P&S Heterogeneous Systems

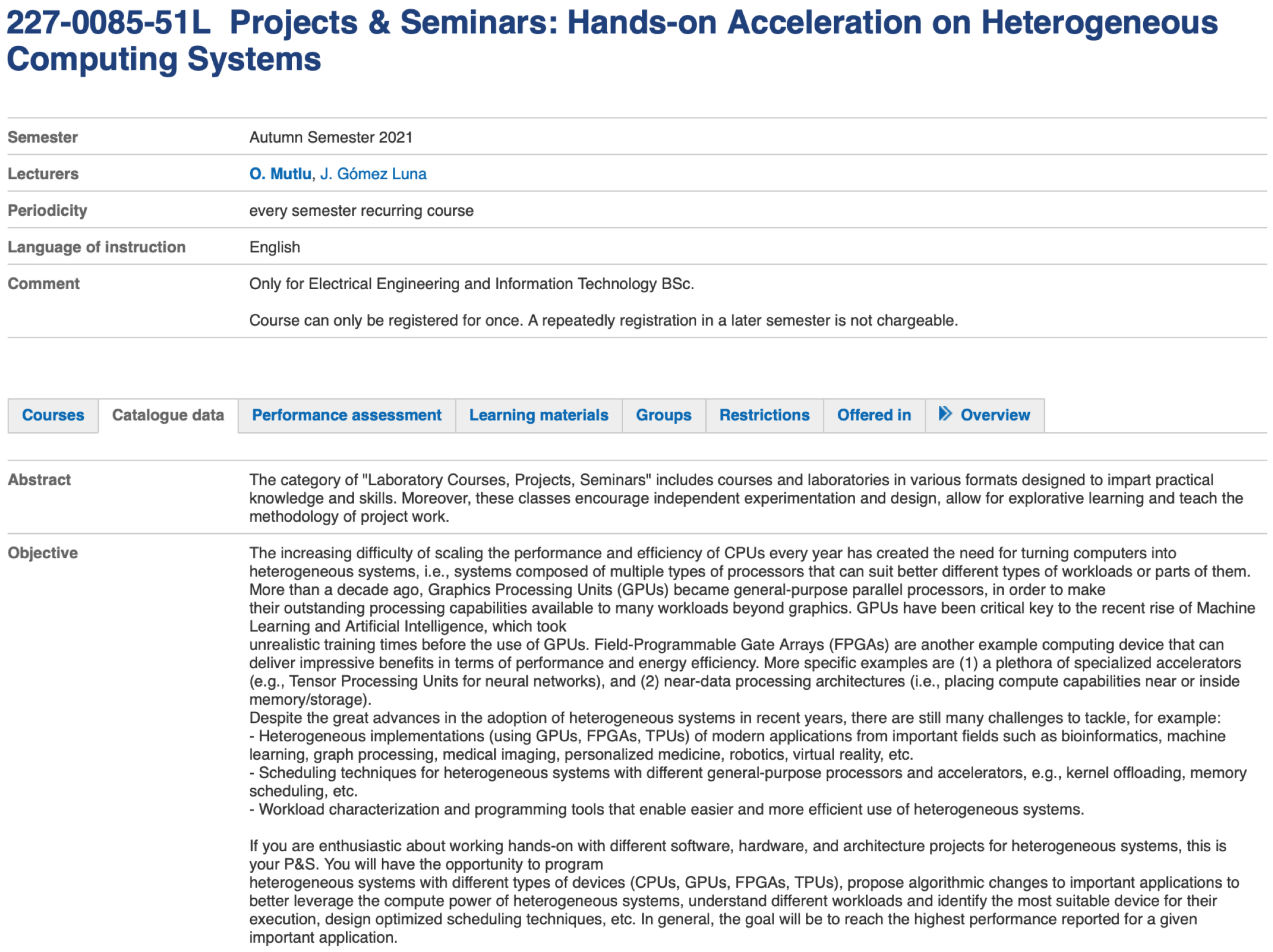
Hands-on Acceleration on Heterogeneous Computing Systems

Dr. Juan Gómez Luna
Prof. Onur Mutlu
ETH Zürich

Fall 2021

7 October 2021

P&S: Heterogeneous Systems (I)



227-0085-51L Projects & Seminars: Hands-on Acceleration on Heterogeneous Computing Systems

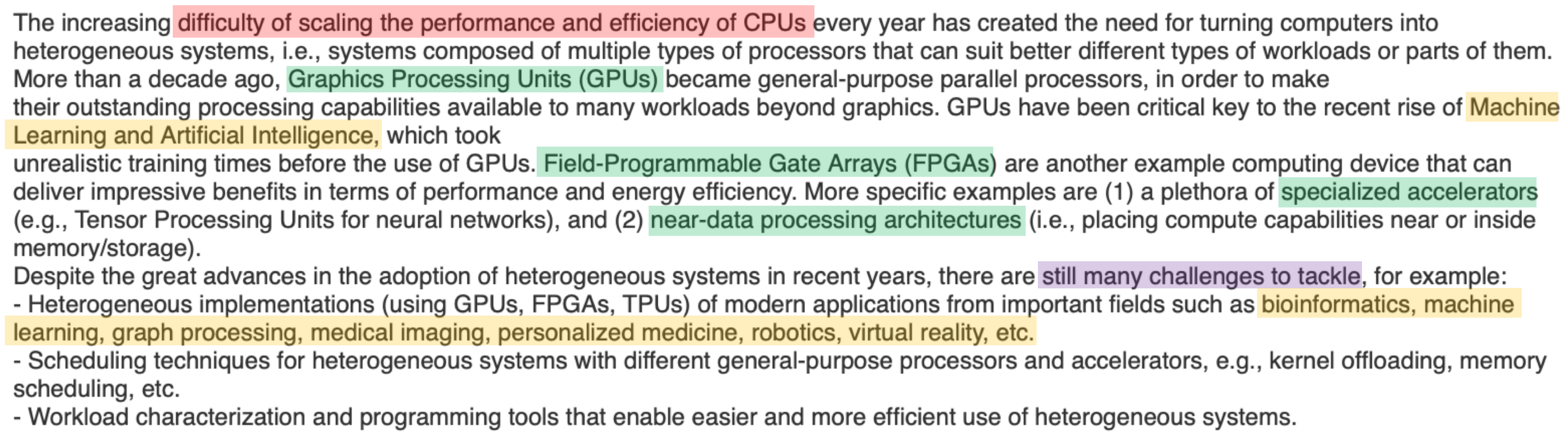
Semester	Autumn Semester 2021
Lecturers	**O. Mutlu**, J. Gómez Luna
Periodicity	every semester recurring course
Language of instruction	English
Comment	Only for Electrical Engineering and Information Technology BSc. Course can only be registered for once. A repeatedly registration in a later semester is not chargeable.

Courses | Catalogue data | Performance assessment | Learning materials | Groups | Restrictions | Offered in | Overview

Abstract	The category of "Laboratory Courses, Projects, Seminars" includes courses and laboratories in various formats designed to impart practical knowledge and skills. Moreover, these classes encourage independent experimentation and design, allow for explorative learning and teach the methodology of project work.
Objective	The increasing difficulty of scaling the performance and efficiency of CPUs every year has created the need for turning computers into heterogeneous systems, i.e., systems composed of multiple types of processors that can suit better different types of workloads or parts of them. More than a decade ago, Graphics Processing Units (GPUs) became general-purpose parallel processors, in order to make their outstanding processing capabilities available to many workloads beyond graphics. GPUs have been critical key to the recent rise of Machine Learning and Artificial Intelligence, which took unrealistic training times before the use of GPUs. Field-Programmable Gate Arrays (FPGAs) are another example computing device that can deliver impressive benefits in terms of performance and energy efficiency. More specific examples are (1) a plethora of specialized accelerators (e.g., Tensor Processing Units for neural networks), and (2) near-data processing architectures (i.e., placing compute capabilities near or inside memory/storage). Despite the great advances in the adoption of heterogeneous systems in recent years, there are still many challenges to tackle, for example: - Heterogeneous implementations (using GPUs, FPGAs, TPUs) of modern applications from important fields such as bioinformatics, machine learning, graph processing, medical imaging, personalized medicine, robotics, virtual reality, etc. - Scheduling techniques for heterogeneous systems with different general-purpose processors and accelerators, e.g., kernel offloading, memory scheduling, etc. - Workload characterization and programming tools that enable easier and more efficient use of heterogeneous systems. If you are enthusiastic about working hands-on with different software, hardware, and architecture projects for heterogeneous systems, this is your P&S. You will have the opportunity to program heterogeneous systems with different types of devices (CPUs, GPUs, FPGAs, TPUs), propose algorithmic changes to important applications to better leverage the compute power of heterogeneous systems, understand different workloads and identify the most suitable device for their execution, design optimized scheduling techniques, etc. In general, the goal will be to reach the highest performance reported for a given important application.

http://www.vvz.ethz.ch/Vorlesungsverzeichnis/lerneinheit.view?semkez=2021W&ansicht=KATALOGDATEN&lerneinheitId=154599&lang=en

P&S: Heterogeneous Systems (II)



The increasing difficulty of scaling the performance and efficiency of CPUs every year has created the need for turning computers into heterogeneous systems, i.e., systems composed of multiple types of processors that can suit better different types of workloads or parts of them. More than a decade ago, Graphics Processing Units (GPUs) became general-purpose parallel processors, in order to make their outstanding processing capabilities available to many workloads beyond graphics. GPUs have been critical key to the recent rise of Machine Learning and Artificial Intelligence, which took unrealistic training times before the use of GPUs. Field-Programmable Gate Arrays (FPGAs) are another example computing device that can deliver impressive benefits in terms of performance and energy efficiency. More specific examples are (1) a plethora of specialized accelerators (e.g., Tensor Processing Units for neural networks), and (2) near-data processing architectures (i.e., placing compute capabilities near or inside memory/storage).

Despite the great advances in the adoption of heterogeneous systems in recent years, there are still many challenges to tackle, for example:

- Heterogeneous implementations (using GPUs, FPGAs, TPUs) of modern applications from important fields such as bioinformatics, machine learning, graph processing, medical imaging, personalized medicine, robotics, virtual reality, etc.
- Scheduling techniques for heterogeneous systems with different general-purpose processors and accelerators, e.g., kernel offloading, memory scheduling, etc.
- Workload characterization and programming tools that enable easier and more efficient use of heterogeneous systems.

Flynn's Taxonomy of Computers

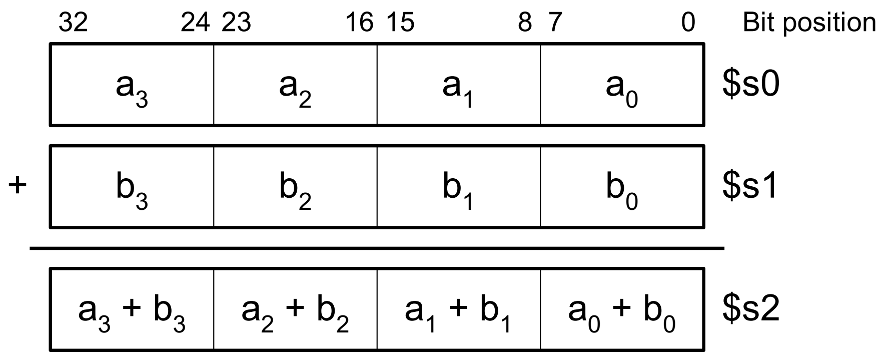
- Mike Flynn, "Very High-Speed Computing Systems," Proc. of IEEE, 1966

- 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

SIMD ISA Extensions

- Single Instruction Multiple Data (SIMD) extension instructions
  - Single instruction acts on multiple pieces of data at once
  - Common application: graphics
  - Perform short arithmetic operations (also called *packed arithmetic*)
- For example: add four 8-bit numbers
- Must modify ALU to eliminate carries between 8-bit values

```
padd8 $s2, $s0, $s1
```



Intel Pentium MMX Operations

- Idea: One instruction operates on multiple data elements simultaneously
  - *À la* array processing (yet much more limited)
  - Designed with multimedia (graphics) operations in mind

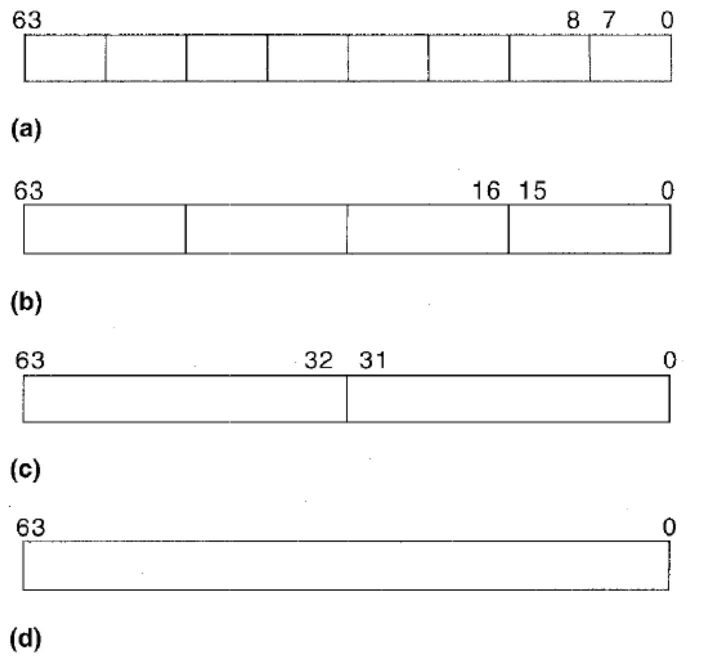


Figure 1. MMX technology data types: packed byte (a), packed word (b), packed doubleword (c), and quadword (d).

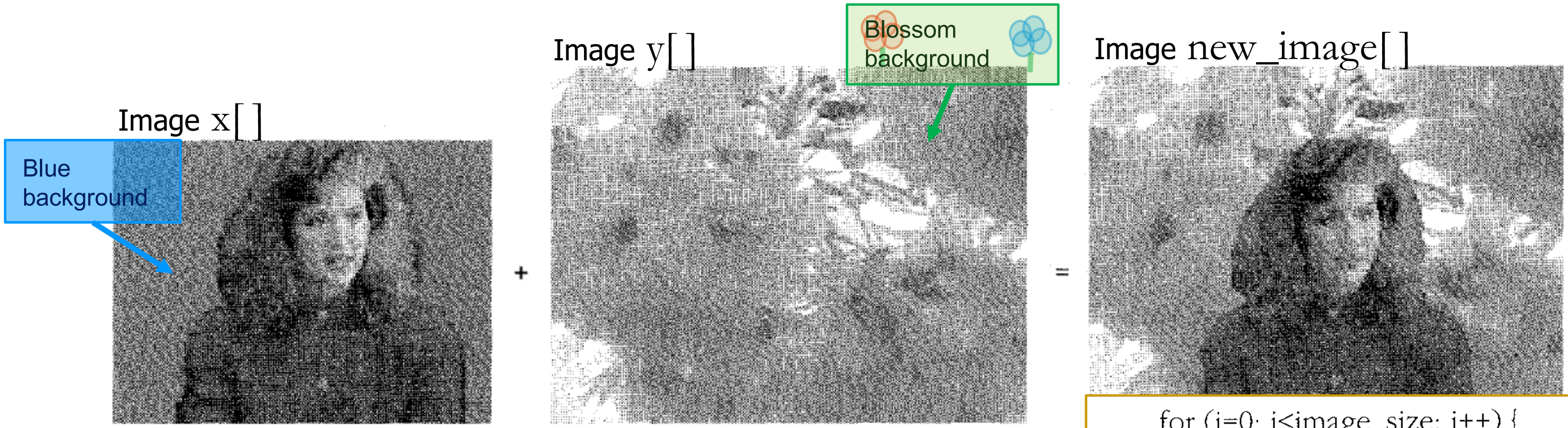
No VLEN register
Opcode determines data type:
8 8-bit bytes
4 16-bit words
2 32-bit doublewords
1 64-bit quadword

Stride is always equal to 1.

