UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

The Road to Widely Deploying Processing-in-Memory

Challenges and Opportunities

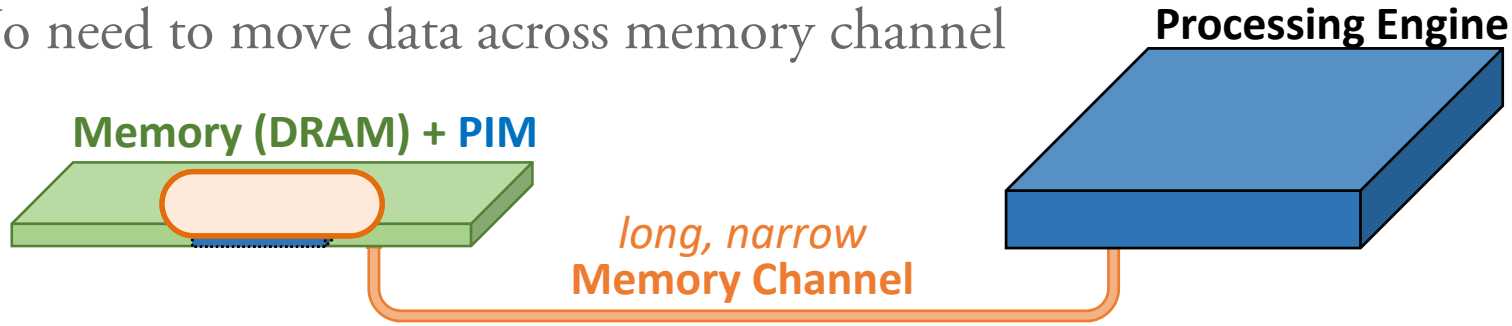
Saugata Ghose

https://ghose.cs.illinois.edu/

ISVLSI 2022 Special Session on In-Memory Processing • July 4, 2022

Can We Avoid Moving Data Around?

- **Processing-in-memory (PIM), or near-data processing (NDP)**
  - Add some compute capability to memory
  - No need to move data across memory channel

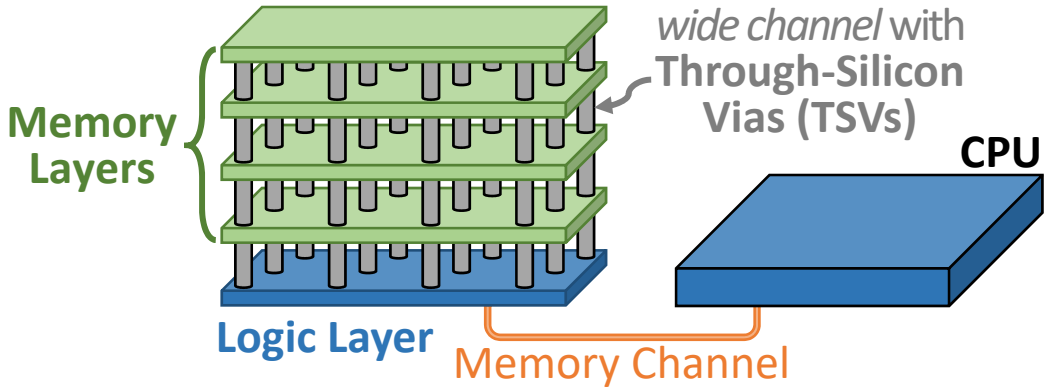


- **Recent innovations have made PIM hardware viable**
  - New **memory architectures and heterogeneous integration approaches** (e.g., 3D-stacked DRAM, silicon interposers)
  - New ways of **having memory devices interact with each other** (e.g., Boolean logic, multi-bit multiplies)

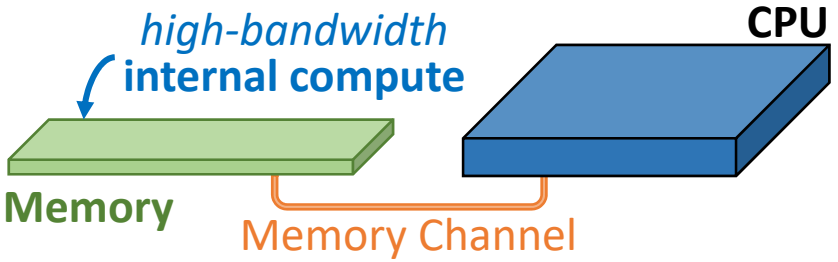
Many PIM-capable architectures and memory devices are being explored today

Real PIM Products Finally Exist!

