P&S Processing-in-Memory

Real-World Processing-in-Memory Architectures:
Samsung HBM-PIM Architecture

Dr. Juan Gómez Luna

Prof. Onur Mutlu

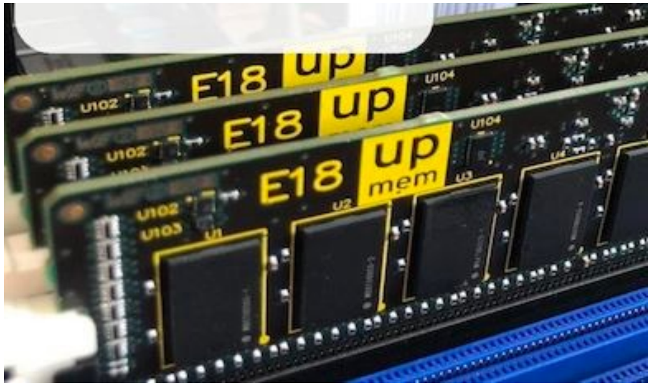
ETH Zürich

Spring 2022

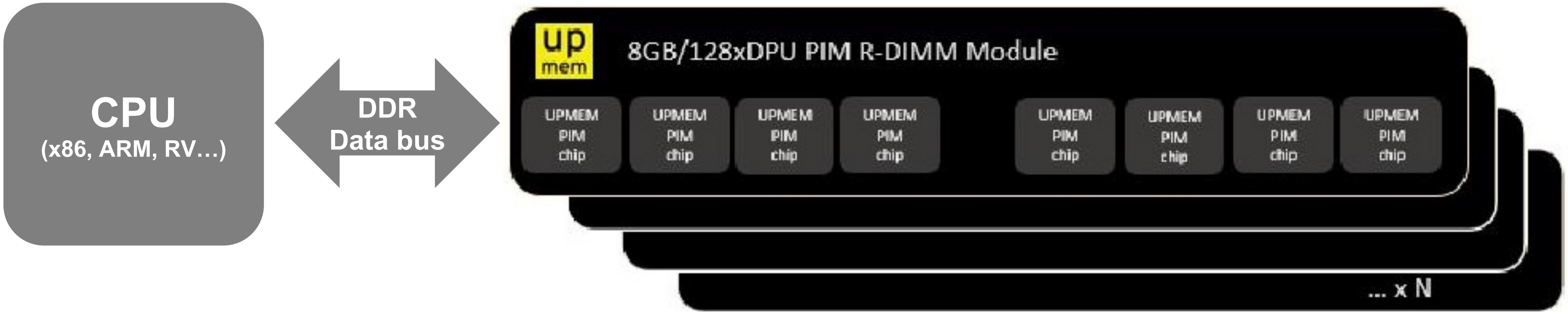
31 March 2022

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



CPU
(x86, ARM, RV…)

DDR
Data bus

up mem
8GB/128xDPU PIM R-DIMM Module

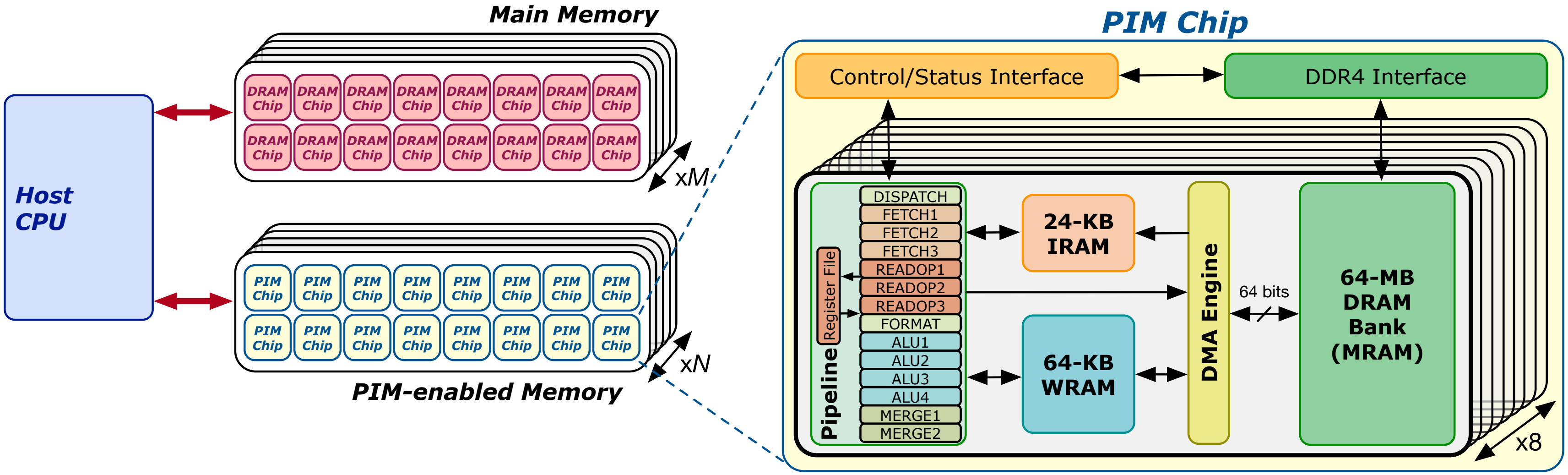
UPMEM PIM chip | UPMEM PIM chip | UPMEM PIM chip | UPMEM PIM chip | UPMEM PIM chip | UPMEM PIM chip | UPMEM PIM chip | UPMEM PIM chip

… x N

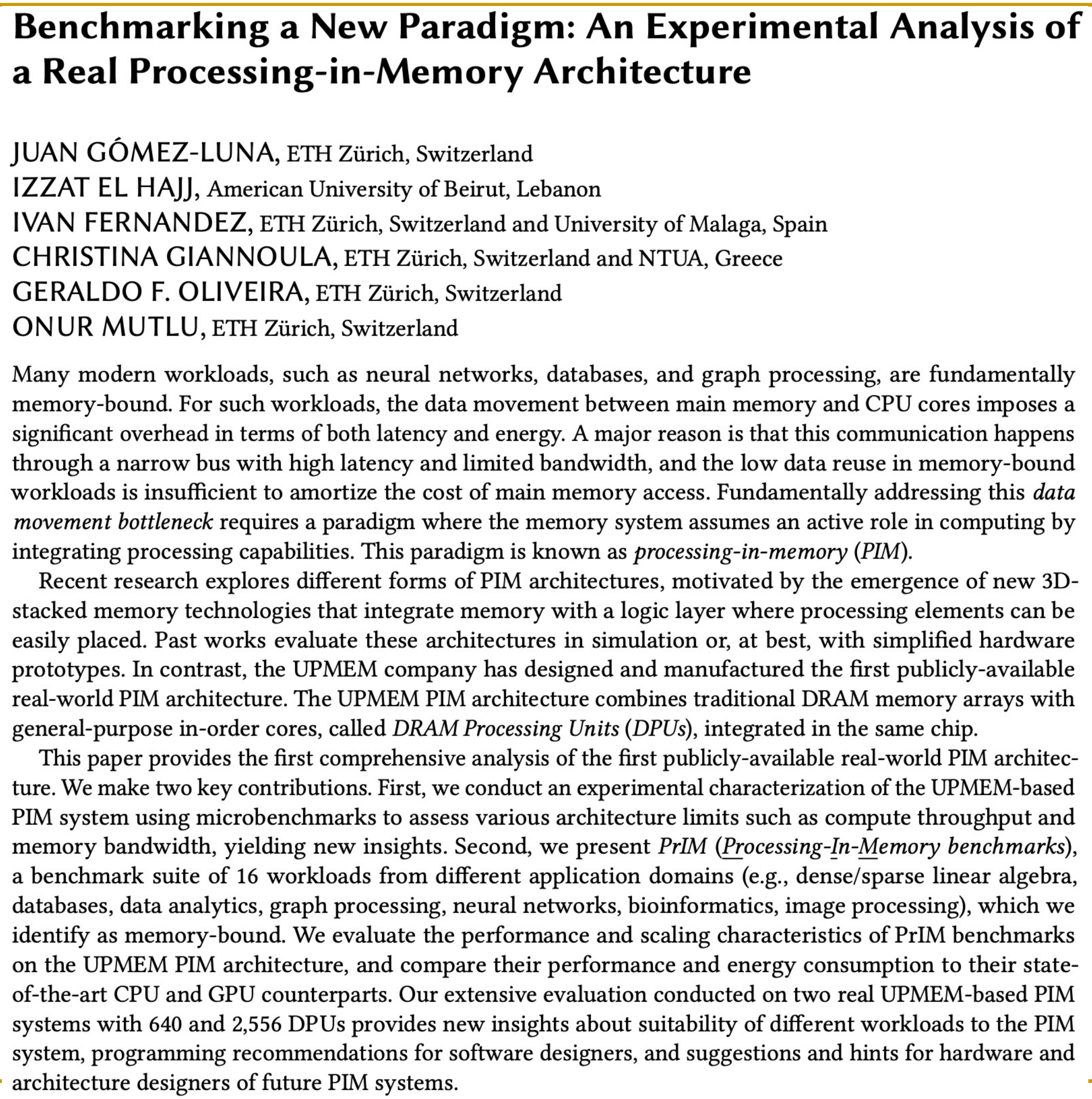
https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem
https://www.upmem.com/video-upmem-presenting-its-true-processing-in-memory-solution-hot-chips-2019/

Recall: UPMEM PIM System Organization

- A UPMEM DIMM contains 8 or 16 chips
  - Thus, 1 or 2 ranks of 8 chips each
- Inside each PIM chip there are:
  - 8 64MB banks per chip: Main RAM (MRAM) banks
  - 8 DRAM Processing Units (DPUs) in each chip, 64 DPUs per rank



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

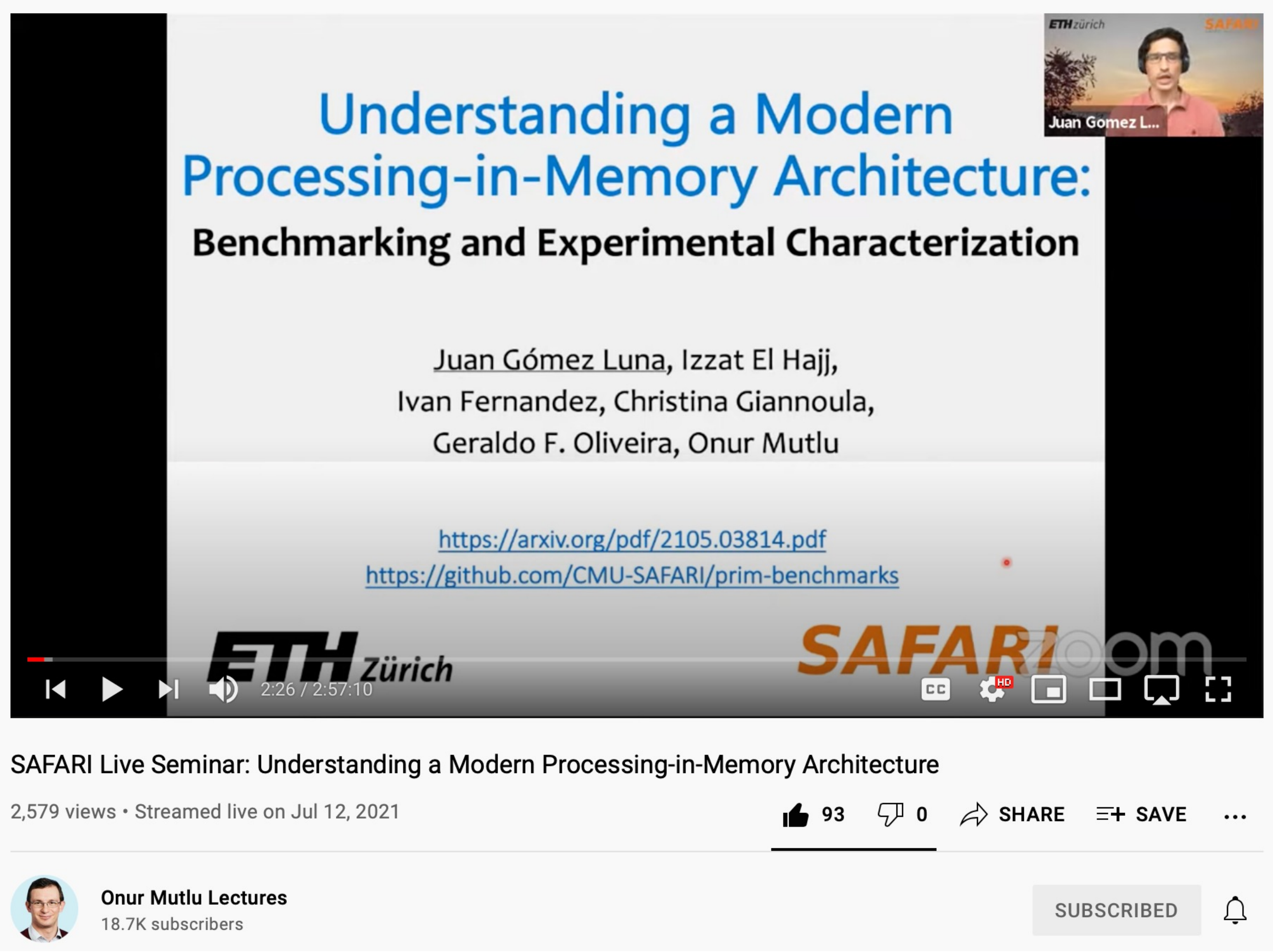
Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens 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.

https://arxiv.org/pdf/2105.03814.pdf

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://arxiv.org/pdf/2105.03814.pdf
https://github.com/CMU-SAFARI/prim-benchmarks

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

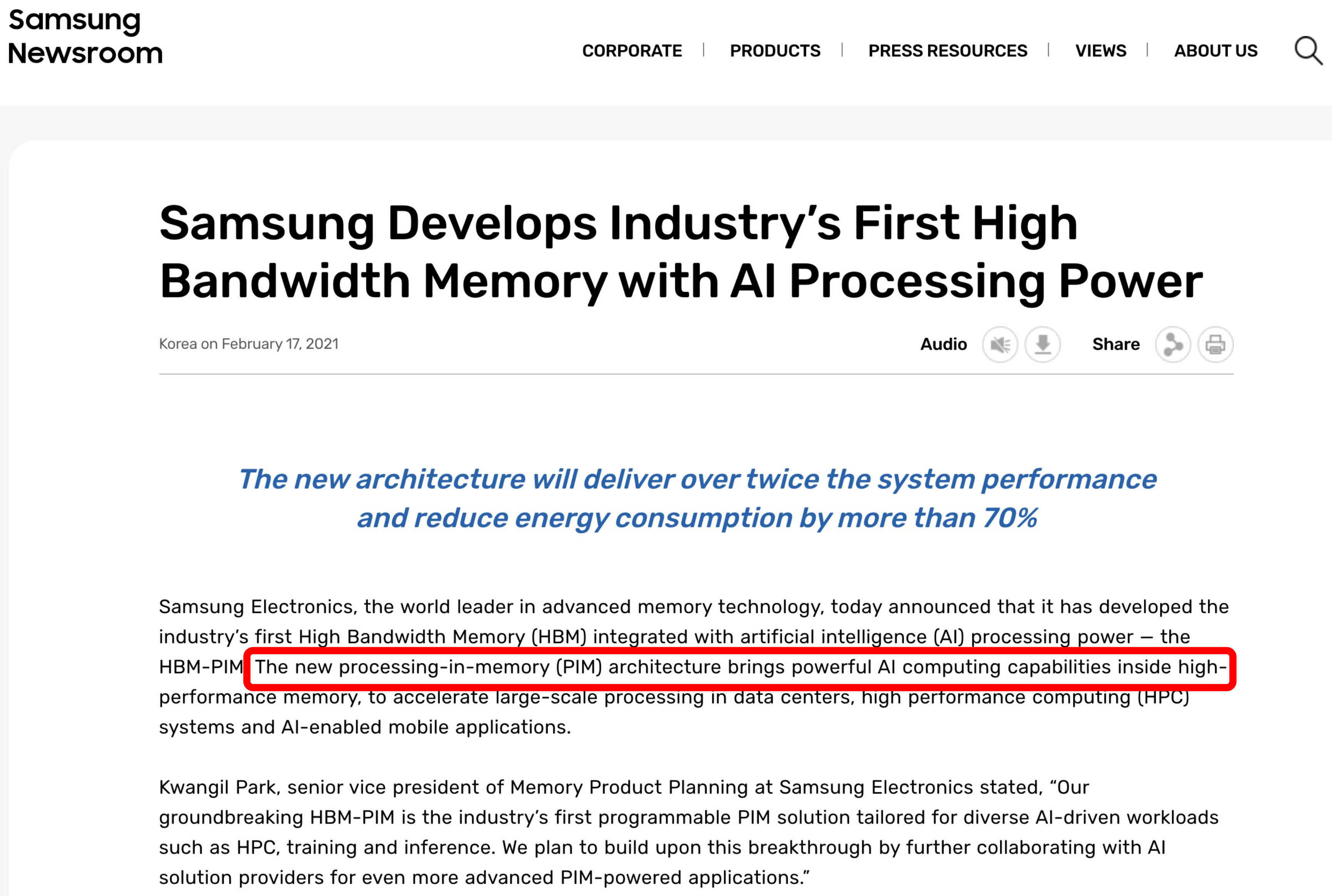
2,579 views • Streamed live on Jul 12, 2021

Onur Mutlu Lectures
18.7K subscribers

https://www.youtube.com/watch?v=D8Hjy2iU9l4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9

Samsung HBM-PIM, a.k.a. FIMDRAM

Samsung Function-in-Memory DRAM (2021)