Peleg and Weiser, "MMX Technology Extension to the Intel Architecture," IEEE Micro, 1996.

MMX Example: Image Overlaying (I)

- Goal: Overlay the human in image x on top of the background in image y



```
for (i=0; i<image_size; i++) {
if (x[i] == Blue)  new_image[i] =y[i];
        else new_image[i] = x[i];
}
```

Figure 8. Chroma keying: image overlay using a background color.

PCMPEQB MM1, MM3

	MM1	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Image x[ ]	MM3	X7!=blue	X6!=blue	X5=blue	X4=blue	X3!=blue	X2!=blue	X1=blue	X0=blue
Bit mask	MM1	0x0000	0x0000	0xFFFF	0xFFFF	0x0000	0x0000	0xFFFF	0xFFFF

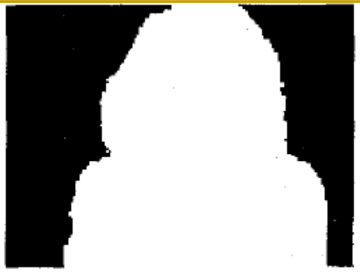
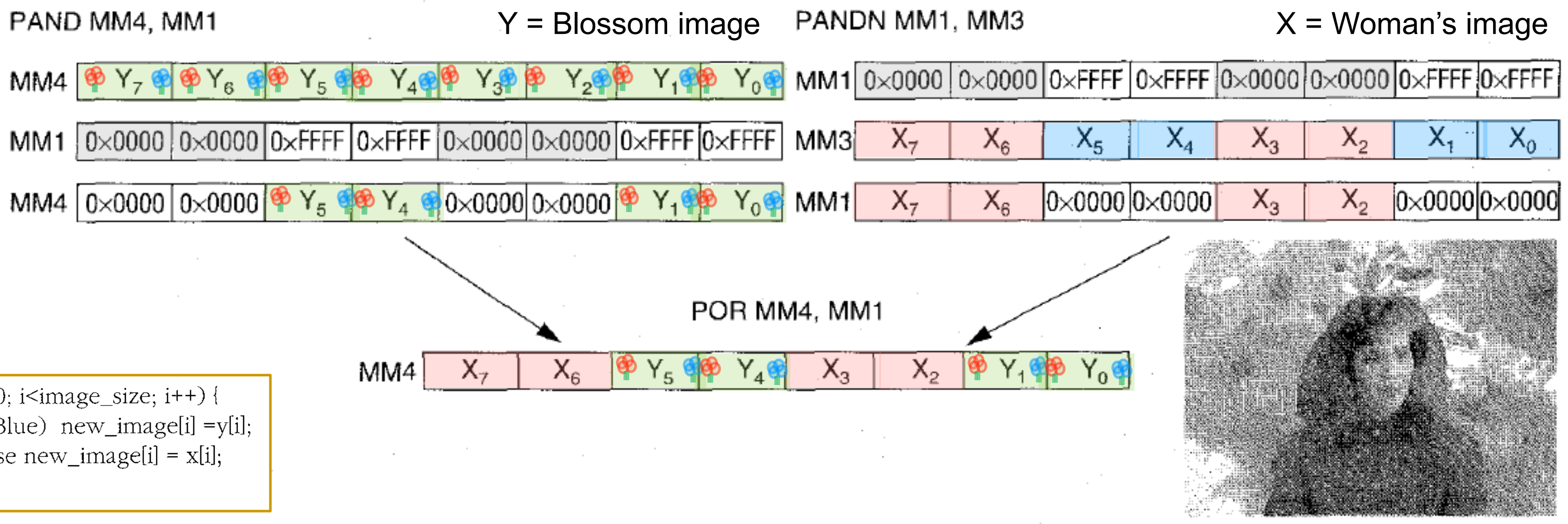

Bitmask

Figure 9. Generating the selection bit mask.

MMX Example: Image Overlaying (II)


PAND MM4, MM1
Y = Blossom image
PANDN MM1, MM3
X = Woman's image
MM4 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
MM1 0x0000 0x0000 0xFFFF 0xFFFF 0x0000 0x0000 0xFFFF 0xFFFF
MM4 0x0000 0x0000 Y5 Y4 0x0000 0x0000 Y1 Y0
MM1 0x0000 0x0000 0xFFFF 0xFFFF 0x0000 0x0000 0xFFFF 0xFFFF
MM3 X7 X6 X5 X4 X3 X2 X1 X0
MM1 X7 X6 0x0000 0x0000 X3 X2 0x0000 0x0000
POR MM4, MM1
MM4 X7 X6 Y5 Y4 X3 X2 Y1 Y0

```
for (i=0; i<image_size; i++) {
if (x[i] == Blue)  new_image[i] =y[i];
        else new_image[i] = x[i];
    }
```

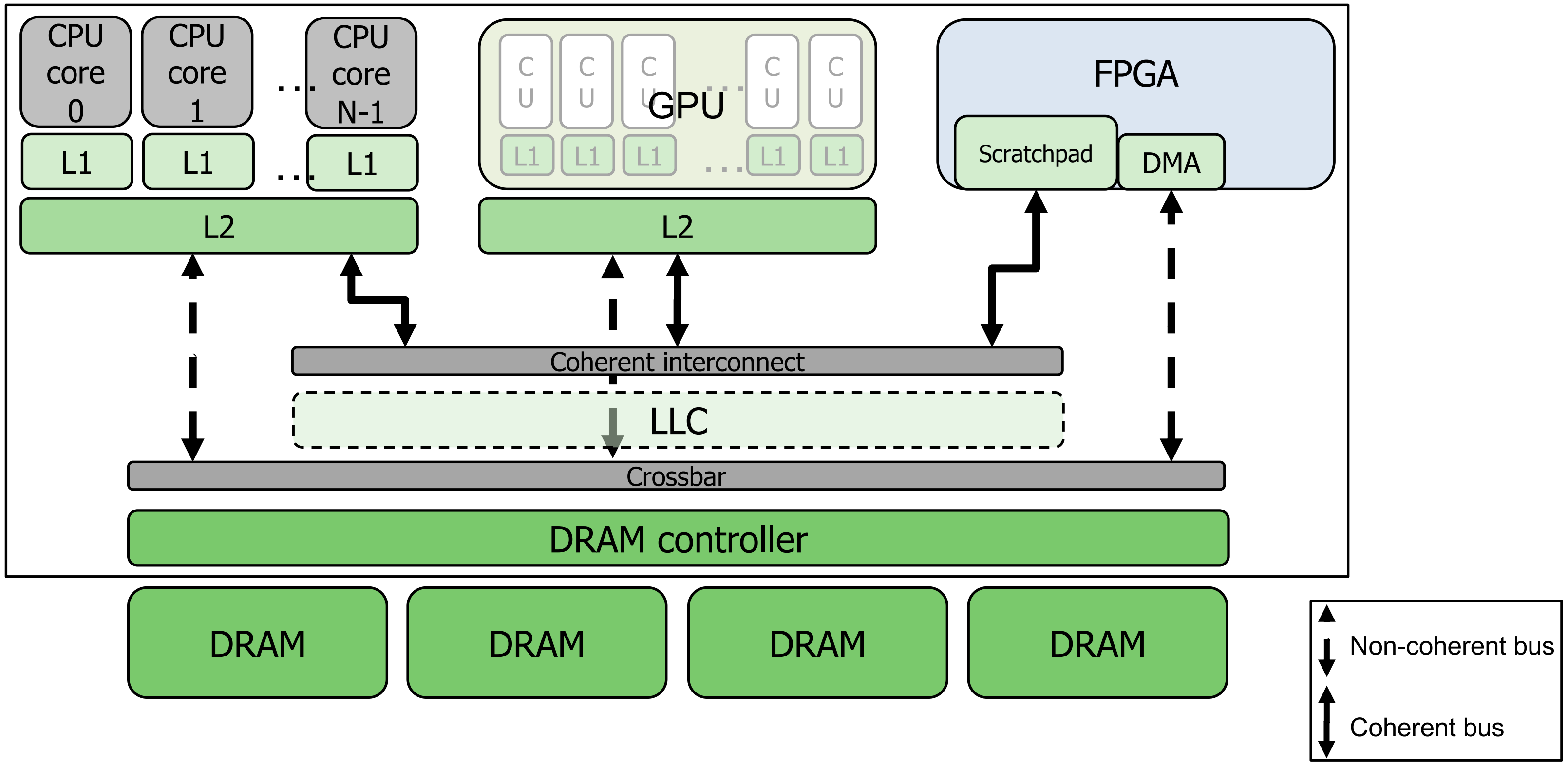
Figure 10. Using the mask with logical MMX instructions to perform a conditional select.

```
Movq       mm3, mem1    /* Load eight pixels from
                           woman's image
Movq       mm4, mem2    /* Load eight pixels from the
                           blossom image
Pcmpeqb    mm1, mm3
Pand       mm4, mm1
Pandn      mm1, mm3
Por        mm4, mm1
```

Figure 11. MMX code sequence for performing a conditional select.

Heterogeneous Computing Systems

- The end of Moore's law created the need for heterogeneous systems
  - More suitable devices for each type of workload
  - Increased performance and energy efficiency



Goals of this P&S Course

P&S Heterogeneous Systems: Contents

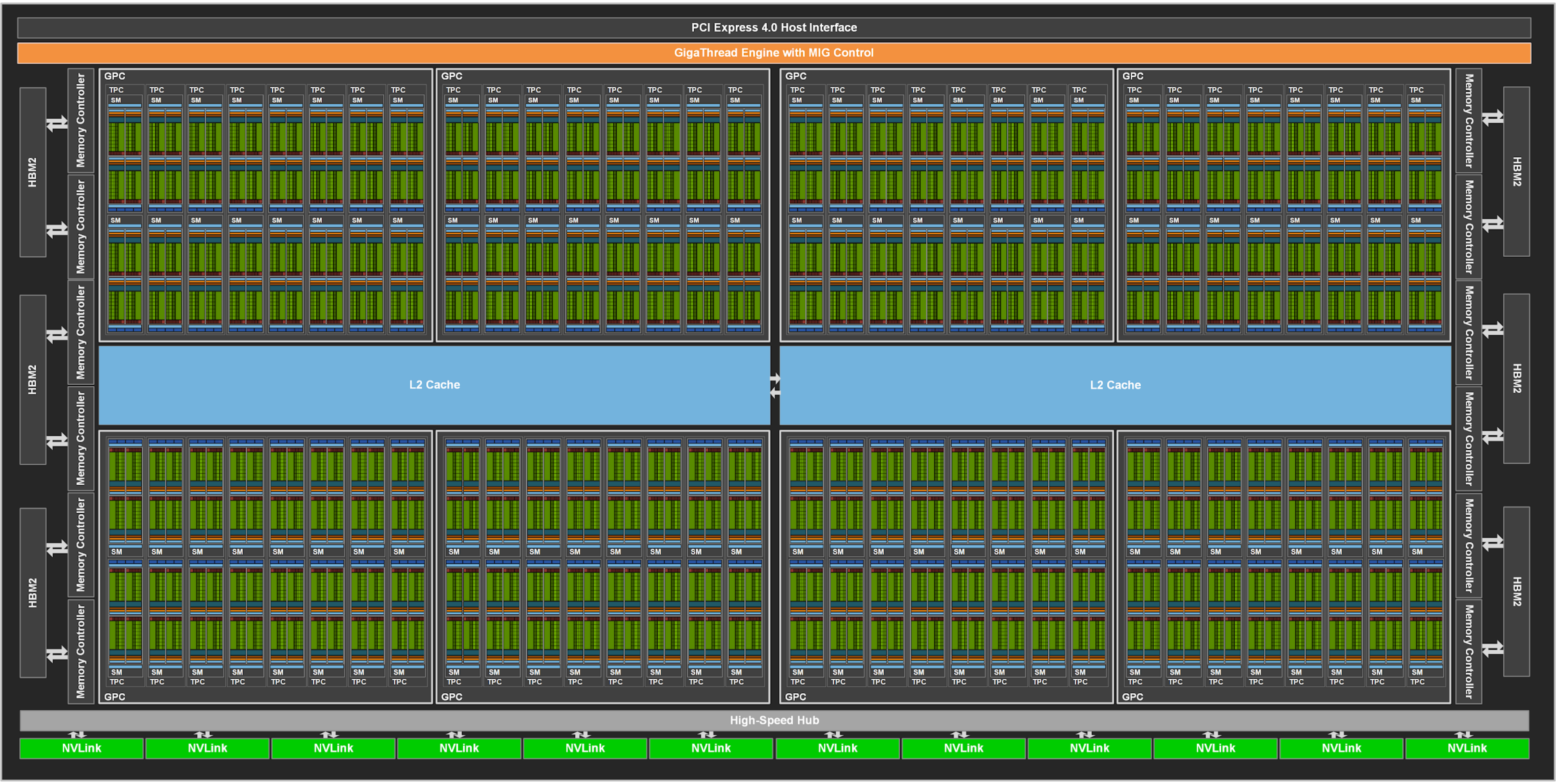
- We will introduce the need for heterogeneity in current computing systems, in order to achieve high performance and energy efficiency

- You will get familiar with some of the different heterogeneous devices that are available in computing systems

- You will learn workload distribution and parallelization strategies that leverage heterogeneous devices

- You will work hands-on: analyzing workloads, programming heterogeneous architectures, proposing scheduling/offloading mechanisms, etc.