- Processing-Near-Memory (PNM)
  - Discrete logic in or near the memory chip
  - Examples
    - » UPMEM PIM–DRAM Big Data Accelerator
    - » Samsung Aquabolt-XL (a.k.a. HBM-PIM, AxDIMM)
    - » SK hynix GDDR6-AiM



- Processing-Using-Memory (PUM)
  - Electrical interactions between memory cells w/ little additional logic
  - Examples
    - » Mythic Analog Matrix Processor
    - » IBM Hermes

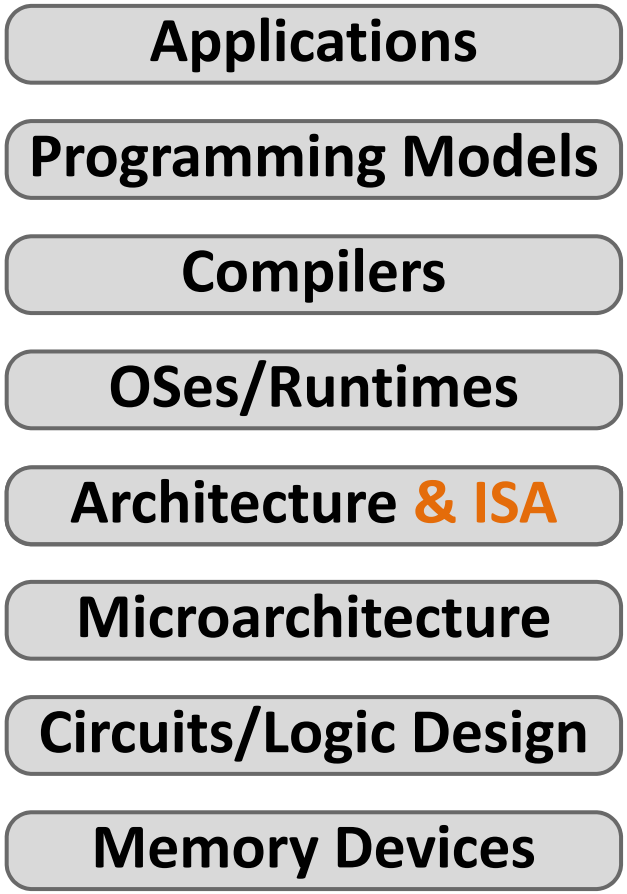


So Why Can't We Declare Victory Just Yet?

- Novel architectures and devices alone don't cut it
  - Widespread adoption happens only when **many people can make use of the architecture and underlying devices**
  - A new architecture may be very difficult to make use of in programs

We need to focus on completing the stack

- How do we translate all of our cutting-edge hardware ideas into **practical-to-manufacture architectures**?
- How can we get **end-to-end support to run a program** on a PIM architecture?



A Roadmap for Completing a Full Stack for PIM

- **Emerging/novel hardware requires cooperation between architects, circuit designers, and device physicists**
  - Need to consider two types of device limits
    - » Fundamental limits of materials
    - » Evolving limits of devices based on manufacturing technologies
  - **RACER** [Truong+ MICRO 2021, Truong+ JETCAS 2022]
    - » **Cross-stack PUM** that architects around device limits
    - » Fully-synthesized front end, peripherals, interconnects
- **We need a PIM-compatible ISA to build reusable software**
- **We need to tackle difficult problems in the software stack**
  - Data mapping
  - Virtual memory
  - Programming models

Introduction

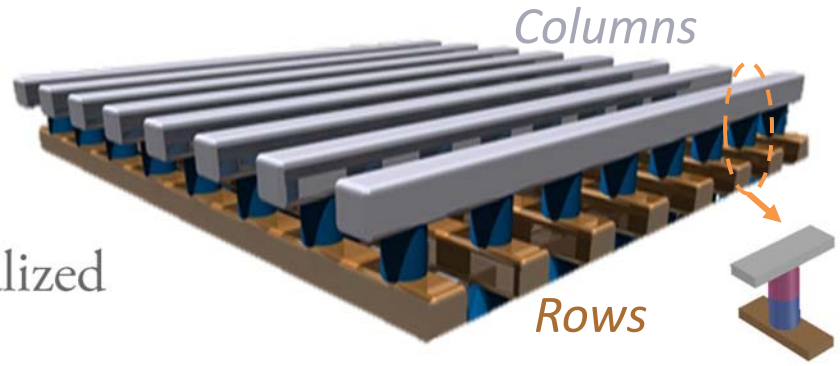
Cooperatively Designing Practical PIM Architectures

Using ISAs to Enable a Reusable Software Stack

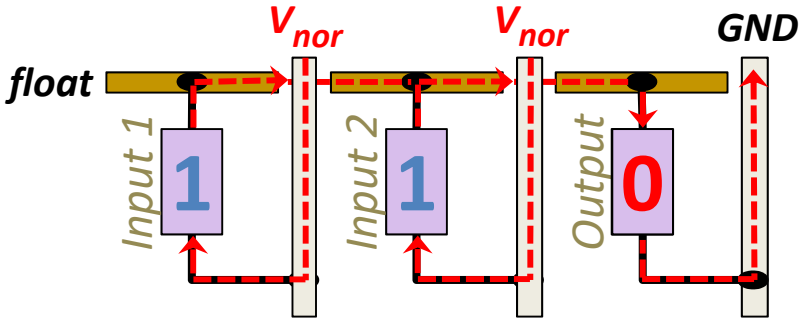
Closing Thoughts

Resistive RAM (ReRAM): A Potential PUM Candidate

- **Data stored as a resistance**
  - 1S1R: a **selector device** in series with a **resistive memory switch**
  - Resistive switch programmed using localized thermal events
  - Typically organized as a **crosspoint array**

