EPFL Center for Imaging

Basics of GPU Acceleration for Image Processing

Imaging Lunch

Joan Rué Queralt — 17/04/24

Preview

EPFL Center for Imaging

What we will see today

- CPU Architecture
- GPU Architecture
- GPU Execution Model
- CUDA programming with Python: Numba, CuPy and cuCIM

What we won't see today

- GPU Memory (hierarchy, shared memory, unified memory)
- Multi-GPU
- DL / ML
- AMD / Intel GPUs & OpenCL
- Other CUDA Python Libraries: PyCUDA, PyTorch, TensorFlow, scikit-cuda, ...

Poll

EPFL Center for Imaging

Join at menti.com | use code 62 57 58 2

The Evolution of GPUs

Major Milestones

Early Graphics Hardware

Integrated Graphics Processors

The Dawn of GPUs

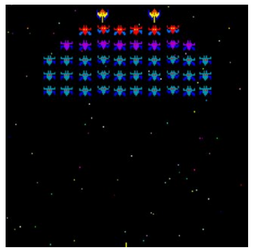
Programmable Shading and Multi-GPU Setups

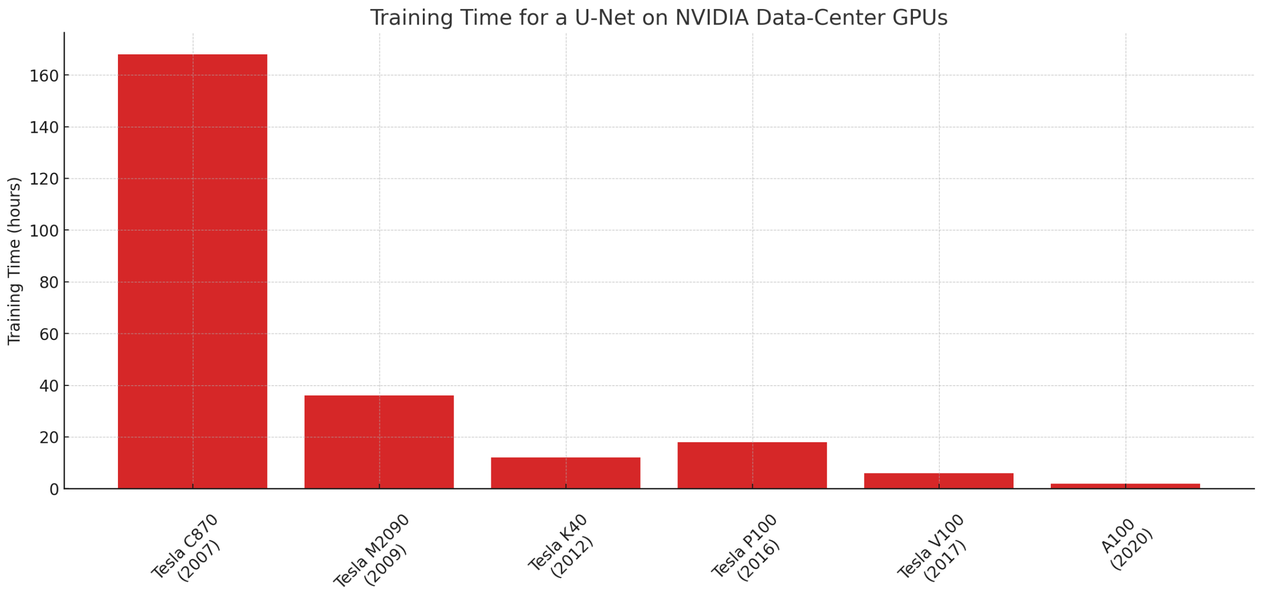
