P&S Heterogeneous Systems

Hands-on Acceleration on Heterogeneous Computing Systems

Dr. Juan Gómez Luna
Prof. Onur Mutlu

ETH Zürich
Spring 2022
11 March 2022
P&S: Heterogeneous Systems (I)

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

Semester  
Spring Semester 2022

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.

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.
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


- **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
```

<table>
<thead>
<tr>
<th>32</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
<th>Bit position</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_3</td>
<td>a_2</td>
<td>a_1</td>
<td>a_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$s0</td>
</tr>
<tr>
<td>b_3</td>
<td>b_2</td>
<td>b_1</td>
<td>b_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$s1</td>
</tr>
<tr>
<td>a_3 + b_3</td>
<td>a_2 + b_2</td>
<td>a_1 + b_1</td>
<td>a_0 + b_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$s2</td>
</tr>
</tbody>
</table>
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

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.

MMX Example: Image Overlaying (I)

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

![Image overlay diagram](image)

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

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

<table>
<thead>
<tr>
<th>PCMP EQB MM1, MM3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image x[ ]</strong></td>
</tr>
<tr>
<td><strong>Bit mask</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MM1</th>
<th>Blue</th>
<th>Blue</th>
<th>Blue</th>
<th>Blue</th>
<th>Blue</th>
<th>Blue</th>
<th>Blue</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM3</td>
<td>X7</td>
<td>=blue</td>
<td>X6</td>
<td>=blue</td>
<td>X5</td>
<td>=blue</td>
<td>X4</td>
<td>=blue</td>
</tr>
</tbody>
</table>

| MM1 | 0x0000 | 0x0000 | 0xFFFF | 0xFFFF | 0x0000 | 0x0000 | 0xFFFF | 0xFFFF |

**Figure 9.** Generating the selection bit mask.
MMX Example: Image Overlaying (II)

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

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

Figure 11. MMX code sequence for performing 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
Pand mm1, mm3
Por mm4, mm1
```
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)

108 cores on the A100
(Up to 128 cores in the full-blown chip)

40MB L2 cache

https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/
NVIDIA A100 Core

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 mm²

**Largest GPU**
21.1 Billion transistors
815 mm²
*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.

Google TPU Generation II (2017)

- 4 TPU chips vs 1 chip in TPU1
- High Bandwidth Memory vs DDR3
- Floating point operations vs FP16
- 45 TFLOPS per chip vs 23 TOPS
- Designed for training and inference vs only inference

Google TPU Generation III (2019)

32GB HBM per chip vs 16GB HBM in TPU2

90 TFLOPS per chip vs 45 TFLOPS in TPU2

4 Matrix Units per chip vs 2 Matrix Units 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

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.

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

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

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

Samsung AxDIMM (2021)

- DIMM-based PIM
  - DLRM recommendation system

SK hynix Develops PIM, Next-Generation AI Accelerator

February 16, 2022

Seoul, February 16, 2022

SK hynix (or "the Company", [www.skhynix.com](http://www.skhynix.com)) announced on February 16 that it has developed PIM*, a next-generation memory chip with computing capabilities.

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

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

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

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

For the first product that adopts the PIM technology, SK hynix has developed a sample of GDDR6-AIM (Accelerator in memory). The GDDR6-AIM adds computational functions to GDDR6 memory chips, which process data at 16Gbps. A combination of GDDR6-AIM with CPU or GPU instead of a typical DRAM makes certain computation speed 16 times faster. GDDR6-AIM is widely expected to be adopted for machine learning, high-performance computing, and big data computation and storage.

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
  - ...

[Image of a person]
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/](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-january-2021/

40+ Researchers

Think BIG, Aim HIGH!

https://safari.ethz.ch
SAFARI Newsletter December 2021 Edition

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

Think Big, Aim High

ETHzürich

View in your browser
December 2021
SAFARI Live Seminars (1)

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

12
Man Jul 2021

19
Mo Jul 2021

What’s happening in solid state memories?
Longer slowing down? Increasingorldhammer susceptibility

SAFARI Live Seminars in Computer Architecture
Dr. Andrew Walker, Schiliton Corporation & Nexgen Power Systems
An Addiction to Low Cost Per Memory Bit – How to Recognize it and What to Do About It

22
Do Jul 2021

Near-Data Processing (2/2)

The goal of Near-Data Processing (NDP) is to mitigate data movement

SAFARI Live Seminars in Computer Architecture
Gennady Pelekhmenko, University of Toronto
Efficient DNN Training at Scale: from Algorithms to Hardware

5
Do Aug 2021

DNN Training vs. Inference

SAFARI Live Seminars in Computer Architecture
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

