveira
ETH Zürich

Onur Mutlu
ETH Zürich
## PrIM Benchmarks: Application Domains

<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-SAFAI/prim-benchmarks](https://github.com/CMU-SAFAI/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 all 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 PIM Architecture

Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System

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}, AND 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

Corresponding author: Juan Gómez-Luna (e-mail: juang@ethz.ch).

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

Onur Mutlu Lectures
18.7K subscribers

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 place 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=PL5Q2soXY2Zj_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*

* Boroumand et al., “Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks,” ASPLOS 2018
* Kester et al., “Quantifying the Energy Cost of Data Movement in Scientific Applications,” ISWC 2013
* Pandiany and Wu, “Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms,” ISWC 2014
ML Training on a Real PIM System

Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna\textsuperscript{1}  Yuxin Guo\textsuperscript{1}  Sylvan Brocard\textsuperscript{2}  Julien Legriel\textsuperscript{2}
Remy Cimadomo\textsuperscript{2}  Geraldo F. Oliveira\textsuperscript{1}  Gagandeep Singh\textsuperscript{1}  Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich  \textsuperscript{2}UPMEM

An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna\textsuperscript{1}  Yuxin Guo\textsuperscript{1}  Sylvan Brocard\textsuperscript{2}  Julien Legriel\textsuperscript{2}
Remy Cimadomo\textsuperscript{2}  Geraldo F. Oliveira\textsuperscript{1}  Gagandeep Singh\textsuperscript{1}  Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich  \textsuperscript{2}UPMEM

\url{https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s}
ML Training on a Real PIM System

• Need to optimize data representation
  (1) fixed-point
  (2) quantization
  (3) hybrid precision

• Use lookup tables (LUTs) to implement complex functions (e.g., sigmoid)

• Optimize data placement & layout for streaming

• Large speedups: 2.8X/27X vs. CPU, 1.3x/3.2x vs. GPU
ML Training on Real PIM Talk Video

https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s
SpMV Multiplication on Real PIM Systems

- Appears in SIGMETRICS 2022

SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems

CHRISTINA GIANNIOULA, ETH Zürich, Switzerland and National Technical University of Athens, Greece
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
NECTARIOS KOZIRIS, National Technical University of Athens, Greece
GEORGIOS GOUMAS, National Technical University of Athens, Greece
ONUR MUTLU, ETH Zürich, Switzerland

https://github.com/CMU-SAIFARI/SparseP

SAFARI https://www.youtube.com/watch?v=5kaOsJKIGrE
SparseP
Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula
Ivan Fernandez, Juan Gomez-Luna,
Nectarios Koziris, Georgios Goumas, Onur Mutlu
SparseP Summary

Efficient Algorithmic Designs

The first open-source Sparse Matrix Vector Multiplication (SpMV) software package, SparseP, for real Processing-In-Memory (PIM) systems

Extensive Characterization

The first comprehensive analysis of SpMV on the first real commercial PIM architecture

SparseP is Open-Source

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

Recommendations for Architects and Programmers

SparseP: SpMV Library for Real PIMs

Our Contributions:

1. Design **efficient SpMV kernels** for current and future PIM systems
   - 25 SpMV kernels
     - 4 compressed matrix formats (CSR, COO, BCSR, BCOO)
     - 6 data types
     - 4 data partitioning techniques
     - Various load balancing schemes among PIM cores/threads
     - 3 synchronization approaches

2. Provide a **comprehensive analysis** of SpMV on the first commercially-available real PIM system
   - 26 sparse matrices
   - Comparisons to state-of-the-art CPU and GPU systems
   - **Recommendations** for software, system and hardware designers
SparseP Talk Video

Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula
Ivan Fernandez, Juan Gomez-Luna, Nectarios Koziris, Georgios Goumas, Onur Mutlu

Processing-in-Memory Course: Lecture 11: SpMV on a Real PIM Architecture - Spring 2022

https://www.youtube.com/watch?v=5kaOsJKlGrE
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 29nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Heeun Kwon1, Suk Han Lee1, Jaehoon Lee1, Sang-Hyuk Kwon1, Je Min Ryu1, Jong-Pil Son1, Geongil O1, Hak-Soo Yu1, Haseuk Lee2, Soo Young Kim1, Youngmin Cho1, Jin Guk Kim1, Jongyoon Choi1, Hyeon-Sung Shin1, Jin Kim1, Beng-Seng Phua1, HyoYoung Kim1, Myeong Jun Song1, Ahn Choi1, Daeho Kim1, SooYoung Kim1, Eun-Bong Kim1, David Wang1, Shinh-Joong Kang1, Yuhwan Ro1, Seungwoo Seo1, JoonHo Song1, Jaryoun Yoon1, Kyomin Sohn1, Nam Sung Kim1