Ray Tracing & AI Acceleration

Advancements in AI & Real-Time Rendering

1970 | 1980 | 1990 | 2000 | 2010 | 2020

2007: NVIDIA launches CUDA

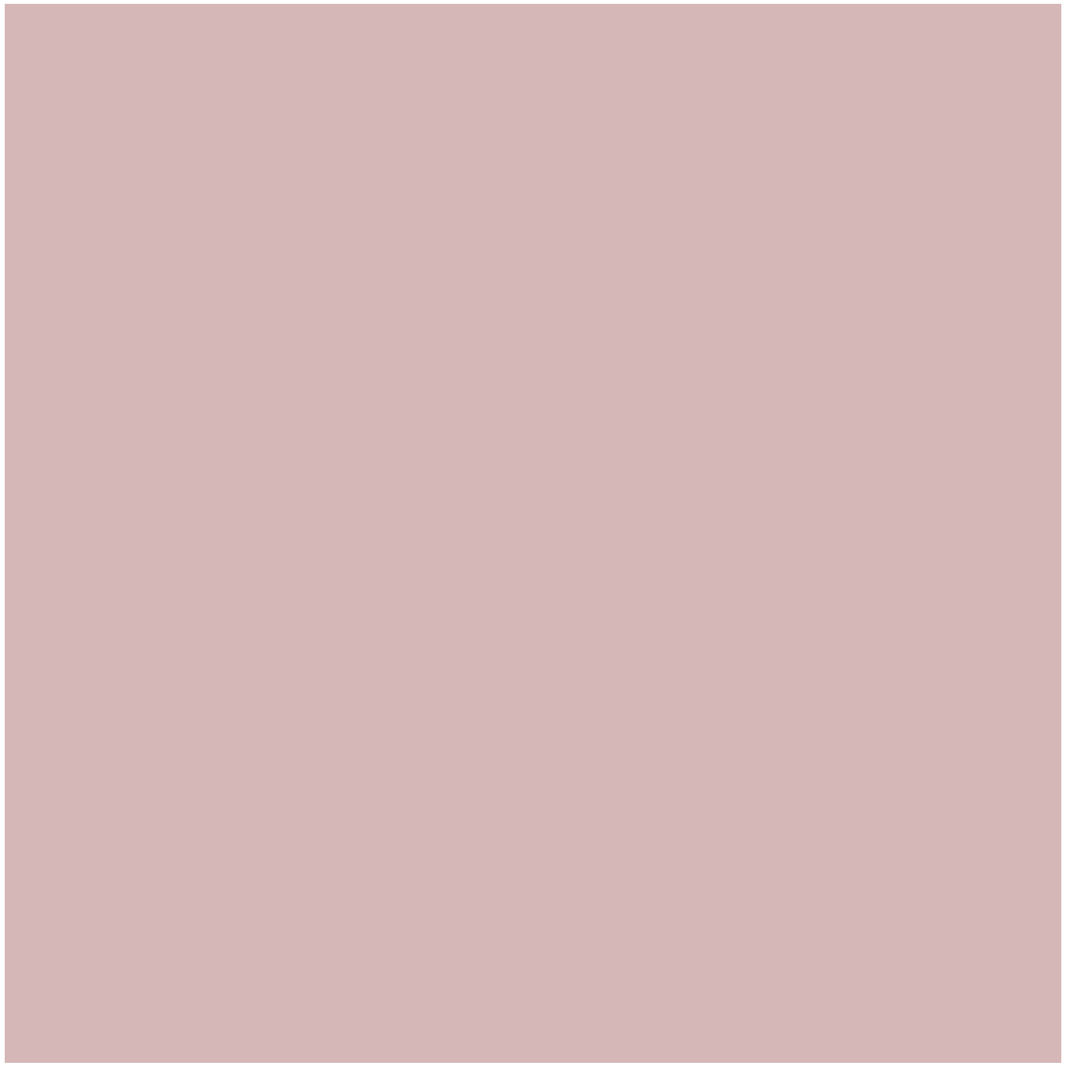




https://shadow.tech/en-GB/blog/history-of-gaming-gpus

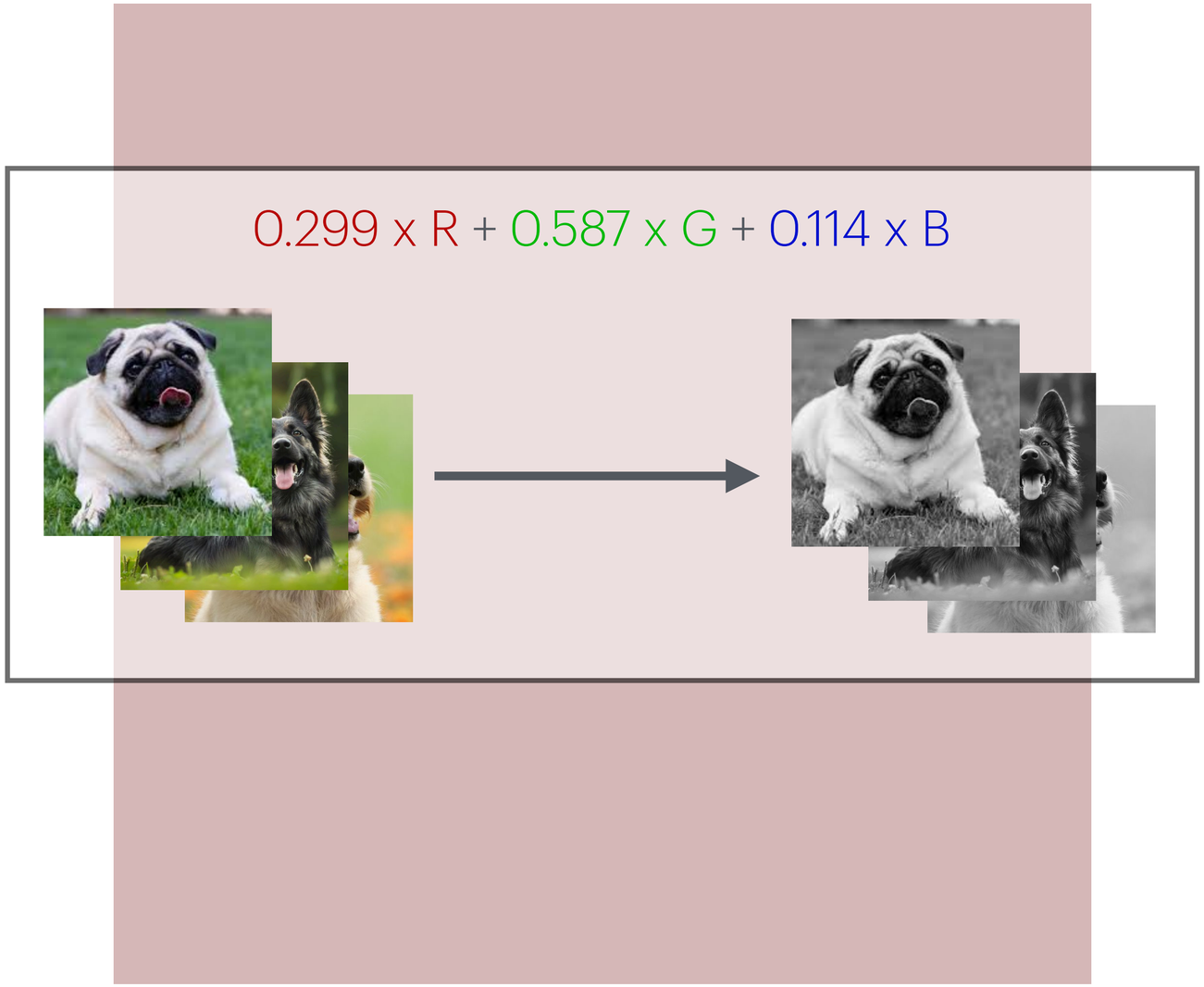
CPU chip architecture

- General purpose processor



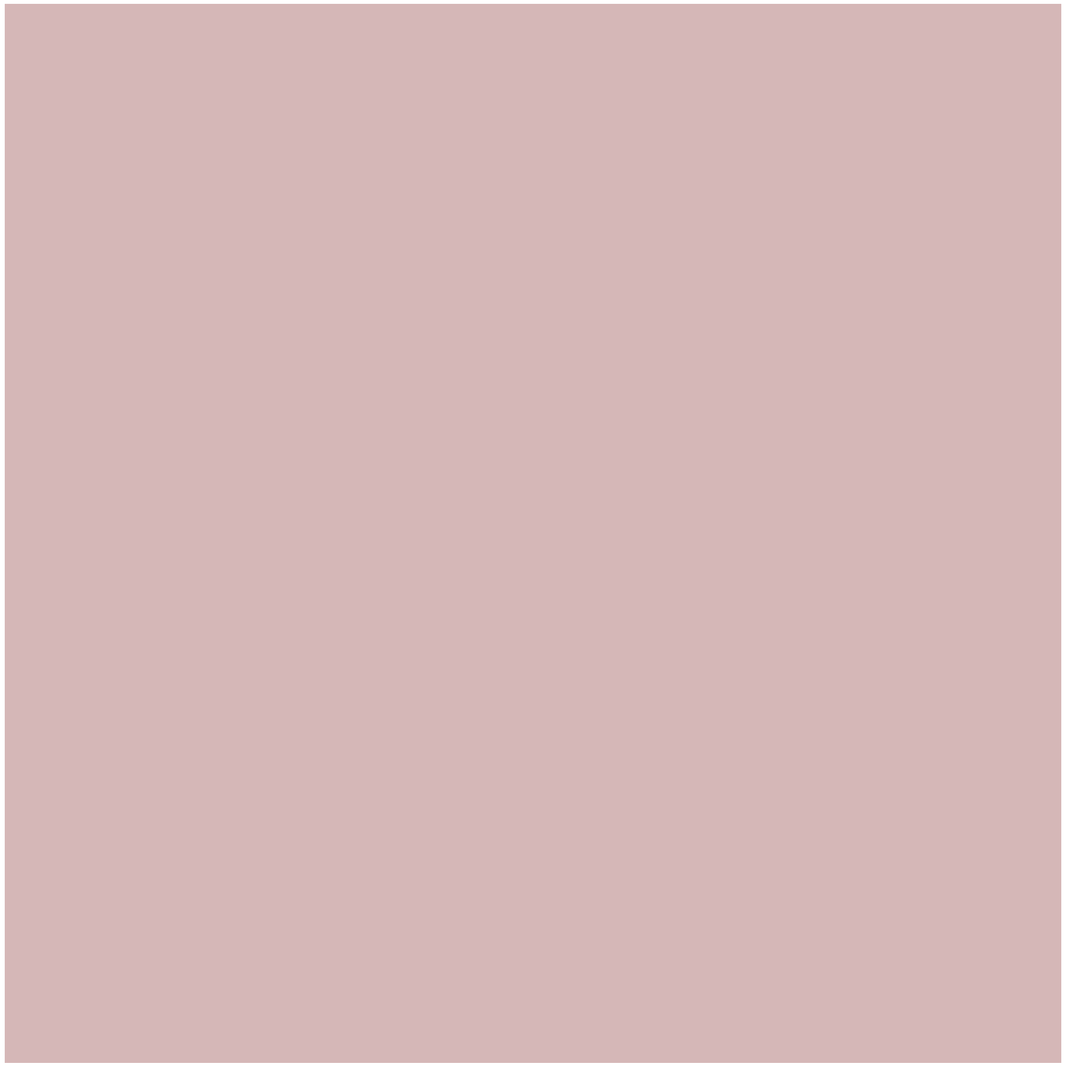
CPU chip architecture

- General purpose processor



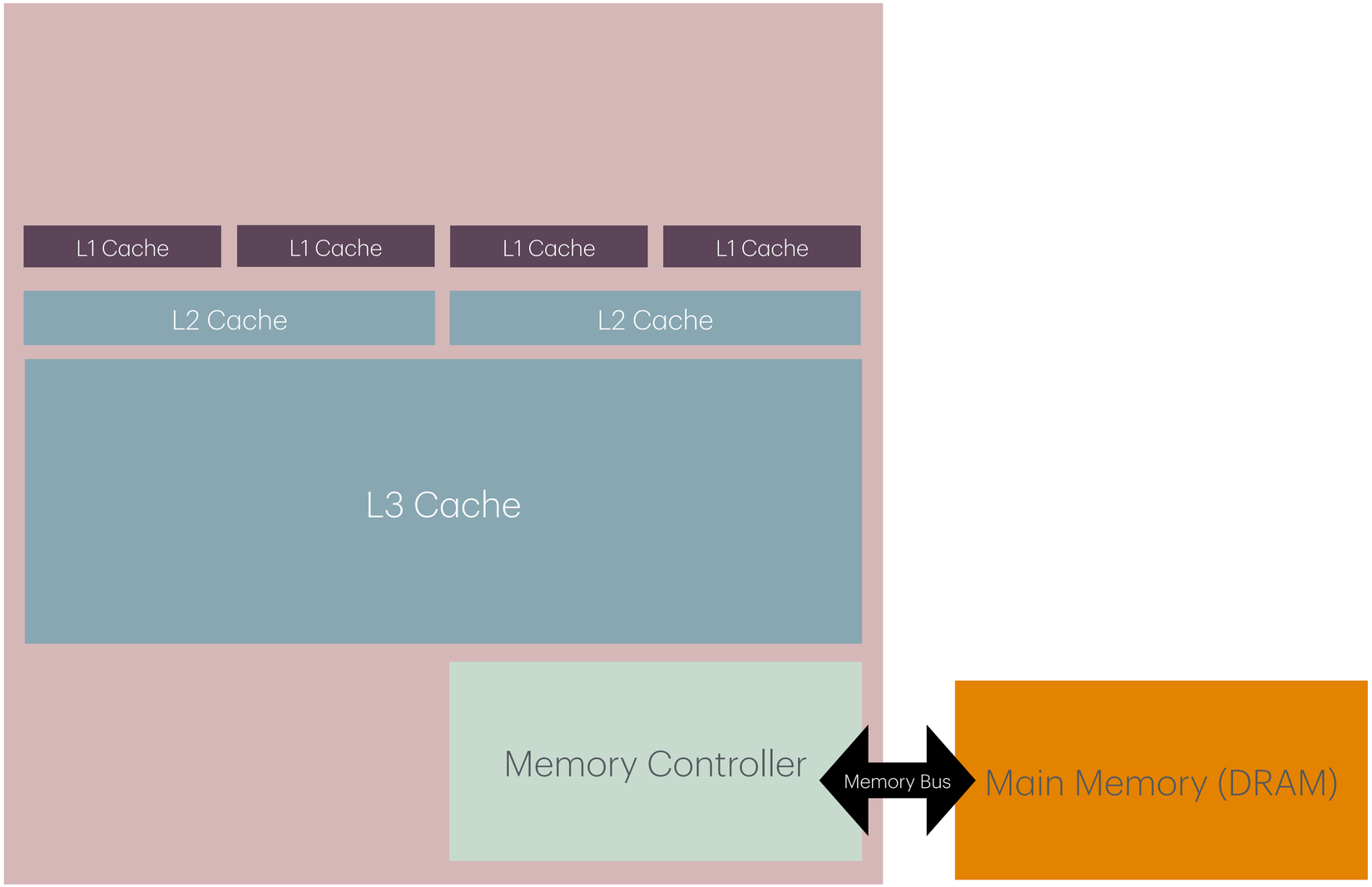
CPU chip architecture

- General purpose processor



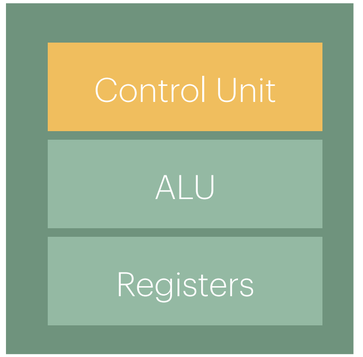
CPU chip architecture

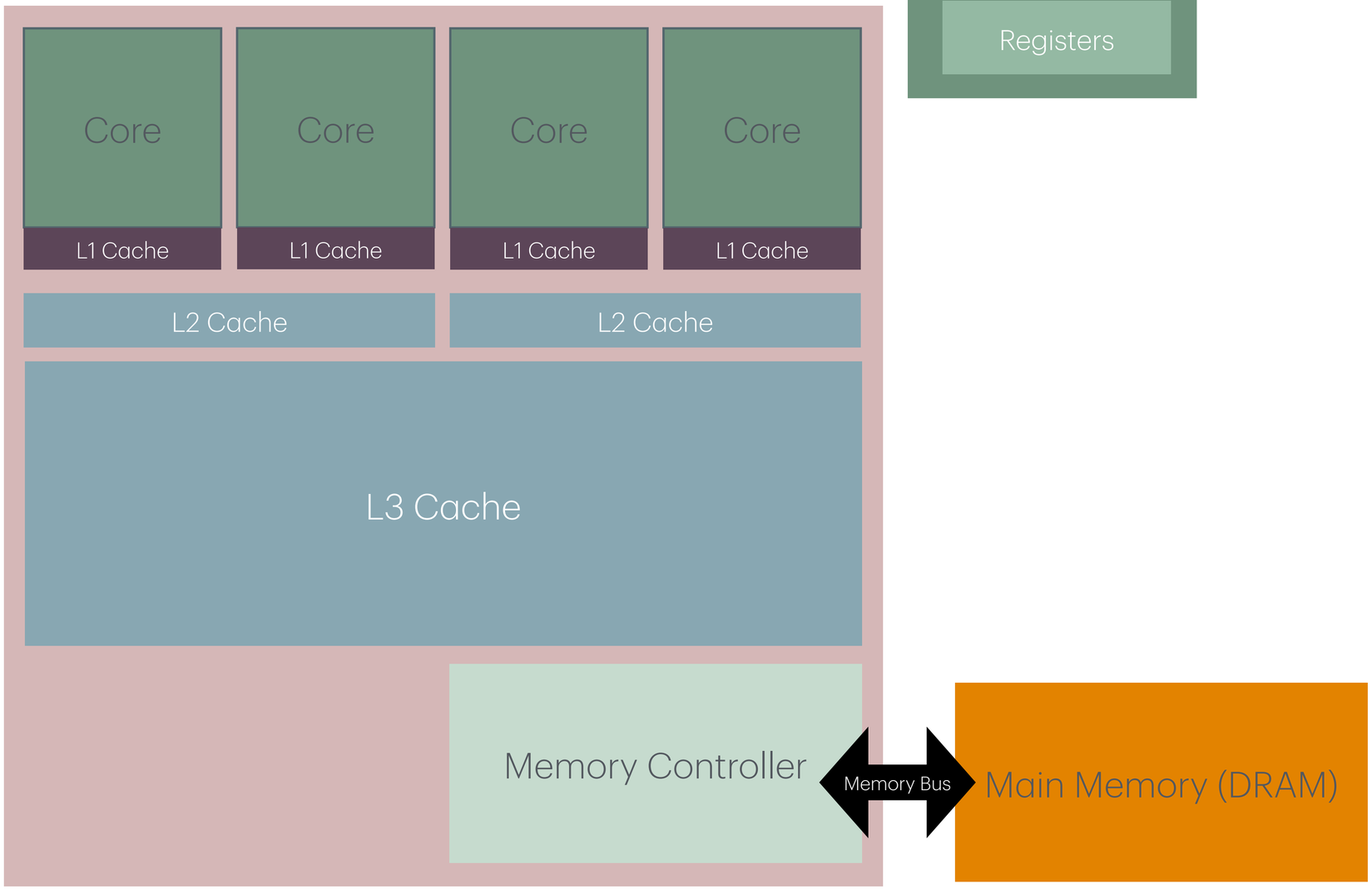
- General purpose processor
- Memory Controller is integrated
- Different Cache Levels
  The lower level the smaller memory and faster access