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

Korea on February 17, 2021

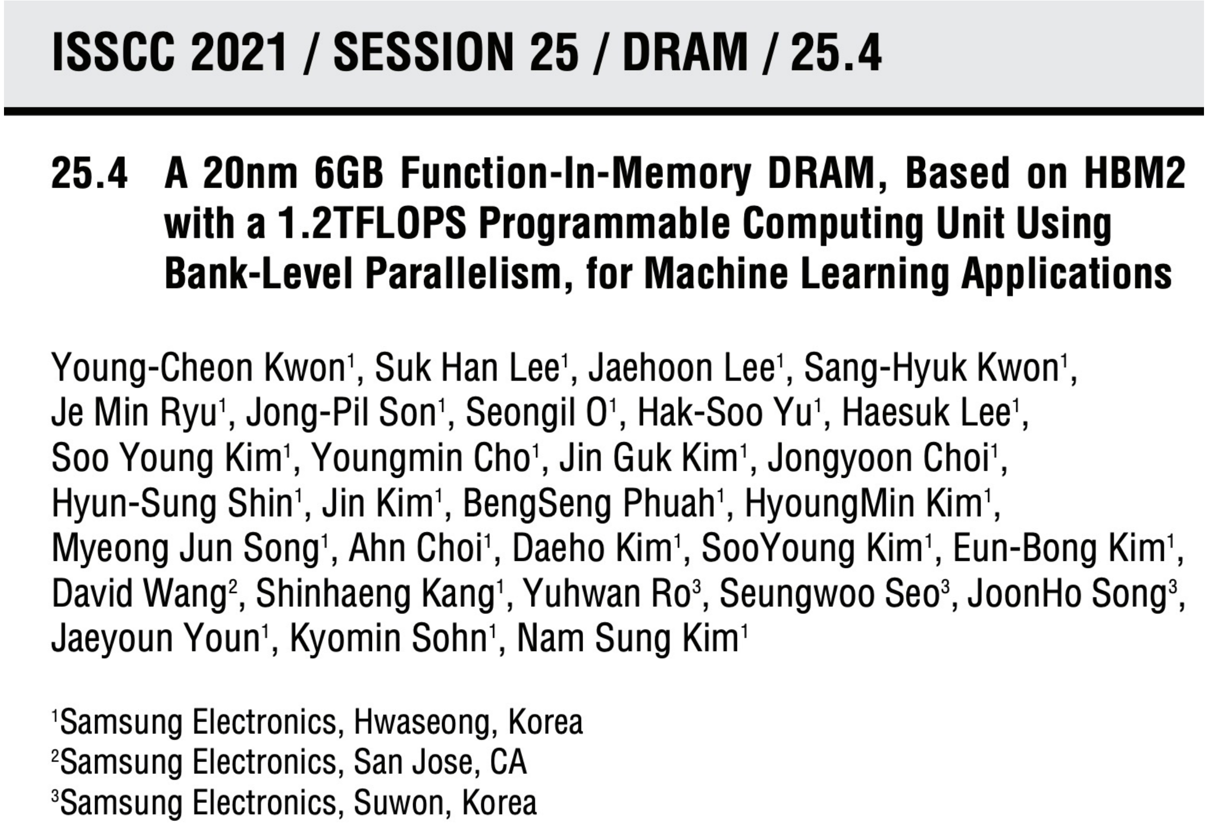
The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

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."

https://news.samsung.com/global/samsung-develops-industrys-first-high-bandwidth-memory-with-ai-processing-power

Function-in-Memory DRAM (ISSCC 2021)



ISSCC 2021 / SESSION 25 / DRAM / 25.4

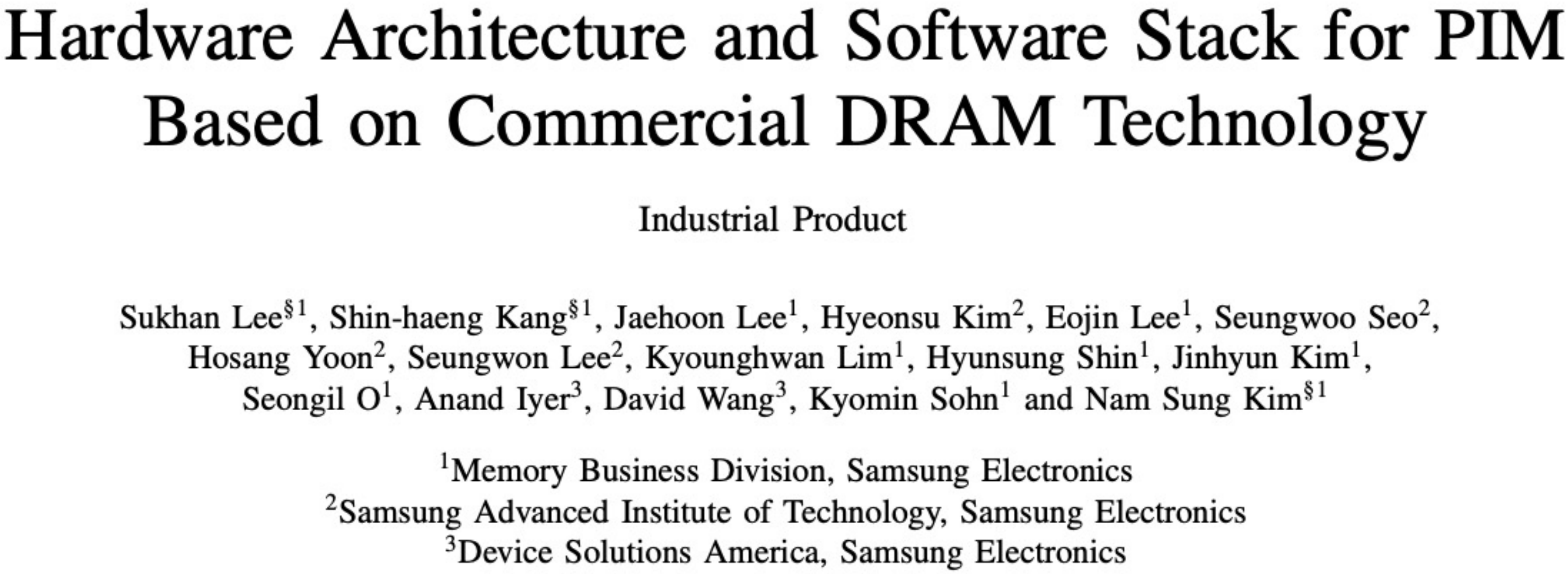
25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

https://doi.org/10.1109/ISSCC42613.2021.9365862

PIM based on Commercial DRAM (ISCA 2021)



Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology

Industrial Product

Sukhan Lee[§1], Shin-haeng Kang[§1], Jaehoon Lee[1], Hyeonsu Kim[2], Eojin Lee[1], Seungwoo Seo[2],
Hosang Yoon[2], Seungwon Lee[2], Kyounghwan Lim[1], Hyunsung Shin[1], Jinhyun Kim[1],
Seongil O[1], Anand Iyer[3], David Wang[3], Kyomin Sohn[1] and Nam Sung Kim[§1]

[1]Memory Business Division, Samsung Electronics
[2]Samsung Advanced Institute of Technology, Samsung Electronics
[3]Device Solutions America, Samsung Electronics

https://doi.org/10.1109/ISCA52012.2021.00013

Aquabolt-XL: Samsung HBM2-PIM (HCS 2021)

Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond

Jin Hyun Kim, Shin-haeng Kang, Sukhan Lee, Hyeonsu Kim, Woongjae Song, Yuhwan Ro, Seungwon Lee, David Wang, Hyunsung Shin, Bengseng Phuah, Jihyun Choi, Jinin So, YeonGon Cho, JoonHo Song, Jangseok Choi, Jeonghyeon Cho, Kyomin Sohn, Youngsoo Sohn, Kwangil Park, and Nam Sung Kim

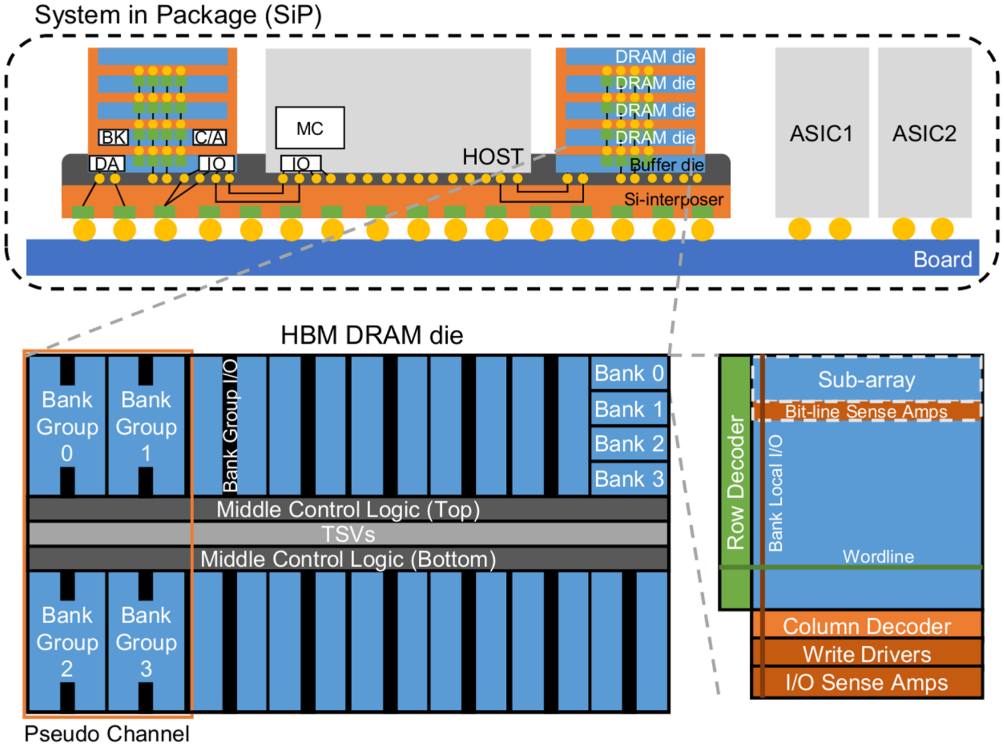
Samsung Electronics

https://doi.org/10.1109/HCS52781.2021.9567191

Background: High Bandwidth Memory (HBM)

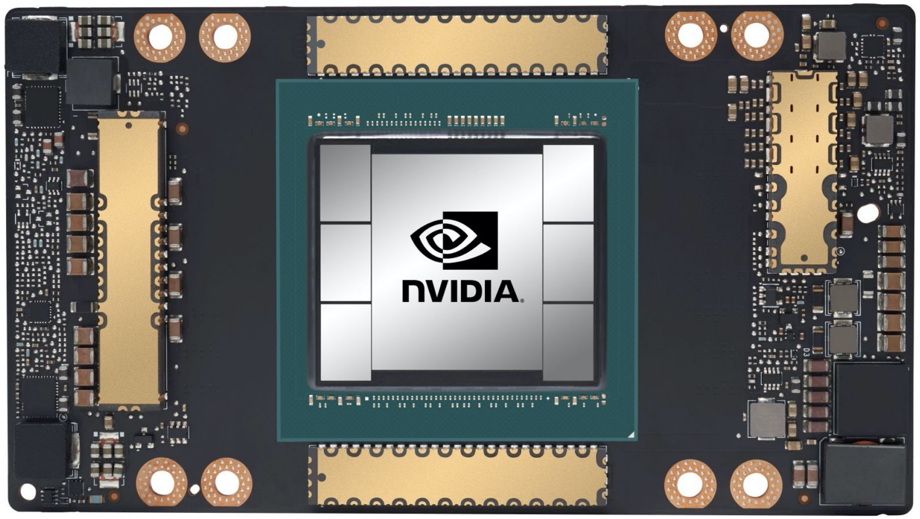
- HBM stacks DRAM layers and a buffer layer
  - The buffer layer contains I/O circuitry, self-test, test/debug
- DRAM layers and buffer layer communicate using Through Silicon Vias (TSVs)

- The buffer layer is connected to a host processor via a silicon interposer
- 1 HBM2 die comprises 4 pseudo channels (pCHs) each with 4 bank groups
  - An access transfers a 256-bit data block over 4 64-bit bursts over one pCH



Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

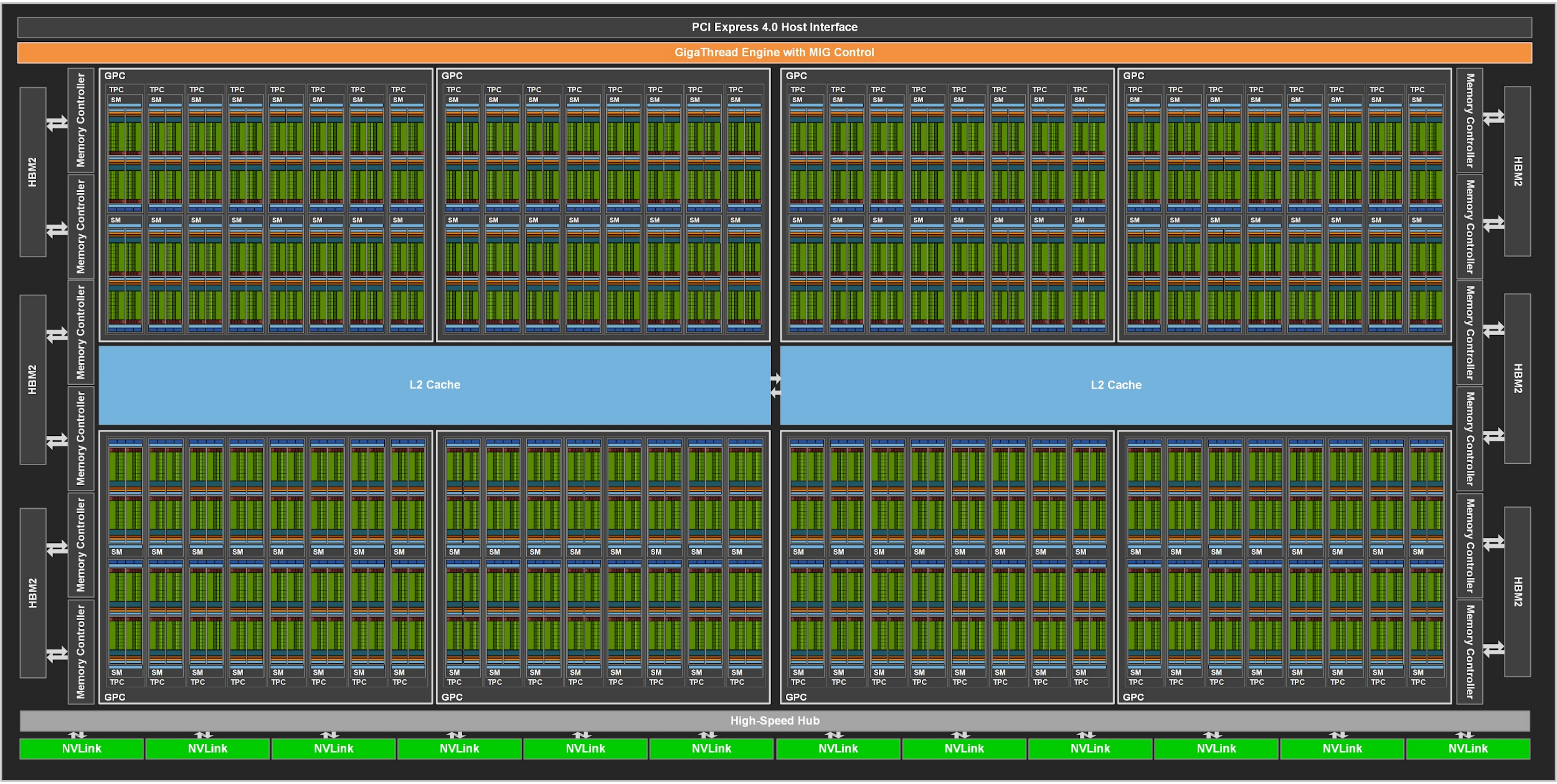
NVIDIA A100 GPU



- NVIDIA-speak:
  - 6912 stream processors
  - “SIMT execution”
- Generic speak:
  - 108 cores
  - 64 SIMD functional units per core
  - Tensor cores for Machine Learning
    - Support for sparsity
    - New floating point data type (TF32)

https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

NVIDIA A100 Block Diagram



https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

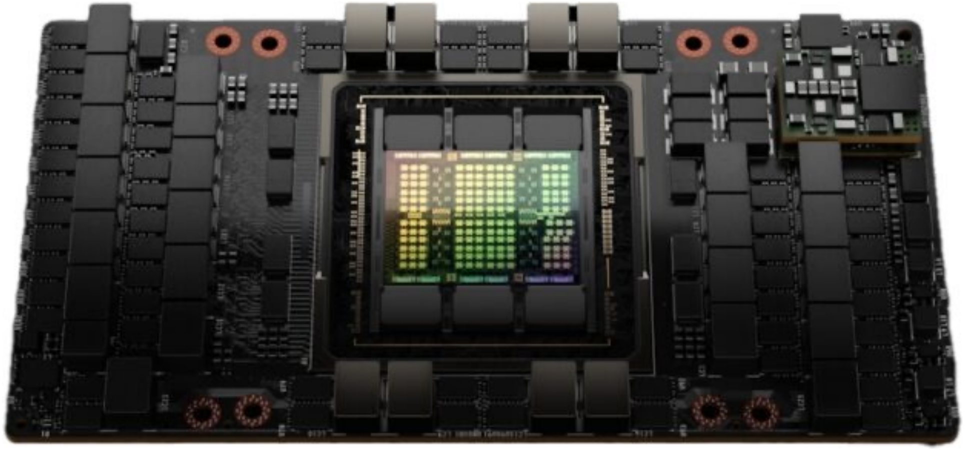
108 cores on the A100

(Up to 128 cores in the full-blown chip)

40MB L2 cache

NVIDIA H100 GPU

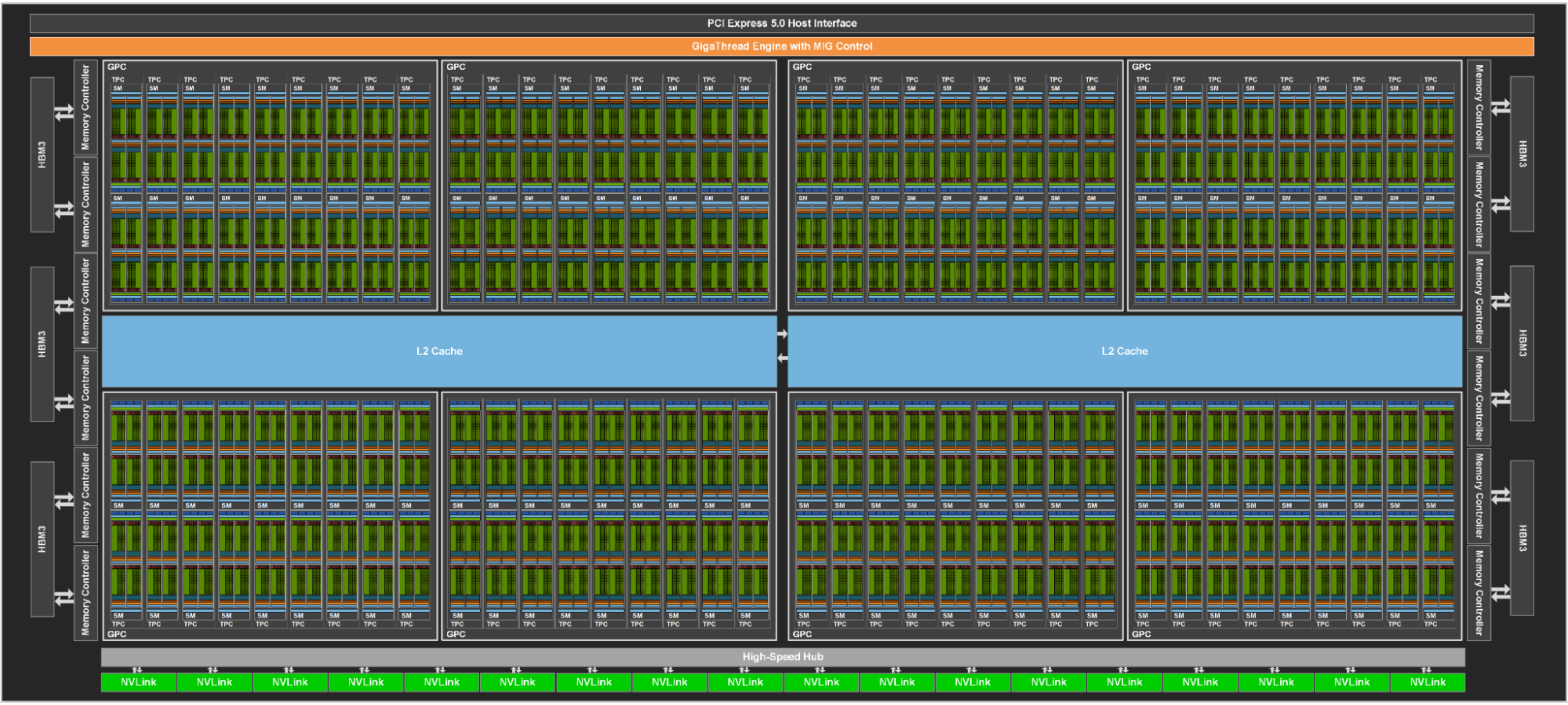
- NVIDIA-speak:
  - 14592 stream processors
  - "SIMT execution"