1Samsung Electronics, Hwasung, Korea
2Samsung Electronics, San Jose, CA
3Samsung Electronics, Suwon, Korea
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]
Samsung Function-in-Memory DRAM (2021)

<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
Samsung AxDIMM (2021)

- DDRx-PIM
  - DLRM recommendation system

SK hynix Develops PIM, Next-Generation AI Accelerator

February 16, 2022

Seoul, February 16, 2022

SK hynix (or “the Company”, www.skhynix.com) announced on February 16 that it has developed PIM*, a next-generation memory chip with computing capabilities.

*PIM (Processing in Memory): A next-generation technology that provides a solution for data congestion issues for AI and big data by adding computational functions to semiconductor memory

It has been generally accepted that memory chips store data and CPU or GPU, like human brain, process data. SK hynix, following its challenge to such notion and efforts to pursue innovation in the next-generation smart memory, has found a breakthrough solution with the development of the latest technology.

SK hynix plans to showcase its PIM development at the world’s most prestigious semiconductor conference, 2022 ISSCC*, in San Francisco at the end of this month. The company expects continued efforts for innovation of this technology to bring the memory-centric computing, in which semiconductor memory plays a central role, a step closer to the reality in devices such as smartphones.

*ISSCC: The International Solid-State Circuits Conference will be held virtually from Feb. 20 to Feb. 24 this year with a theme of “Intelligent Silicon for a Sustainable World”

For the first product that adopts the PIM technology, SK hynix has developed a sample of GDDR6-AiM (Accelerator in memory). The GDDR6-AiM adds computational functions to GDDR6` memory chips, which process data at 16Gbps. A combination of GDDR6-AiM with CPU or GPU instead of a typical DRAM makes certain computation speed 16 times faster. GDDR6-AiM is widely expected to be adopted for machine learning, high-performance computing, and big data computation and storage.
29.1 184QPS/W 64Mb/mm² 3D Logic-to-DRAM Hybrid Bonding with Process-Near-Memory Engine for Recommendation System

Dimin Niu¹, Shuangchen Li¹, Yuhao Wang¹, Wei Han¹, Zhe Zhang², Yijin Guan², Tianchan Guan³, Fei Sun¹, Fei Xue¹, Lide Duan¹, Yuanwei Fang¹, Hongzhong Zheng¹, Xiping Jiang⁴, Song Wang⁴, Fengguo Zuo⁴, Yubing Wang⁴, Bing Yu⁴, Qiwei Ren⁴, Yuan Xie¹
Eliminating the Adoption Barriers

Processing-in-Memory in the Real World
Processing in Memory: Two Approaches

1. Processing near Memory
2. Processing using Memory
Mindset: Memory as an Accelerator

Memory similar to a “conventional” accelerator
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
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

Transfer row

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 in Off-the-Shelf DRAM Chips

- Idea: Violate DRAM timing parameters to mimic RowClone

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao  
feig@princeton.edu  
Department of Electrical Engineering  
Princeton University

Georgios Tziantzioulis  
georgios.tziantzioulis@princeton.edu  
Department of Electrical Engineering  
Princeton University

David Wentzlaff  
wentzlaf@princeton.edu  
Department of Electrical Engineering  
Princeton University

Real Processing Using Memory Prototype

- End-to-end RowClone & TRNG using off-the-shelf DRAM chips
- Idea: Violate DRAM timing parameters to mimic RowClone

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun§†  Juan Gómez Luna§  Konstantinos Kanellopoulos§  Behzad Salami§*  Hasan Hassan§  Oğuz Ergin†  Onur Mutlu§  §ETH Zürich  †TOBB ETÜ  *BSC

https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s
Real Processing-using-Memory Prototype

https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s
Building a PiDRAM Prototype

To build PiDRAM’s prototype on Xilinx ZC706 boards, developers need to use the two sub-projects in this directory. `fpga-zynq` is a repository branched off of UCB-BAR’s `fpga-zynq` repository. We use `fpga-zynq` to generate rocket chip designs that support end-to-end DRAM PuM execution. `controller-hardware` is where we keep the main Vivado project and Verilog sources for PiDRAM’s memory controller and the top level system design.