CPU chip architecture

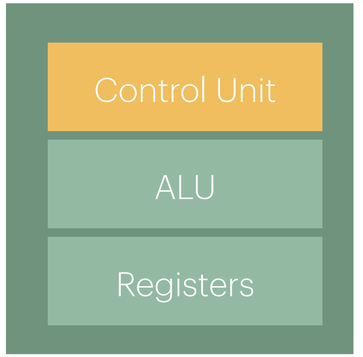
- General purpose processor
- Memory Controller is integrated
- Different Cache Levels
  The lower level the smaller memory and faster access
- Between 2-20 Cores
- Each core is capable of executing program instructions independently of the others
- If the software can run in multiple cores, having more cores will result in more instructions run in parallel

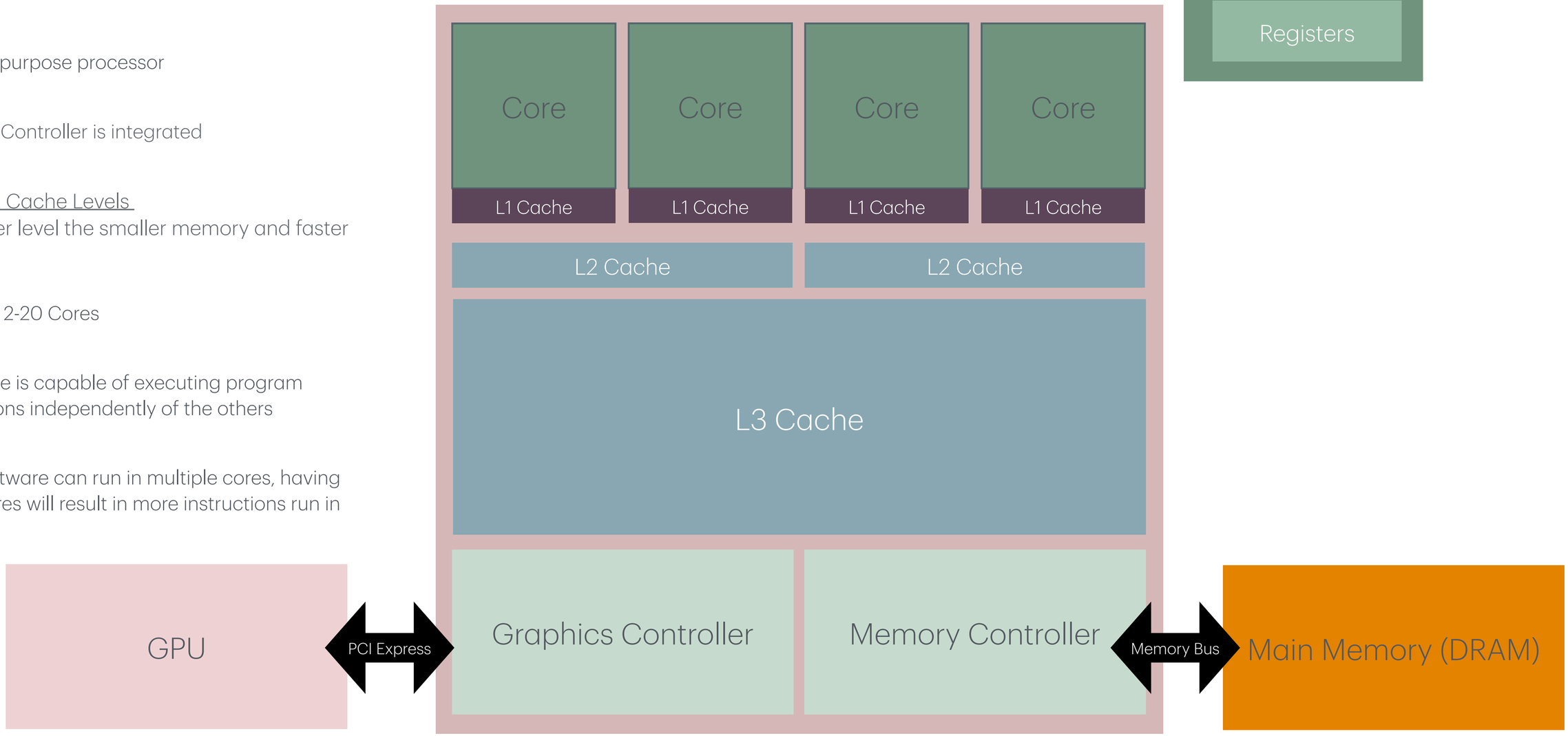




CPU chip architecture

- General purpose processor
- Memory Controller is integrated
- Different Cache Levels
  The lower level the smaller memory and faster access
- Between 2-20 Cores
- Each core is capable of executing program instructions independently of the others
- If the software can run in multiple cores, having more cores will result in more instructions run in parallel

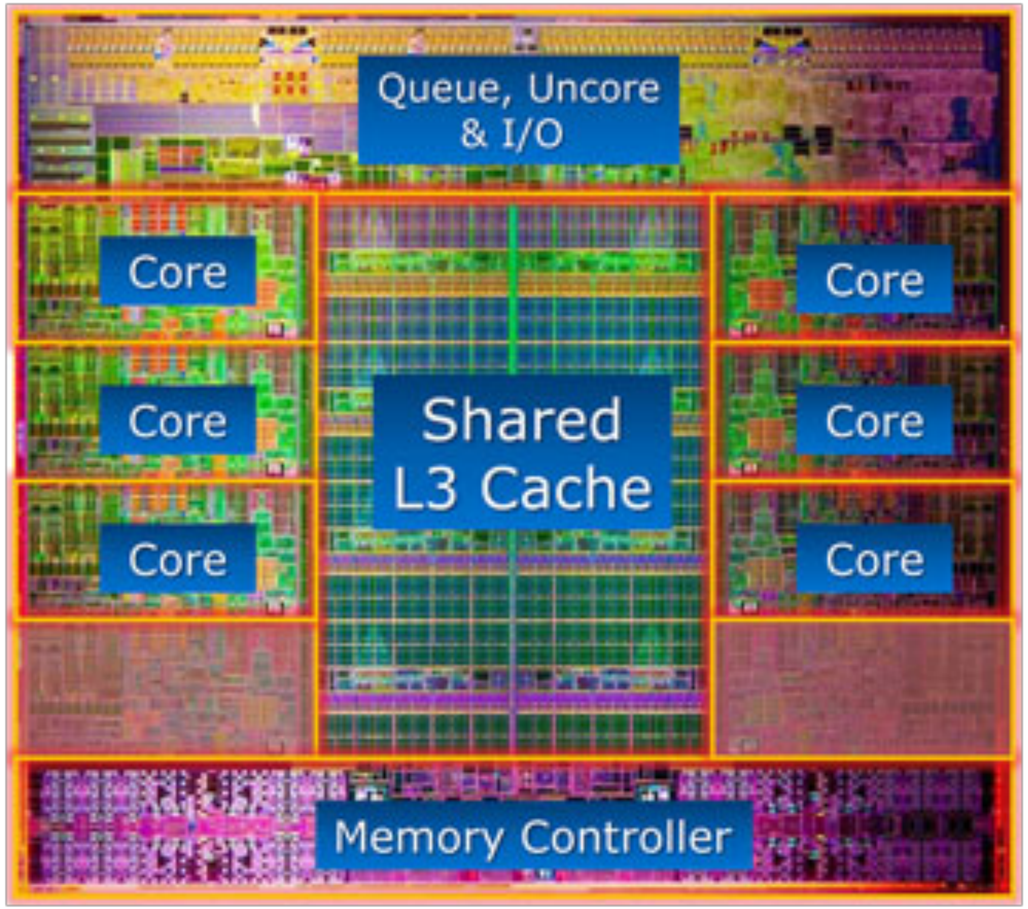




Intel Core i7-3960X

EPFL Center for Imaging

Release year: 2011
6 Cores
HyperThreading (12 threads)
15 MB of L3 cache
Clock speed: 3.3 GHz (boost up to 3.9 GHz)