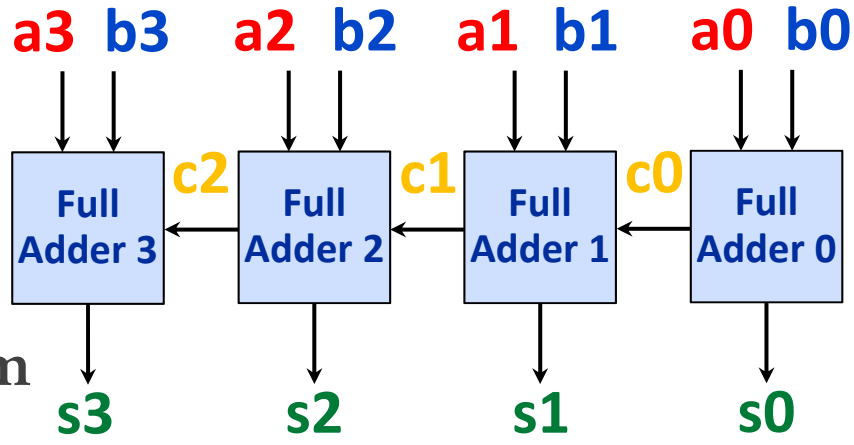


- **Analog interaction between multiple ReRAM cells can perform meaningful computation**
  - Multi-level ReRAM cells can perform dot product, low-precision multiply
  - **Single-level ReRAM cells can perform NOR**



Bit-Serial Operations Work Well for ReRAM Arrays

- Bit-serial operations
  - Perform operations one bit at a time
  - Example: ripple-carry add



- NOR-capable ReRAM can perform many bit-serial functions
  - Addition/subtraction
  - Content search (like a content-addressable memory, or CAM)

- Bit-serial operations incur long latencies

To compensate for long latencies, ReRAM can compute on whole columns of data at once to exploit data-level parallelism

Issues with Processing-Using-ReRAM

- The larger the crossbar size, the bigger the throughput for whole-column operations

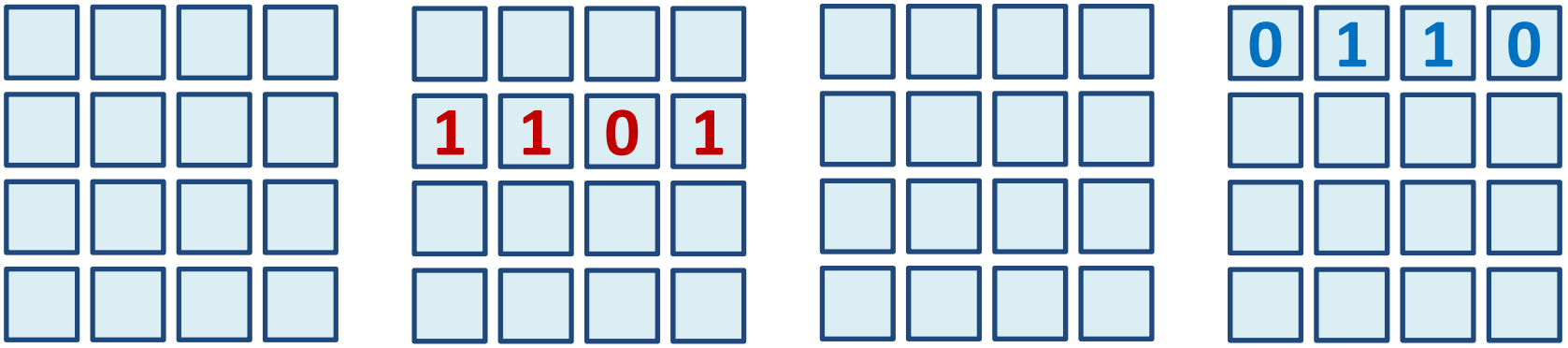
- Whole-column operations limit the crossbar size
  - Each extra cell in a column adds current → grows **proportionally** to crossbar size
  - Limited by the current carrying capacity of a wire
  - Large currents **permanently damage** the metal wires *(see MICRO 2021 paper for analysis)*

Whole-column operations are realistically possible only when column length < 200 cells!