Rebuilding Steps

1. Navigate into `fpga-zynq` and read the README file to understand the overall workflow of the repository
   - Follow the readme in `fpga-zynq/rocket-chip/riscv-tools` to install dependencies
2. Create the Verilog source of the rocket chip design using the `ZynqCopyFPAGConfig`
   - Navigate into zc706, then run `make rocket CONFIG=ZynqCopyFPAGConfig -j number of cores`
3. Copy the generated Verilog file (should be under zc706/src) and overwrite the same file in `controller-hardware/source/hdl/impl/rocket-chip`
4. Open the Vivado project in `controller-hardware/Vivado_Project` using Vivado 2016.2
5. Generate a bitstream
6. Copy the bitstream (`system_top.bit`) to `fpga-zynq/zc706`
7. Use the `./build_script.sh` to generate the new `boot.bin` under `fpga-images-zc706`, you can use this file to program the FPGA using the SD-Card
   - For details, follow the relevant instructions in `fpga-zynq/README.md`

You can run programs compiled with the RISC-V Toolchain supplied within the `fpga-zynq` repository. To install the toolchain, follow the instructions under `fpga-zynq/rocket-chip/riscv-tools`.

Generating DDR3 Controller IP sources

We cannot provide the sources for the Xilinx Phy IP we use in PiDRAM’s memory controller due to licensing issues. We describe here how to regenerate them using Vivado 2016.2. First, you need to generate the IP RTL files:

1. Open IP Catalog
2. Find “Memory Interface Generator (MIG 7 Series)” IP and double click

https://github.com/cmu-safari/pidram
https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s
In-DRAM Copy and Initialization improve throughput by 119x and 89x, respectively.
PiDRAM is the first flexible end-to-end framework that enables system integration studies and evaluation of real Processing-using-Memory (PuM) techniques. PiDRAM, at a high level, comprises a RISC-V system and a custom memory controller that can perform PuM operations in real DDR3 chips. This repository contains all sources required to build PiDRAM and develop its prototype on the Xilinx ZC706 FPGA boards.

https://github.com/CMU-SAFARI/PiDRAM
PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun, Juan Gómez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oğuz Ergin, Onur Mutlu

Processing-using-memory (PuM) techniques leverage the analog operation of memory cells to perform computation. Several recent works have demonstrated PuM techniques in off-the-shelf DRAM devices. Since DRAM is the dominant memory technology as main memory in current computing systems, these PuM techniques represent an opportunity for alleviating the data movement bottleneck at very low cost. However, system integration of PuM techniques imposes non-trivial challenges that are yet to be solved. Design space exploration of potential solutions to the PuM integration challenges requires appropriate tools to develop necessary hardware and software components. Unfortunately, current specialized DRAM-testing platforms, or system simulators do not provide the flexibility and/or the holistic system view that is necessary to deal with PuM integration challenges.

We design and develop PiDRAM, the first flexible end-to-end framework that enables system integration studies and evaluation of real PuM techniques. PiDRAM provides software and hardware components to rapidly integrate PuM techniques across the whole system software and hardware stack (e.g., necessary modifications in the operating system, memory controller). We implement PiDRAM on an FPGA-based platform along with an open-source RISC-V system. Using PiDRAM, we implement and evaluate two state-of-the-art PuM techniques: in-DRAM (i) copy and initialization, (ii) true random number generation. Our results show that the in-memory copy and initialization techniques can improve the performance of bulk copy operations by 12.6x and bulk initialization operations by 14.6x on a real system. Implementing the true random number generator requires only 190 lines of Verilog and 74 lines of C code using PiDRAM’s software and hardware components.
Alloc_align Example

A = alloc_align(16*1024, 0);  B = alloc_align(16*1024, 0);

Array A

16 KBs

4 KB

Virtual Addresses: 0x0000 0x1000 0x2000

Array B

16 KBs

0x7000

Row 1

Bank 0  Bank 1  Bank 2

Row 0

https://youtu.be/s_z_S6FYpC8
(Truly) In-Memory Computation

- We can 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
\[ AB + BC + AC \]

\[ C(A + B) + \neg C(AB) \]
Bulk Bitwise Operations in Workloads

- Bitmap indices (database indexing)
- Set operations
- Encryption algorithms
- BitWeaving (database queries)
- BitFunnel (web search)
- DNA sequence mapping

[1] Li and Patel, BitWeaving, SIGMOD 2013
In-DRAM Acceleration of Database Queries

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

More on Ambit

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,

"Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
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 Hajinazar\textsuperscript{1,2}
Nika Mansouri Ghiasi\textsuperscript{1}