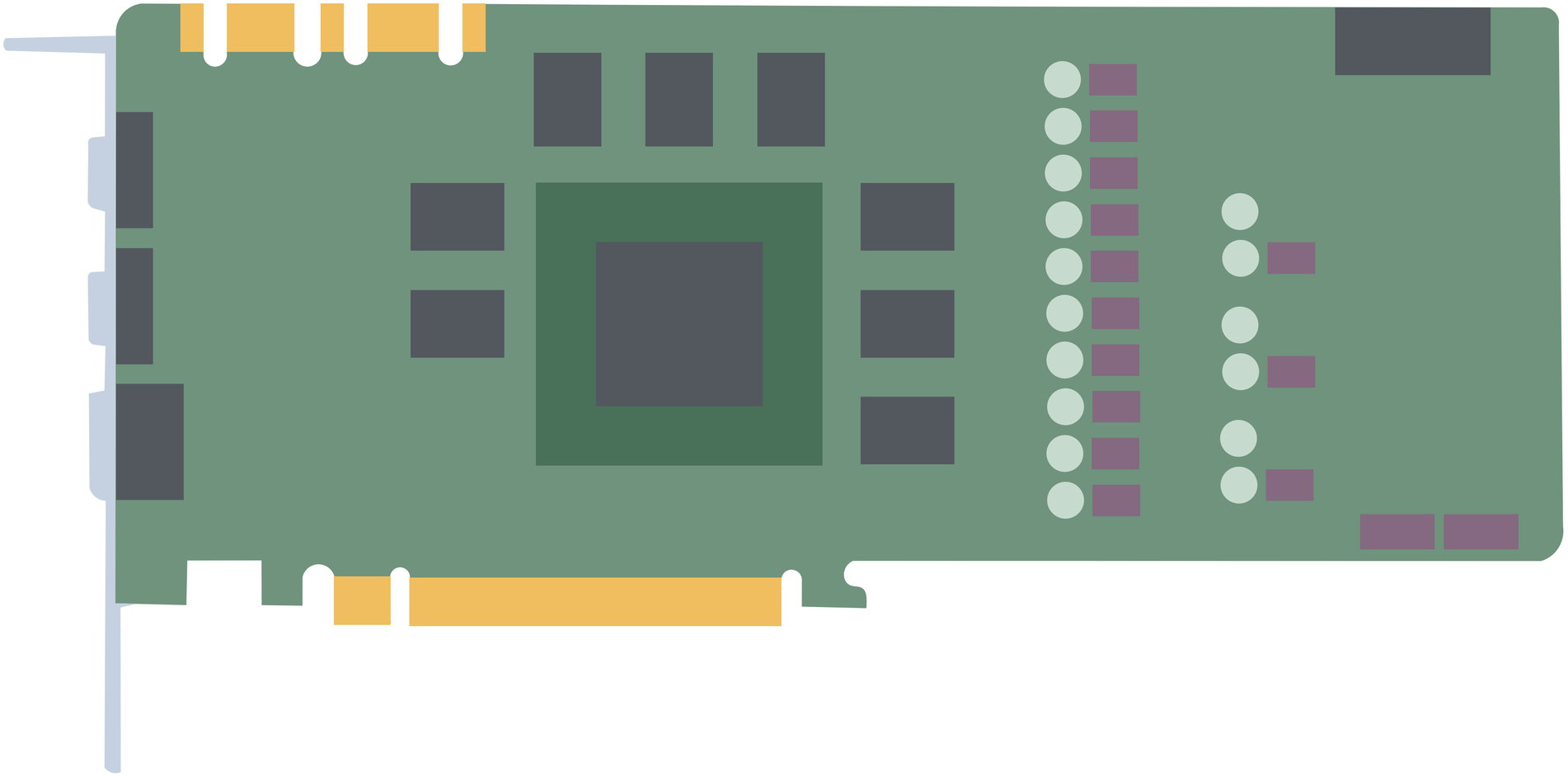
Graphics card

Circuit board to handle graphics operations

EPFL Center for Imaging

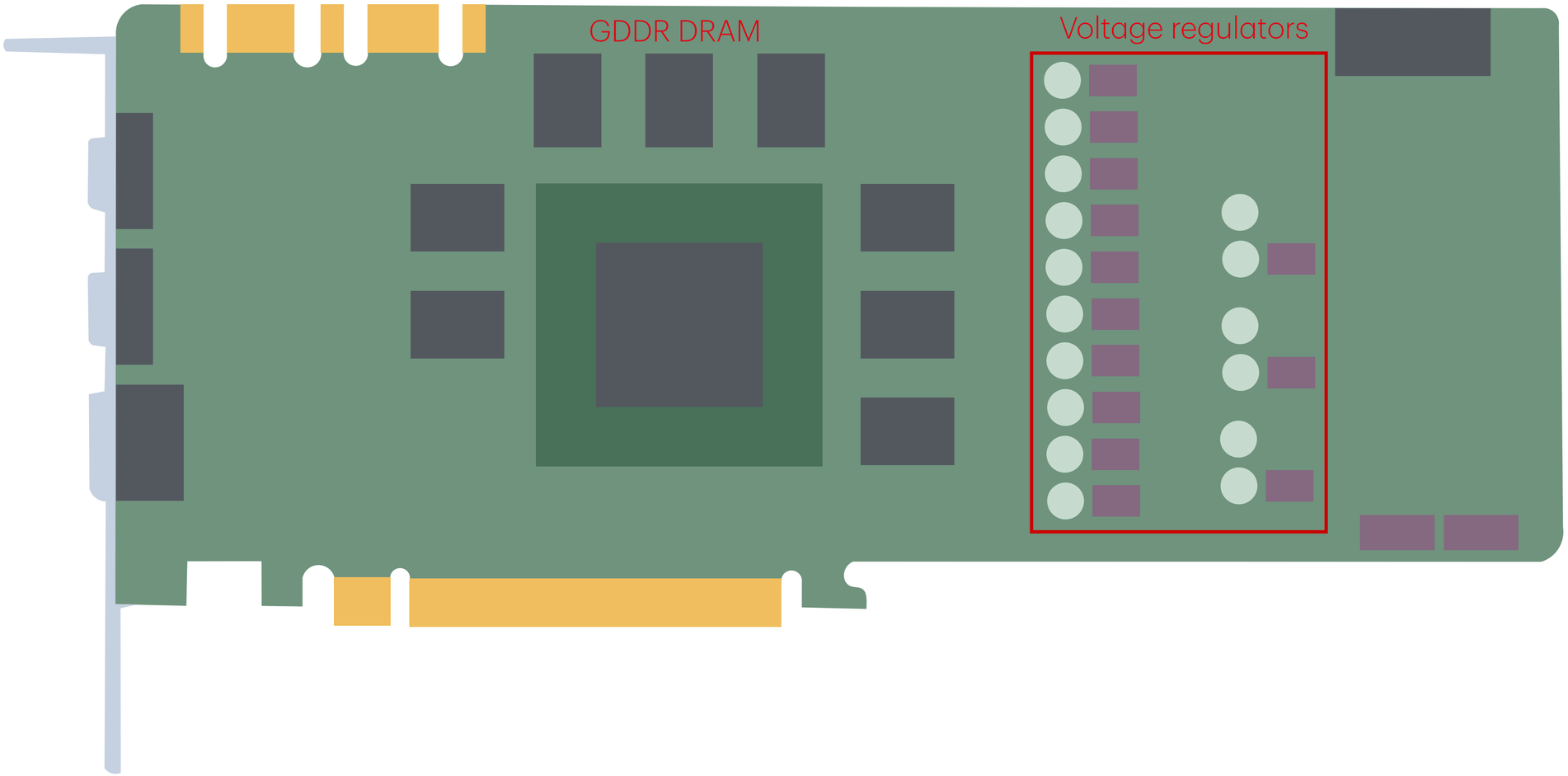
Graphics card

Circuit board to handle graphics operations



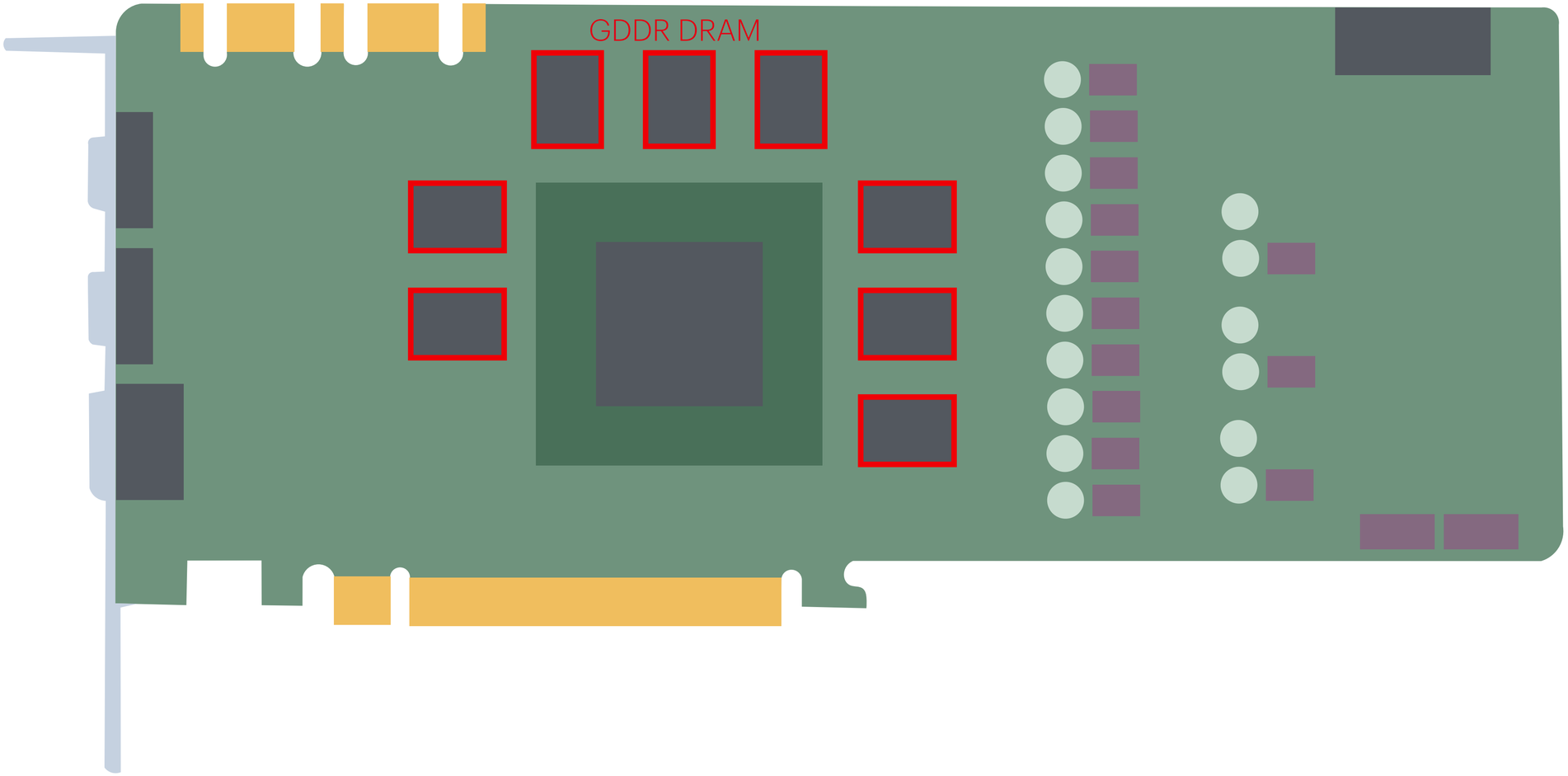
Graphics card

Circuit board to handle graphics operations