- Generic speak:
  - 144 cores
  - 64 SIMD functional units per core
  - Tensor cores for Machine Learning
    - New 8-bit floating point formats

https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

NVIDIA H100 Block Diagram

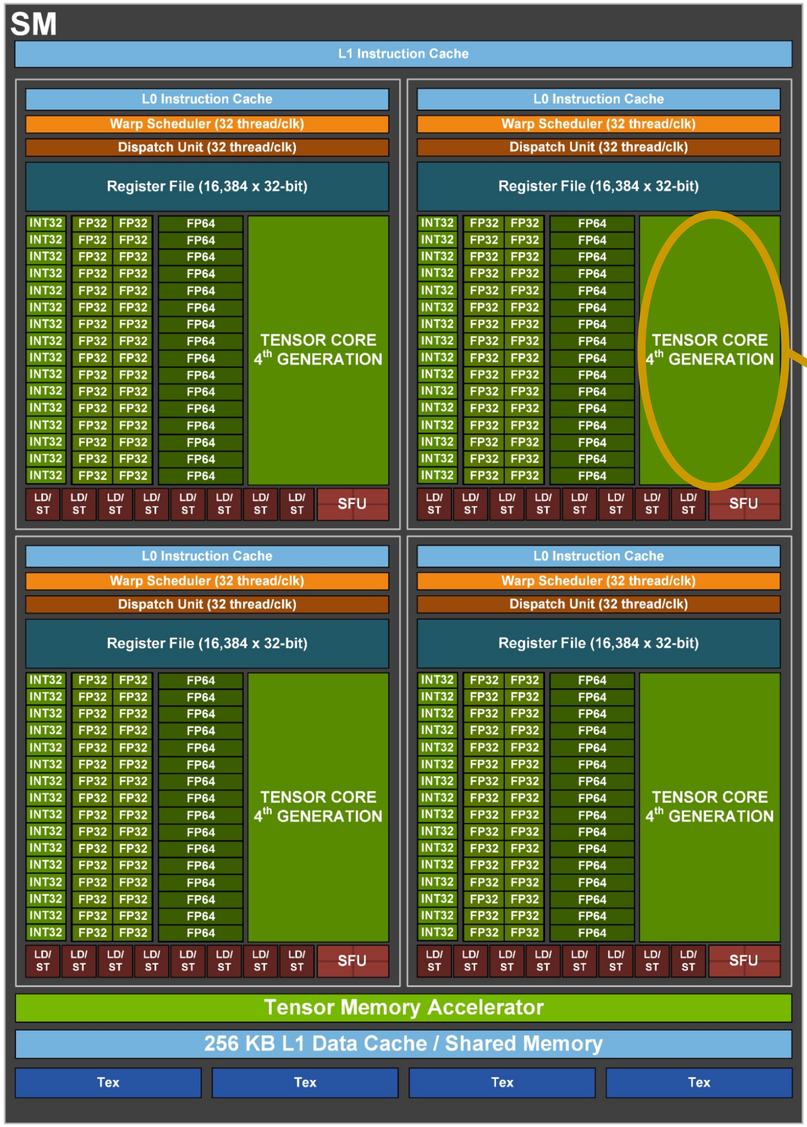


https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

144 cores on the full GH100

60MB L2 cache

NVIDIA H100 Core

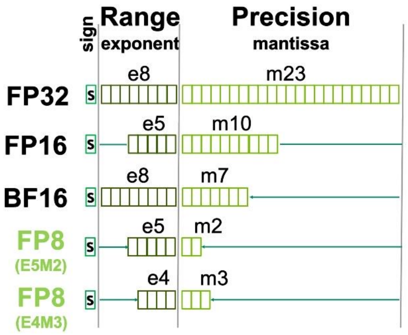


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

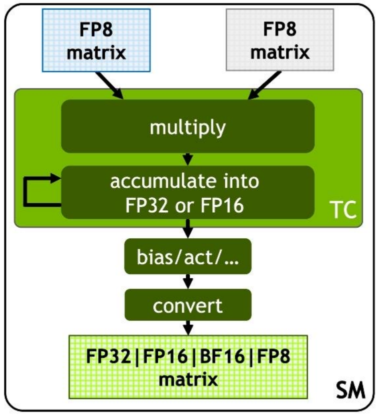
48 TFLOPS Single Precision*

24 TFLOPS Double Precision*

800 TFLOPS (FP16, Tensor Cores)*



Allocate 1 bit to either range or precision

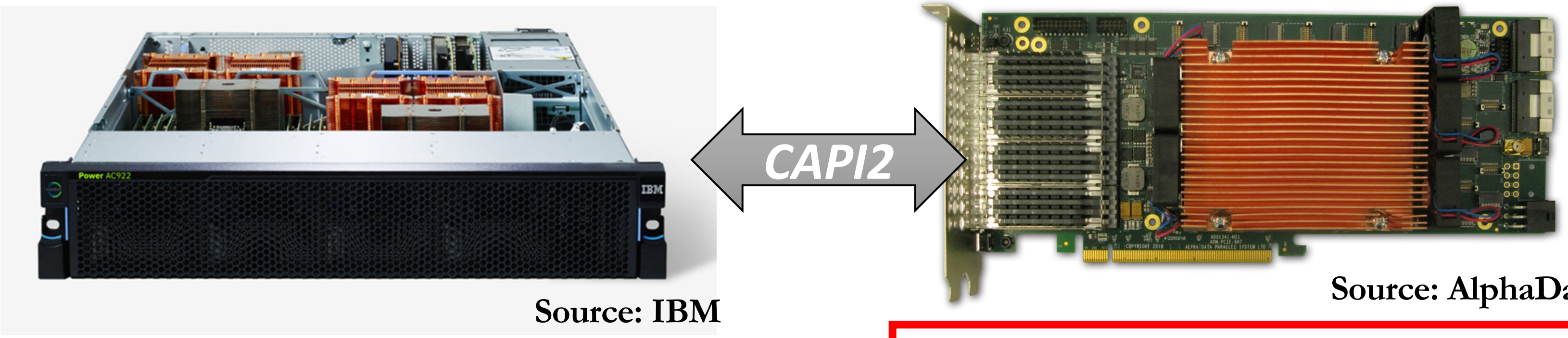


Support for multiple accumulator and output types

https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

* Preliminary performance estimates

Heterogeneous System: CPU+FPGA



Source: IBM

Source: AlphaData

POWER9 AC922

HBM-based AD9H7 board

We evaluate two POWER9+FPGA systems:

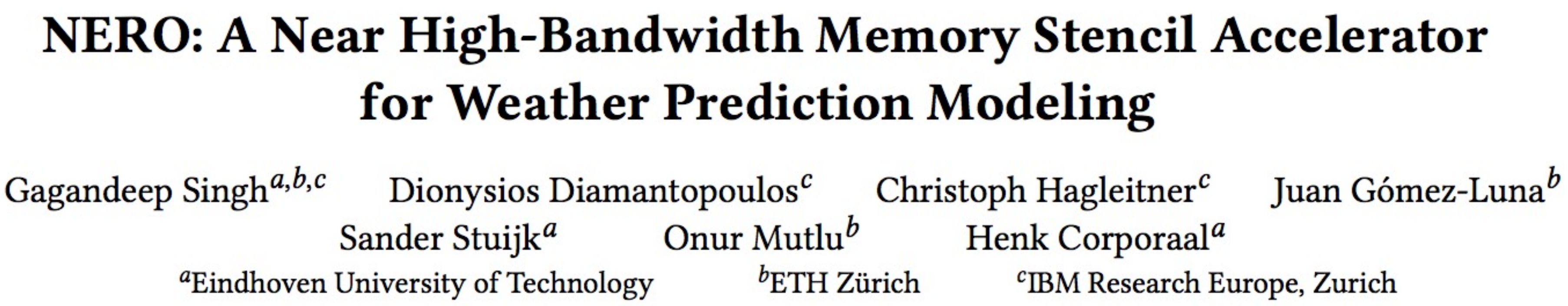
1. HBM-based board AD9H7
Xilinx Virtex Ultrascale+™ XCVU37P-2

2. DDR4-based board AD9V3
Xilinx Virtex Ultrascale+™ XCVU3P-2

Singh et al., NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling, FPL 2020

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.

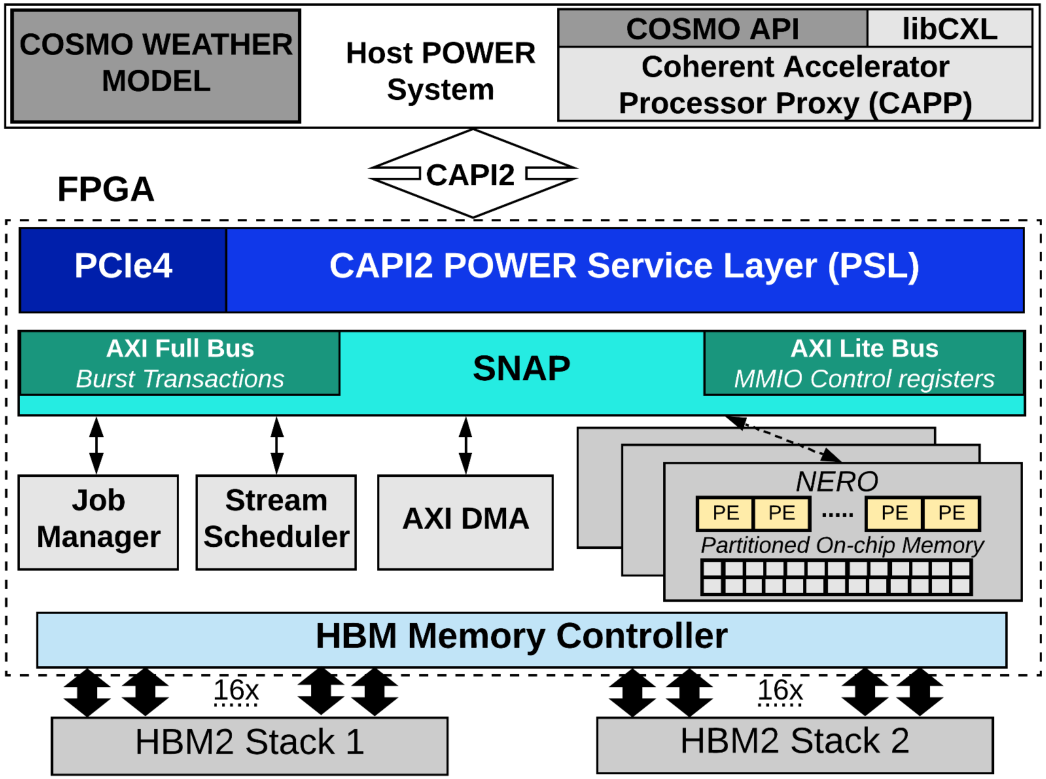


NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

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

NERO Application Framework

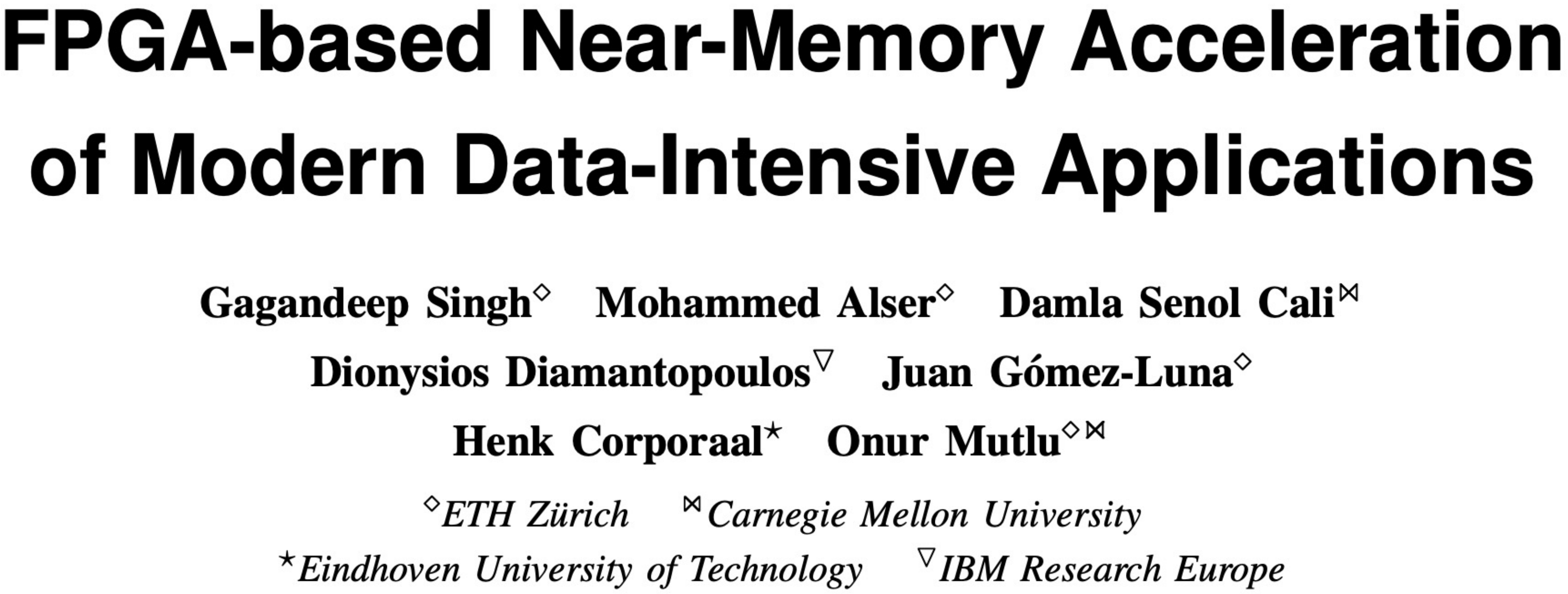
- NERO communicates to Host over **CAPI2** (Coherent Accelerator Processor Interface)
- **COSMO API** handles offloading jobs to NERO
- **SNAP** (Storage, Network, and Analytics Programming) allows for seamless integration of the COSMO API



https://github.com/open-power/snap

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"**
IEEE Micro (***IEEE MICRO***), 2021.



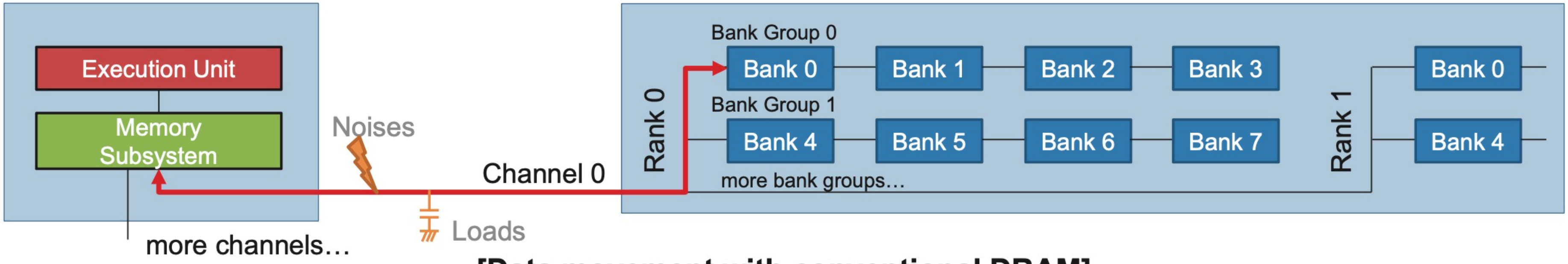
FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] **Mohammed Alser**[◇] **Damla Senol Cali**[⋈]
Dionysios Diamantopoulos[▽] **Juan Gómez-Luna**[◇]
Henk Corporaal[⋆] **Onur Mutlu**[◇⋈]

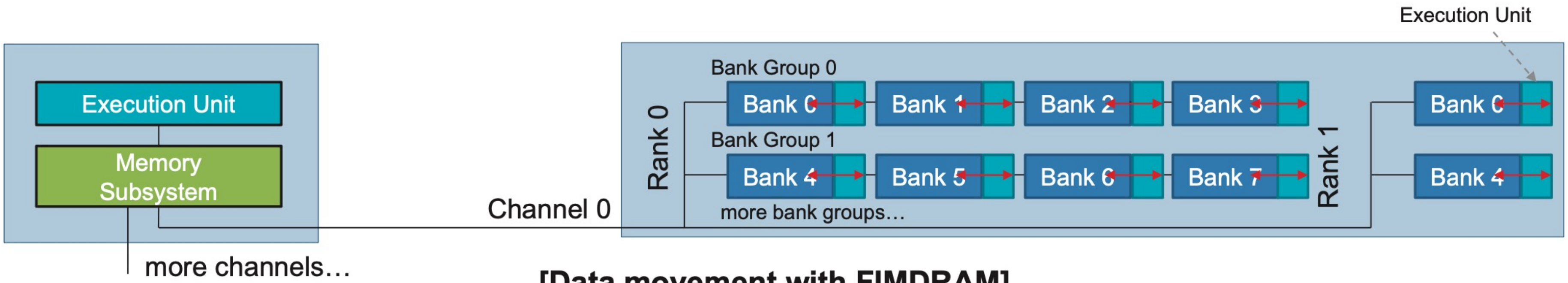
[◇]*ETH Zürich* [⋈]*Carnegie Mellon University*
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

FIMDRAM: Exploiting Bank Parallelism

- HBM bandwidth is not enough for many ML workloads
  - BLAS-1 and BLAS-2 are typically memory bound



[Data movement with conventional DRAM]