*Geraldo F. Oliveira\textsuperscript{1}
Minesh Patel\textsuperscript{1}
Juan Gómez-Luna\textsuperscript{1}

Sven Gregorio\textsuperscript{1}
Mohammed Alser\textsuperscript{1}
Onur Mutlu\textsuperscript{1}
João Dinis Ferreira\textsuperscript{1}
Saugata Ghose\textsuperscript{3}

\textsuperscript{1}ETH Zürich \textsuperscript{2}Simon Fraser University \textsuperscript{3}University of Illinois at Urbana–Champaign
SIMDRAM Framework: Overview

**User Input**

Desired operation

- AND/OR/NOT logic

**SIMDRAM Output**

New SIMDRAM μProgram

- μProgram

Main memory

- bbop_new

New SIMDRAM instruction

**Step 1: Generate MAJ logic**

MAJ

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

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

**Step 3: Execution according to μProgram**

SIMDRAM-enabled application

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

Control Unit

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

Memory Controller

- ACT/PRE

Instruction result in memory
SIMDRAM Key Results

Evaluated on:
- 16 complex in-DRAM operations
- 7 commonly-used real-world applications

SIMDRAM provides:

• 88× and 5.8× the throughput of a CPU and a high-end GPU, respectively, over 16 operations

• 257× and 31× the energy efficiency of a CPU and a high-end GPU, respectively, over 16 operations

• 21× and 2.1× the performance of a CPU an a high-end GPU, over seven real-world applications
More on SIMDRAM

- 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
Juan Gómez-Luna¹

Sven Gregorio¹
Mohammed Alser¹
Onur Mutlu¹
João Dinis Ferreira¹
Saugata Ghose³

¹ETH Zürich  ²Simon Fraser University  ³University 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)]

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
In-DRAM True Random Number Generation


[Slides (pptx) (pdf)]
[Full Talk Video (21 minutes)]
[Full Talk Lecture Video (27 minutes)]
Top Picks Honorable Mention by IEEE Micro.
In-DRAM True Random Number Generation


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Talk Video (25 minutes)]
[SAFARI Live Seminar Video (1 hr 26 mins)]

QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

Ataberk Olgun§† Minesh Patel§ A. Giray Yağlıkçı§ Haocong Luo§
Jeremie S. Kim§ F. Nisa Bostancı§† Nandita Vijaykumar§⊙ Oguz Ergin† Onur Mutlu§

§ETH Zürich †TOBB University of Economics and Technology ⊙University of Toronto
In-DRAM True Random Number Generation

- F. Nisa Bostanci, Ataberk Olgun, Lois Orosa, A. Giray Yaglikci, Jeremie S. Kim, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
  "DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators"
  Proceedings of the 28th International Symposium on High-Performance Computer Architecture (HPCA), Virtual, April 2022.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]

DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators

F. Nisa Bostancı†§  Ataberk Olgun†§  Lois Orosa§  A. Giray Yağlıkçı§
Jeremie S. Kim§  Hasan Hassan§  Oğuz Ergin†  Onur Mutlu§

†TOBB University of Economics and Technology  §ETH Zürich

Processing in Memory: Adoption Challenges

1. Processing using Memory
2. Processing near Memory
Eliminating the Adoption Barriers

How to Enable Adoption of Processing in Memory
Potential Barriers to Adoption of PIM

1. **Applications & software** for PIM

2. Ease of **programming** (interfaces and compiler/HW support)

3. **System** and **security** support: coherence, synchronization, virtual memory, isolation, communication interfaces, ...

4. **Runtime** and **compilation** systems for adaptive scheduling, data mapping, access/sharing control, ...

5. **Infrastructures** to assess benefits and feasibility

All can be solved with change of mindset
We Need to Revisit the Entire Stack

We can get there step by step
Challenge and Opportunity for Future

Fundamentally Energy-Efficient (Data-Centric) Computing Architectures
Fundamentally High-Performance (Data-Centric) Computing Architectures
Challenge and Opportunity for Future Computing Architectures with Minimal Data Movement
Concluding Remarks

- We must design systems to be balanced, high-performance, energy-efficient (all at the same time) → intelligent systems
  - Data-centric, data-driven, data-aware

- Enable computation capability inside and close to memory
  - This can
    - Lead to orders-of-magnitude improvements
    - Enable new applications & computing platforms
    - Enable better understanding of nature
    - ...