NVIDIA A100 (2020)



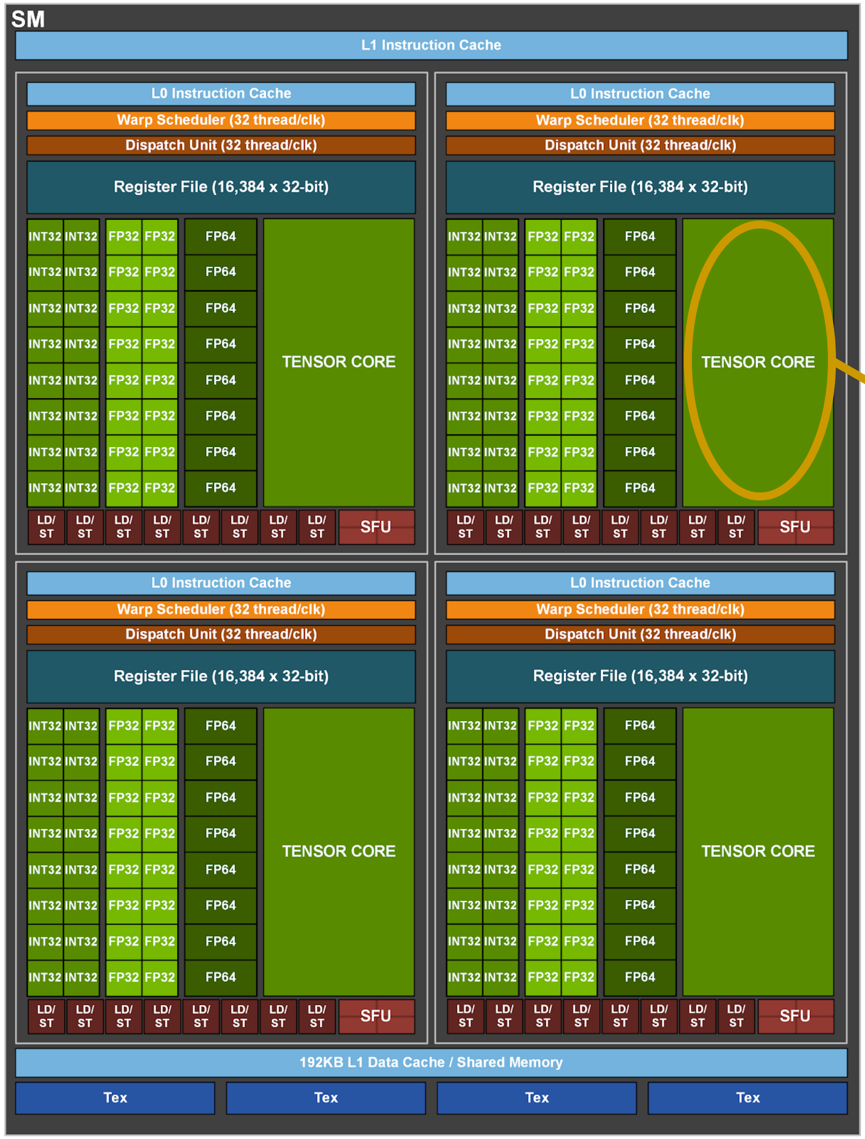
https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

108 cores on the A100

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

40MB L2 cache

NVIDIA A100 Core

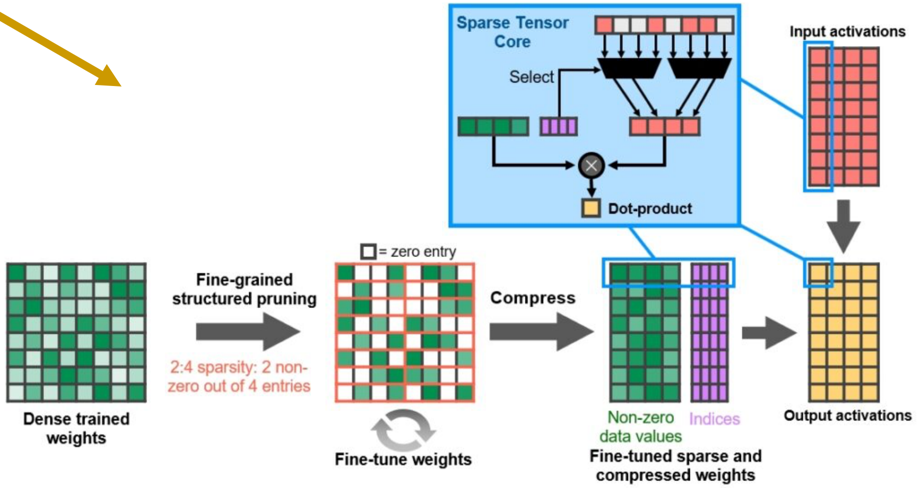

SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE
LD/ ST
SFU
192KB L1 Data Cache / Shared Memory
Tex

GPU compute throughput:

19.5 TFLOPS Single Precision

9.7 TFLOPS Double Precision

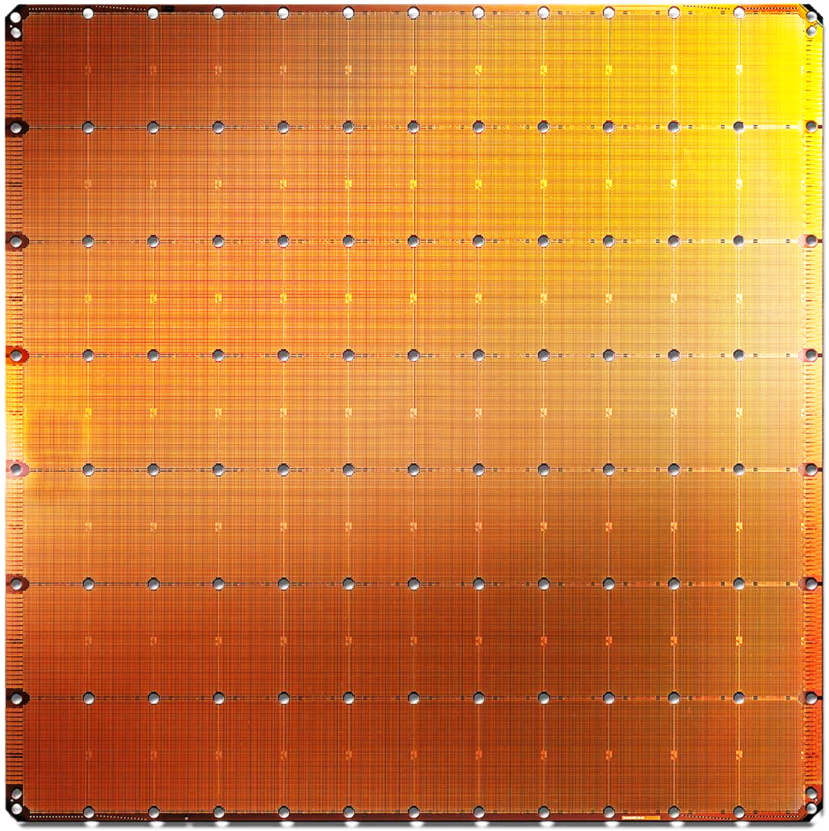
312 TFLOPS for Deep Learning (Tensor cores)



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

Cerebras's Wafer Scale Engine (2019)

- The largest ML accelerator chip (2019)
- 400,000 cores



Cerebras WSE
1.2 Trillion transistors
46,225 mm^2

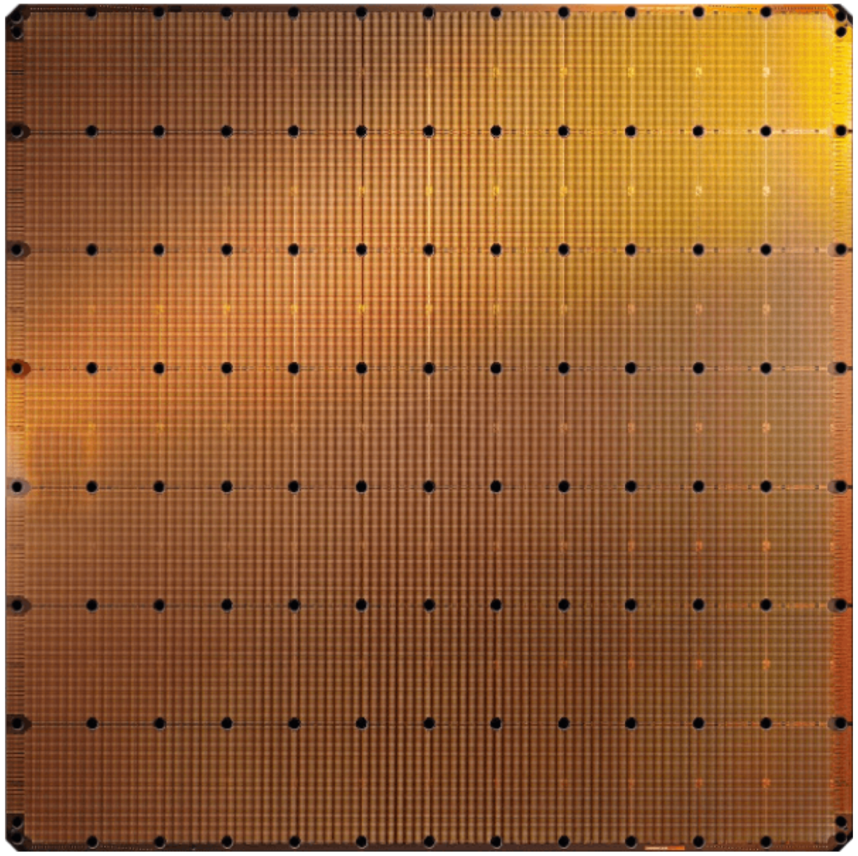
Largest GPU
21.1 Billion transistors
815 mm^2
NVIDIA TITAN V

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Cerebras's Wafer Scale Engine-2 (2021)

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



Cerebras WSE-2
2.6 Trillion transistors
46,225 mm^2

Largest GPU
54.2 Billion transistors
826 mm^2
NVIDIA Ampere GA100

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Google TPU Generation I (~2016)

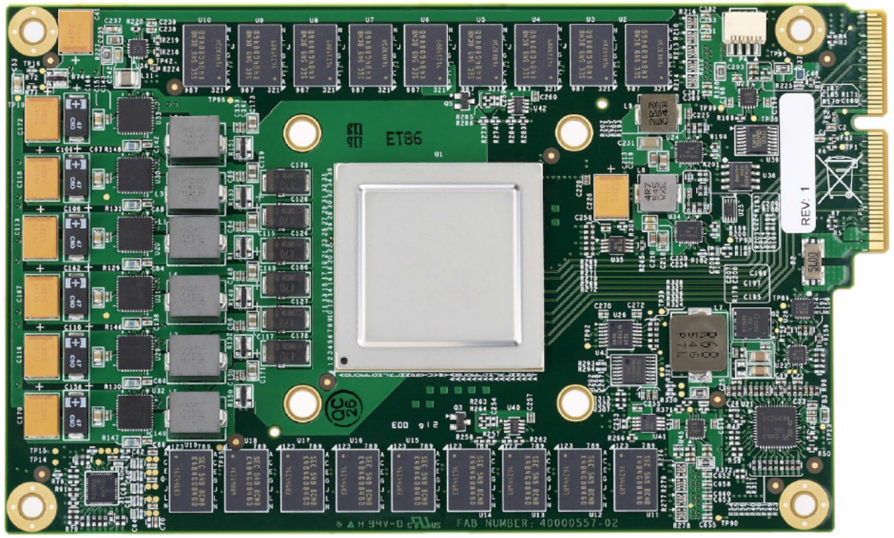


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

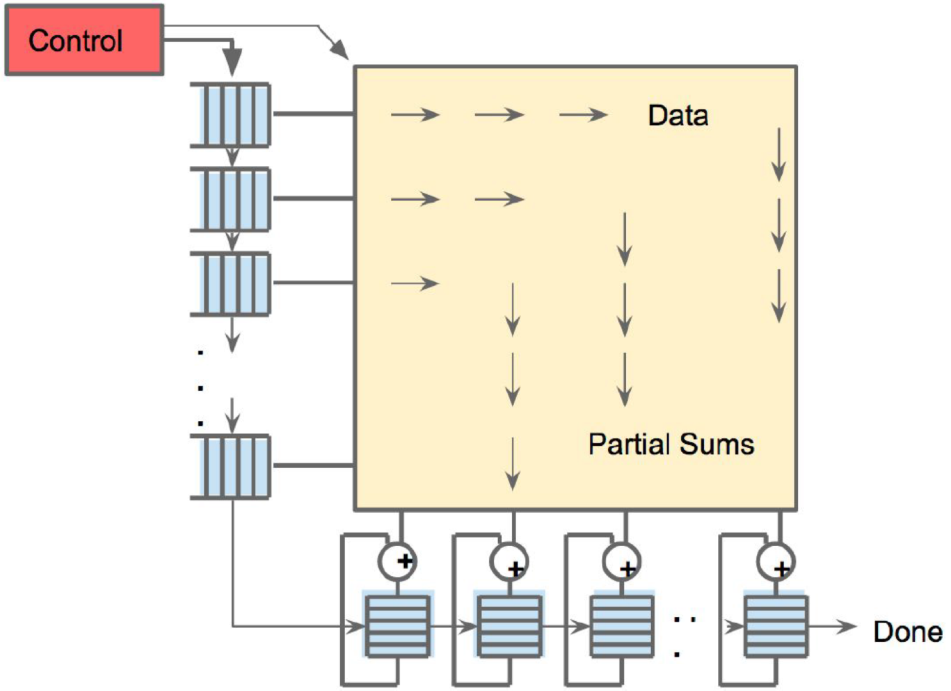
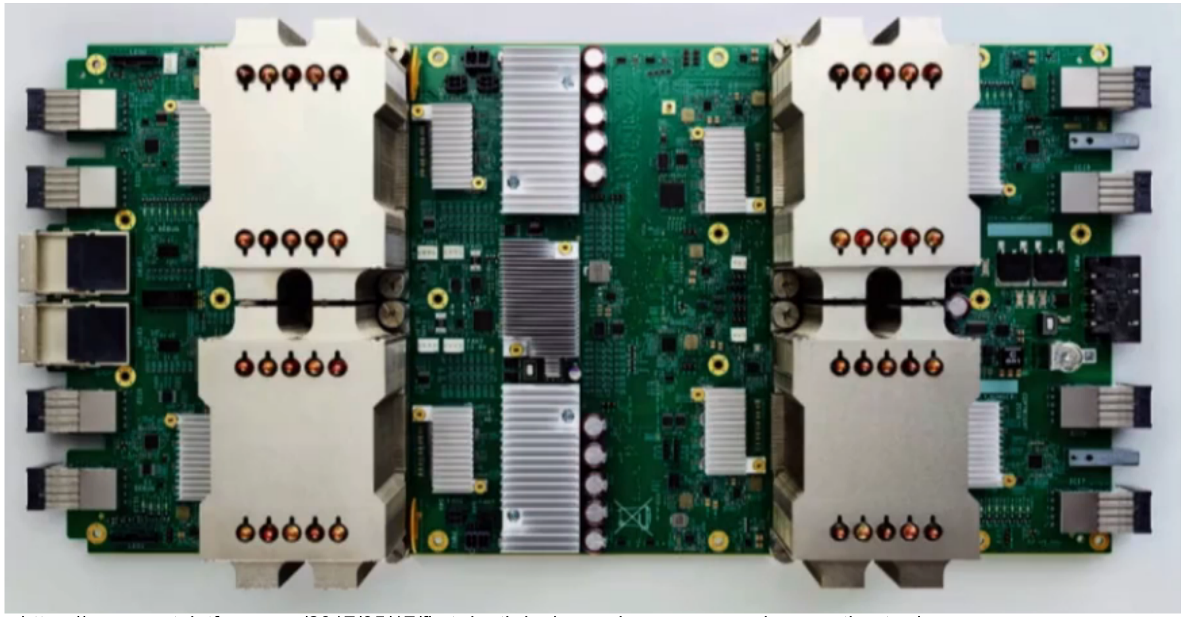


Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

Google TPU Generation II (2017)


https://www.nextplatform.com/2017/05/17/first-depth-look-googles-new-second-generation-tpu/