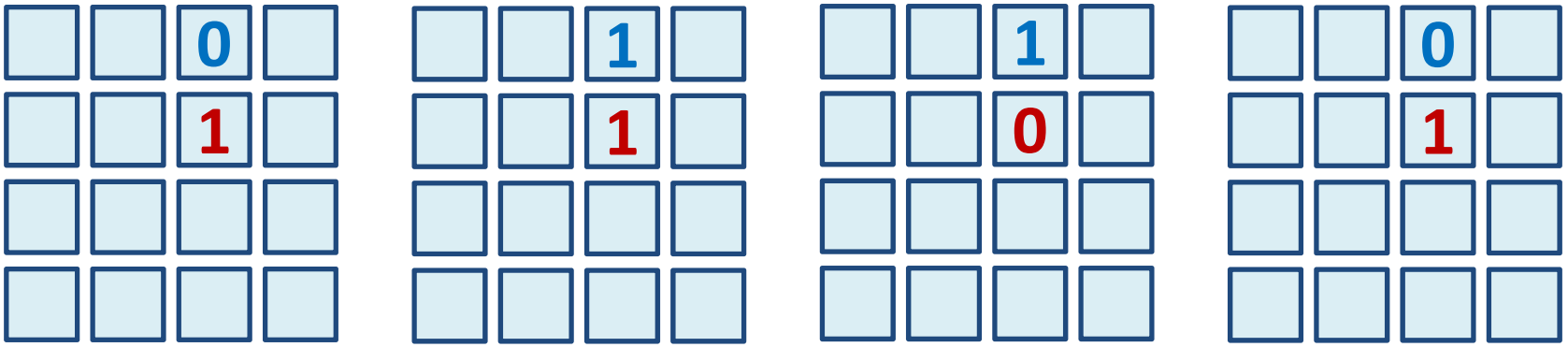
How can small tiles deliver high throughput at low overhead?

RACER: Optimizing PUM for Small Memory Tiles

- State-of-the-art processing-using-memory architectures keep all of a word in a single tile

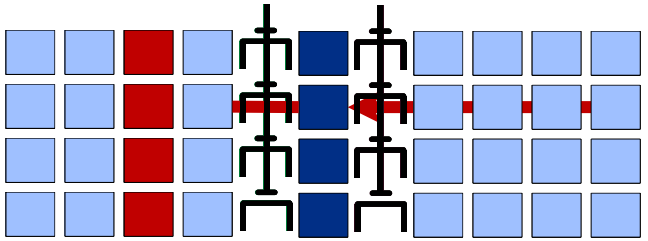


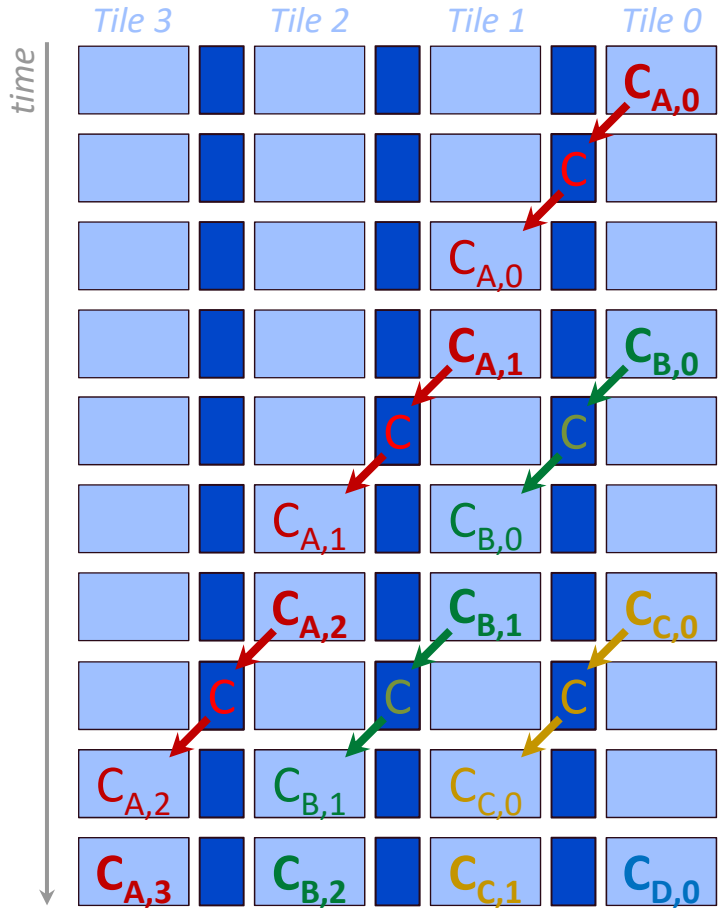
- In RACER, we distribute each bit of a word to a different tile



RACER: Dealing with Bit-Serial Ops Across Tiles

- We add 1x64 ReRAM **column buffers**
  - Enables tile-to-tile communication
  - Connects to an adjacent tile using pass gates





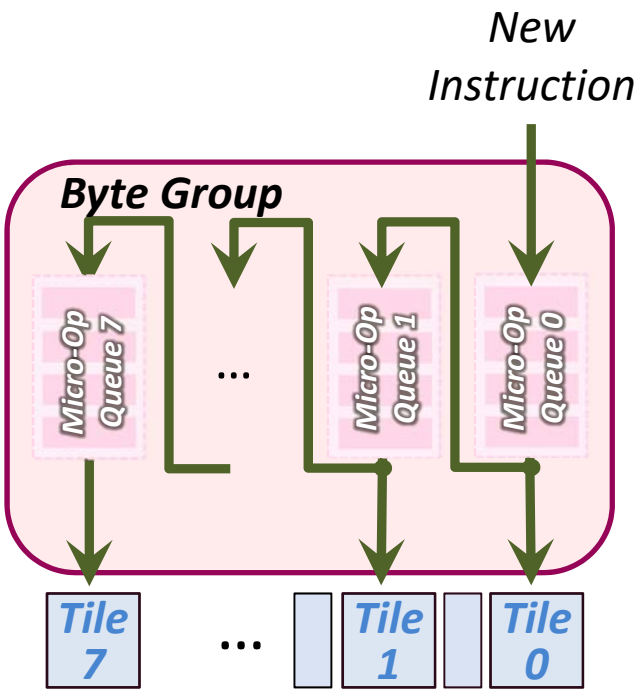
- We enable a new technique that we call **bit-pipelining**
  - Treat each tile as a pipeline stage
  - With *t* tiles, we can operate on *t* columns of words at once