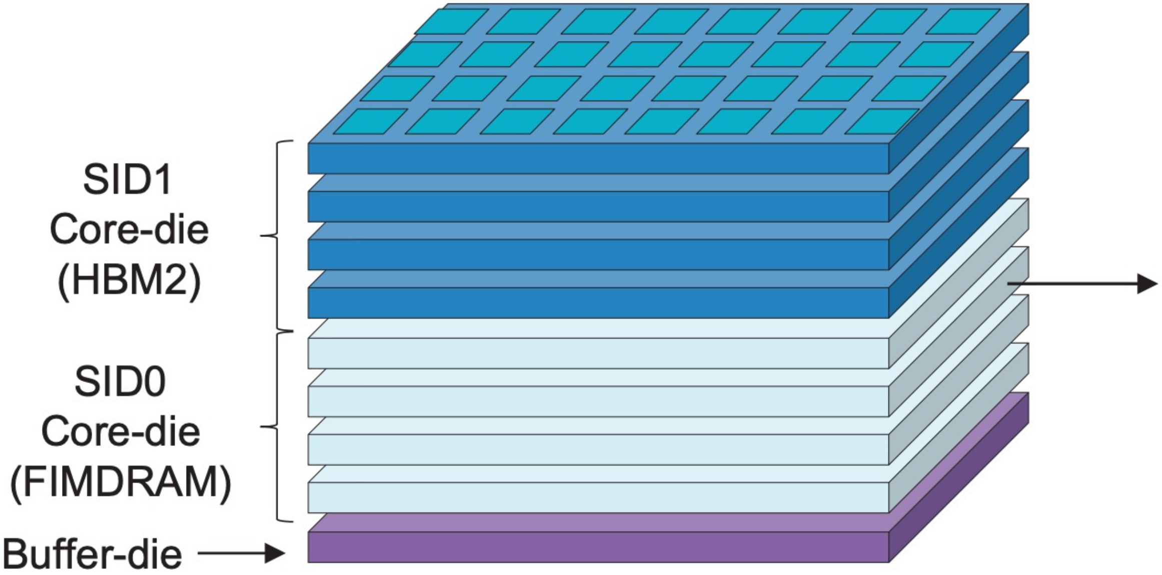


[Data movement with FIMDRAM]

Kwon et al., 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, ISSCC 2021

FIMDRAM: Chip Structure

- FIMDRAM based on HBM2



[3D Chip Structure of HBM with FIMDRAM]

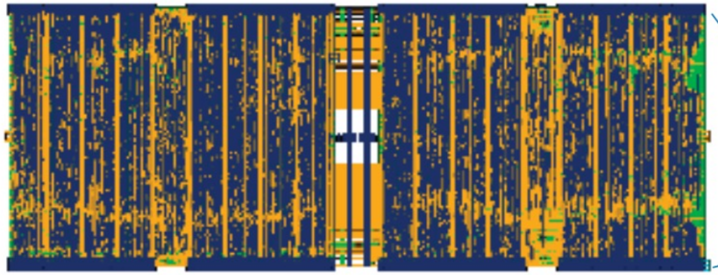
Chip Specification
128DQ / 8CH / 16 banks / BL4
32 PCU blocks (1 FIM block/2 banks)
1.2 TFLOPS (4H)
FP16 ADD / Multiply (MUL) / Multiply-Accumulate (MAC) / Multiply-and- Add (MAD)

Kwon et al., 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, ISSCC 2021

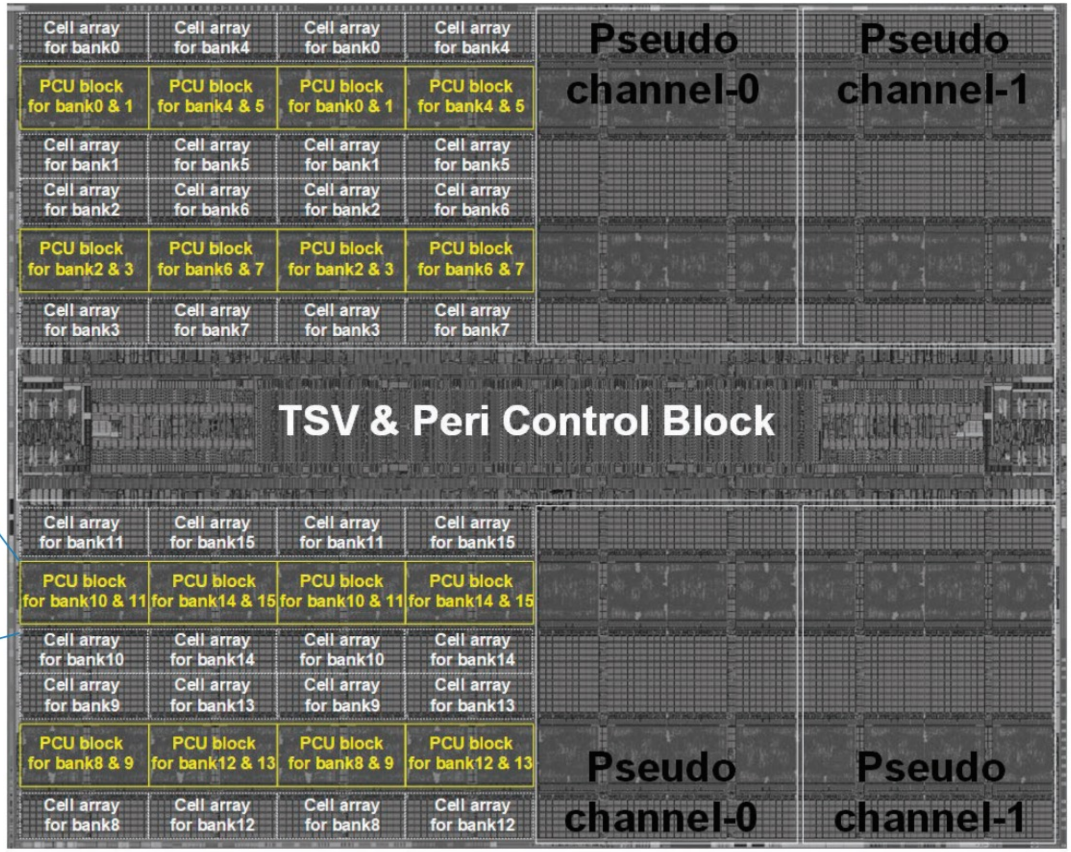
FIMDRAM: Chip Implementation

Chip Implementation

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



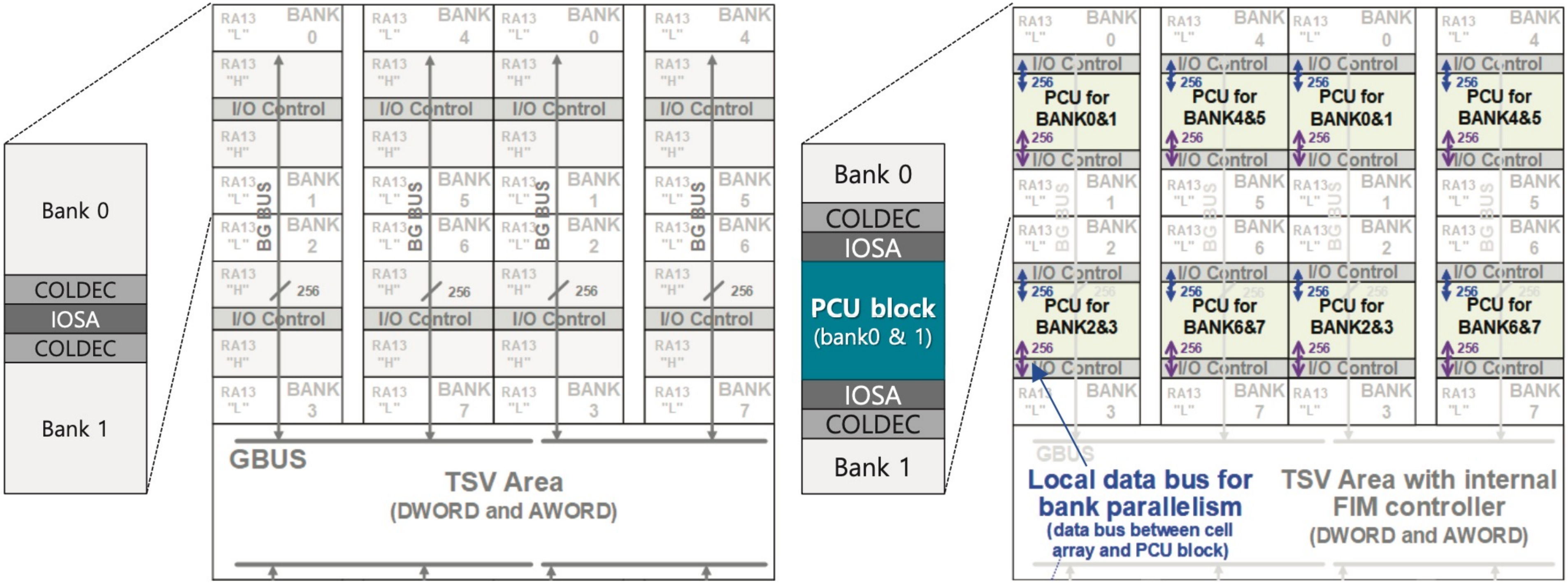
[Digital RTL design for PCU block]



Kwon et al., 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, ISSCC 2021

FIMDRAM: System Organization (I)

- HBM2 vs. FIMDRAM



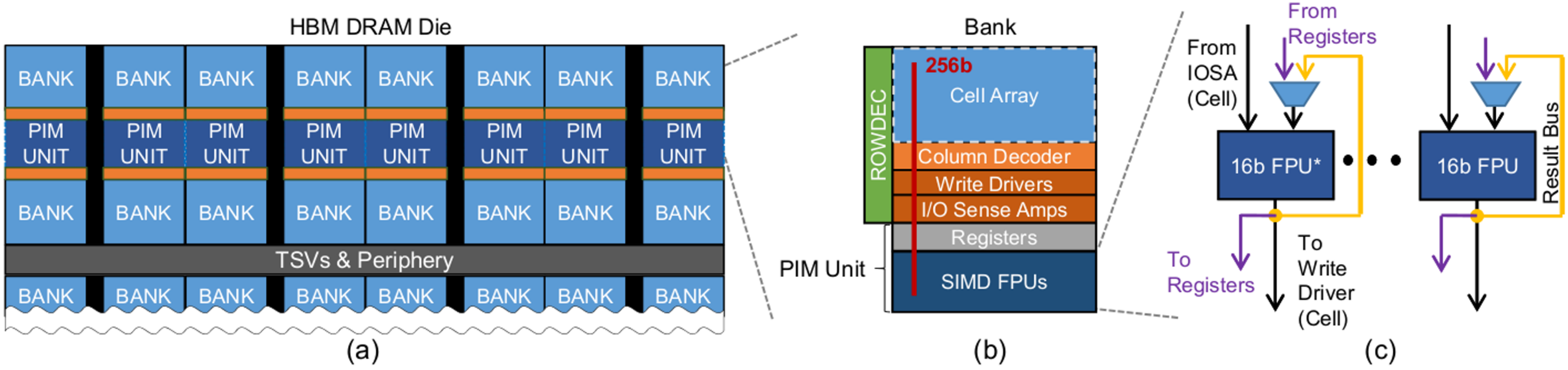
[HBM2]

[FIMDRAM]

Kwon et al., 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, ISSCC 2021

FIMDRAM: System Organization (II)

- Design goals:
  - 1. Support DRAM and PIM-DRAM mode for versatility
  - 2. Minimize the engineering cost of redesigning DRAM banks and sub-arrays
- Thus, PIM unit at I/O boundary of bank
  - 1 PIM unit for each 2 banks
  - 16 16-bit SIMD floating-point units (FPUs) per PIM unit



Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

SIMD Processing and GPUs

Flynn's Taxonomy of Computers

- Mike Flynn, "Very High-Speed Computing Systems," Proc. of IEEE, 1966

- SISD: Single instruction operates on single data element
- SIMD: Single instruction operates on multiple data elements
  - Array processor
  - Vector processor
- MISD: Multiple instructions operate on single data element
  - Closest form: systolic array processor, streaming processor
- MIMD: Multiple instructions operate on multiple data elements (multiple instruction streams)
  - Multiprocessor
  - Multithreaded processor

Data Parallelism

- Concurrency arises from performing the same operation on different pieces of data
  - Single instruction multiple data (SIMD)
  - E.g., dot product of two vectors
- Contrast with data flow
  - Concurrency arises from executing different operations in parallel (in a data driven manner)
- Contrast with thread (“control”) parallelism
  - Concurrency arises from executing different threads of control in parallel
- SIMD exploits operation-level parallelism on different data
  - Same operation concurrently applied to different pieces of data
  - A form of ILP where instruction happens to be the same across data

SIMD Processing

- Single instruction operates on multiple data elements
  - In time or in space
- Multiple processing elements (PEs), i.e., execution units

- Time-space duality

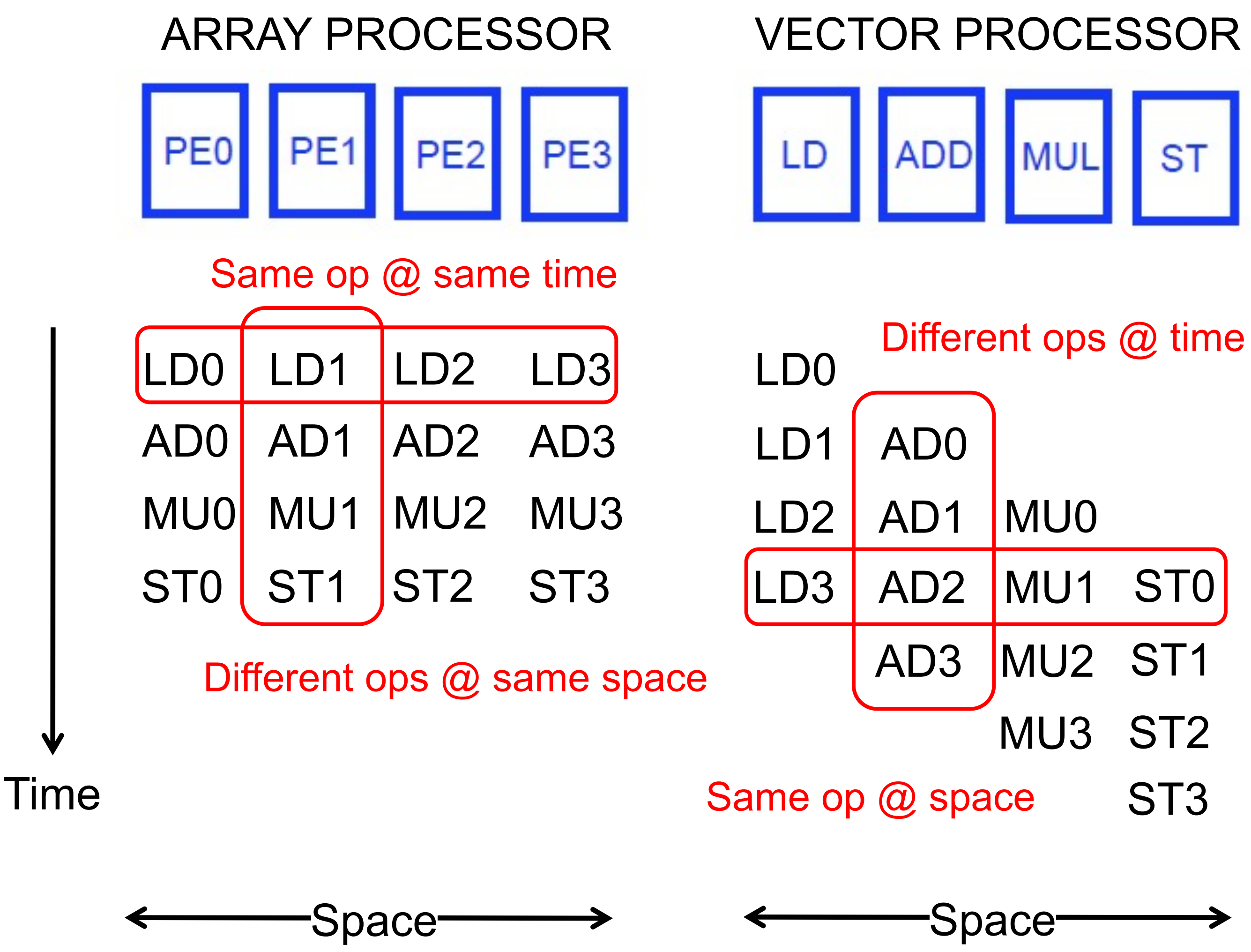
  - Array processor: Instruction operates on multiple data elements at the same time using different spaces (PEs)

  - Vector processor: Instruction operates on multiple data elements in consecutive time steps using the same space (PE)

Array vs. Vector Processors

Instruction Stream

```
LD   VR ← A[3:0]
ADD  VR ← VR, 1
MUL  VR ← VR, 2
ST   A[3:0] ← VR
```



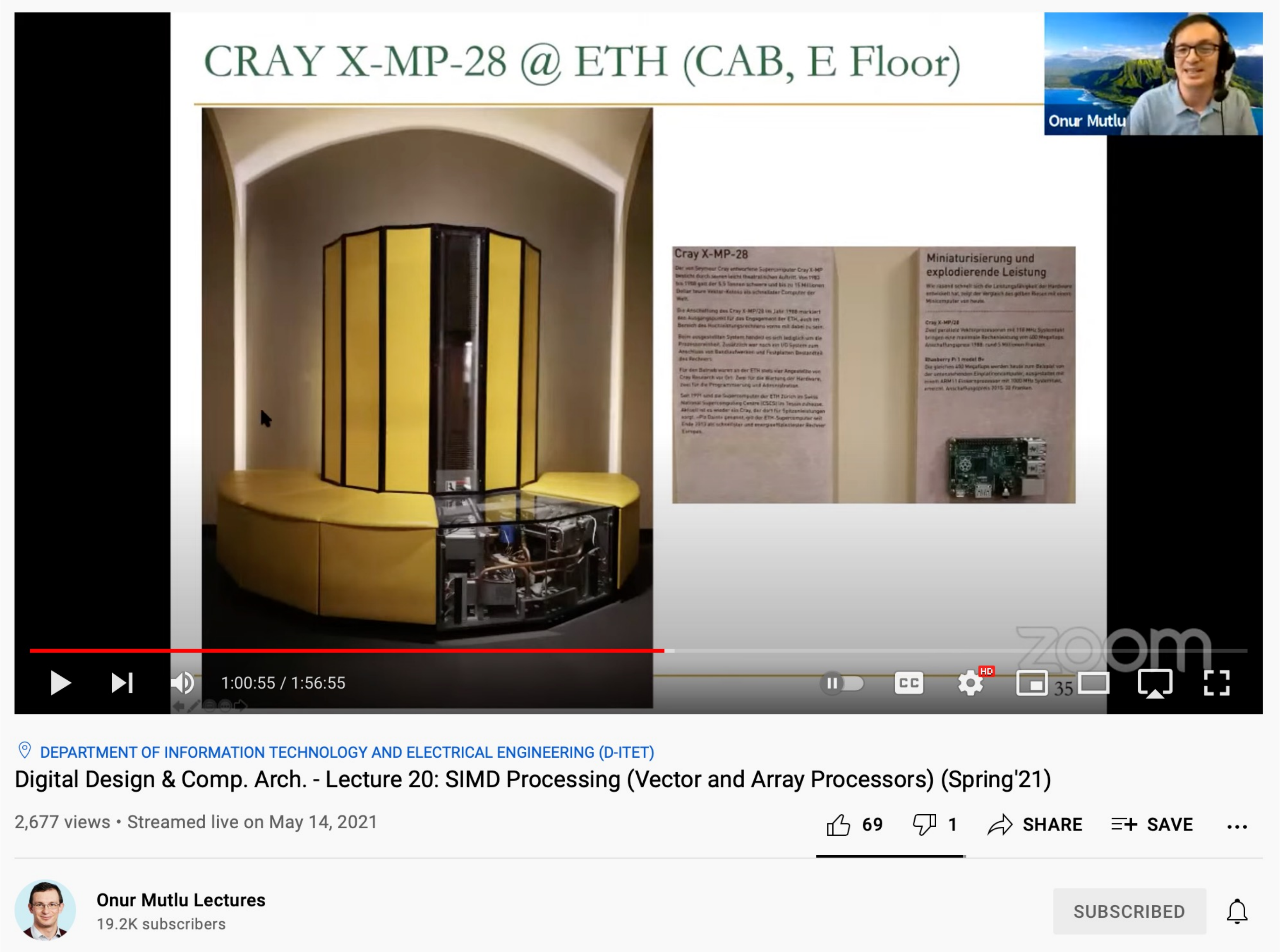
ARRAY PROCESSOR

PE0	PE1	PE2	PE3
LD0	LD1	LD2	LD3
AD0	AD1	AD2	AD3
MU0	MU1	MU2	MU3
ST0	ST1	ST2	ST3

VECTOR PROCESSOR

LD	ADD	MUL	ST
LD0			
LD1	AD0		
LD2	AD1	MU0	
LD3	AD2	MU1	ST0
	AD3	MU2	ST1
		MU3	ST2
			ST3

Lecture on SIMD Processing


CRAY X-MP-28 @ ETH (CAB, E Floor)
Cray X-MP-28
Miniaturisierung und explodierende Leistung
Onur Mutlu
zoom
1:00:55 / 1:56:55
HD
35
69
1
SHARE
SAVE
SUBSCRIBED