4 TPU chips
vs 1 chip in TPU1

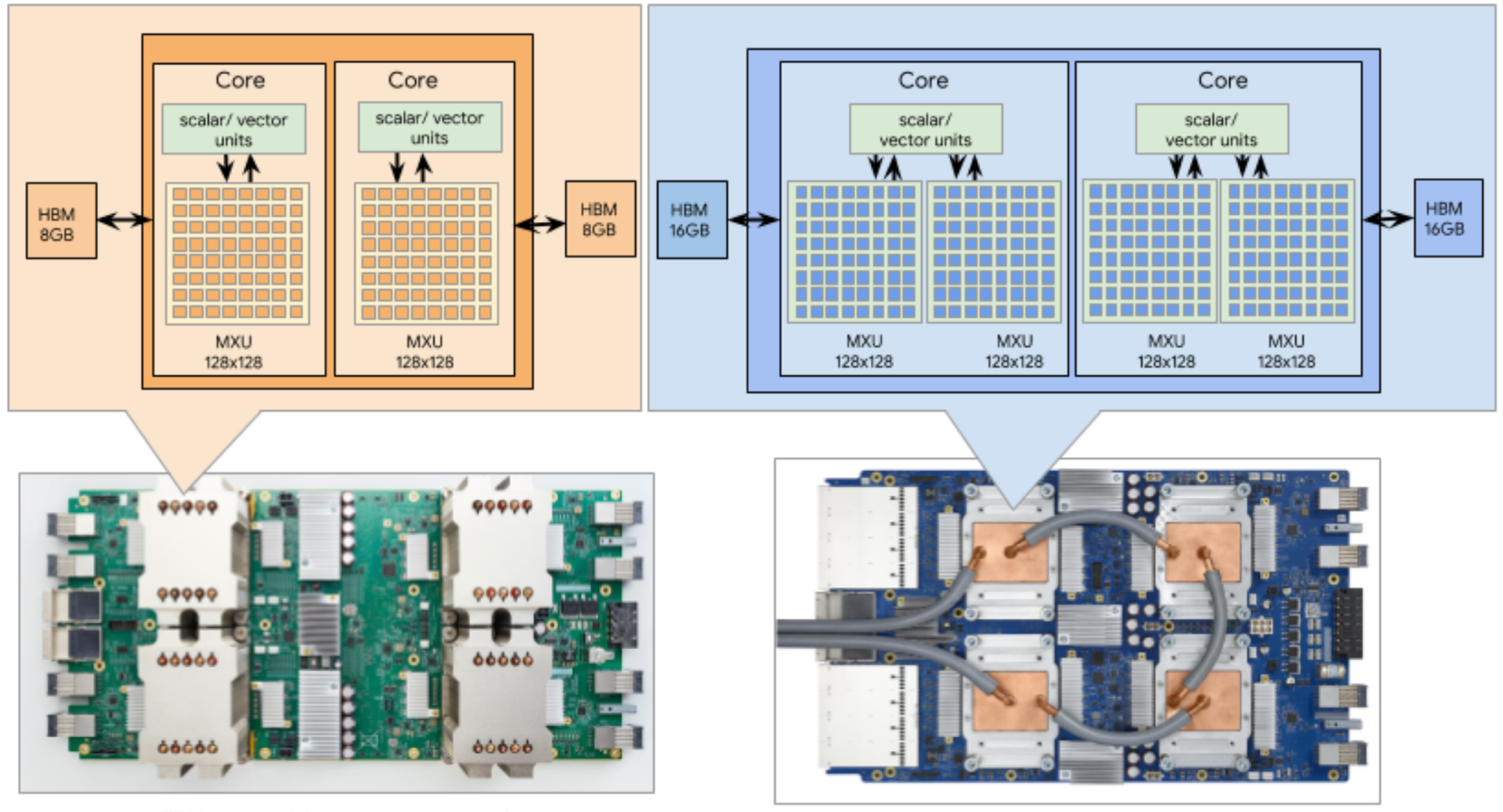
High Bandwidth Memory
vs DDR3

Floating point operations
vs FP16

45 TFLOPS per chip
vs 23 TOPS

Designed for training and inference
vs only inference

Google TPU Generation III (2019)



TPU v2 - 4 chips, 2 cores per chip

TPU v3 - 4 chips, 2 cores per chip

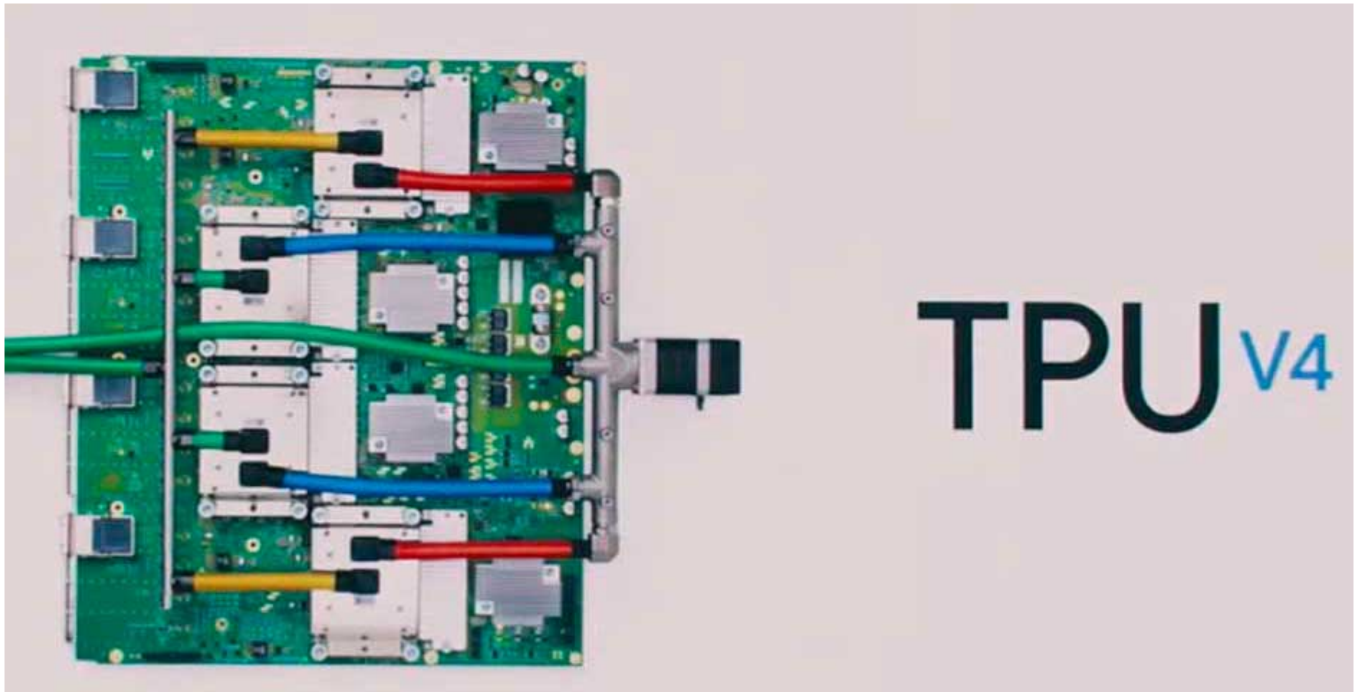
32GB HBM per chip
vs 16GB HBM in TPU2

4 Matrix Units per chip
vs 2 Matrix Units in TPU2

90 TFLOPS per chip
vs 45 TFLOPS in TPU2

https://cloud.google.com/tpu/docs/system-architecture

Google TPU Generation IV (2019)



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

250 TFLOPS per chip in 2021
vs 90 TFLOPS in TPU3

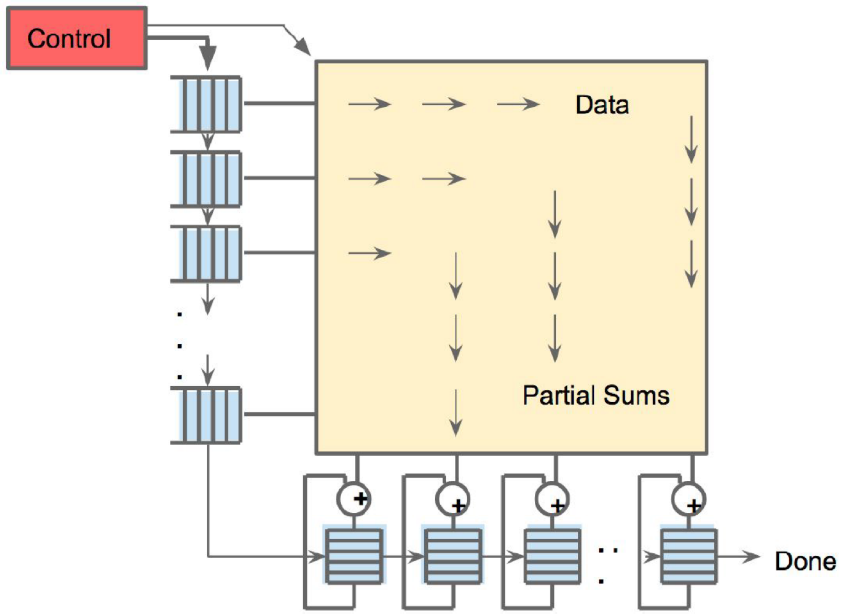
$\Downarrow$

1 ExaFLOPS per board

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

An Example Modern Systolic Array: TPU (II)

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



Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

An Example Modern Systolic Array: TPU (III)

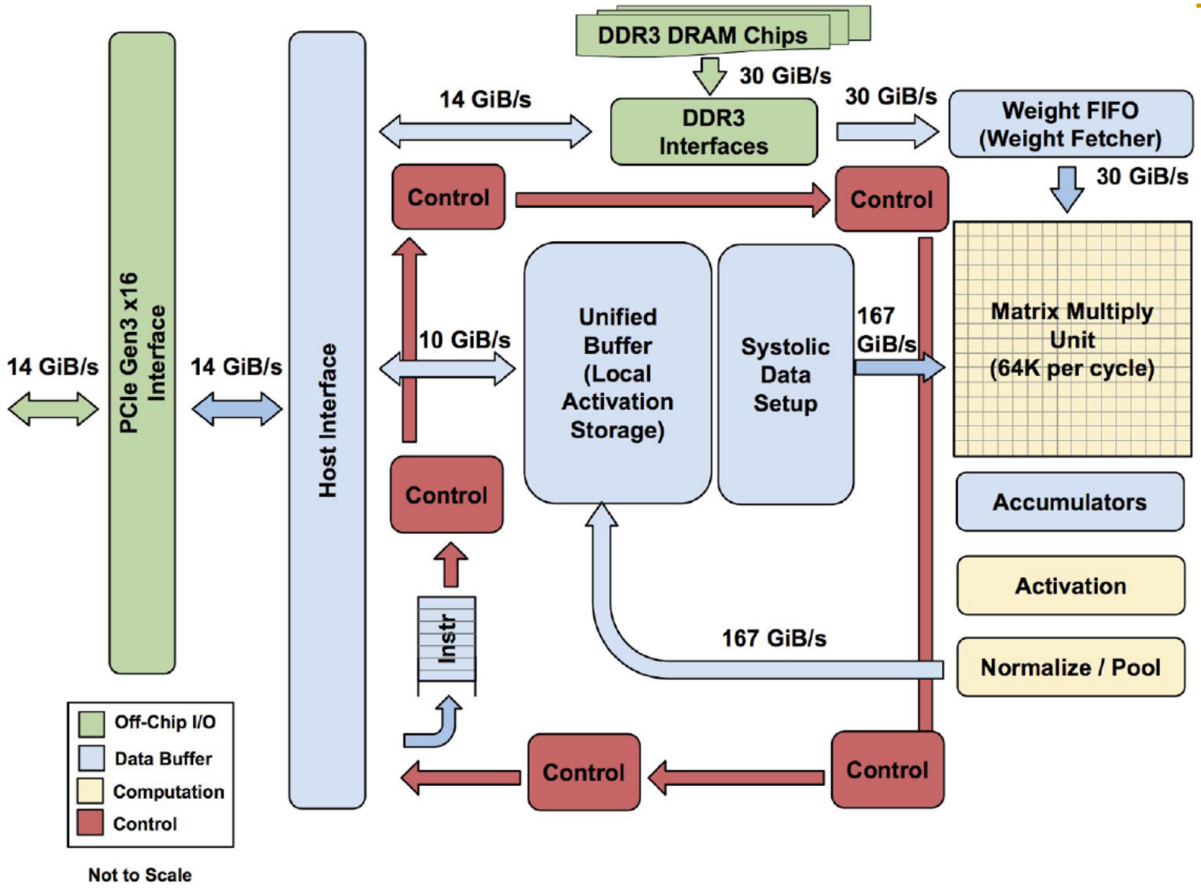
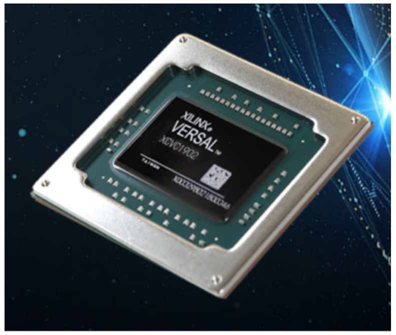
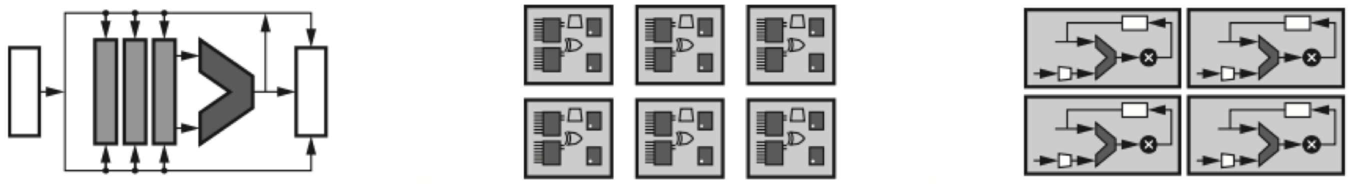


Figure 1. TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on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

Xilinx Versal ACAP (2020) (I)

- Three compute engines inside the same chip
  - Different workloads, different devices

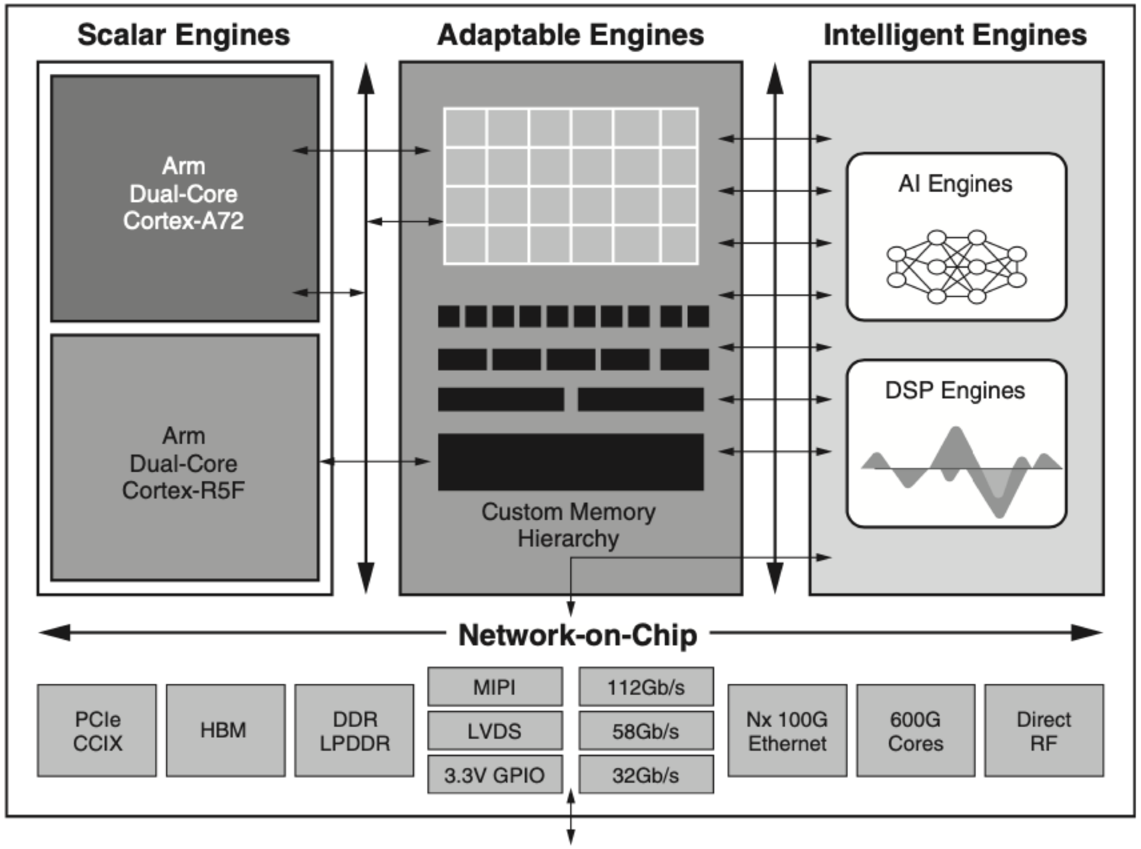




Scalar Processing	Adaptable Hardware	Vector Processing *(e.g., GPU, DSP)*
Complex Algorithms and Decision Making	Processing of Irregular Data Structures *Genomic Sequencing*	Domain-specific Parallelism
	Latency Critical Workloads *Real-Time Control*	Signal Processing *Complex Math, Convolutions*
	Sensor Fusion *Pre-processing, Programmable I/O*	Video and Image Processing