Byte Group: Core Control Circuitry for Bit-Pipelining

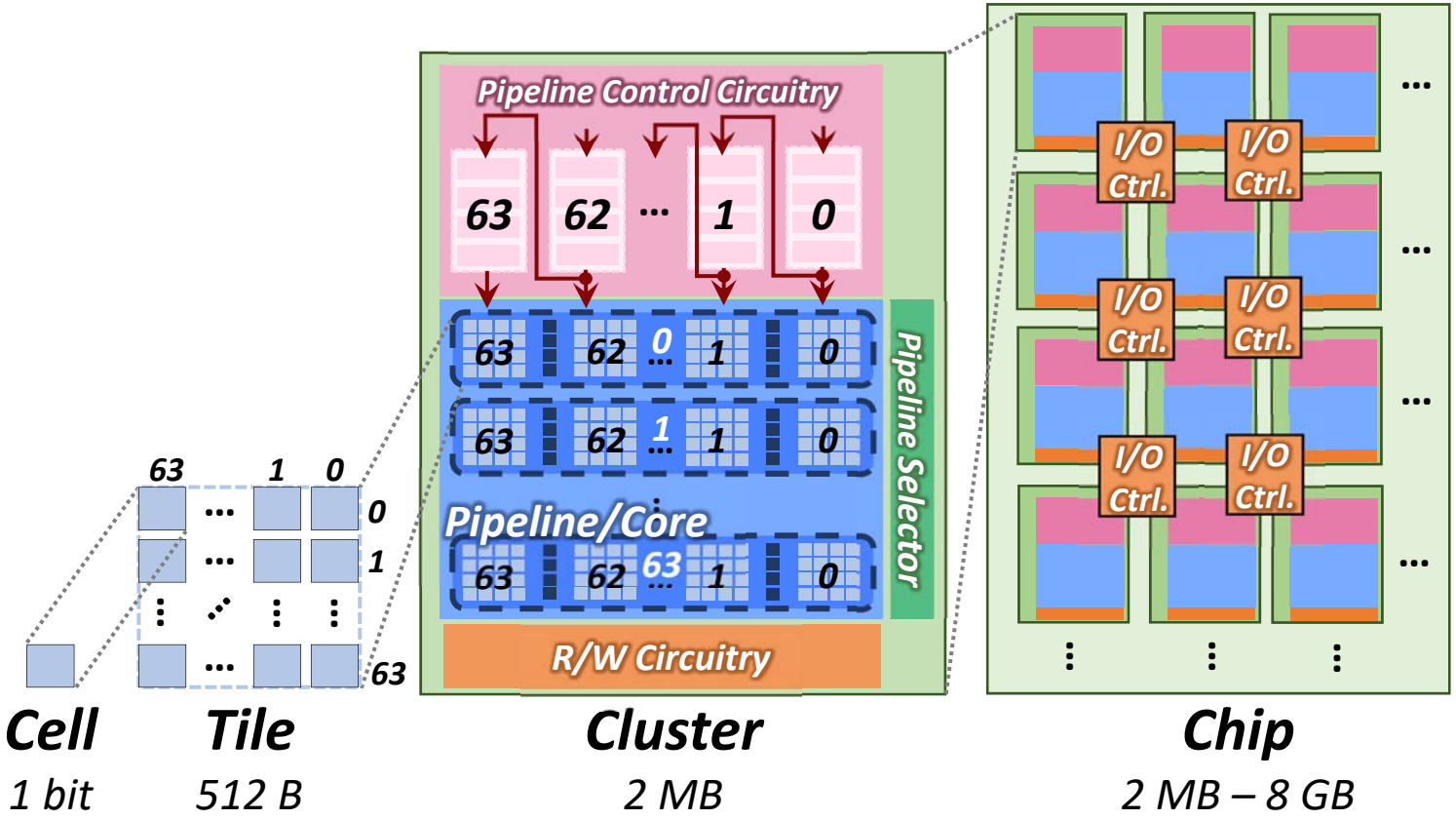
- Each bit (i.e., each tile) **repeats the same exact operations**

- NOR instructions (micro-ops) are stored in **micro-op queues**
  - Each tile has a dedicated queue
  - Queue *i* sends its micro-ops to Queue *i*+1