Graphics card

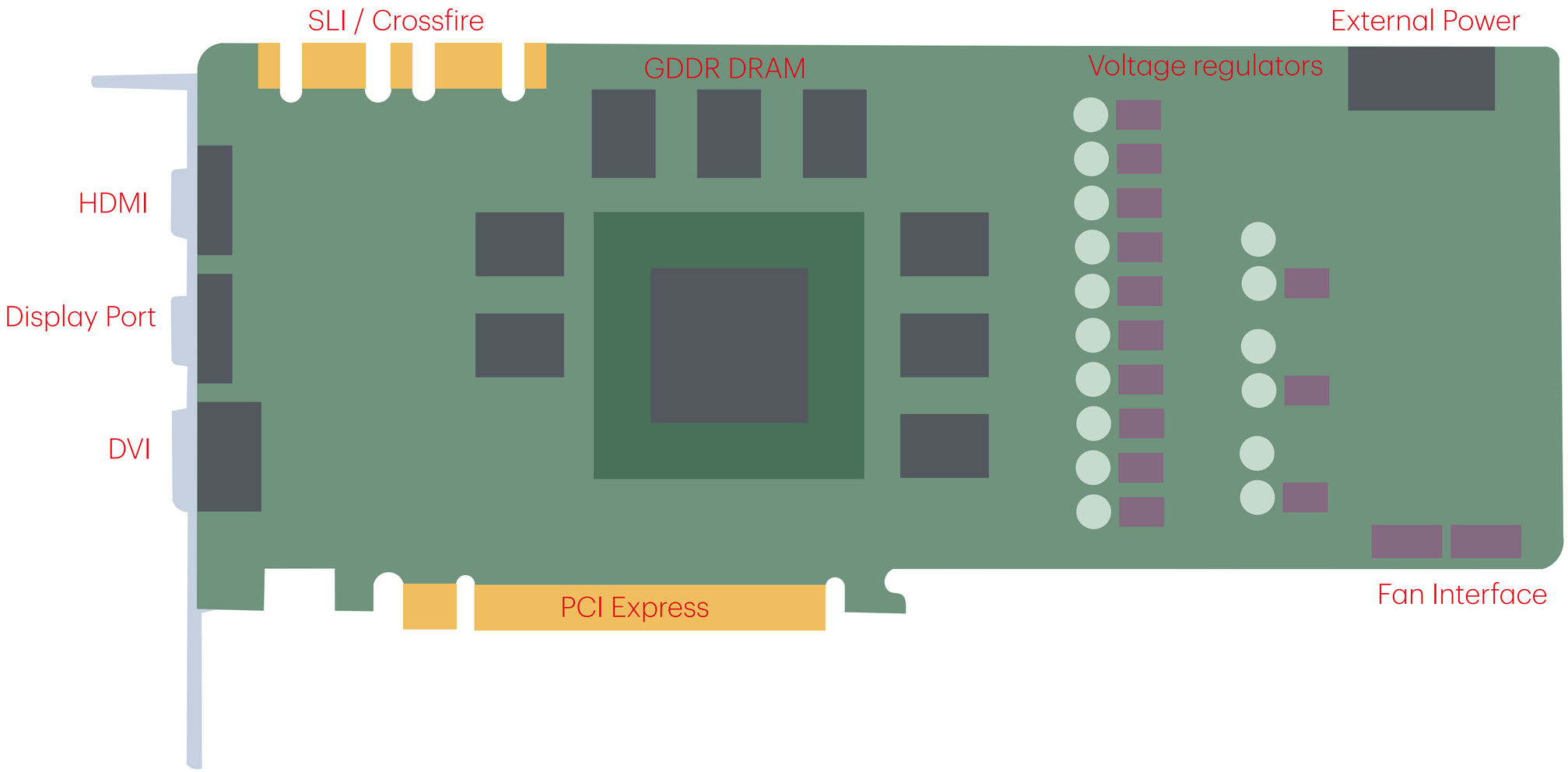
Circuit board to handle graphics operations



EPFL Center for Imaging

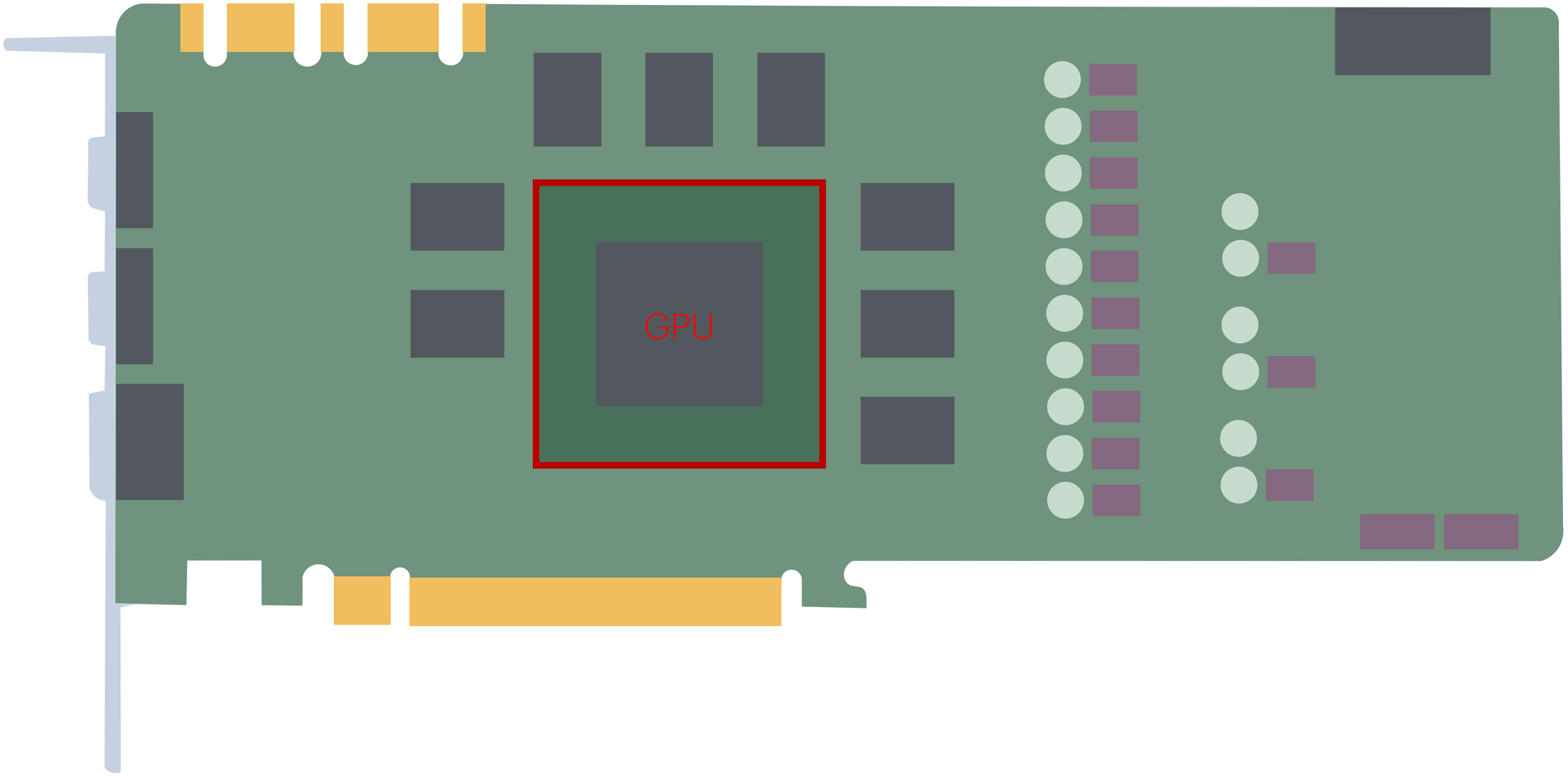
Graphics card

Circuit board to handle graphics operations



Graphics card

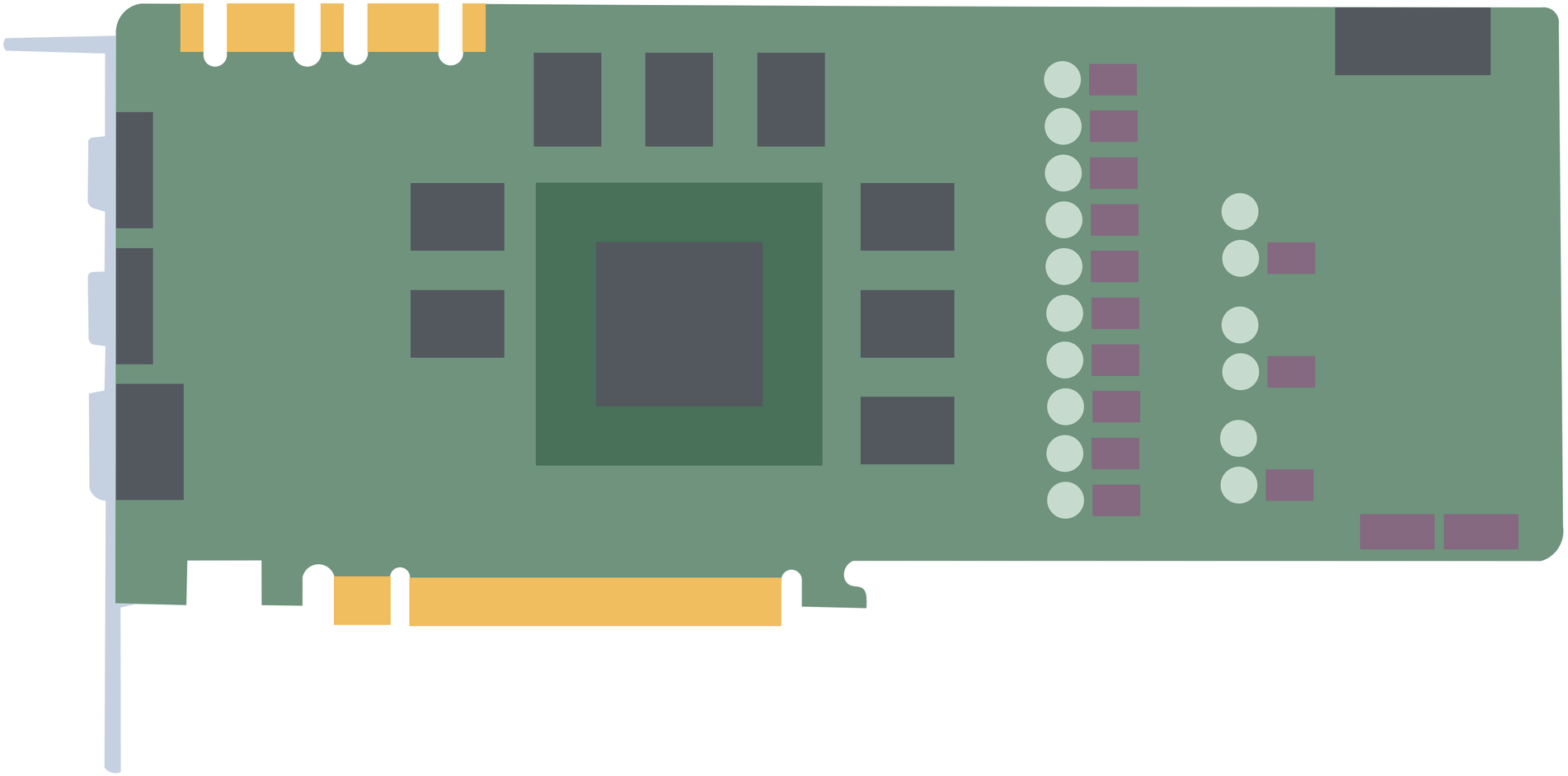
Circuit board to handle graphics operations