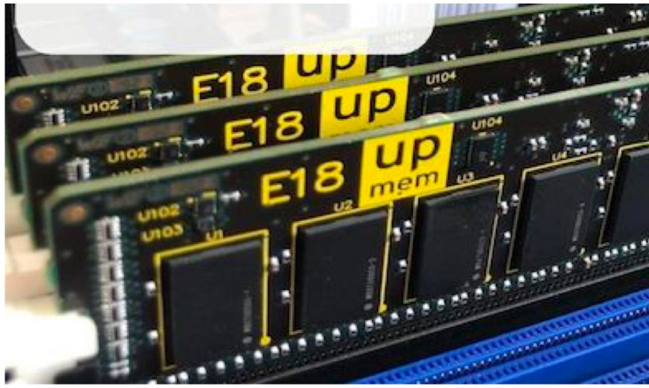
Xilinx Versal ACAP (2020) (II)

- Three compute engines inside the same chip
  - Scalar cores, reconfigurable engines, vector processors

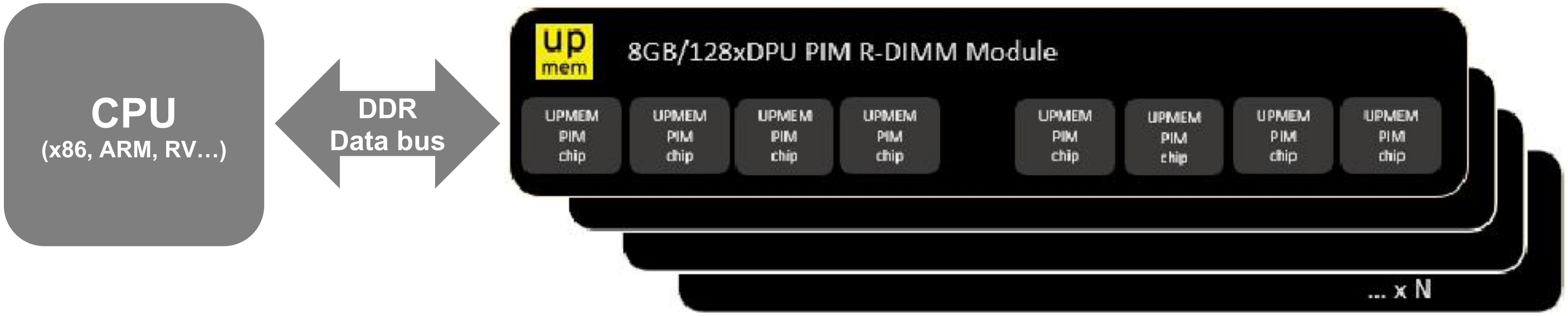


UPMEM Processing-in-DRAM Engine (2019)

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



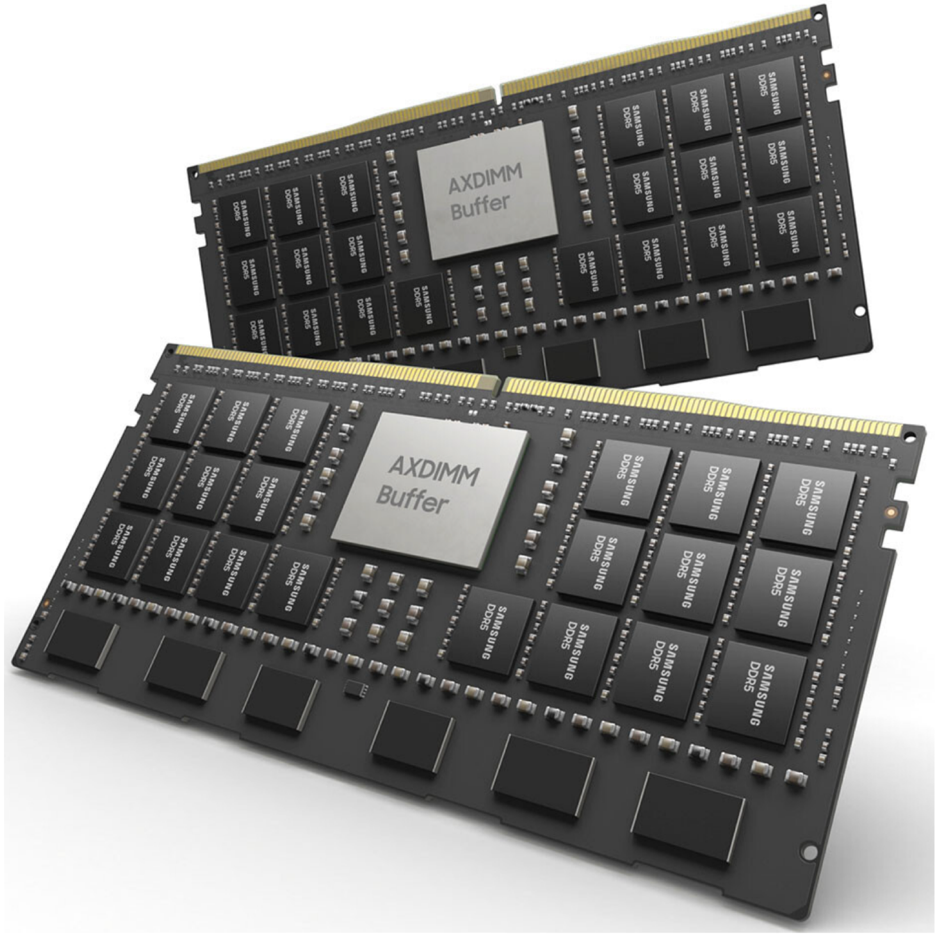
- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth

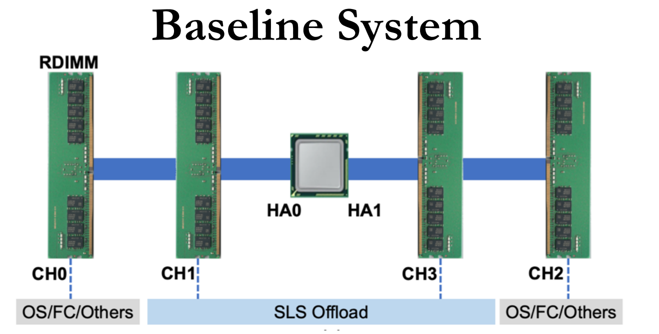


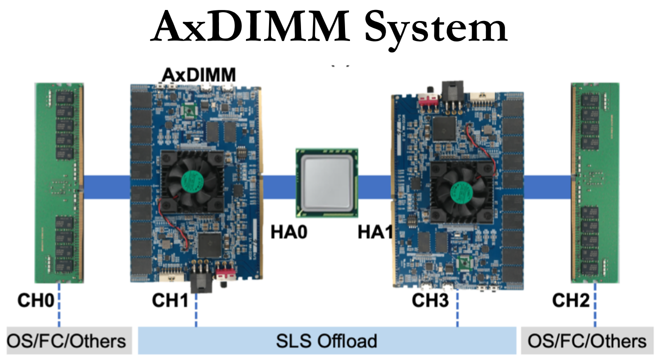
https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem
https://www.upmem.com/video-upmem-presenting-its-true-processing-in-memory-solution-hot-chips-2019/

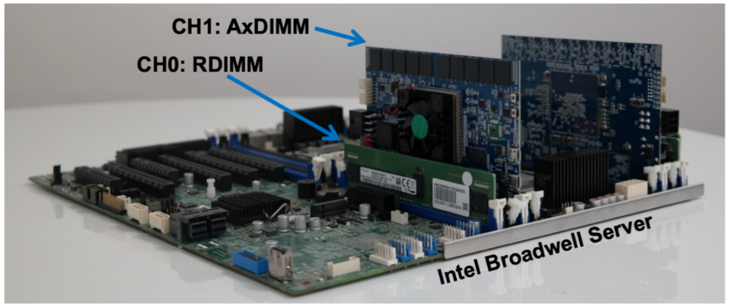
Samsung AxDIMM (2021)

- DDR5-PIM
  - DLRM recommendation system









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

Key Takeaways

- This P&S is aimed at improving your
  - Knowledge in Computer Architecture and Heterogeneous Systems
  - Technical skills in programming heterogeneous architectures
  - Critical thinking and analysis
  - Interaction with a nice group of researchers
  - Familiarity with key research directions
  - Technical presentation of your project

Key Goal

(Learn how to) take advantage of existing heterogeneous devices by programming them, analyzing workloads, proposing offloading/scheduling techniques…

Prerequisites of the Course

- Digital Design and Computer Architecture (or equivalent course)

- Familiarity with C/C++ programming
  - FPGA implementation or GPU programming (desirable)

- Interest in
  - computer architectures and computing paradigms
  - discovering why things do or do not work and solving problems
  - making systems efficient and usable

Course Info: Who Are We? (I)

- Onur Mutlu
  - Full Professor @ ETH Zurich ITET (INFK), since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-…
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - https://people.inf.ethz.ch/omutlu/
  - omutlu@gmail.com (Best way to reach me)
  - https://people.inf.ethz.ch/omutlu/projects.htm

- Research and Teaching in:
  - Computer architecture, computer systems, hardware security, bioinformatics
  - Memory and storage systems
  - Hardware security, safety, predictability
  - Fault tolerance
  - Hardware/software cooperation
  - Architectures for bioinformatics, health, medicine
  - …

Course Info: Who Are We? (II)

- Lead Supervisor:
  - Dr. Juan Gómez Luna

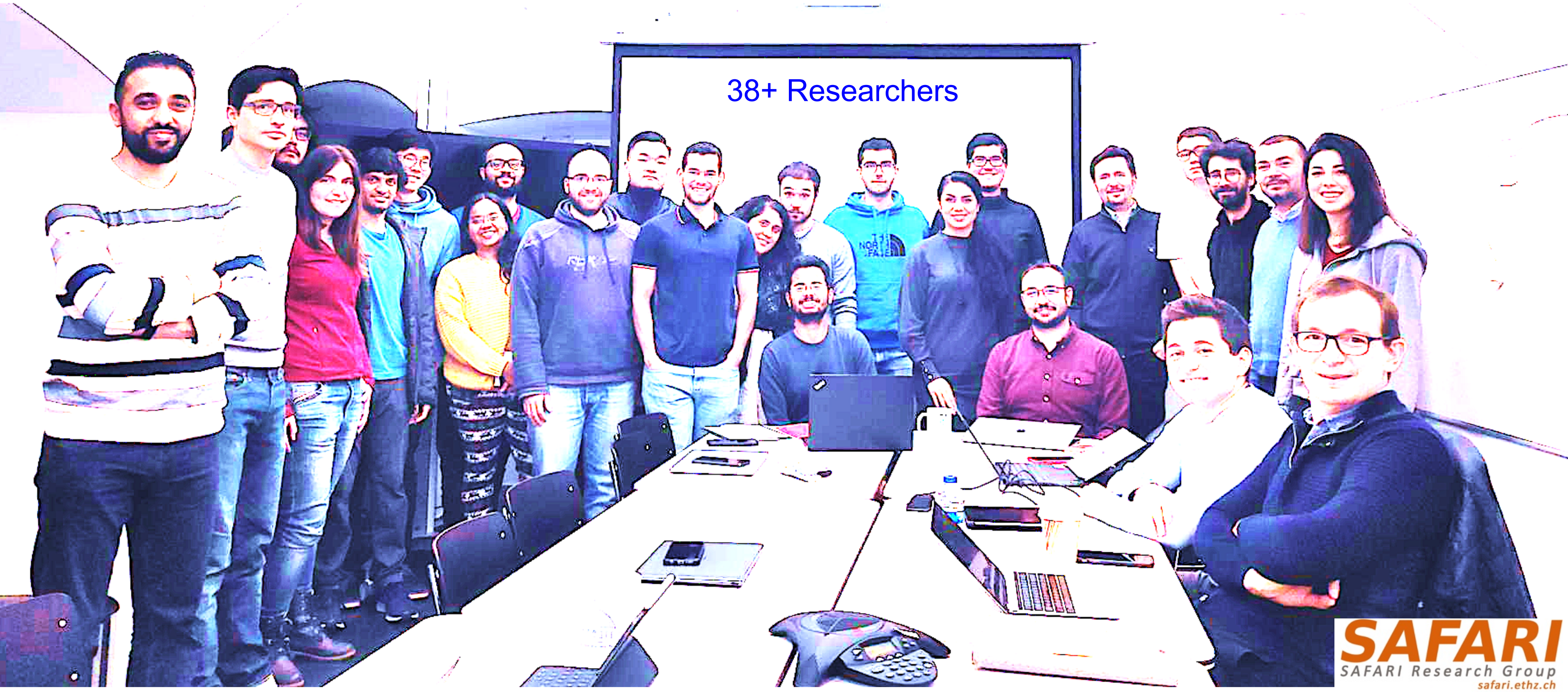
- Supervisors:
  - Dr. Mohammed Alser
  - Dr. Behzad Salami
  - Dr. Gagandeep Singh

- Get to know us and our research
  - https://safari.ethz.ch/safari-group/

Onur Mutlu's SAFARI Research Group

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

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



Think BIG, Aim HIGH!

https://safari.ethz.ch

SAFARI Newsletter January 2021 Edition

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

SAFARI
SAFARI Research Group

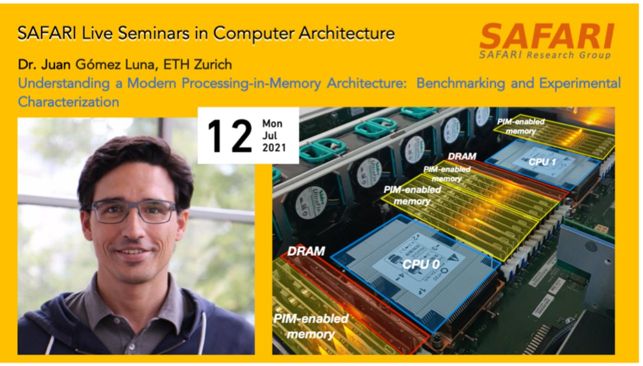
Newsletter
January 2021

Think Big, Aim High, and Have a Wonderful 2021!