- Enables efficient support of 8-/16-/32-/64-bit operands



RACER Cluster: A Scalable Module for PUM

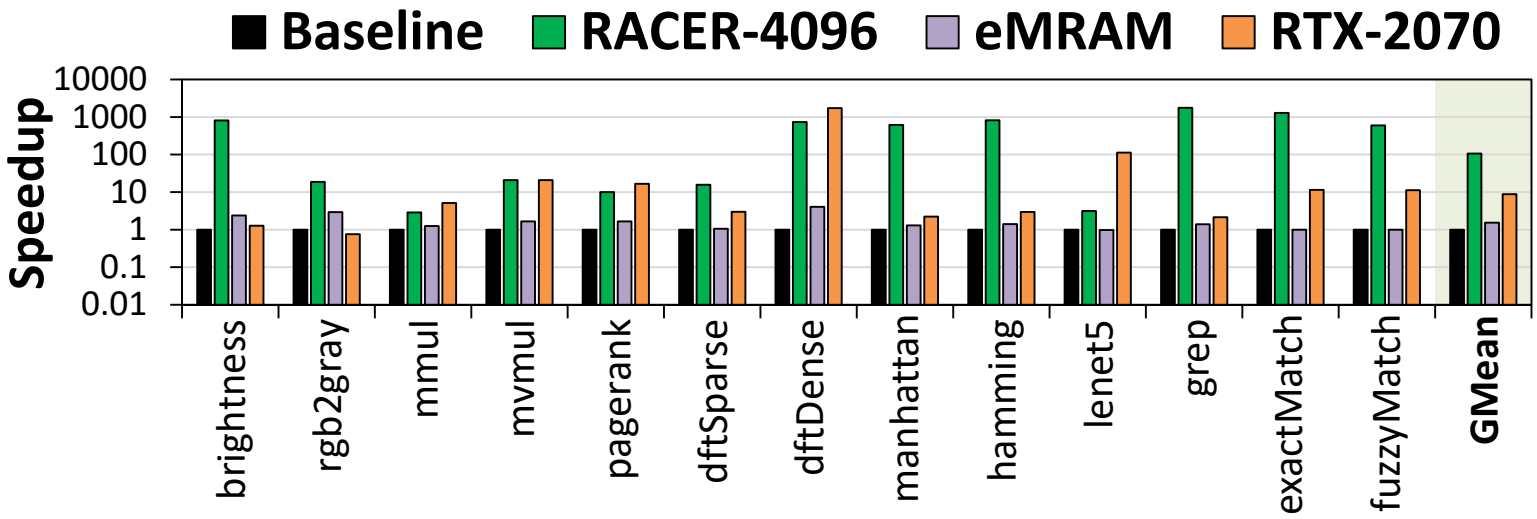


- Cluster: group of pipelines
  - Clusters operate independently
  - Only one set of control queues and peripheral circuits per cluster
- Chip can contain however few/many clusters as needed

Methodology

- **Iso-area comparisons to four state-of-the-art platforms**
  - **Baseline:** 16-core Xeon 8253 CPU + 8GB off-chip DRAM
  - **eMRAM:** 16-core Xeon 8253 CPU + 8GB on-chip MRAM
  - **RTX-2070:** GeForce RTX 2070 GPU
  - DC: Duality Cache, a compute-in-SRAM architecture
- **We model RACER at multiple levels of the stack**
  - Device-level ReRAM characteristics modeled using VerilogA with in-house device measurements
  - Control and peripheral circuits synthesized using FreePDK 15 nm
  - RACER ISA microbenchmarks executed using in-house simulator
  - Baseline modeled using MARSSx86 + DRAMSim2 + McPAT
- **Full paper:** https://ghose.cs.illinois.edu/papers/21micro_racer.pdf
- **Simulation framework open-sourced:** https://doi.org/10.5281/zenodo.5495803

RACER Increases Performance vs. CPU/GPU



107× speedup vs. CPU

thanks to RACER's tile-/pipeline-/cluster-level parallelism

71× speedup vs. eMRAM

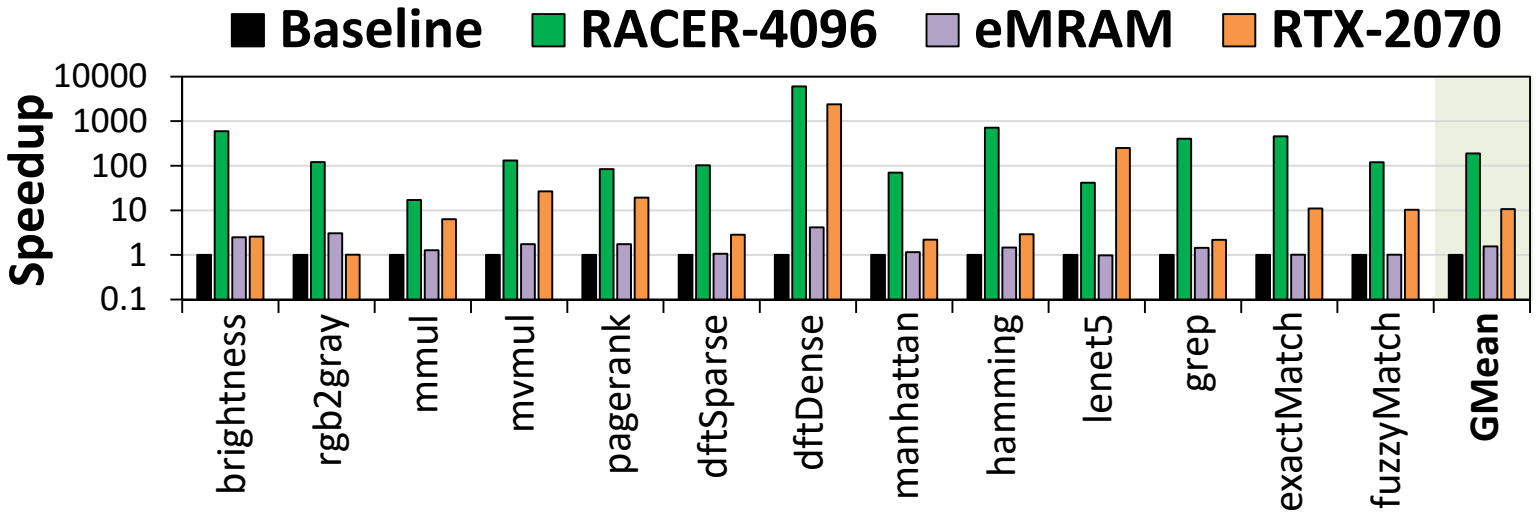
as embedded memory does not reduce frequent data movement