EPFL Center for Imaging

Graphics card

Circuit board to handle graphics operations

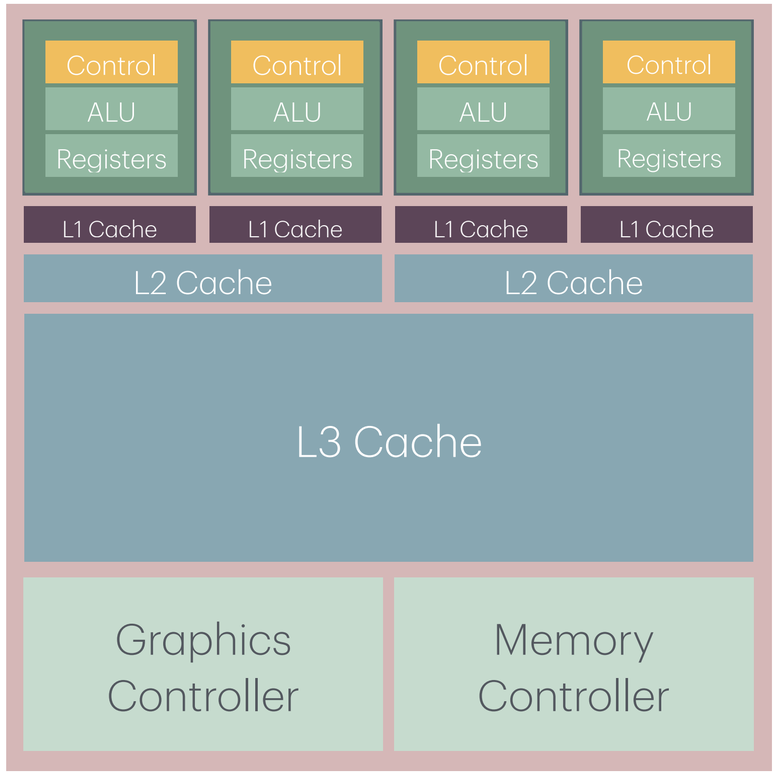


CPU vs GPU

EPFL Center for Imaging

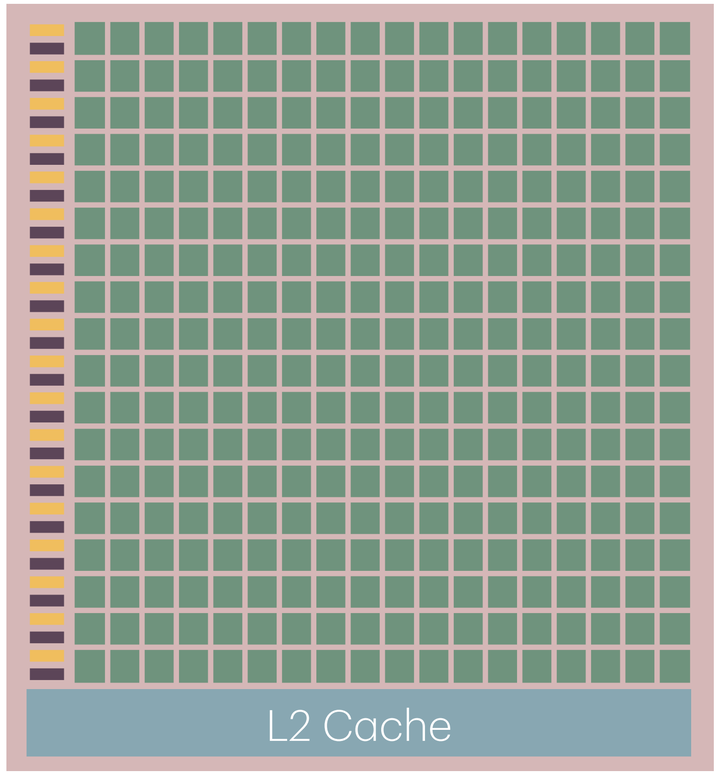
Low-Latency Optimization

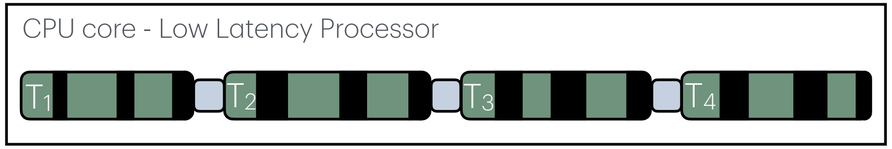
Advanced Control Logic

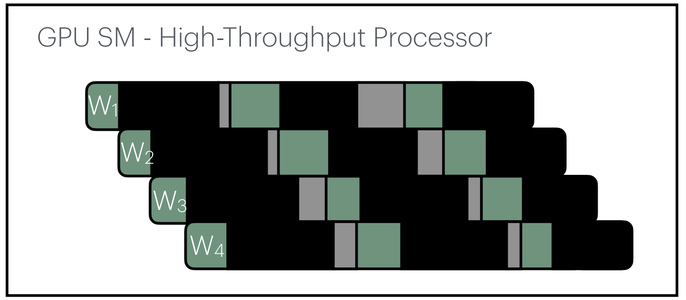


High Throughput

Computation-Heavy Design





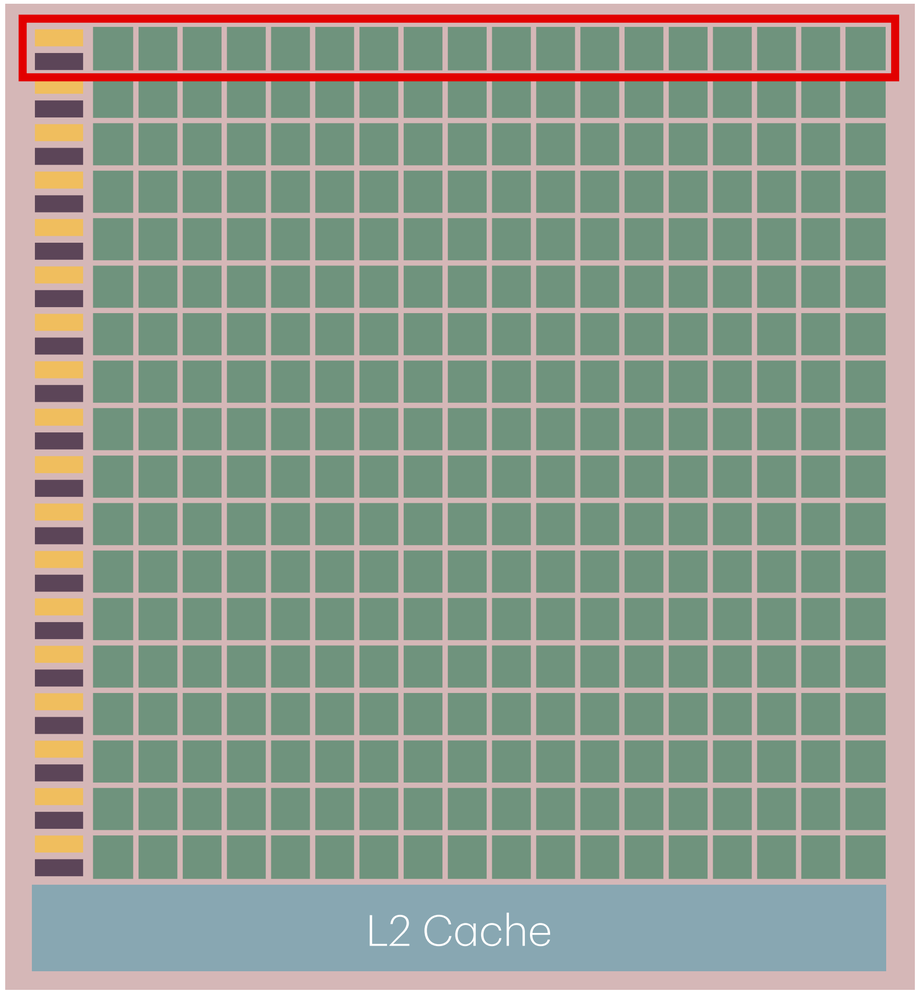


GPU chip architecture

Execution model

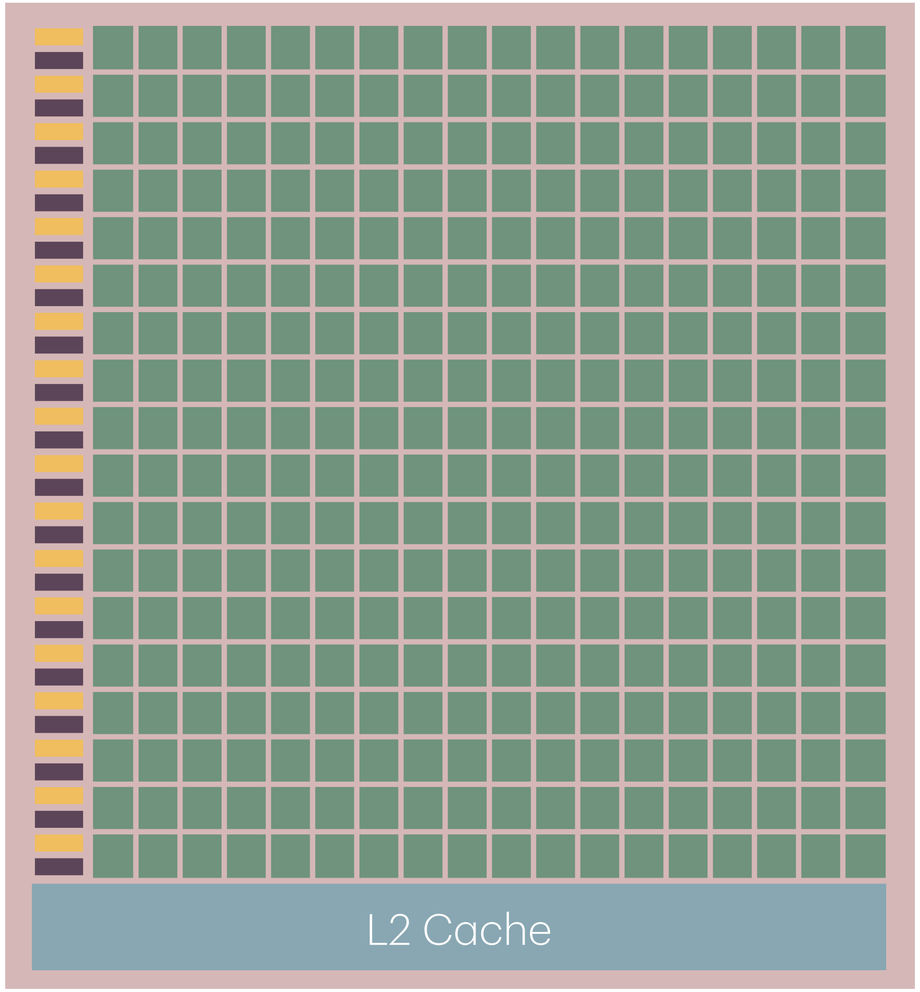
- **Stream processor (SP)**
  Equivalent to a CPU core. Designed to perform few operations but very quickly.