DEPARTMENT OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING (D-ITET)

Digital Design & Comp. Arch. - Lecture 20: SIMD Processing (Vector and Array Processors) (Spring'21)

2,677 views • Streamed live on May 14, 2021

Onur Mutlu Lectures

19.2K subscribers

https://youtu.be/fP4kZ2Zx_84

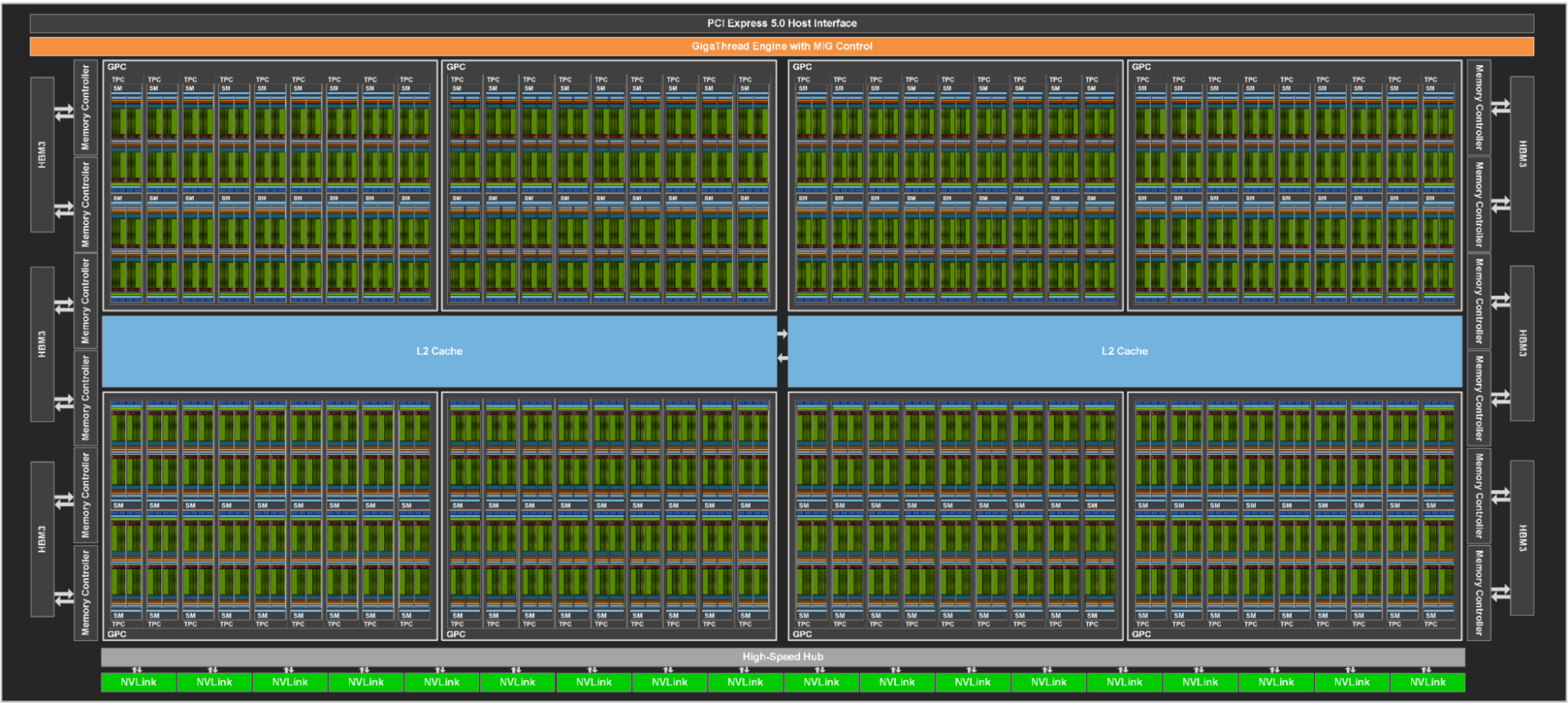
A GPU is a SIMD (SIMT) Machine

- Except it is **not** programmed using SIMD instructions

- It is programmed using threads (SPMD programming model)
  - Each thread executes the same code but operates a different piece of data
  - Each thread has its own context (i.e., can be treated/restarted/executed independently)

- A set of threads executing the same instruction are dynamically grouped into a **warp (wavefront)** by the hardware
  - A warp is essentially a SIMD operation formed by hardware!

NVIDIA H100 Block Diagram

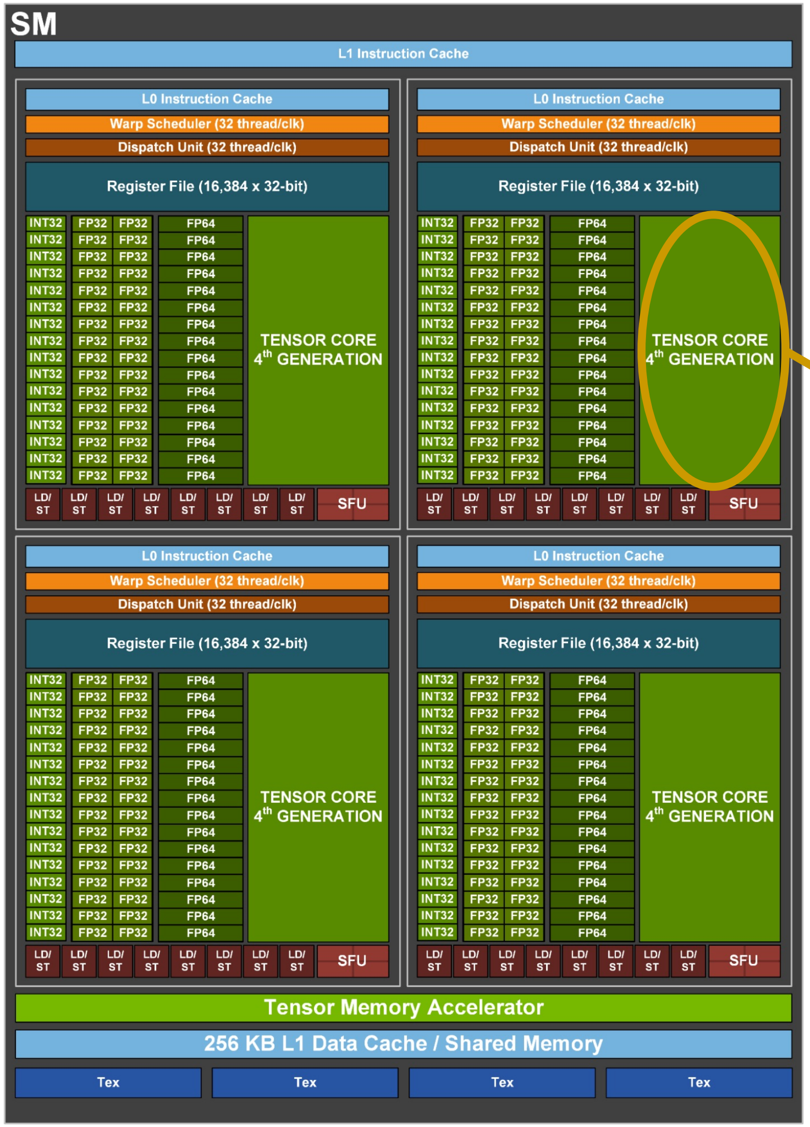


https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

144 cores on the full GH100

60MB L2 cache

NVIDIA H100 Core

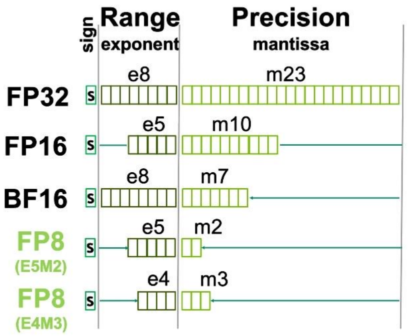


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

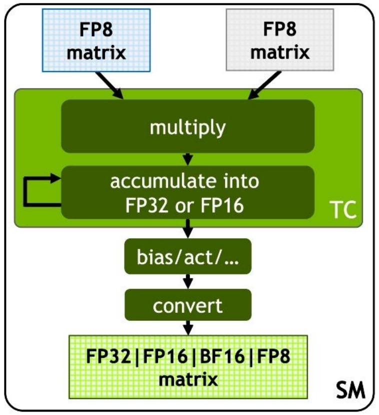
48 TFLOPS Single Precision*

24 TFLOPS Double Precision*

800 TFLOPS (FP16, Tensor Cores)*



Allocate 1 bit to either range or precision

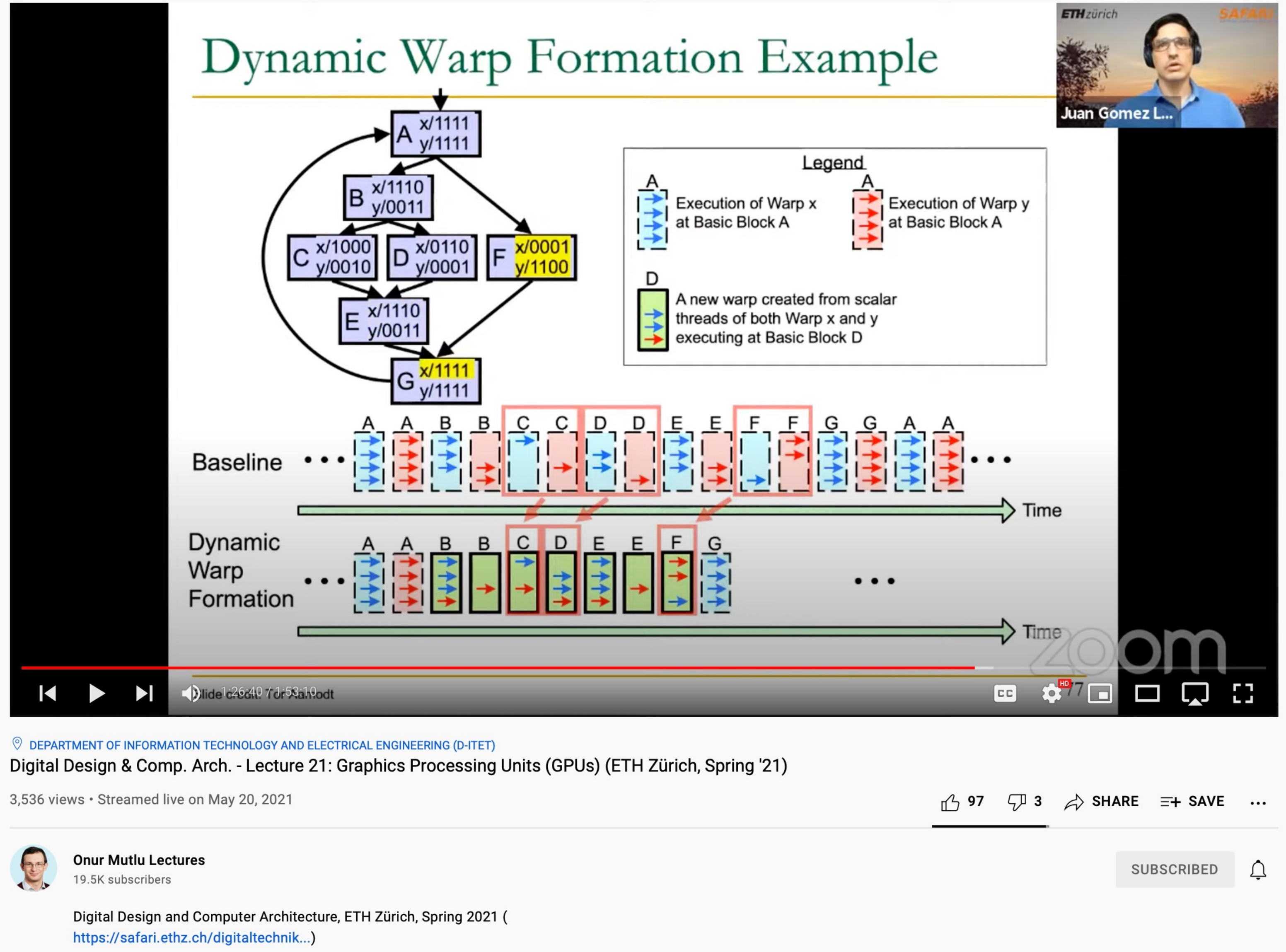


Support for multiple accumulator and output types

https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

* Preliminary performance estimates

Lecture on Graphics Processing Units



Dynamic Warp Formation Example

Execution of Warp x at Basic Block A

Execution of Warp y at Basic Block A

A new warp created from scalar threads of both Warp x and y executing at Basic Block D

DEPARTMENT OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING (D-ITET)

Digital Design & Comp. Arch. - Lecture 21: Graphics Processing Units (GPUs) (ETH Zürich, Spring '21)

3,536 views • Streamed live on May 20, 2021

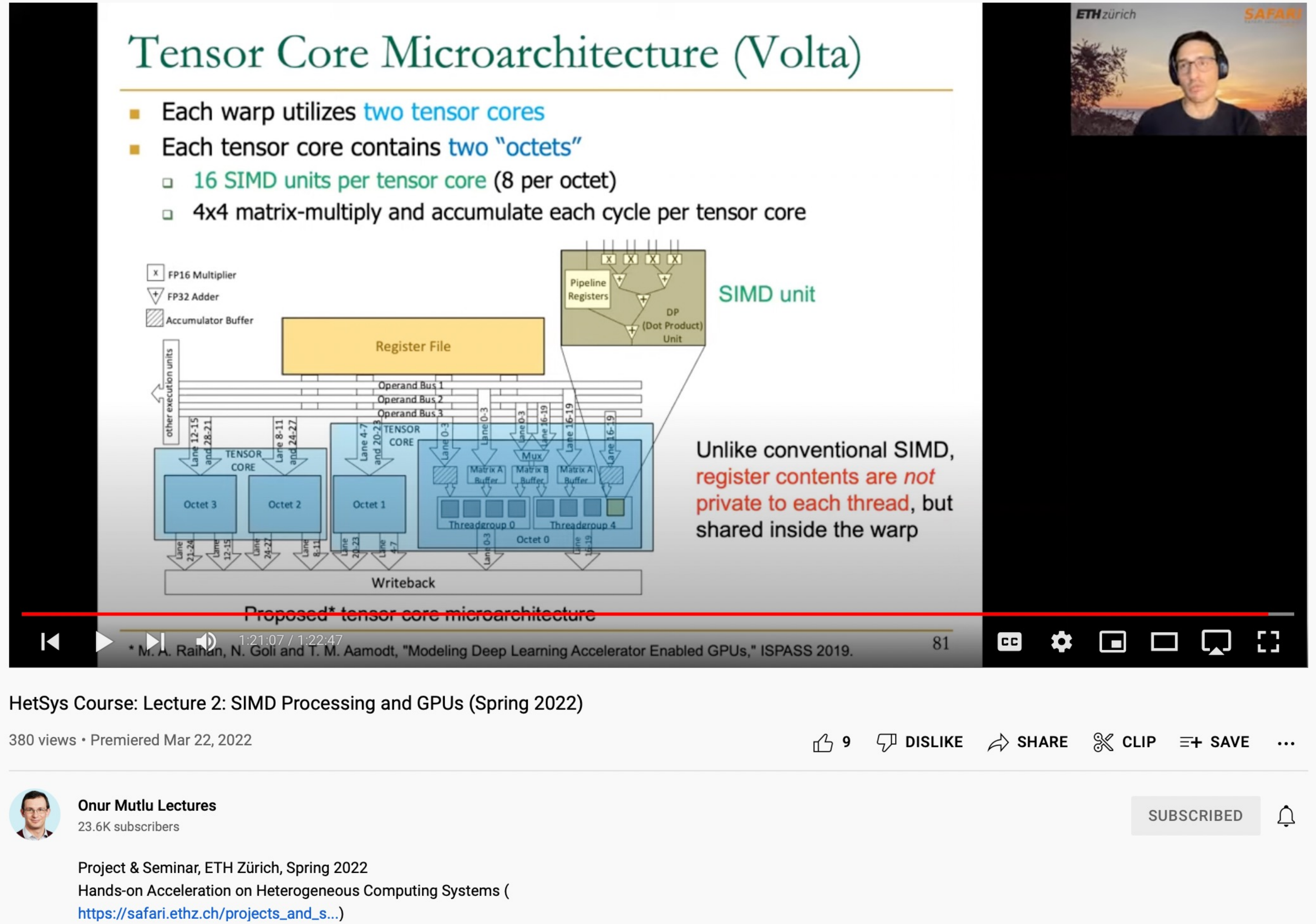
Onur Mutlu Lectures

19.5K subscribers

Digital Design and Computer Architecture, ETH Zürich, Spring 2021 (
https://safari.ethz.ch/digitaltechnik...)

https://youtu.be/eaxGCv0wRrU

Lecture on SIMD Processing and GPUs



Tensor Core Microarchitecture (Volta)

- Each warp utilizes two tensor cores
- Each tensor core contains two "octets"
  - 16 SIMD units per tensor core (8 per octet)
  - 4x4 matrix-multiply and accumulate each cycle per tensor core

Unlike conventional SIMD, register contents are *not* private to each thread, but shared inside the warp

Proposed* tensor core microarchitecture

81

* M. A. Raihan, N. Goli and T. M. Aamodt, "Modeling Deep Learning Accelerator Enabled GPUs," ISPASS 2019.