Dear SAFARI friends,

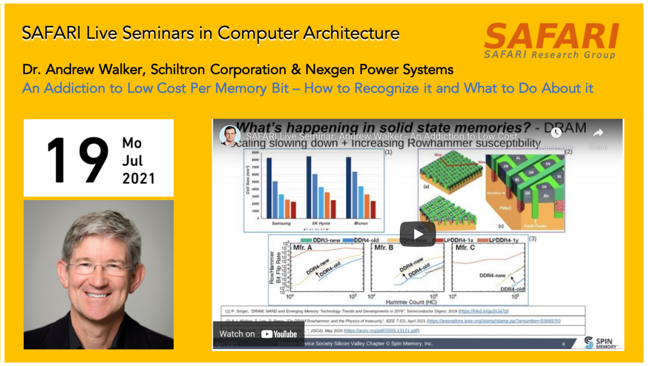
Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 **here**). 2020 has

SAFARI Live Seminars (I)


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Dr. Juan Gómez Luna, ETH Zurich
Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

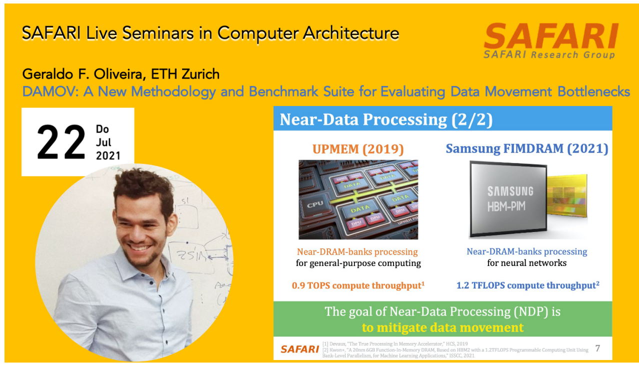
12 Mon Jul 2021


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Dr. Andrew Walker, Schiltron Corporation & Nexgen Power Systems
An Addiction to Low Cost Per Memory Bit – How to Recognize it and What to Do About it

19 Mo Jul 2021

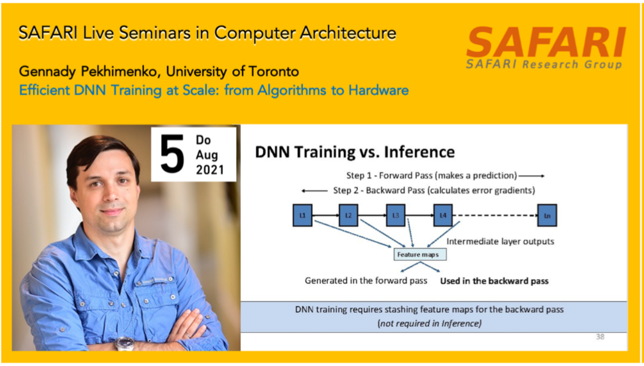
What's happening in solid state memories? - DRAM scaling slowing down + Increasing Rowhammer susceptibility
Watch on YouTube
SPIN MEMORY


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Geraldo F. Oliveira, ETH Zurich
DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

22 Do Jul 2021

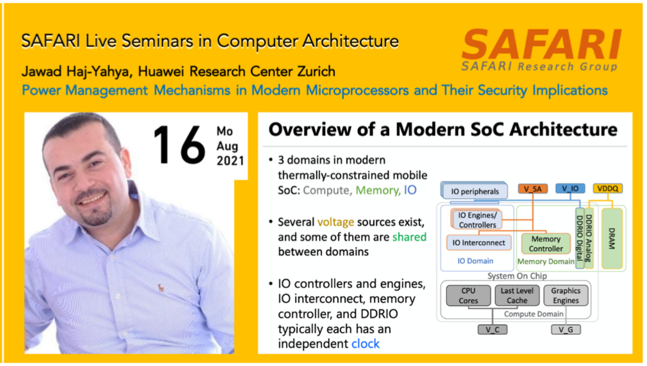
Near-Data Processing (2/2)
UPMEM (2019)
Samsung FIMDRAM (2021)
Near-DRAM-banks processing for general-purpose computing
Near-DRAM-banks processing for neural networks
0.9 TOPS compute throughput
1.2 TFLOPS compute throughput
The goal of Near-Data Processing (NDP) is to mitigate data movement


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Gennady Pekhimenko, University of Toronto
Efficient DNN Training at Scale: from Algorithms to Hardware

5 Do Aug 2021

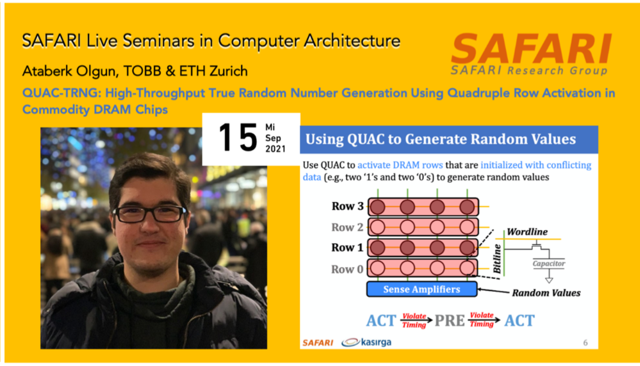
DNN Training vs. Inference
Step 1 - Forward Pass (makes a prediction)
Step 2 - Backward Pass (calculates error gradients)
Intermediate layer outputs
Feature maps
Generated in the forward pass
Used in the backward pass
DNN training requires stashing feature maps for the backward pass
(not required in Inference)


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Jawad Haj-Yahya, Huawei Research Center Zurich
Power Management Mechanisms in Modern Microprocessors and Their Security Implications

16 Mo Aug 2021

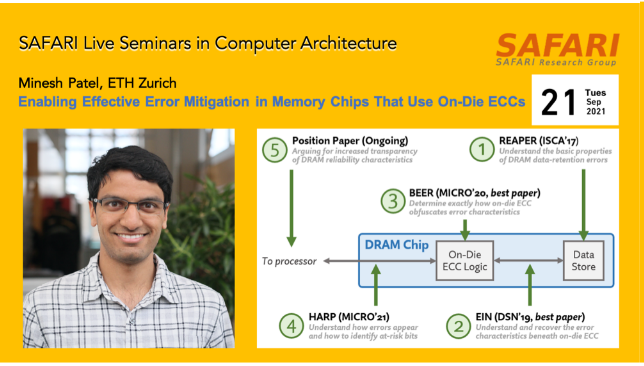
Overview of a Modern SoC Architecture
3 domains in modern thermally-constrained mobile SoC: Compute, Memory, IO
Several voltage sources exist, and some of them are shared between domains
IO controllers and engines, IO interconnect, memory controller, and DDRIO typically each has an independent clock
IO peripherals
IO Engines/Controllers
IO Interconnect
IO Domain
Memory Controller
Memory Domain
DDRIO Analog
DDRIO Digital
DRAM
System On Chip
CPU Cores
Last Level Cache
Graphics Engines
Compute Domain


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Ataberk Olgun, TOBB & ETH Zurich
QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

15 Mi Sep 2021

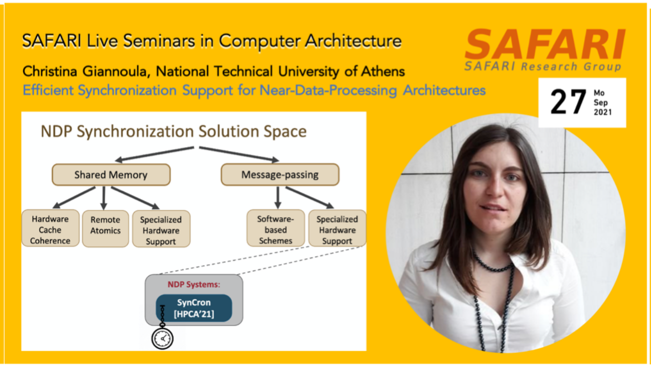
Using QUAC to Generate Random Values
Use QUAC to activate DRAM rows that are initialized with conflicting data (e.g., two '1's and two '0's) to generate random values
Row 3
Row 2
Row 1
Row 0
Wordline
Bitline
Capacitor
Sense Amplifiers
Random Values
ACT
Violate Timing
PRE
Violate Timing
ACT


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Minesh Patel, ETH Zurich
Enabling Effective Error Mitigation in Memory Chips That Use On-Die ECCs

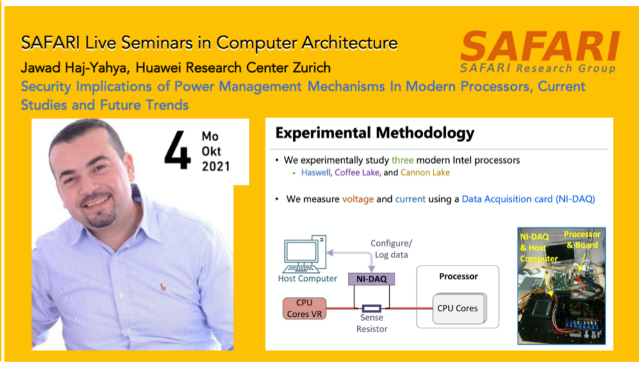
21 Tues Sep 2021

5 Position Paper (Ongoing)
Arguing for increased transparency of DRAM reliability characteristics
1 REAPER (ISCA'17)
Understand the basic properties of DRAM data-retention errors
3 BEER (MICRO'20, best paper)
Determine exactly how on-die ECC obfuscates error characteristics
DRAM Chip
On-Die ECC Logic
Data Store
To processor
4 HARP (MICRO'21)
Understand how errors appear and how to identify at-risk bits
2 EIN (DSN'19, best paper)
Understand and recover the error characteristics beneath on-die ECC


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

Christina Giannoula, National Technical University of Athens
Efficient Synchronization Support for Near-Data-Processing Architectures

27 Mo Sep 2021


SAFARI Live Seminars in Computer Architecture — SAFARI Research Group

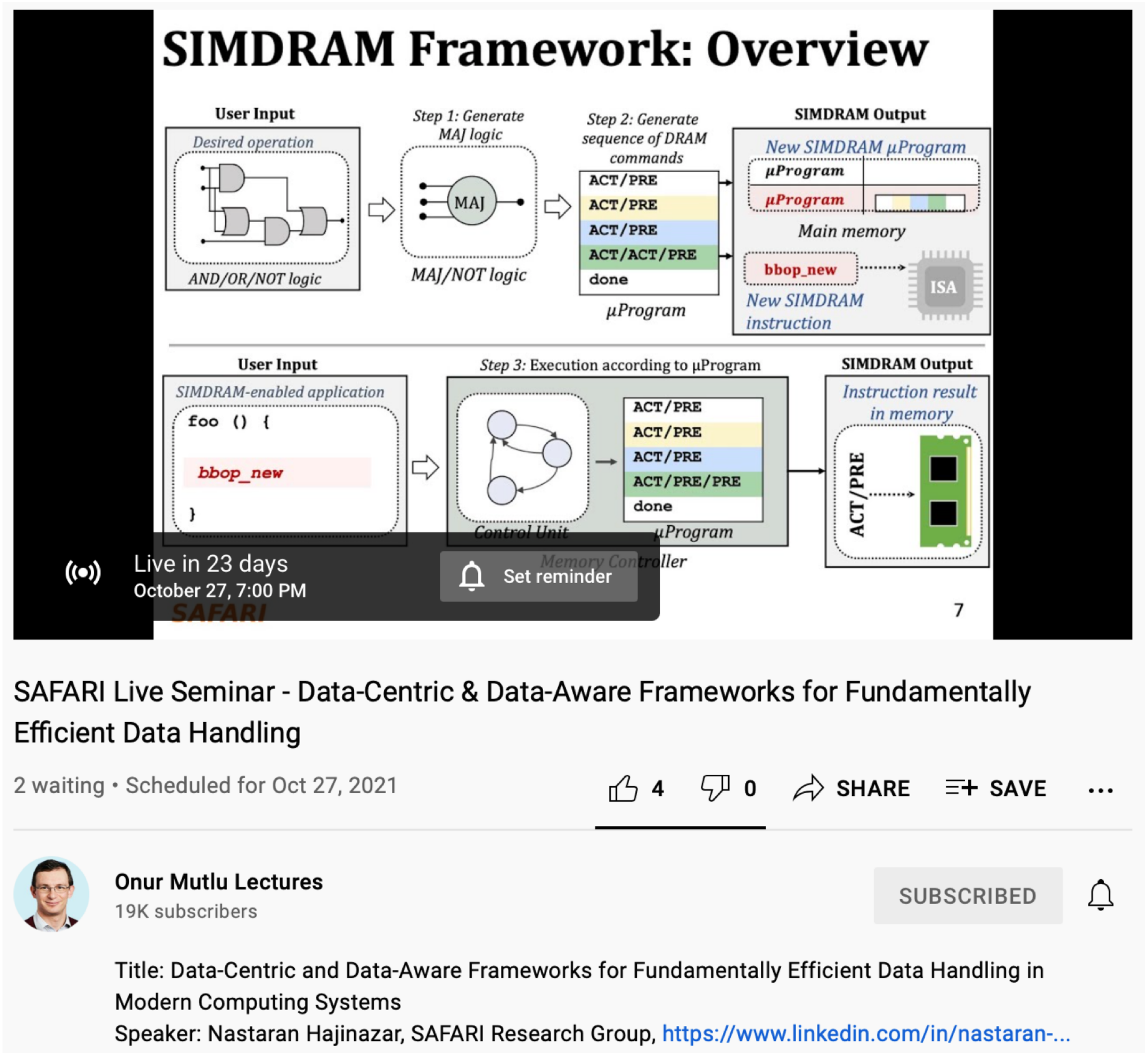
Jawad Haj-Yahya, Huawei Research Center Zurich
Security Implications of Power Management Mechanisms In Modern Processors, Current Studies and Future Trends

4 Mo Okt 2021

Experimental Methodology
We experimentally study three modern Intel processors
Haswell, Coffee Lake, and Cannon Lake
We measure voltage and current using a Data Acquisition card (NI-DAQ)
Host Computer
Configure/Log data
NI-DAQ
Processor
CPU Cores VR
Sense Resistor
CPU Cores
NI-DAQ & Host Computer
Processor & Board

https://safari.ethz.ch/safari-seminar-series/

SAFARI Live Seminars (II)



SAFARI Live Seminar - Data-Centric & Data-Aware Frameworks for Fundamentally Efficient Data Handling

2 waiting • Scheduled for Oct 27, 2021

Onur Mutlu Lectures
19K subscribers

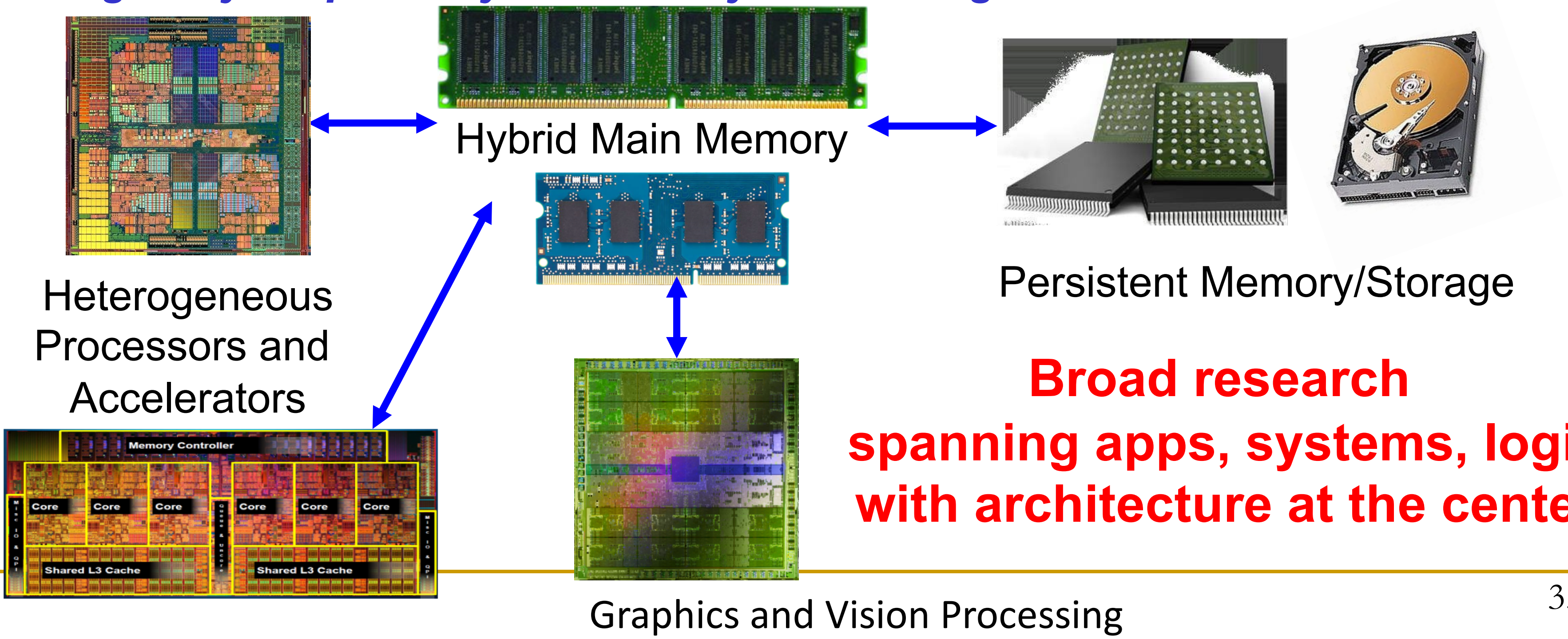
Title: Data-Centric and Data-Aware Frameworks for Fundamentally Efficient Data Handling in Modern Computing Systems
Speaker: Nastaran Hajinazar, SAFARI Research Group, https://www.linkedin.com/in/nastaran-...

https://youtu.be/XIfPHtvA9rw

Current Research Focus Areas

Research Focus: ***Computer architecture, HW/SW, bioinformatics***

- ***Memory and storage (DRAM, flash, emerging), interconnects***
- ***Heterogeneous & parallel systems, GPUs, systems for data analytics***
- ***System/architecture interaction, new execution models, new interfaces***
- ***Energy efficiency, fault tolerance, hardware security, performance***
- ***Genome sequence analysis & assembly algorithms and architectures***
- ***Biologically inspired systems & system design for bio/medicine***


Hybrid Main Memory
Persistent Memory/Storage
Heterogeneous Processors and Accelerators
Memory Controller
Core
Shared L3 Cache
Graphics and Vision Processing

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

Course Info: How About You?