- **Stream Multiprocessor (SM)**
  A cluster of cores sharing a control unit and local memory (L1)
  (Single Instruction Multiple Data, SIMD paradigm)



GPU chip architecture

Execution Hierarchy

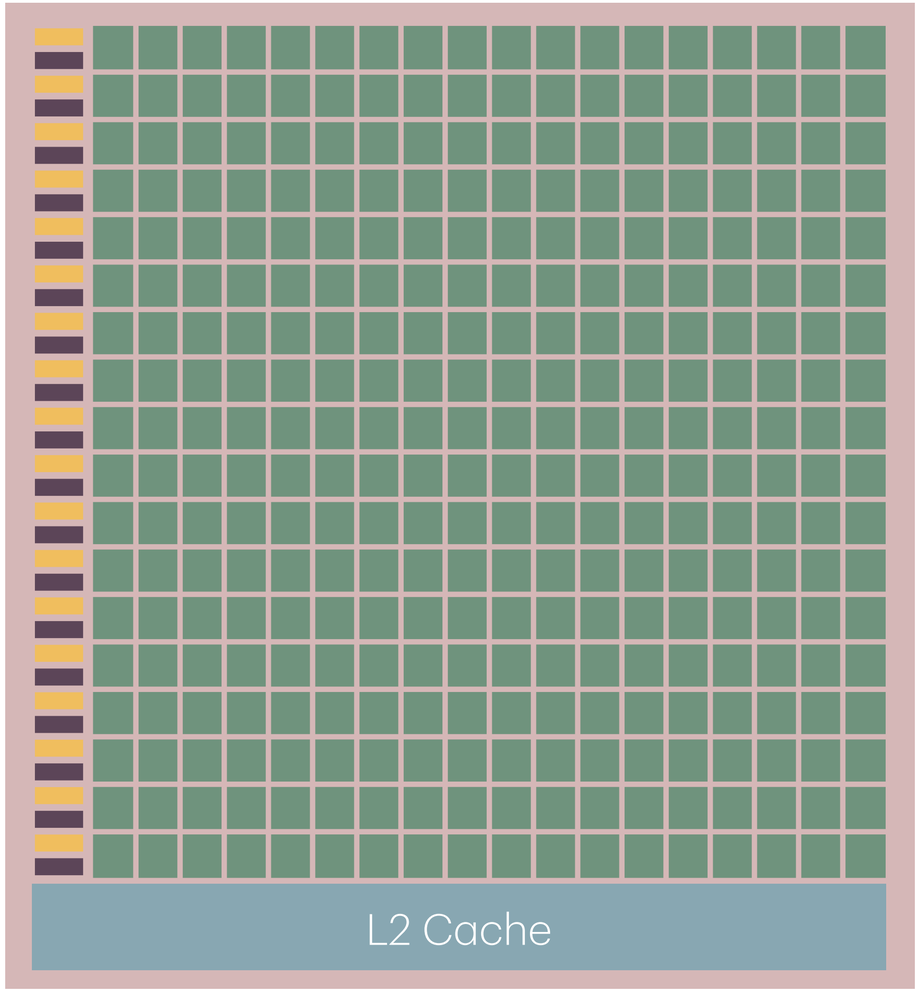


EPFL Center for Imaging

GPU chip architecture

Execution Hierarchy

- **Thread** (Gray[i] = 0.299 * R[i] + 0.587 * G[i] + 0.114 * B[i])
  Is the single execution of a kernel in a single piece of data.
  Runs on one CORE
  One CORE can run multiple threads (one after the other).
  Threads do not execute at the same rate.

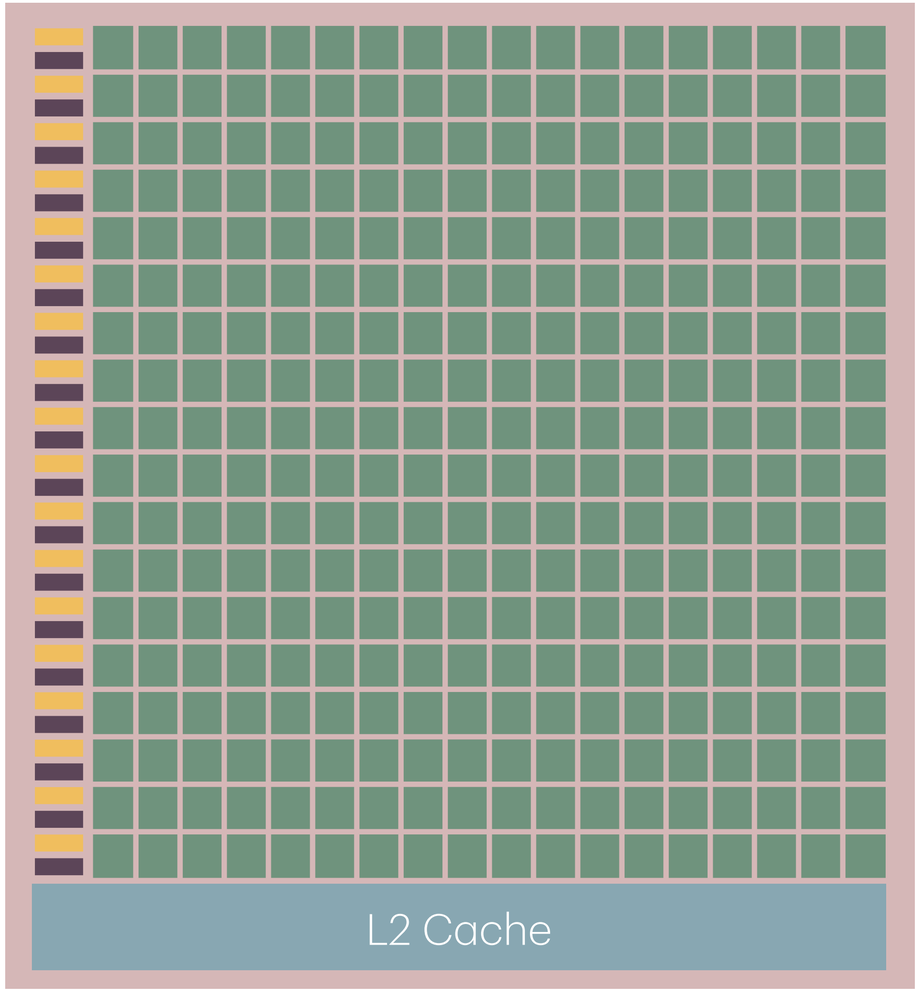


GPU chip architecture

Execution Hierarchy

- **Thread** (Gray[i] = 0.299 * R[i] + 0.587 * G[i] + 0.114 * B[i])
  Is the single execution of a kernel in a single piece of data.
  Runs on one CORE
  One CORE can run multiple threads (one after the other).
  Threads do not execute at the same rate.

- **Block**: Group of threads that physically share memory.
  Runs on one SM, one SM can run multiple blocks



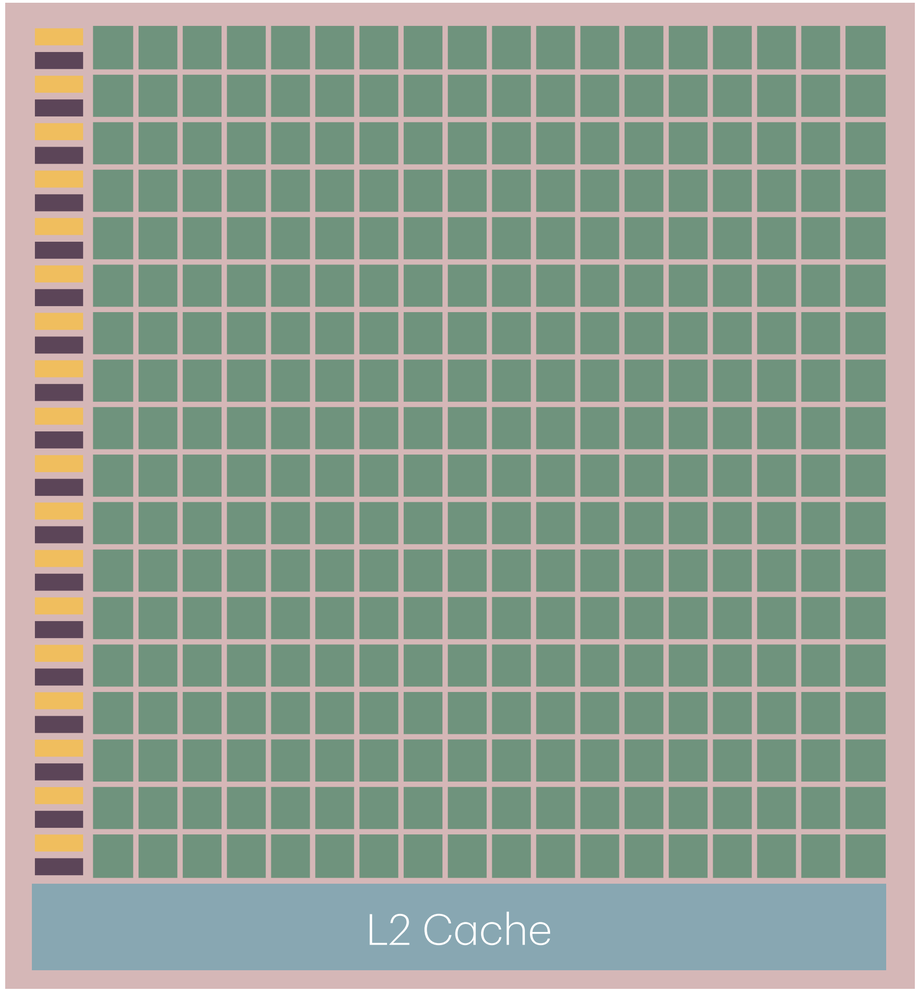
GPU chip architecture

Execution Hierarchy

- **Thread** (Gray[i] = 0.299 * R[i] + 0.587 * G[i] + 0.114 * B[i])
  Is the single execution of a kernel in a single piece of data.
  Runs on one CORE
  One CORE can run multiple threads (one after the other).
  Threads do not execute at the same rate.

- **Block**: Group of threads that physically share memory.
  Runs on one SM, one SM can run multiple blocks

- **Grid**: Group of blocks.