12× speedup vs. GPU

7× speedup vs. DC *(not shown)*

thanks to RACER's *in-situ* computation and tile-/pipeline-/cluster-level parallelism

RACER Significantly Reduces Energy



189× savings vs. CPU

thanks to RACER's *in-situ* computation and fast low-power circuitry

94× savings vs. eMRAM

as embedded memory mostly reduces only the energy used by off-chip network

17× savings vs. GPU

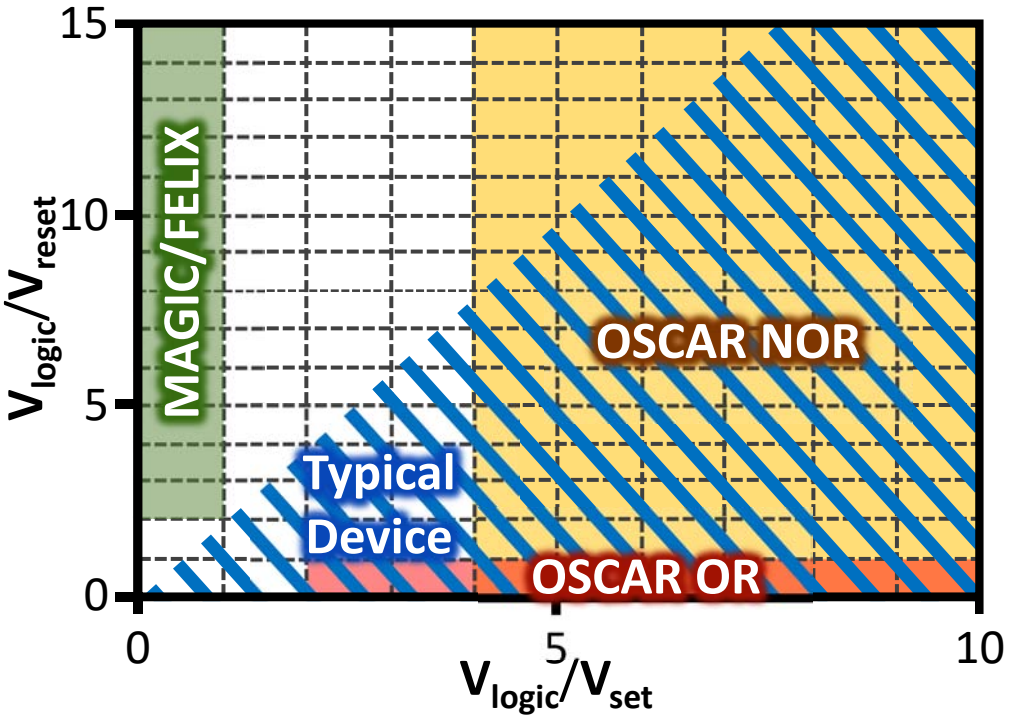
1.3× savings vs. DC *(not shown)*

5× savings vs. DC for applications that trigger frequent data swapping

OK Fine – We'll Co-Design the Hardware. Now What?

- **We need architects to serve as a bridge**
  - Be **better informed** about circuit design and emerging devices (through collaboration, not by reading tables in papers!)
  - **Drive the requirements of future devices**

- **My anecdotal example with ReRAM**
  - Many device physicists focusing on lower voltages/currents
  - What we really need is
    - » **More realistic switching tolerances** for PUM-capable devices [Truong+ JETCAS 2022]
    - » Significantly improved lifetimes (10^{14}+ writes, currently at 10^{12})



Introduction

Cooperatively Designing Practical PIM Architectures

Using ISAs to Enable a Reusable Software Stack

Closing Thoughts

What's Stopping Us From Designing PIM Software?

- We don't really know which hardware design(s) will win
  - Processing-near-memory or PUM?
  - What underlying memory devices?
  - What primitives?
  - What microarchitecture?
- Difficult to make software with so many unknowns
- Wait! We've solved this before for conventional computers…

We need to think about general abstractions
that can easily be ported to a wide range of microarchitectures

It's now time for a PIM ISA!