SAFARI Live Seminars in Computer Architecture
Ataberk Olgun, TOBB & ETH Zurich
GIAC–PRNG: High-Throughput True Random Number Generation Using Quad-Port RAM Activation in Commercial DRAM Chips

15
Tu Aug 2021

Using GIAC to Generate Random Values

SAFARI Live Seminars in Computer Architecture
Minesh Patel, ETH Zurich
Enabling Effective Error Mitigation in Memory Chips That Use On-Die ECCs

21
Tue Sep 2021

NPD Synchronization Solution Space

SAFARI Live Seminars in Computer Architecture
Christina Giannouli, National Technical University of Athens
Efficient Synchronization: Support for Near-Data-Processing Architectures

27
Mo Oct 2021

Experimental Methodology

SAFARI Live Seminars in Computer Architecture
Jawad Haj-Yahya, Huawei Research Center Zurich

4
Mo Oct 2021

SAFARI Live Seminars in Computer Architecture
Geraldo F. Oliveira, ETH Zurich
DAMON: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

https://safari.ethz.ch/safari-seminar-series/
SAFARI Live Seminars (II)

Overview

- CODIC substrate enables greater control over DRAM internal circuit timings
- CODIC is an efficient and low-cost way to enable new functionalities and optimizations in DRAM
- CODIC controls four key signals that orchestrate DRAM internal circuit timings
  - wordline (wl): Connects DRAM cells to bitlines
  - sense_p and sense_n: Trigger sense amplifiers
  - EQ: Triggers the logic that prepares a DRAM bank for the next access

SAFARI Live Seminar: Lois Orosa, 10 Feb 2022
Posted on January 16, 2022 by ewent

Join us for our next SAFARI Live Seminar with Lois Orosa. Thursday, February 10 at 5:00 pm Zurich time (CET)

Lois Orosa, SAFARI Research Group, ETH Zurich
CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations

Livestream on YouTube Link

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

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](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](https://youtu.be/uhQjXbNo6Cc?t=3040)
     [https://chai-benchmarks.github.io](https://chai-benchmarks.github.io)
     [https://github.com/chai-benchmarks/chai](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"
     [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

- 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
Hetero. Systems (Fall’21)

- **Fall 2021 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/fall2021/doku.php?id=heterogeneous_systems](https://safari.ethz.ch/projects_and_seminars/fall2021/doku.php?id=heterogeneous_systems)

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=QYbjwzsfMM&list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737](https://www.youtube.com/watch?v=QYbjwzsfMM&list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - GPU and Parallelism lectures
  - Hands-on research exploration
  - Many research readings

---

### Fall 2021 Meetings/Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Livestream</th>
<th>Meeting</th>
<th>Learning Materials</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>07.10 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M1: P&amp;S Course Presentation (PPT)</td>
<td>Required Materials Recommended Materials</td>
<td>HW 0 Out</td>
</tr>
<tr>
<td>W2</td>
<td>14.10 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M2: SIMD Processing and GPUs (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td>21.10 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M3: GPU Software Hierarchy (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td>28.10 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M4: GPU Memory Hierarchy (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W5</td>
<td>04.11 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M5: GPU Performance Considerations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W6</td>
<td>11.11 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M6: Parallel Patterns: Reduction (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W7</td>
<td>18.11 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M7: Parallel Patterns: Histogram (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W8</td>
<td>25.11 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M8: Parallel Patterns: Convolution (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W9</td>
<td>02.12 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M9: Parallel Patterns: Prefix Sum (Scan) (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W10</td>
<td>09.12 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M10: Parallel Patterns: Sparse Matrices (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W11</td>
<td>16.12 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M11: Parallel Patterns: Graph Search (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W12</td>
<td>23.12 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M12: Dynamic Parallelism (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W13</td>
<td>06.01 Thu</td>
<td><a href="https://www.youtube.com/watch?v=QYbjwzsfMM&amp;list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737">Live</a></td>
<td>M13: Collaborative Computing (PDF)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exploiting Data Parallelism: SIMD Processors and GPUs
Recall: Flynn’s Taxonomy of Computers


- **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 $y$

![Image x[]](image1.png) + ![Image y[]](image2.png) = ![Image new_image[]](image3.png)

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

**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

Time

Space

ARRAY PROCESSOR

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

VECTOR PROCESSOR

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

Same op @ same time
Different ops @ time
Different ops @ same space
Same op @ space
NVIDIA A100 Core

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

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

```c
for (ii = 0; ii < 100000; ++ii) {
}
```

CUDA code

```c
// 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;
}
```
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
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
P&S Heterogeneous Systems

Hands-on Acceleration on Heterogeneous Computing Systems

Dr. Juan Gómez Luna
Prof. Onur Mutlu

ETH Zürich
Spring 2022
11 March 2022