HetSys Course: Lecture 2: SIMD Processing and GPUs (Spring 2022)

380 views • Premiered Mar 22, 2022

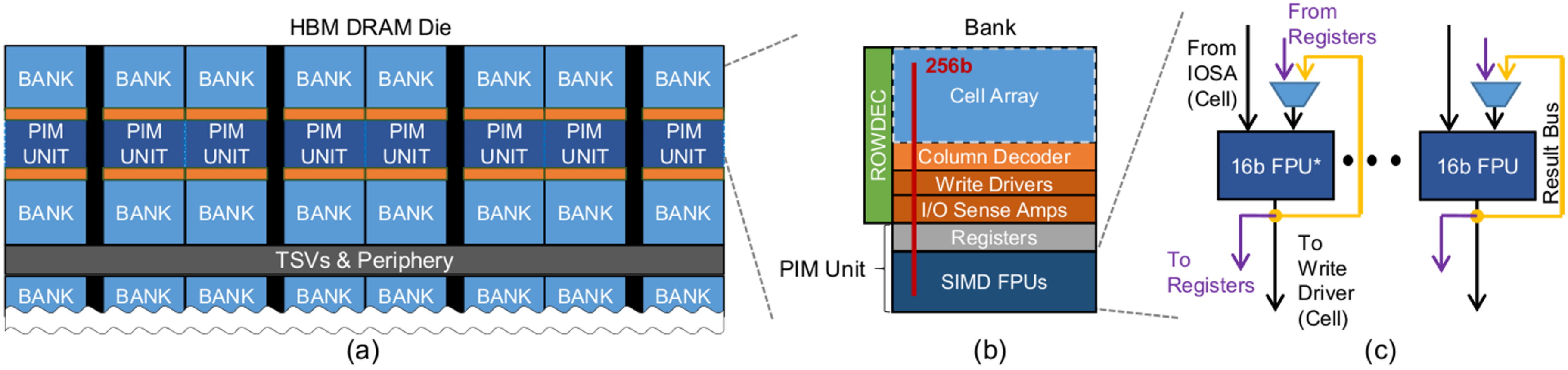
Onur Mutlu Lectures
23.6K subscribers

Project & Seminar, ETH Zürich, Spring 2022
Hands-on Acceleration on Heterogeneous Computing Systems (
https://safari.ethz.ch/projects_and_s...)

https://youtu.be/hEqk6UMQT0U

FIMDRAM: System Organization (III)

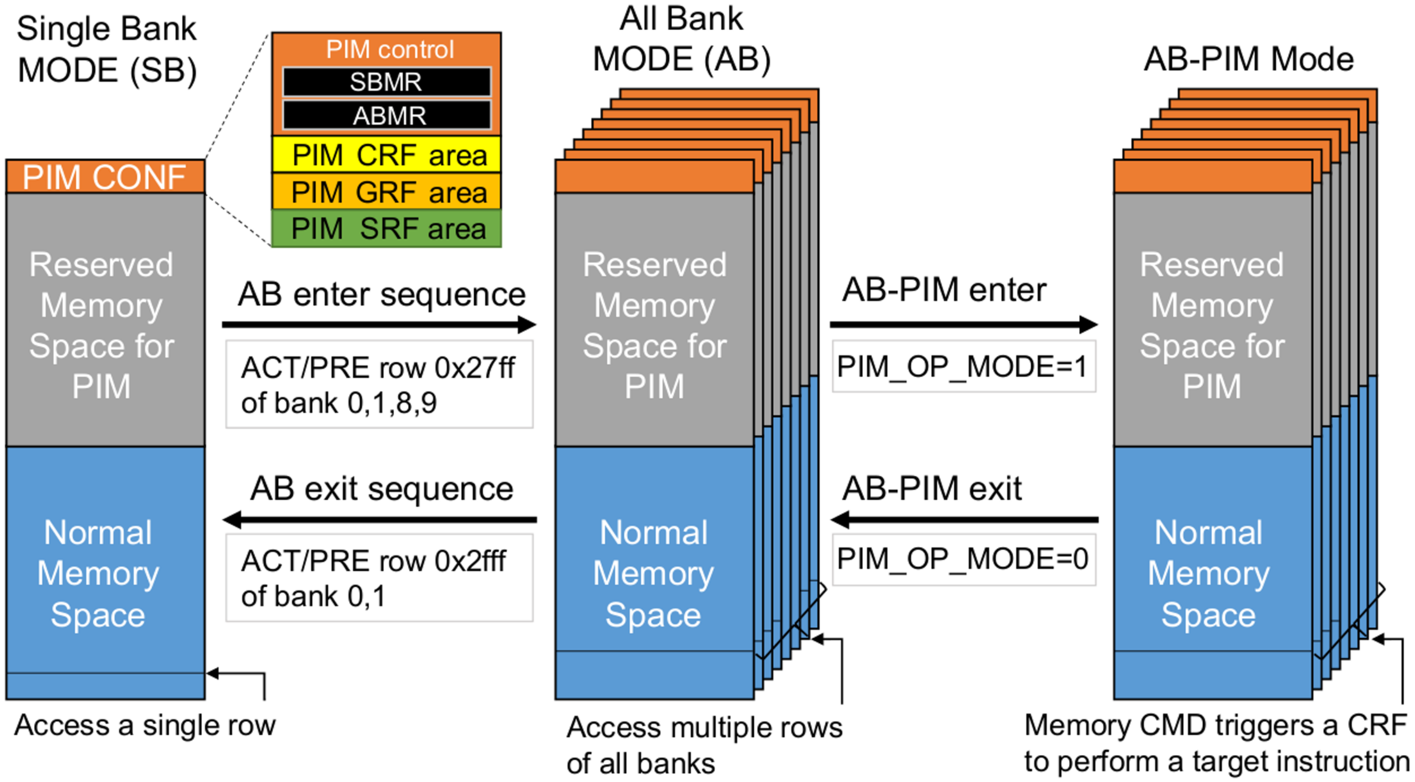
- PIM units respond to standard DRAM column commands (RD or WR)
  - Compliant with unmodified JEDEC controllers
- They execute one wide-SIMD operation commanded by a PIM instruction with deterministic latency in a lock-step manner
- A PIM unit can get 16 16-bit operands from IOSAs, a register, and/or the result bus



Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

FIMDRAM: Bank-level Parallelism

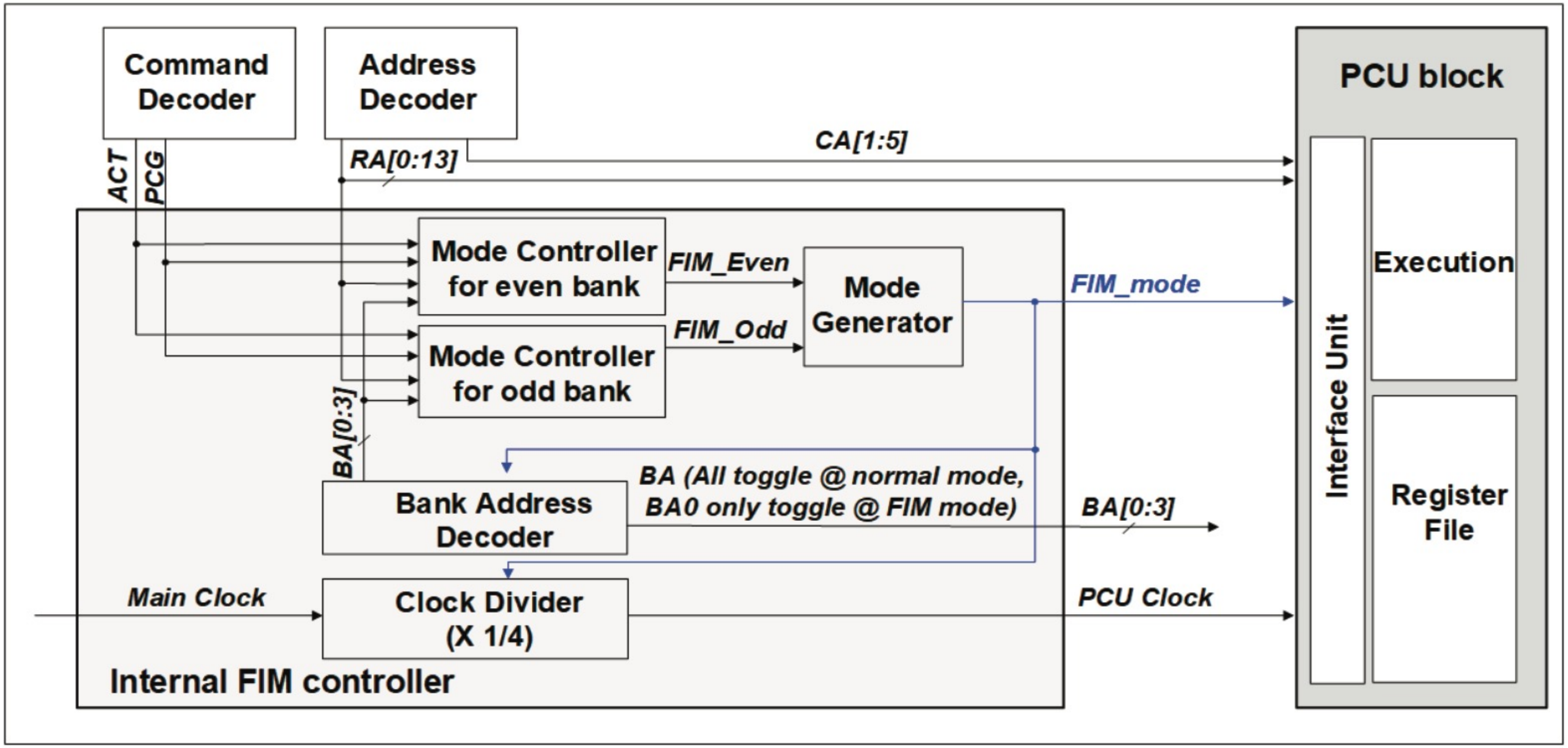
- Unlike standard DRAM devices, all banks can be accessed concurrently for 8x higher bandwidth (with 16 pCHs)
- In AB-PIM mode, a memory command triggers a PIM instruction in the CRF



Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

FIMDRAM: Internal FIM Controller

- The internal FIM controller controls FIM mode without any modification of the host processor hardware

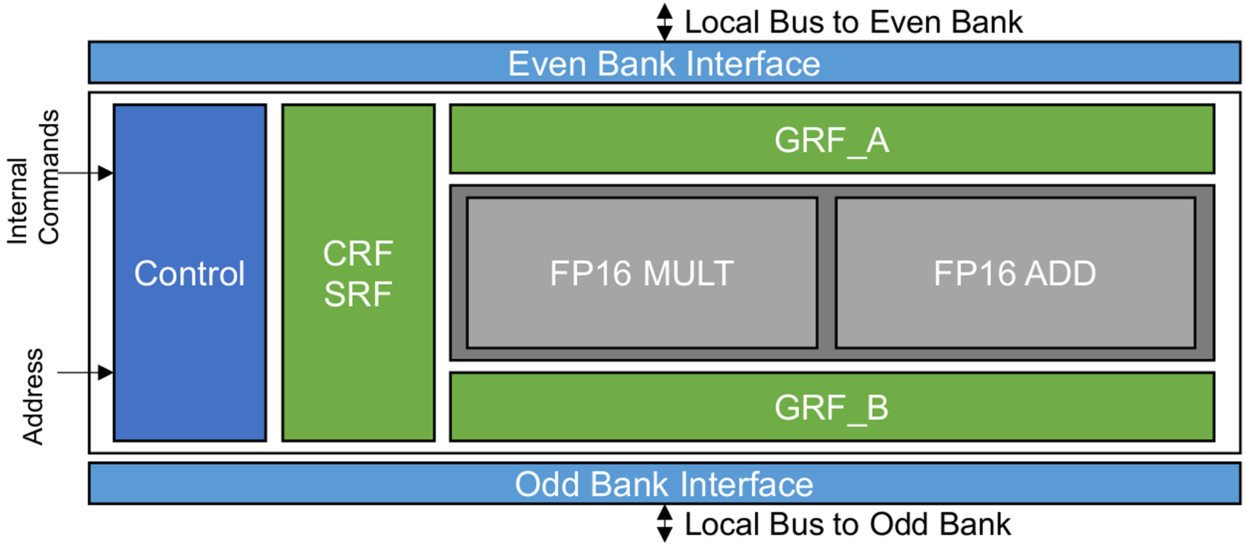


[Block diagram of control circuit for FIM operation]

Kwon et al., 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, ISSCC 2021

FIMDRAM: Programmable Computing Unit (I)

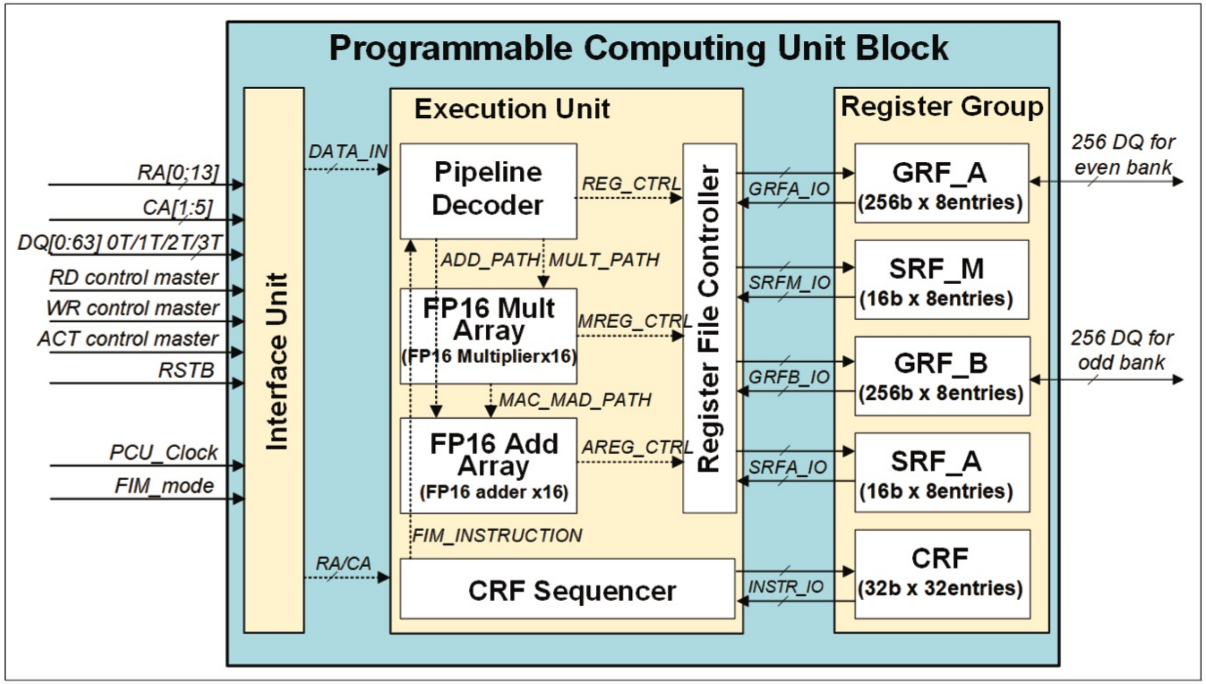
- Control: Instruction sequence manager
- Pipeline of 5 stages
  - 1. Fetch/decode
  - 2. Load 256-bit data from even or odd bank (optional)
  - 3. MUL
  - 4. ADD
  - 5. Writeback to GRF



Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

FIMDRAM: Programmable Computing Unit (II)

- Interface unit to control data flow
- Execution unit
- Register group
  - CRF (command): Instruction buffer
  - GRF (general): Weights and accumulation
  - SRF (source): Constants for MAC

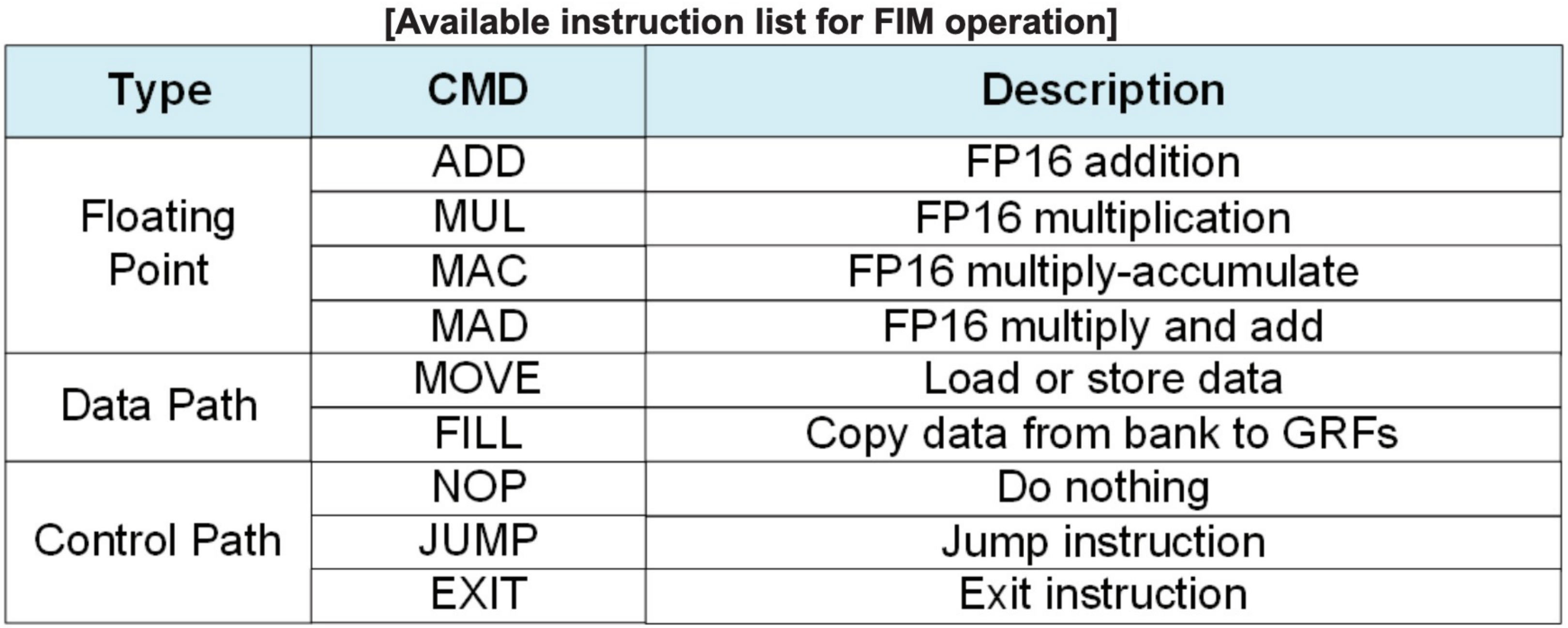


[Block diagram of PCU in FIMDRAM]

Kwon et al., 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, ISSCC 2021

FIMDRAM: Instruction Set Architecture (I)