An Architectural Abstraction for RACER

- **RACER core: pipeline w/ 32 kB of data**
  - Corresponds to 64 tiles
  - Each core has local access to data from 512 cores
  - A global network gives each core access to data in the entire chip (up to 8 GB)
  - **Vectorized ISA** for easy programmability

Table 2: RACER ISA (†: non-bit-pipelined operations).

Op.	Description/Notes	Op.	Description/Notes
Arithmetic Operations			
ADD	Two's complement add	ABS	Absolute value
SUB	Two's complement subtract	MUX	Multiplex (i.e., choose)
POPC	Population count	RELU	Rectified linear unit
CMPEQ	Check equality	LSHIFT	Left shift by 1
FUZZY	Fuzzy search	RSHIFT	Right shift by 1
MUL†	Multiply (only 8-/16-/32-bit)	SQRT†	CORDIC square root
MAC†	Multiply–accumulate	SIN†	CORDIC sine
DIV†	Division (returns quotient & remainder)	COS†	CORDIC cosine
MAX	Searches for the maximum number	EXP†	CORDIC exponent
MIN	Searches for the minimum number	CAS	Compare and swap
Boolean Operations			
NOR	Bitwise NOR	OR	Bitwise OR
NAND	Bitwise NAND	AND	Bitwise AND
NOT	Bitwise NOT	XOR	Bitwise XOR
Data Transfer Operations			
MOV	<**MOV** *buff*[*dst*] = *buff*[*src*]> Moves data stored in buffers of core *src* to buffers of core *dst*	SHIFT	<**SHIFT** *stride*> Parallel data shift *dst* = *src* + *stride*
Configuration Operations			
SET	<**SET** *start*, *stop*, *stride*> Turns on RACER core *i* for *i* ∈ *range*(*start*, *stop*, *stride*)	UNSET	Turns off all RACER cores that are active

- **This of course isn't the only way**
  - PEI: specialized atomic instructions [Ahn+ ISCA 2015]
  - SIMDRAM: customizable μPrograms [Hajinazar+ ASPLOS 2021]
  - CAPE: RISC-V Vector Extension instructions [Caminal+ HPCA 2021]

Programmers Need More Than Just a PIM ISA

- Once PIM hardware exists, programmers must be able to use it
  - Tough sell: force them to **learn a new programming model**
  - Path to broad adoption: **adapt PIM to existing models**
- Easiest for programmers: **Let's treat PIM like another core**
- What do programmers expect when they use multiple cores?

Support for Multithreading

Access to Shared Memory

Virtual Memory

Data Partitioning Support

Cache Coherence

[Boroumand+ ISCA 2019]

We Need to Think About Integration

- **How do we write programs?**
  - Provide **friendly programming models**
  - Develop **compilers** that
    - » Automatically identify opportunities for PIM
    - » Generate PIM-compatible binaries
  - Provide **support to partition data** across PIM cores and/or memory arrays

- **How do we use PIM in the context of a whole system?**
  - Develop an **OS or runtime** that can manage PIM execution
  - Enable support for **multitenancy**
    - » Spatial multiplexing support
    - » Context switching
    - » Memory virtualization & protection

Introduction

Cooperatively Designing Practical PIM Architectures

Using ISAs to Enable a Reusable Software Stack

Closing Thoughts

Conclusion

- **Emerging/novel hardware requires cooperation between architects, circuit designers, and device physicists**
  - Example approach: **RACER** [Truong+ MICRO 2021, Truong+ JETCAS 2022]
    - » **Cross-stack PUM** that architects around device limits
    - » Fully-synthesized front end, peripherals, interconnects
  - Architects/circuit designers need to be informed of & drive device research
  - We need more practical PIM-capable devices, longer lifetimes
- **It's time to build a PIM ISA to enable reusable software**
- **We need to start tackling difficult problems in the software stack**
  - Provide/enable friendly programming models
  - Develop support for compilation and data mapping
  - Integrate PIM hardware into OSes/runtimes
  - Provide support for multitenancy

Thanks to My Collaborators

- **ARCANA Research Group:**
  https://arcana.cs.illinois.edu/

  - Minh S. Q. Truong
  - Amirali Boroumand
  - Damla Senol Cali
  - Eric Chen
  - Deanyone Su
  - Alex Glass
  - Ali Hoffmann

- **SAFARI Research Group:**
  https://safari.ethz.ch/

  - Onur Mutlu
  - Juan Gómez-Luna
  - Geraldo F. Oliveira
  - Rachata Ausavarungnirun
  - … and many others

- **CMU Data Storage Systems Center**

  - James A. Bain
  - L. Richard Carley
  - Liting Shen

- **Industrial Collaborators**
  - Intel
  - Google
  - Samsung

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

The Road to Widely Deploying Processing-in-Memory

Challenges and Opportunities

Saugata Ghose

https://ghose.cs.illinois.edu/

ISVLSI 2022 Special Session on In-Memory Processing • July 4, 2022