- Let us know your background, interests
- Why did you join this P&S?

Course Requirements and Expectations

- Attendance required for all meetings
- Study the learning materials
- Each student will carry out a hands-on project
  - Build, implement, code, and design with close engagement from the supervisors
- Participation
  - Ask questions, contribute thoughts/ideas
  - Read relevant papers

We will help in all projects!
If your work is really good, you may get it published!

Course Website

- https://safari.ethz.ch/projects_and_seminars/doku.php?id=heterogeneous_systems
- Useful information about the course
- Check your email frequently for announcements
- We also have Moodle for Q&A

Meeting 1

- Required materials:

  1. An introduction to SIMD processors and GPUs (Dr. Juan Gomez Luna, lecture).
  (PDF) (PPT)
  Video

  2. An introduction to GPUs and heterogeneous programming (Dr. Juan Gomez Luna, lecture).
  (PDF) (PPT)
  Video

- Recommended materials:

  3. Programming heterogeneous collaborative systems (Dr. Juan Gomez Luna, lecture):
  (PDF) (PPT)
  https://youtu.be/uhQjXbNo6Cc?t=3040

  4. Juan Gomez-Luna, Izzat El Hajj, Li-Wen Chang, Victor Garcia-Flores, Simon Garcia de Gonzalo, Thomas B. Jablin, Antonio J. Peña and Wen-mei Hwu,
  "Chai: Collaborative Heterogeneous Applications for Integrated-architectures"
  Proceedings of the 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Santa Rosa, California, April 2017.
  https://chai-benchmarks.github.io
  https://github.com/chai-benchmarks/chai

  5. 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)]

  6. Mohammed Alser, Taha Shahroodi, Juan Gomez Luna, Can Alkan, and Onur Mutlu,
  "SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs"
  Bioinformatics, 26 December 2020.
  [Source Code] [Online link at Bioinformatics Journal]

  7. Real Processing-in-DRAM with UPMEM (Dr. Juan Gomez Luna, lecture, Fall 2020).
  (PDF) (PPT)Video

Meeting 2 (October 14th)

- We will announce the projects and will give you some description about them
- We will give you a chance to select a project
- Then, we will have 1-1 meetings to match your interests, skills, and background with a suitable project
- It is important that you study the learning materials before our next meeting!

Next Meetings

- Individual meetings with your mentor/s
- Tutorials and short talks
  - GPU/FPGA programming
  - Recent research works
- Presentation of your work

Exploiting Data Parallelism: SIMD Processors and GPUs

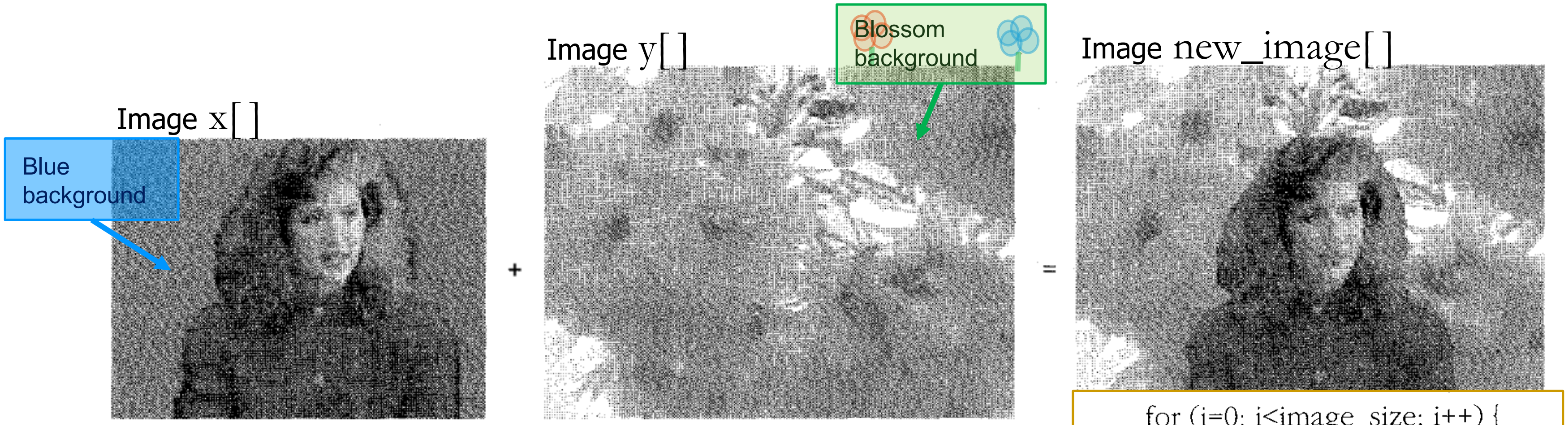
Recall: Flynn’s Taxonomy of Computers

- 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

Recall: MMX Example: Image Overlaying (I)

- Goal: Overlay the human in image x on top of the background in image y

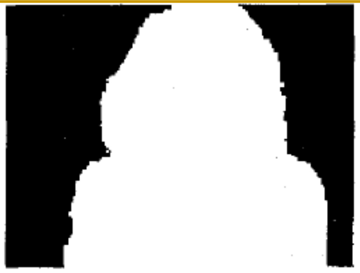


```
for (i=0; i<image_size; i++) {
if (x[i] == Blue)  new_image[i] =y[i];
        else new_image[i] = x[i];
}
```

Figure 8. Chroma keying: image overlay using a background color.

PCMPEQB MM1, MM3

	MM1	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Image x[ ]	MM3	X7!=blue	X6!=blue	X5=blue	X4=blue	X3!=blue	X2!=blue	X1=blue	X0=blue
Bit mask	MM1	0x0000	0x0000	0xFFFF	0xFFFF	0x0000	0x0000	0xFFFF	0xFFFF



Bitmask

Figure 9. Generating the selection bit mask.

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

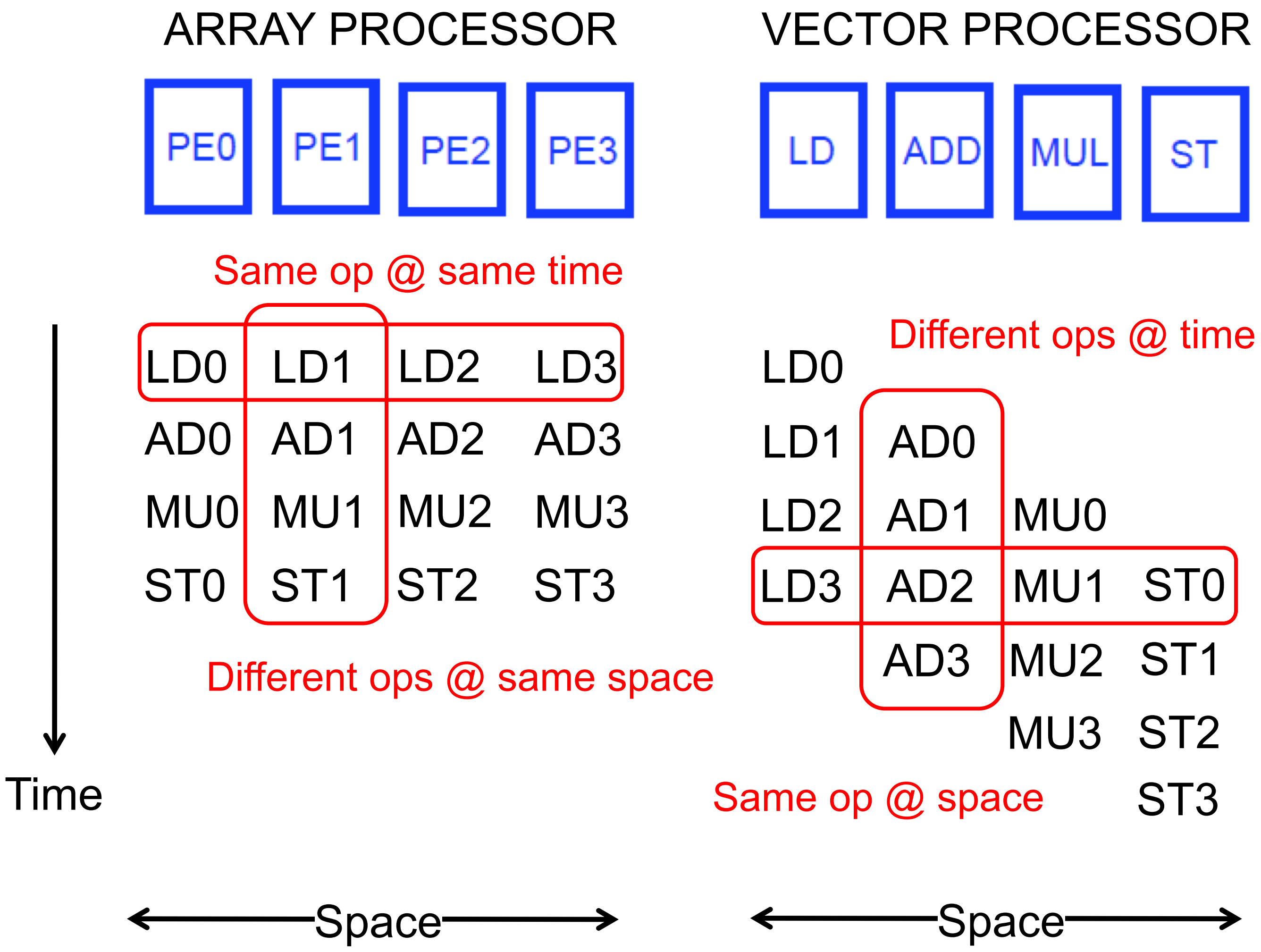
  - Array processor: Instruction operates on multiple data elements at the same time using different spaces (PEs)

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

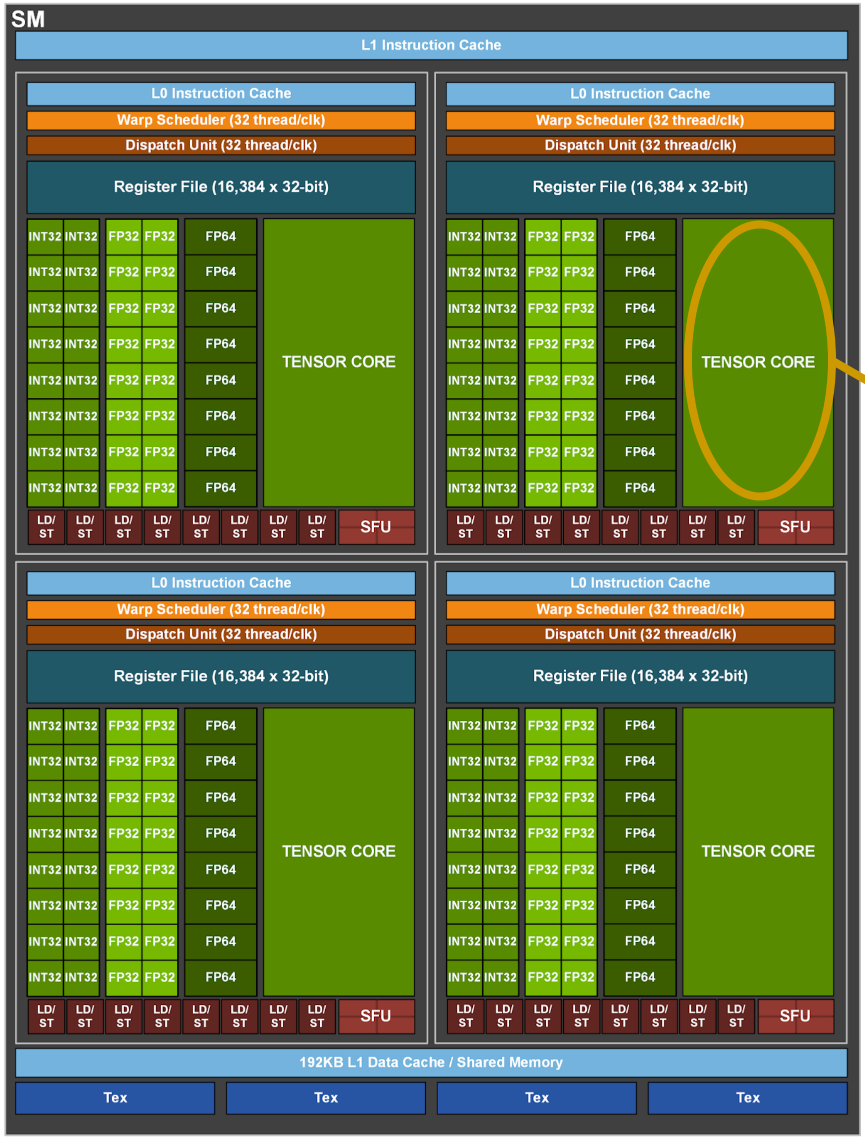
Array vs. Vector Processors

Instruction Stream

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

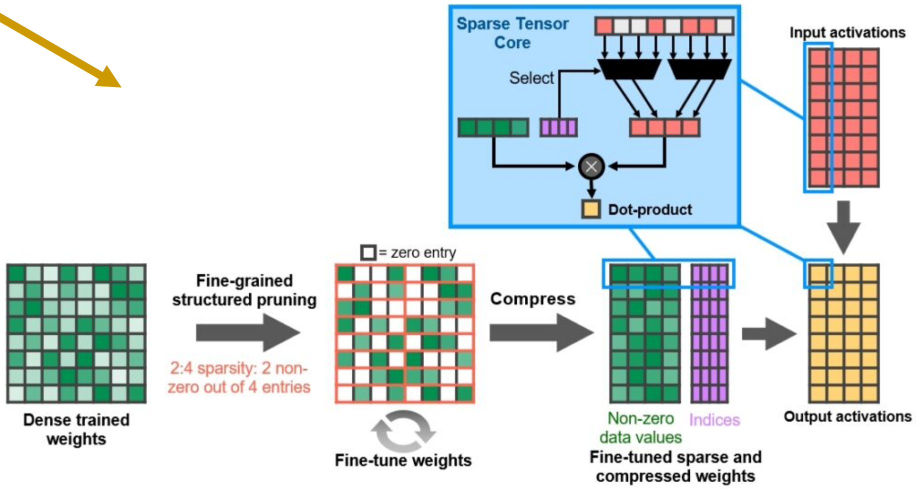


NVIDIA A100 Core


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
LD/ ST
SFU
192KB L1 Data Cache / Shared Memory
Tex