- 9 RISC-style 32-bit instructions



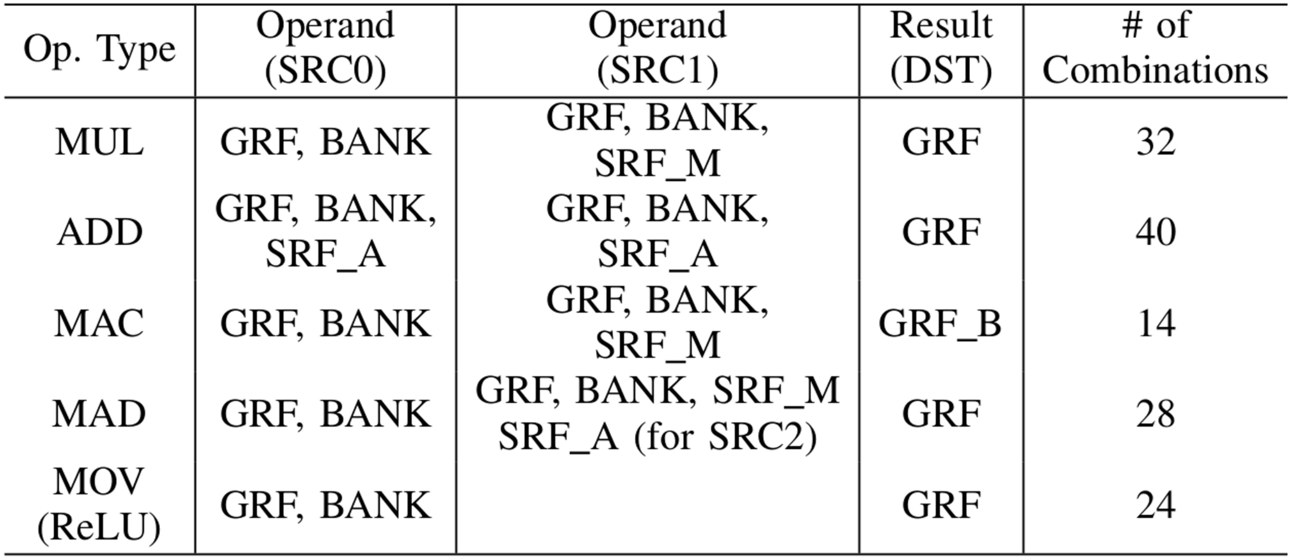
[Available instruction list for FIM operation]

Type	CMD	Description
Floating Point	ADD	FP16 addition
	MUL	FP16 multiplication
	MAC	FP16 multiply-accumulate
	MAD	FP16 multiply and add
Data Path	MOVE	Load or store data
	FILL	Copy data from bank to GRFs
Control Path	NOP	Do nothing
	JUMP	Jump instruction
	EXIT	Exit instruction

Kwon et al., 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, ISSCC 2021

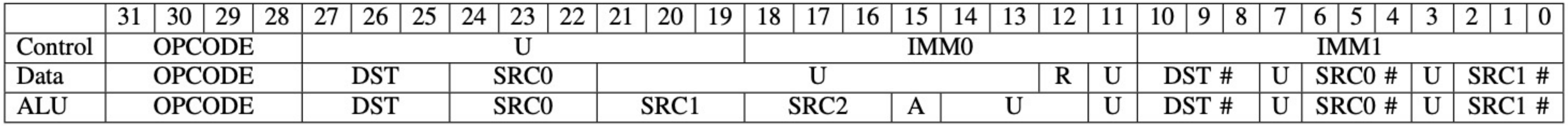
FIMDRAM: Instruction Set Architecture (II)

- Combinations depend on operand sources



Op. Type	Operand (SRC0)	Operand (SRC1)	Result (DST)	# of Combinations
MUL	GRF, BANK	GRF, BANK, SRF_M	GRF	32
ADD	GRF, BANK, SRF_A	GRF, BANK, SRF_A	GRF	40
MAC	GRF, BANK	GRF, BANK, SRF_M	GRF_B	14
MAD	GRF, BANK	GRF, BANK, SRF_M SRF_A (for SRC2)	GRF	28
MOV (ReLU)	GRF, BANK		GRF	24

- Instruction formats

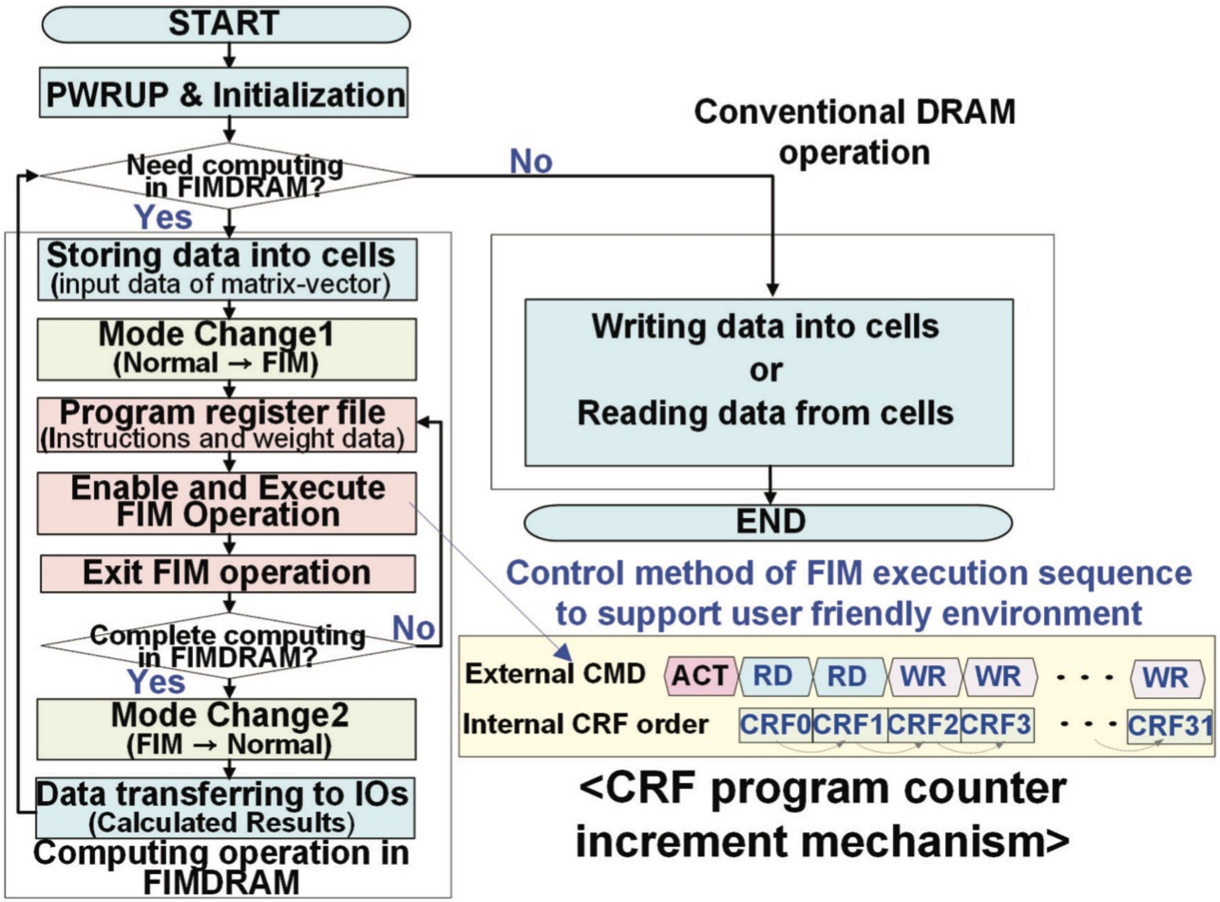


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Control	OPCODE				U									IMM0								IMM1										
Data	OPCODE				DST			SRC0			U									R	U	DST #			U	SRC0 #			U	SRC1 #		
ALU	OPCODE				DST			SRC0			SRC1			SRC2			A	U			U	DST #			U	SRC0 #			U	SRC1 #		

Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

FIMDRAM: Operation Flow

- Operation sequence for matrix vector computing
  - Input and output data are accessible to the host in conventional DRAM operation



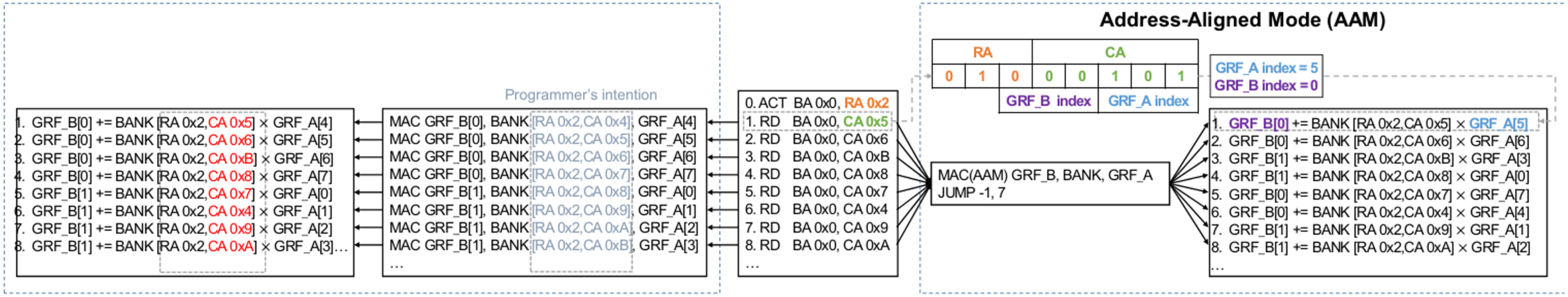
START
PWRUP & Initialization
Need computing in FIMDRAM?
No
Conventional DRAM operation
Yes
Storing data into cells (input data of matrix-vector)
Mode Change1 (Normal → FIM)
Program register file (Instructions and weight data)
Enable and Execute FIM Operation
Exit FIM operation
Complete computing in FIMDRAM?
No
Yes
Mode Change2 (FIM → Normal)
Data transferring to IOs (Calculated Results)
Computing operation in FIMDRAM
Writing data into cells or Reading data from cells
END
Control method of FIM execution sequence to support user friendly environment
External CMD
ACT
RD
RD
WR
WR
· · ·
WR
Internal CRF order
CRF0
CRF1
CRF2
CRF3
· · ·
CRF31

<CRF program counter increment mechanism>

Kwon et al., 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, ISSCC 2021

FIMDRAM: Instruction Ordering

- One challenge is that DRAM commands may be re-ordered, and using fences is costly performance-wise
- Solution: Address Aligned Mode (AAM)
  - 8 MAC operations with 2 PIM instructions



Programmer's intention

```
1. GRF_B[0] += BANK [RA 0x2,CA 0x5] × GRF_A[4]
2. GRF_B[0] += BANK [RA 0x2,CA 0x6] × GRF_A[5]
3. GRF_B[0] += BANK [RA 0x2,CA 0xB] × GRF_A[6]
4. GRF_B[0] += BANK [RA 0x2,CA 0x8] × GRF_A[7]
5. GRF_B[1] += BANK [RA 0x2,CA 0x7] × GRF_A[0]
6. GRF_B[1] += BANK [RA 0x2,CA 0x4] × GRF_A[1]
7. GRF_B[1] += BANK [RA 0x2,CA 0x9] × GRF_A[2]
8. GRF_B[1] += BANK [RA 0x2,CA 0xA] × GRF_A[3]…
```

```
MAC GRF_B[0], BANK [RA 0x2,CA 0x4], GRF_A[4]
MAC GRF_B[0], BANK [RA 0x2,CA 0x5], GRF_A[5]
MAC GRF_B[0], BANK [RA 0x2,CA 0x6], GRF_A[6]
MAC GRF_B[0], BANK [RA 0x2,CA 0x7], GRF_A[7]
MAC GRF_B[1], BANK [RA 0x2,CA 0x8], GRF_A[0]
MAC GRF_B[1], BANK [RA 0x2,CA 0x9], GRF_A[1]
MAC GRF_B[1], BANK [RA 0x2,CA 0xA], GRF_A[2]
MAC GRF_B[1], BANK [RA 0x2,CA 0xB], GRF_A[3]
…
```

```
0. ACT BA 0x0, RA 0x2
1. RD  BA 0x0, CA 0x5
2. RD  BA 0x0, CA 0x6
3. RD  BA 0x0, CA 0xB
4. RD  BA 0x0, CA 0x8
5. RD  BA 0x0, CA 0x7
6. RD  BA 0x0, CA 0x4
7. RD  BA 0x0, CA 0x9
8. RD  BA 0x0, CA 0xA
…
```

Address-Aligned Mode (AAM)

RA			CA				
0	1	0	0	0	1	0	1
		GRF_B index			GRF_A index		

GRF_A index = 5
GRF_B index = 0

```
MAC(AAM) GRF_B, BANK, GRF_A
JUMP -1, 7
```

```
1. GRF_B[0] += BANK [RA 0x2,CA 0x5] × GRF_A[5]
2. GRF_B[0] += BANK [RA 0x2,CA 0x6] × GRF_A[6]
3. GRF_B[1] += BANK [RA 0x2,CA 0xB] × GRF_A[3]
4. GRF_B[1] += BANK [RA 0x2,CA 0x8] × GRF_A[0]
5. GRF_B[0] += BANK [RA 0x2,CA 0x7] × GRF_A[7]
6. GRF_B[0] += BANK [RA 0x2,CA 0x4] × GRF_A[4]
7. GRF_B[1] += BANK [RA 0x2,CA 0x9] × GRF_A[1]
8. GRF_B[1] += BANK [RA 0x2,CA 0xA] × GRF_A[2]
…
```

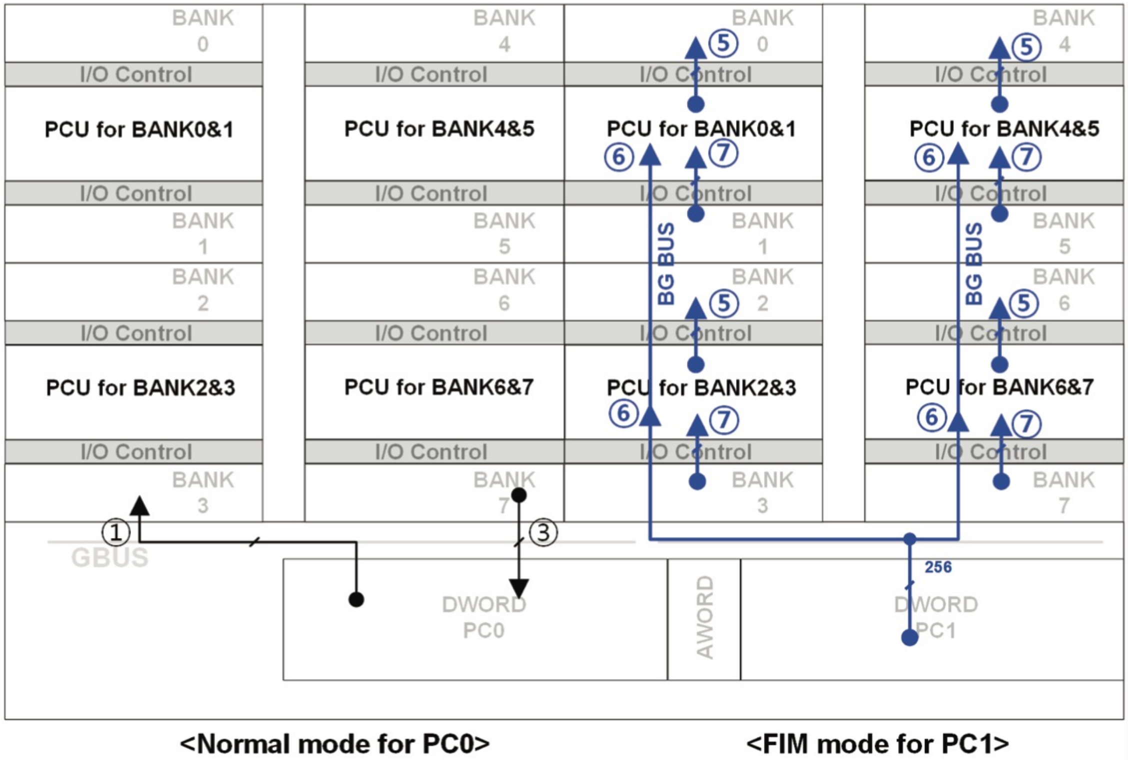
(c) Actual PIM execution (a) CRF instructions (b) Memory commands (d) CRF instructions w/ AAM (e) Actual PIM execution w/ AAM

Lee et al., Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology, ISCA 2021

FIMDRAM: Data Flow

- Data flow controlled by operation mode and bit RA13

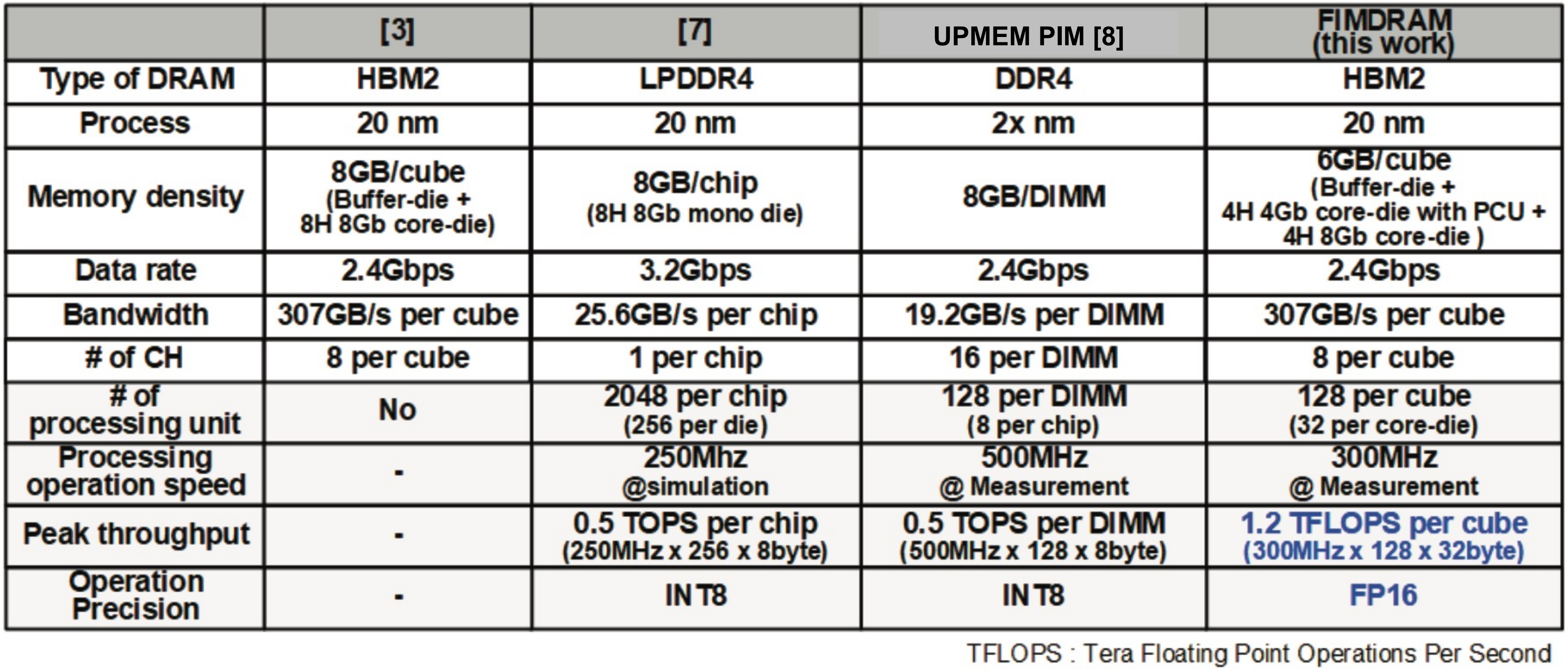
Index	Mode	CMD	RA13	Data Movement
①	Normal	WR	L	Data write to cell array
②		WR	H	Not available
③		RD	L	Data read to cell array
④		RD	H	Not available
⑤	FIM	WR	L	Data movement from PCU block to cell array
⑥		WR	H	Data write to PCU register
⑦		RD	L	Data movement from cell array to PCU block
⑧		RD	H	Not available