- Future of truly memory-centric computing is bright
  - We need to do research & design across the computing stack
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
Funding Acknowledgments

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

SAFARI
SAFARI Research Group
safari.ethz.ch

Think BIG, Aim HIGH!

https://safari.ethz.ch
Referenced Papers, Talks, Artifacts

- All are available at

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

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/
Open Source Tools: SAFARI GitHub

SAFARI Research Group at ETH Zurich and Carnegie Mellon University
Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

ETH Zurich and Carnegie Mellon U...  🌐 https://safari.ethz.ch/  💌 omutlu@gmail.com

Pinned

- **ramulator**  Public
  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...
  🌟 C++  ⭐ 311  📡 161

- **prim-benchmarks**  Public
  PrIM (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publi...
  ⚪ C  ⭐ 53  📡 21

- **DAMOV**  Public
  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 processin...
  🌟 C++  ⭐ 26  📡 4

- **SneakySnake**  Public
  SneakySnake is the first and the only pre-alignment filtering algorithm that works efficiently and fast on modern CPU, FPGA, and GPU architectures. It greatly (by more than two orders of magnitude...
  🌟 VHDL  ⭐ 41  📡 8

- **MQSim**  Public
  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 implement...
  🌟 C++  ⭐ 146  📡 93

- **rowhammer**  Public
  ⚪ C  ⭐ 189  📡 41

https://github.com/CMU-SAFARI/
Special Research Sessions & Courses

- Special Session at ISVLSI 2022: 9 cutting-edge talks

[YouTube Video](https://www.youtube.com/watch?v=qeukNs5XI3g)
Comp Arch (Fall’21)

- **Fall 2021 Edition:**
  - https://safari.ethz.ch/architecture/fall2021/doku.php?id=schedule

- **Fall 2020 Edition:**

- **Youtube Livestream (2021):**
  - https://www.youtube.com/watch?v=4yfkM_5EFg0&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF

- **Youtube Livestream (2020):**
  - https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN

- Master’s level course
  - Taken by Bachelor’s/Masters/PhD students
  - Cutting-edge research topics + fundamentals in Computer Architecture
  - 5 Simulator-based Lab Assignments
  - Potential research exploration
  - Many research readings

https://www.youtube.com/onurmutlulectures
**PIM Course (Spring 2022)**

- **Spring 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory)

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi841fUYYUK9EsXKhQKRPyX](https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi841fUYYUK9EsXKhQKRPyX)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - Processing-in-Memory lectures
  - Hands-on research exploration
  - Many research readings

---

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Livestream</th>
<th>Meeting</th>
<th>Learning Materials</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>19.03</td>
<td>Live</td>
<td>M1: PIM Course Presentation</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td>HW 0 Out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M2: PIM UPMEM PM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td>15.03</td>
<td>Live</td>
<td>M3: MPP &amp; Embedded PIM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td>Live</td>
<td>M4: Real-world PM: Samsung</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24.03</td>
<td>Live</td>
<td>M5: Benchmarking of UPMEM PM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M6: Real-world PM: X86 Hyper-V</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td>31.03</td>
<td>Live</td>
<td>M7: Programming PM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>07.04</td>
<td>Live</td>
<td>M8: Benchmarking and Workload</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.04</td>
<td>Live</td>
<td>M9: Real-world PM: Samsung</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21.04</td>
<td>Live</td>
<td>M10: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28.04</td>
<td>Live</td>
<td>M11: Split Memory: A New Model</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td>12.05</td>
<td>Live</td>
<td>M12: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19.05</td>
<td>Live</td>
<td>M13: End-to-End Framework for</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26.05</td>
<td>Live</td>
<td>M14: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>02.06</td>
<td>Live</td>
<td>M15: Analyzing and Mitigating</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>09.06</td>
<td>Live</td>
<td>M16: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.06</td>
<td>Live</td>
<td>M17: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23.06</td>
<td>Live</td>
<td>M18: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.07</td>
<td>Live</td>
<td>M19: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>09.08</td>
<td>Live</td>
<td>M20: Real-world PM: IBM</td>
<td><a href="#">PDF</a> <a href="#">PPT</a></td>
<td></td>
</tr>
</tbody>
</table>

---

**SAFARI**
Genomics (Spring 2022)

- **Spring 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=bioinformatics](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=bioinformatics)