GPU compute throughput:

- 19.5 TFLOPS Single Precision
- 9.7 TFLOPS Double Precision
- 312 TFLOPS for Deep Learning (Tensor cores)



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

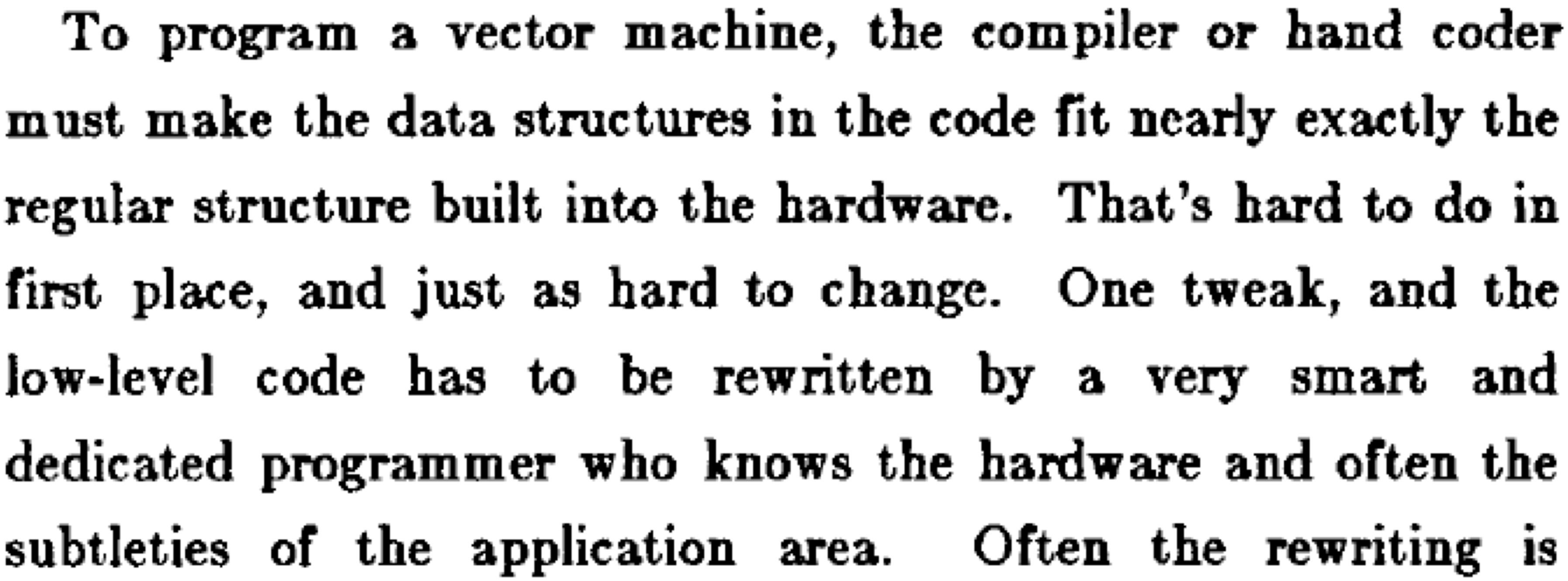
Vector Processor Disadvantages

-- Works (only) if parallelism is regular (data/SIMD parallelism)

++ Vector operations

-- Very inefficient if parallelism is irregular

-- How about searching for a key in a linked list?



To program a vector machine, the compiler or hand coder must make the data structures in the code fit nearly exactly the regular structure built into the hardware. That's hard to do in first place, and just as hard to change. One tweak, and the low-level code has to be rewritten by a very smart and dedicated programmer who knows the hardware and often the subtleties of the application area. Often the rewriting is

Fisher, "Very Long Instruction Word architectures and the ELI-512," ISCA 1983.

Warps *not* Exposed to GPU Programmers

- CPU threads and GPU kernels
  - Sequential or modestly parallel sections on CPU
  - Massively parallel sections on GPU: Blocks of threads

Serial Code (host)

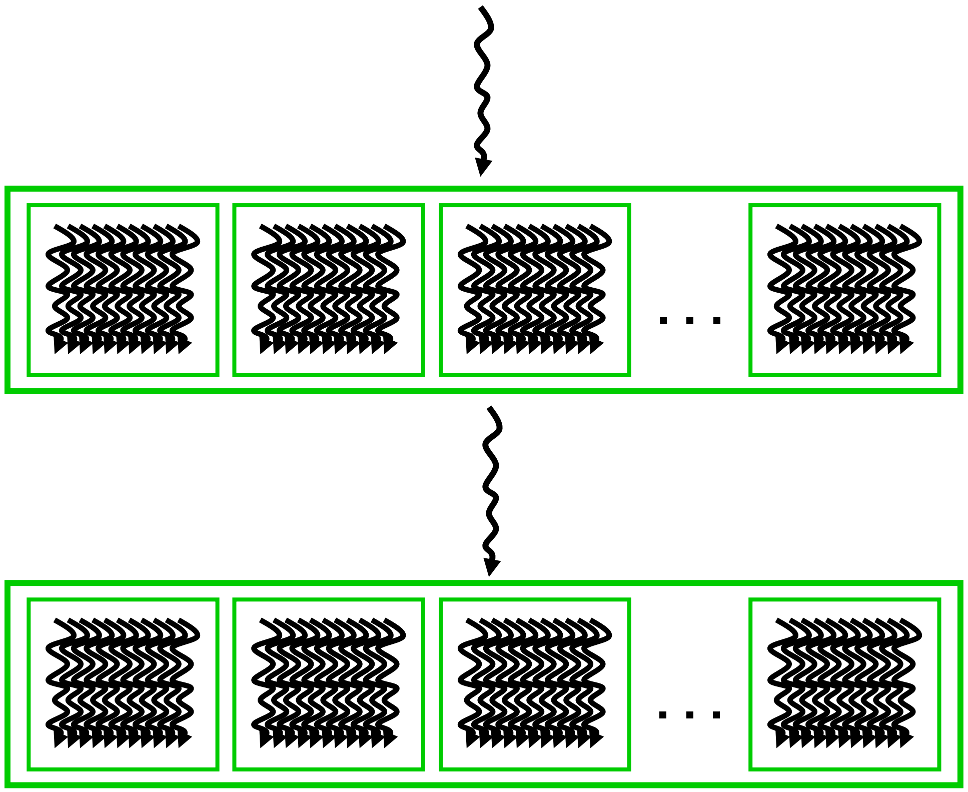
Parallel Kernel (device)

```
KernelA<<<nBlk, nThr>>>(args);
```

Serial Code (host)

Parallel Kernel (device)

```
KernelB<<<nBlk, nThr>>>(args);
```



Slide credit: Hwu & Kirk

Sample GPU SIMT Code (Simplified)

CPU code

```
for (ii = 0; ii < 100000; ++ii) {
C[ii] = A[ii] + B[ii];
}
```

CUDA code

```
// there are 100000 threads
__global__ void KernelFunction(…) {
  int tid = blockDim.x * blockIdx.x + threadIdx.x;
  int varA = aa[tid];
  int varB = bb[tid];
  C[tid] = varA + varB;
}
```

Slide credit: Hyesoon Kim

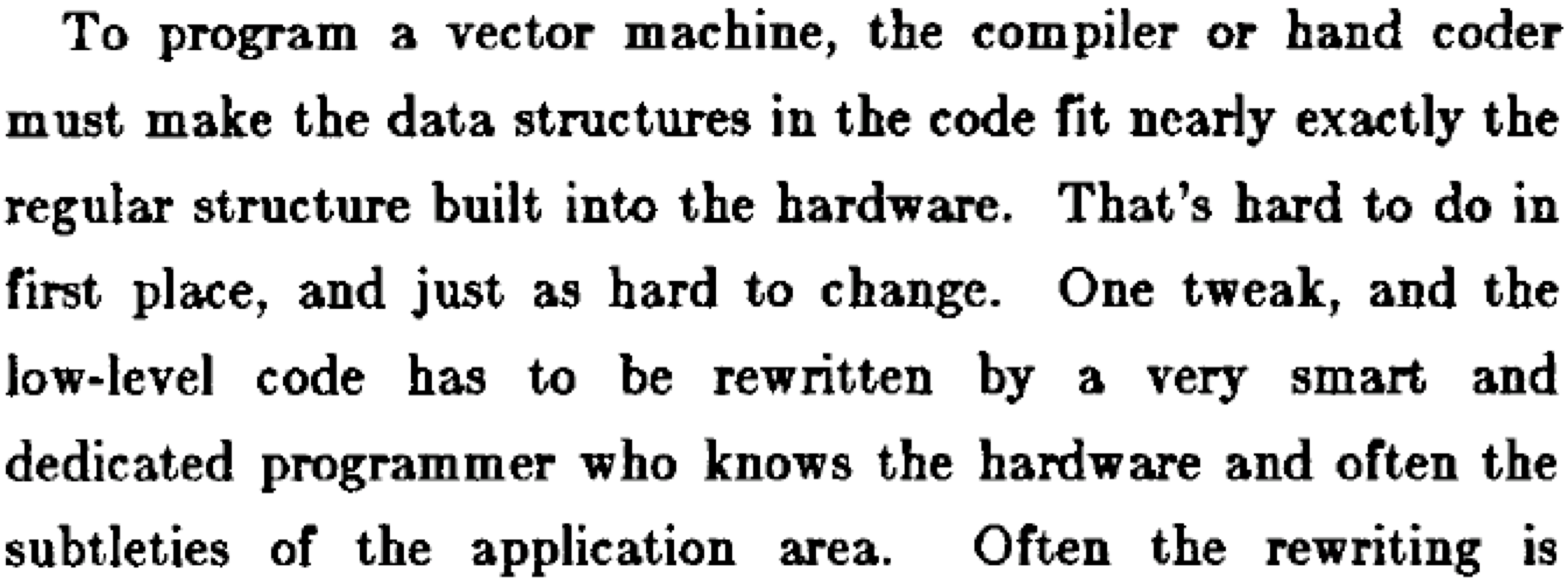
Vector Processor Disadvantages

-- Works (only) if parallelism is regular (data/SIMD parallelism)

  ++ Vector operations

  -- Very inefficient if parallelism is irregular

    -- How about searching for a key in a linked list?

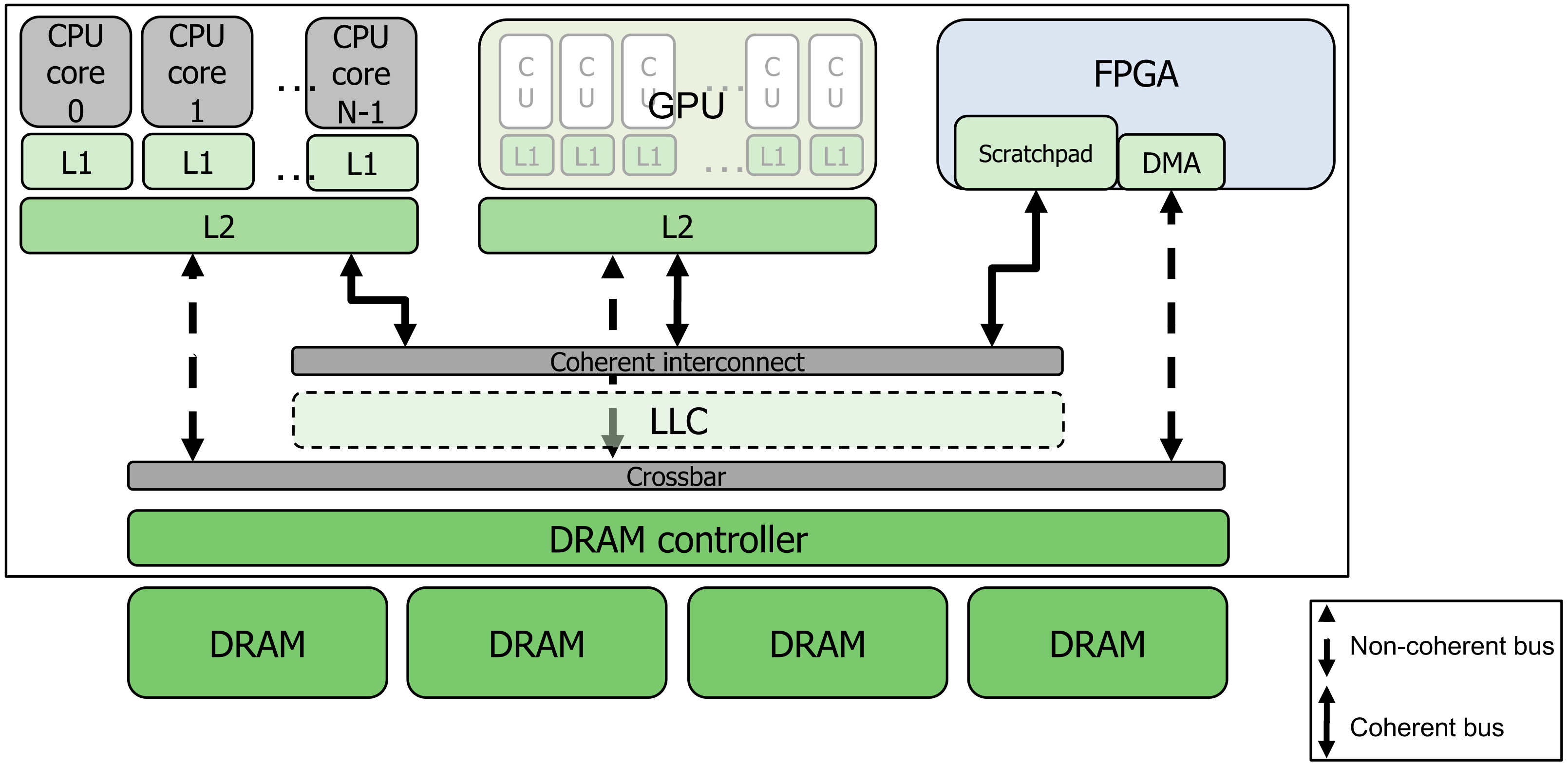


To program a vector machine, the compiler or hand coder must make the data structures in the code fit nearly exactly the regular structure built into the hardware. That's hard to do in first place, and just as hard to change. One tweak, and the low-level code has to be rewritten by a very smart and dedicated programmer who knows the hardware and often the subtleties of the application area. Often the rewriting is

Fisher, "Very Long Instruction Word architectures and the ELI-512," ISCA 1983.

Heterogeneous Computing Systems

- The end of Moore's law created the need for heterogeneous systems
  - More suitable devices for each type of workload
  - Increased performance and energy efficiency



Chai Benchmark Suite

- Heterogeneous execution on CPU, GPU, FPGA
- Collaboration patterns
  - 8 data partitioning benchmarks
  - 3 coarse-grain task partitioning benchmarks
  - 3 fine-grain task partitioning benchmarks
- Discrete (D) and Unified (U) versions
- Chai versions
  - CUDA and OpenCL for CPU+GPU
  - OpenCL for CPU+FPGA
  - CUDA-Sim for Gem5-GPU

https://chai-benchmarks.github.io

Gómez-Luna+, "Chai: Collaborative Heterogenous Applications for Integrated Architectures," ISPASS 2017.

P&S Heterogeneous Systems

Hands-on Acceleration on Heterogeneous Computing Systems

Dr. Juan Gómez Luna
Prof. Onur Mutlu
ETH Zürich

Fall 2021

7 October 2021