Kwon et al., 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, ISSCC 2021

FIMDRAM: Key Feature Summary

- Comparison table



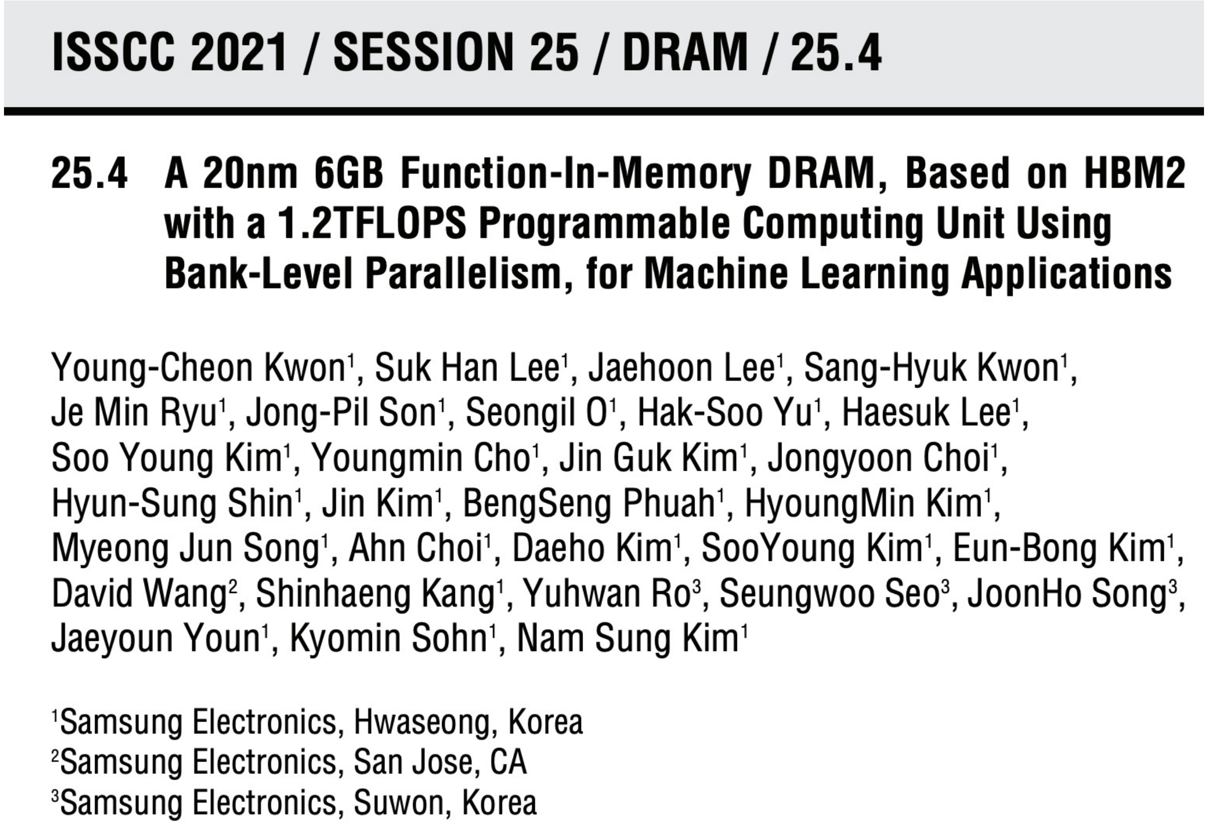
	[3]	[7]	UPMEM PIM [8]	FIMDRAM (this work)
Type of DRAM	HBM2	LPDDR4	DDR4	HBM2
Process	20 nm	20 nm	2x nm	20 nm
Memory density	8GB/cube (Buffer-die + 8H 8Gb core-die)	8GB/chip (8H 8Gb mono die)	8GB/DIMM	6GB/cube (Buffer-die + 4H 4Gb core-die with PCU + 4H 8Gb core-die)
Data rate	2.4Gbps	3.2Gbps	2.4Gbps	2.4Gbps
Bandwidth	307GB/s per cube	25.6GB/s per chip	19.2GB/s per DIMM	307GB/s per cube
# of CH	8 per cube	1 per chip	16 per DIMM	8 per cube
# of processing unit	No	2048 per chip (256 per die)	128 per DIMM (8 per chip)	128 per cube (32 per core-die)
Processing operation speed	-	250Mhz @simulation	500MHz @ Measurement	300MHz @ Measurement
Peak throughput	-	0.5 TOPS per chip (250MHz x 256 x 8byte)	0.5 TOPS per DIMM (500MHz x 128 x 8byte)	1.2 TFLOPS per cube (300MHz x 128 x 32byte)
Operation Precision	-	INT8	INT8	FP16

TFLOPS : Tera Floating Point Operations Per Second

[3] K. Sohn, et al., ISSCC 2016, [7] H. Shin, et al., IEEE TCADICS 2018, [8] F. Devaux, IEEE Hot Chips Symp. 2019

Kwon et al., 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, ISSCC 2021

Function-in-Memory DRAM (ISSCC 2021)



ISSCC 2021 / SESSION 25 / DRAM / 25.4

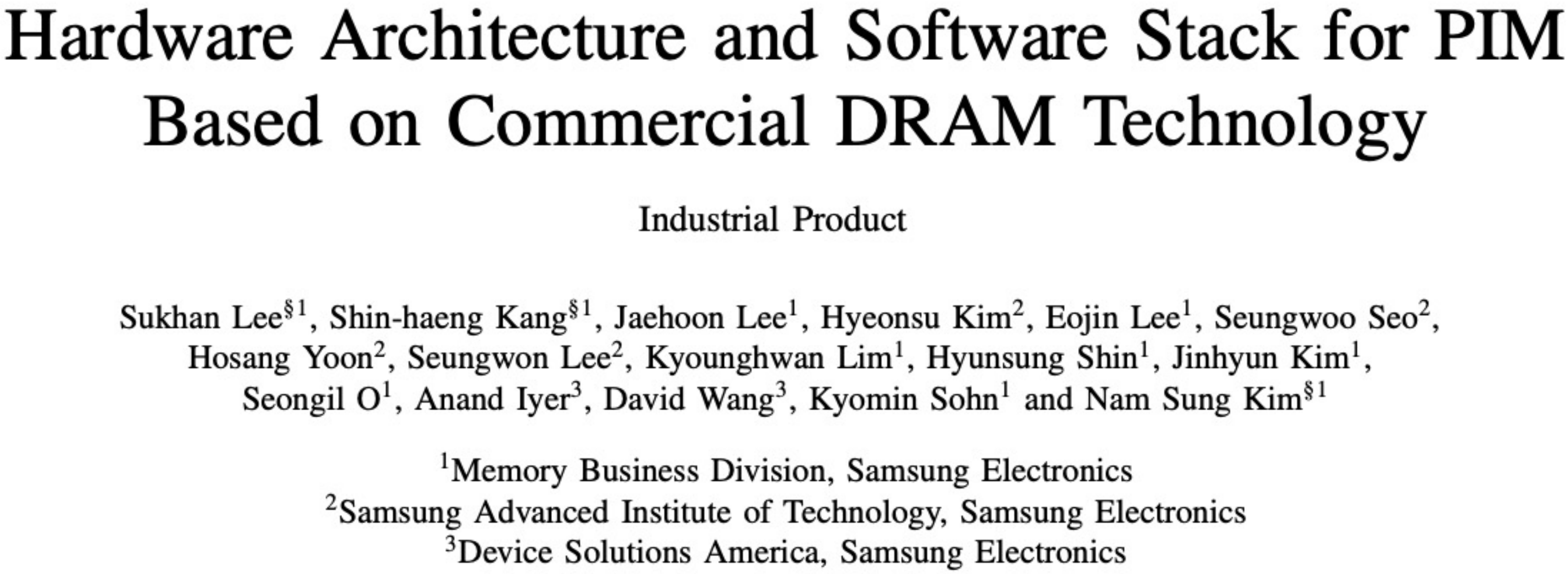
25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

https://doi.org/10.1109/ISSCC42613.2021.9365862

PIM based on Commercial DRAM (ISCA 2021)



Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology

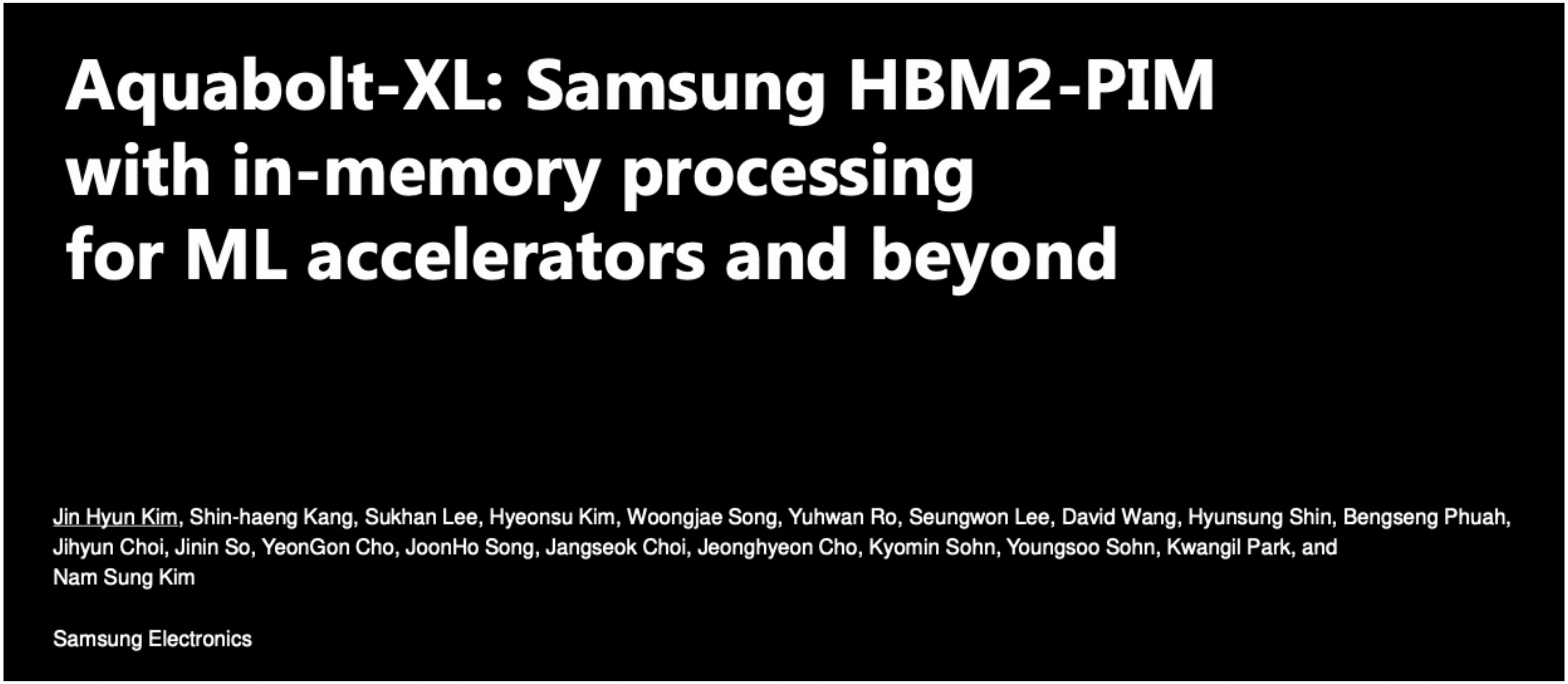
Industrial Product

Sukhan Lee[§1], Shin-haeng Kang[§1], Jaehoon Lee[1], Hyeonsu Kim[2], Eojin Lee[1], Seungwoo Seo[2],
Hosang Yoon[2], Seungwon Lee[2], Kyounghwan Lim[1], Hyunsung Shin[1], Jinhyun Kim[1],
Seongil O[1], Anand Iyer[3], David Wang[3], Kyomin Sohn[1] and Nam Sung Kim[§1]

[1]Memory Business Division, Samsung Electronics
[2]Samsung Advanced Institute of Technology, Samsung Electronics
[3]Device Solutions America, Samsung Electronics

https://doi.org/10.1109/ISCA52012.2021.00013

Aquabolt-XL: Samsung HBM2-PIM (HCS 2021)



Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond

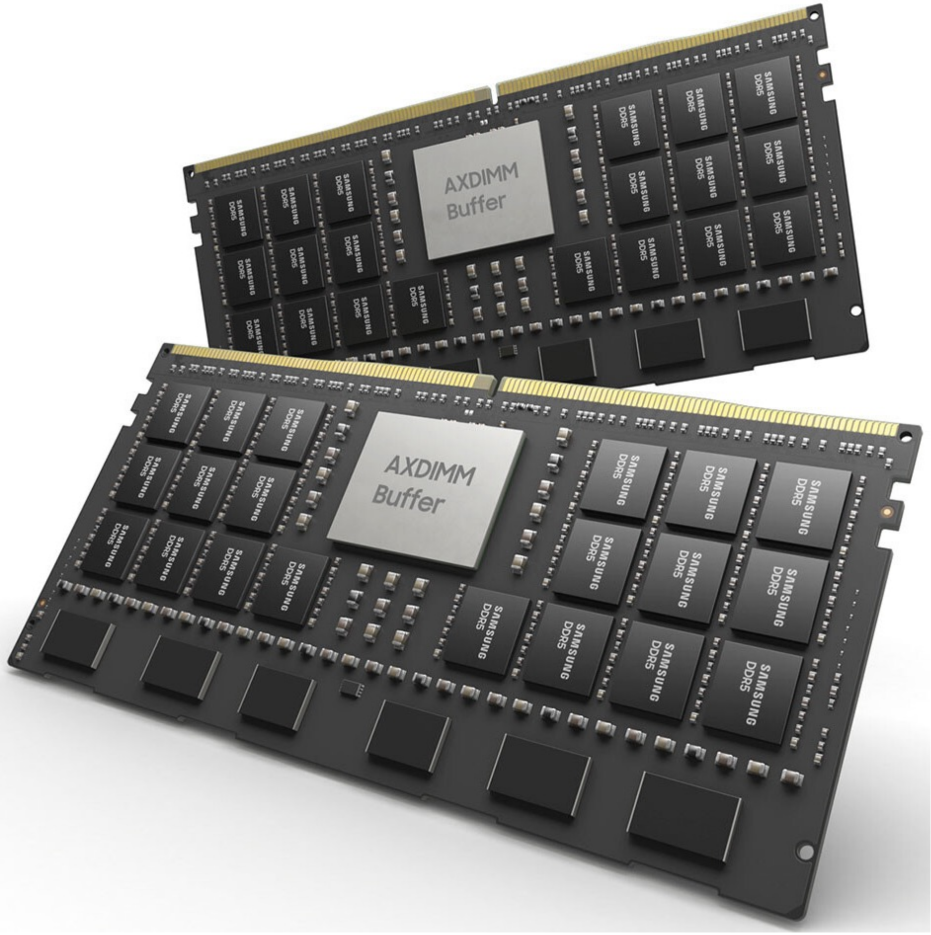
Jin Hyun Kim, Shin-haeng Kang, Sukhan Lee, Hyeonsu Kim, Woongjae Song, Yuhwan Ro, Seungwon Lee, David Wang, Hyunsung Shin, Bengseng Phuah, Jihyun Choi, Jinin So, YeonGon Cho, JoonHo Song, Jangseok Choi, Jeonghyeon Cho, Kyomin Sohn, Youngsoo Sohn, Kwangil Park, and Nam Sung Kim

Samsung Electronics

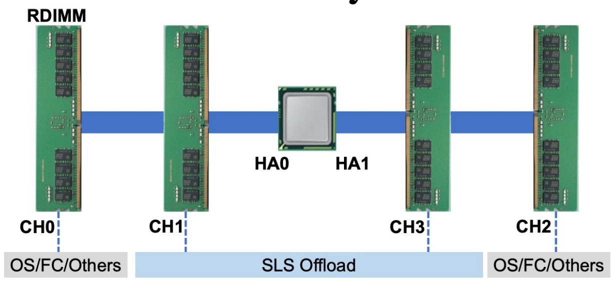
https://doi.org/10.1109/HCS52781.2021.9567191

Samsung AxDIMM (2021)

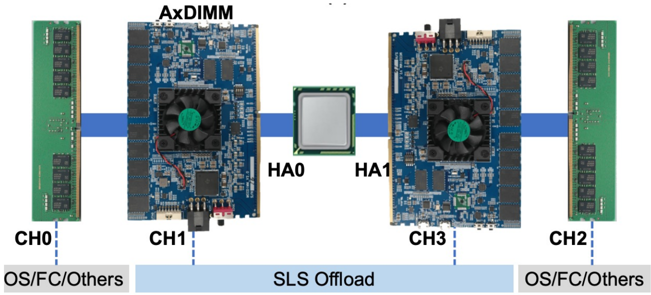
- DIMM-based PIM
  - DLRM recommendation system

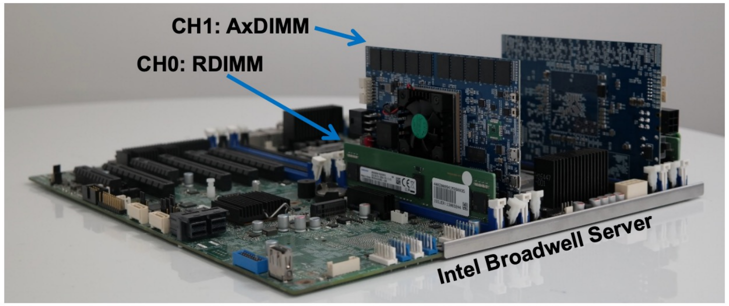


Baseline System



AxDIMM System





Ke et al. "Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM", IEEE Micro (2021)

Upcoming Lectures

- More real-world PIM architectures

- Programming PIM systems

- Workload characterization for PIM suitability

P&S Processing-in-Memory

Real-World Processing-in-Memory Architectures: Samsung HBM-PIM Architecture

Dr. Juan Gómez Luna

Prof. Onur Mutlu

ETH Zürich

Spring 2022

31 March 2022