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=DEL_5A_Y3TI&list=PL5Q2soXY2Zi8NrPDgOR1yRU_Cxxjw-u18](https://www.youtube.com/watch?v=DEL_5A_Y3TI&list=PL5Q2soXY2Zi8NrPDgOR1yRU_Cxxjw-u18)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - Genomics lectures
  - Hands-on research exploration
  - Many research readings
Hetero. Systems (Spring’22)

- **Spring 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=heterogeneous_systems](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=heterogeneous_systems)

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=oFO5fTrqFIY&list=PL5Q2soXY2Zi9XrgXR38IM_FTjmY6h7Gzm](https://www.youtube.com/watch?v=oFO5fTrqFIY&list=PL5Q2soXY2Zi9XrgXR38IM_FTjmY6h7Gzm)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - GPU and Parallelism lectures
  - Hands-on research exploration
  - Many research readings
HW/SW Co-Design (Spring 2022)

- **Spring 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=hw_sw_co_design](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=hw_sw_co_design)

- **Youtube Livestream:**
  - [https://youtube.com/playlist?list=PL5Q2soXY2Zi8nH7un3ghD2nutKWWDk-NK](https://youtube.com/playlist?list=PL5Q2soXY2Zi8nH7un3ghD2nutKWWDk-NK)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - HW/SW co-design lectures
  - Hands-on research exploration
  - Many research readings
SSD Course (Spring 2022)

- **Spring 2022 Edition:**

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=_q4rm71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMI2RAq](https://www.youtube.com/watch?v=_q4rm71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMI2RAq)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - SSD Basics and Advanced Topics
  - Hands-on research exploration
  - Many research readings
Memory-Centric Computing

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

2 September 2022
MATEO 2022
Backup Slides
In-Storage Genomic Data Filtering [ASPLOS 2022]

- Nika Mansourí Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,

"GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"


[Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]

---

**GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis**

Nika Mansourí Ghiasi¹ Jisung Park¹ Harun Mustafa¹ Jeremie Kim¹ Ataberk Olgun¹ Arvid Gollwitzer¹ Damla Senol Cali² Can Firtina¹ Haiyu Mao¹ Nour Almadhoun Alserr¹ Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto
Genome Sequence Analysis

Data Movement from Storage

- Storage System
- Main Memory
- Cache (Computation Unit (CPU or Accelerator))

Alignment

- Computation overhead
- Data movement overhead
Accelerating Genome Sequence Analysis

- Heuristics
- Accelerators
- Filters

- Storage System
- Main Memory
- Cache
- Computation Unit (CPU or Accelerator)

✓ Computation overhead
✗ Data movement overhead
Key Idea

Filter reads that do not require alignment inside the storage system

Exactly-matching reads
Do not need expensive approximate string matching during alignment

Non-matching reads
Do not have potential matching locations and can skip alignment
GenStore

Filter reads that do not require alignment inside the storage system

GenStore-Enabled Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

✓ Computation overhead

✓ Data movement overhead

GenStore provides significant speedup (1.4x - 33.6x) and energy reduction (3.9x – 29.2x) at low cost
## GenStore-EM: Not Finding a Match

### Sorted Read Table

<table>
<thead>
<tr>
<th>Read</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td></td>
</tr>
<tr>
<td>AAAAAG</td>
<td></td>
</tr>
<tr>
<td>AAAAACT</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

### Sorted K-mer Index

<table>
<thead>
<tr>
<th>K-mer</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td></td>
</tr>
<tr>
<td>AAAAAAC</td>
<td></td>
</tr>
<tr>
<td>AAAAAAT</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

### Comparator

Read < K-mer

---

**GenStore: A High-Performance In-Storage Processing System for Genome Analysis -- ASPLOS'22 Talk**

346 views • Premiered Mar 22, 2022

https://www.youtube.com/watch?v=kIU44FxjbFk
In-Storage Genomic Data Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasl, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,

"GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"


[Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]

GenStore: A High-Performance In-Store Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasl\textsuperscript{1} Jisung Park\textsuperscript{1} Harun Mustafa\textsuperscript{1} Jeremie Kim\textsuperscript{1} Ataberk Olgun\textsuperscript{1} Arvid Gollwitzer\textsuperscript{1} Damla Senol Cali\textsuperscript{2} Can Firtina\textsuperscript{1} Haiyu Mao\textsuperscript{1} Nour Almadhoun Alserr\textsuperscript{1} Rachata Ausavarungnirun\textsuperscript{3} Nandita Vijaykumar\textsuperscript{4} Mohammed Alser\textsuperscript{1} Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich \textsuperscript{2}Bionano Genomics \textsuperscript{3}KMUTNB \textsuperscript{4}University of Toronto