GPU chip architecture

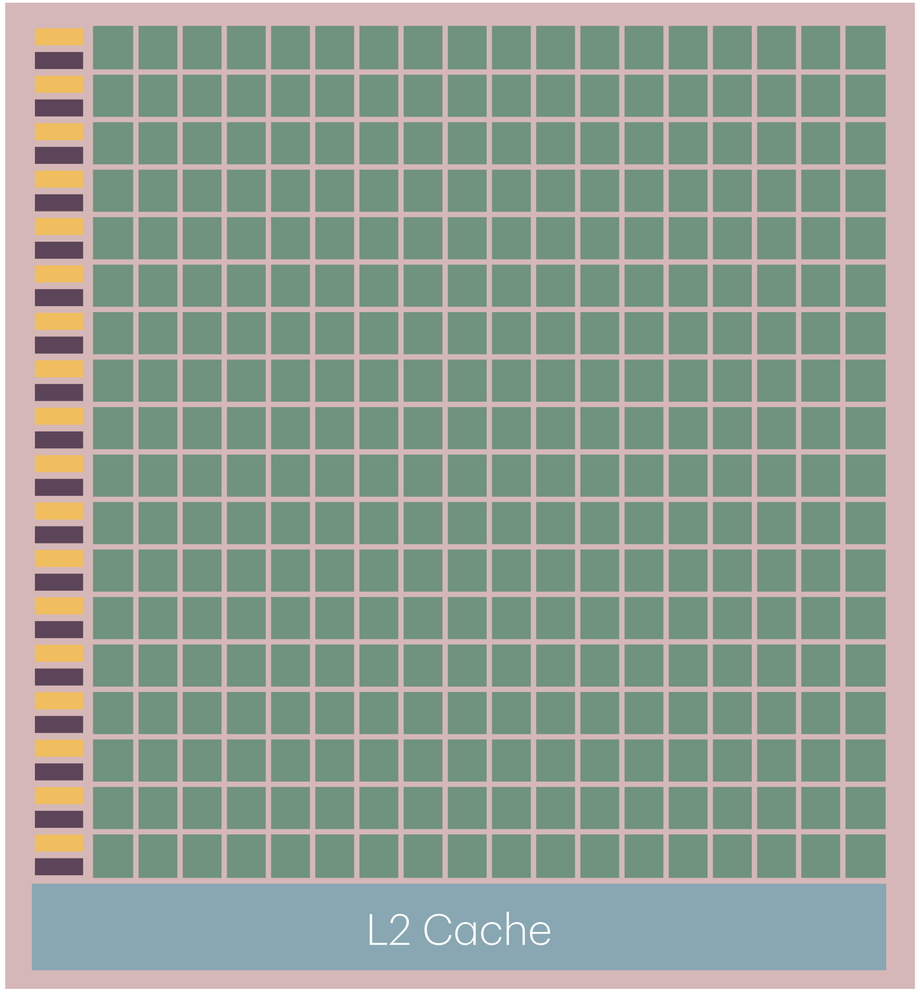
Execution Hierarchy

- **Thread** (Gray[i] = 0.299 * R[i] + 0.587 * G[i] + 0.114 * B[i])
  Is the single execution of a kernel in a single piece of data.
  Runs on one CORE
  One CORE can run multiple threads (one after the other).
  Threads do not execute at the same rate.

- **Block**: Group of threads that physically share memory.
  Runs on one SM, one SM can run multiple blocks

- **Grid**: Group of blocks.

- **Warp**: A fixed number of threads that execute the same instruction at the same time
  (NVIDIA GPUs —> Warp = 32 threads)



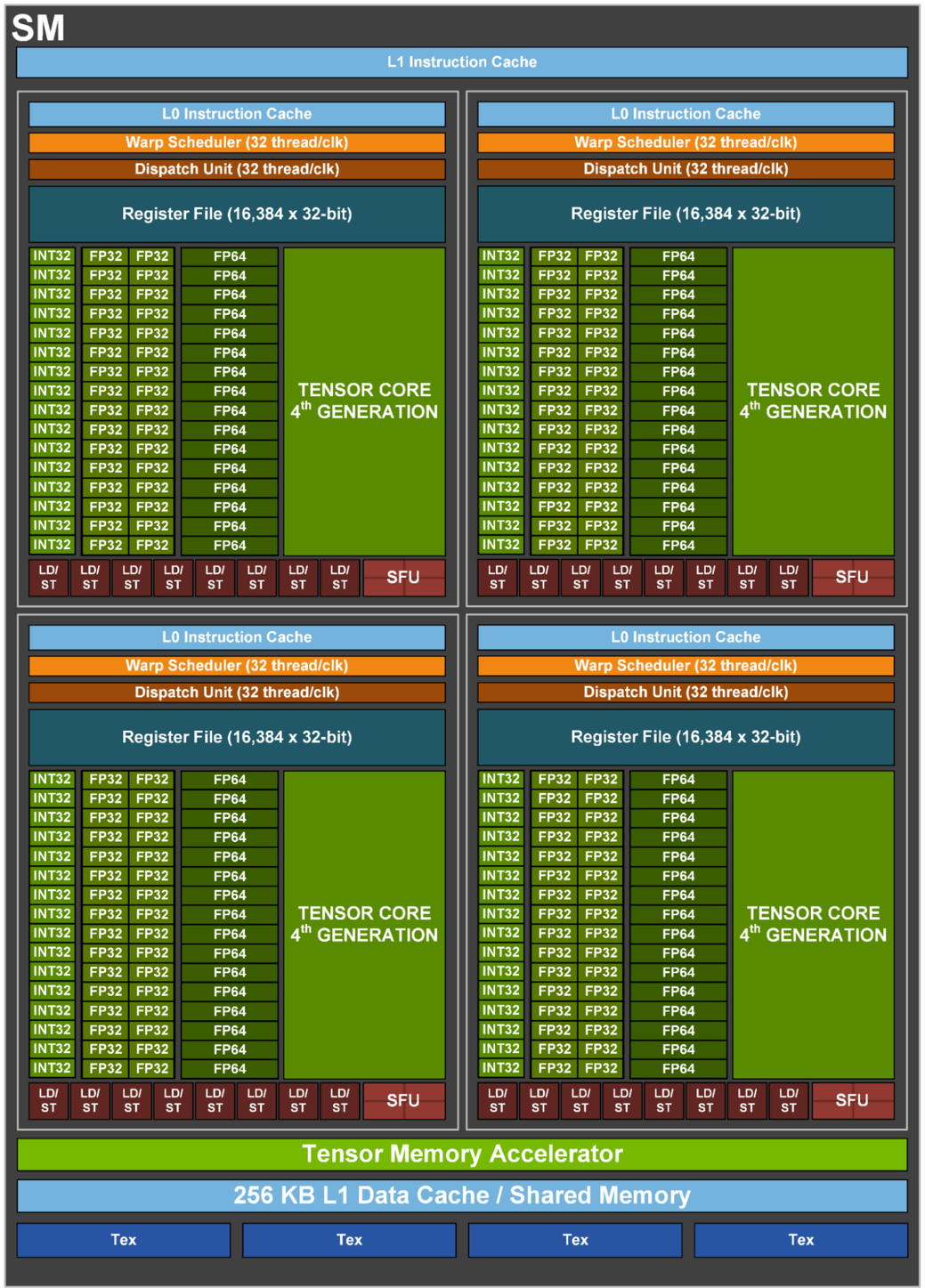
Nvidia H100 Full GPU with 144 SMs

EPFL Center for Imaging

EPFL Center for Imaging

128 FP32 CUDA Cores/SM

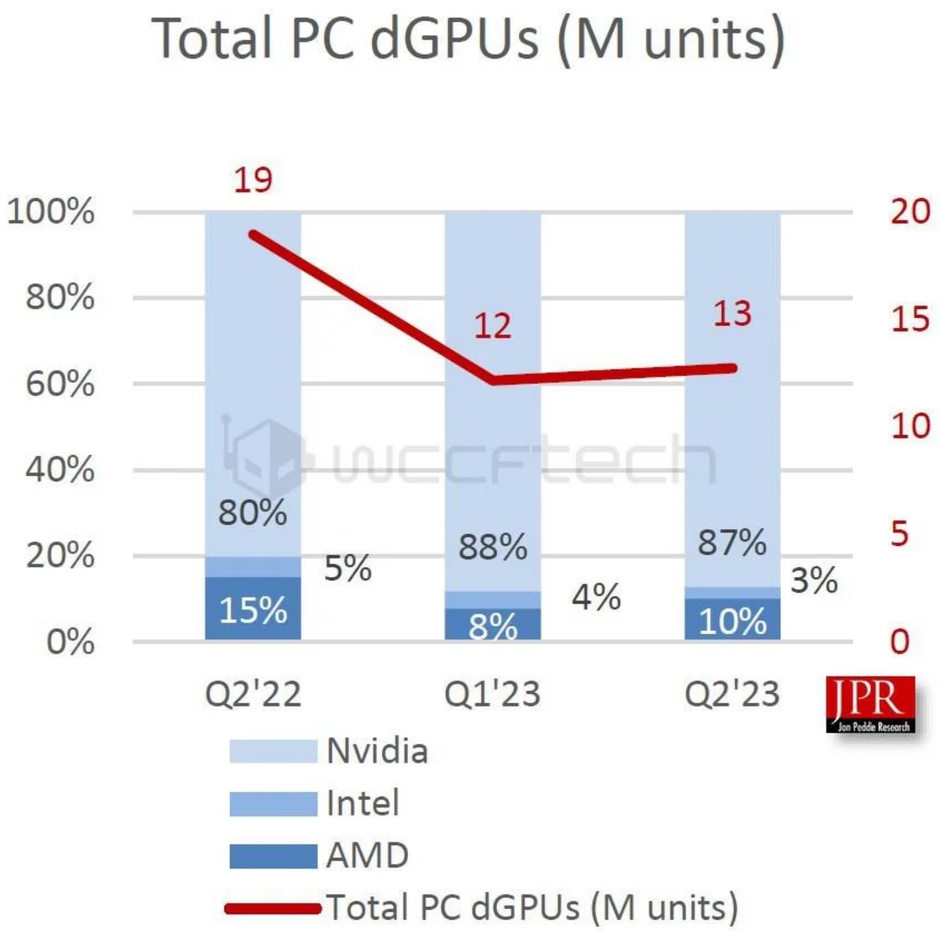
128 Cores/SM
x
144 SM/GPU
=
18,432 Cores/GPU



GPU chip architecture

Execution model

- Host-directed execution
    - CPU sets up and launches instructions on the GPU.
    - CPU manages the memory on the GPU (allocations — transfer in/out)
- Parallelism is needed to cover execution latencies



GPU Marketshare in Q2 2023:

NVIDIA 87%
AMD 10%
Intel 3%.

Desktop dGPUs
NVIDIA 80%
AMD 17%

Laptop dGPUs
NVIDIA 94%
AMD 3%

CUDA

The Basics

- **C**ompute **U**nified **D**evice **A**rchitecture
- Not a programming language —> API / Library for GPU computing
- Can be used from different programming languages
  (C/C+/Python/Matlab/etc.)
- Only for NVIDIA GPUs
  (AMD & Intel —> OpenCL)
- CUDA Kernel: A function to be executed on GPU (e.g., a c++ function)

CUDA

The Basics

- **C**ompute **U**nified **D**evice **A**rchitecture
- Not a programming language —> API / Library for GPU computing
- Can be used from different programming languages (C/C+/Python/Matlab/etc.)
- Only for NVIDIA GPUs (AMD & Intel —> OpenCL)
- CUDA Kernel: A function to be executed on GPU (e.g., a c++ function)

- **CUDA Drivers**: Software components that allow the operating system and applications to access and use the hardware capabilities of NVIDIA GPUs.
- **CUDA Toolkit**: A suite of compilers (nvcc), libraries (cuFFT, cuDNN), APIs (CUDA Runtime), and debuggers/profilers essential for building CUDA applications.

https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html

Python Libraries

for CUDA Programming

Low-level

PyCUDA (https://documen.tician.de/pycuda/)

Numba (https://numba.readthedocs.io/en/stable/)

CuPy (https://cupy.dev/)

cuCIM (https://docs.rapids.ai/api/cucim/stable/)

High-level

Jupyter notebook

https://github.com/EPFL-Center-for-Imaging/basics-GPU-image-proc

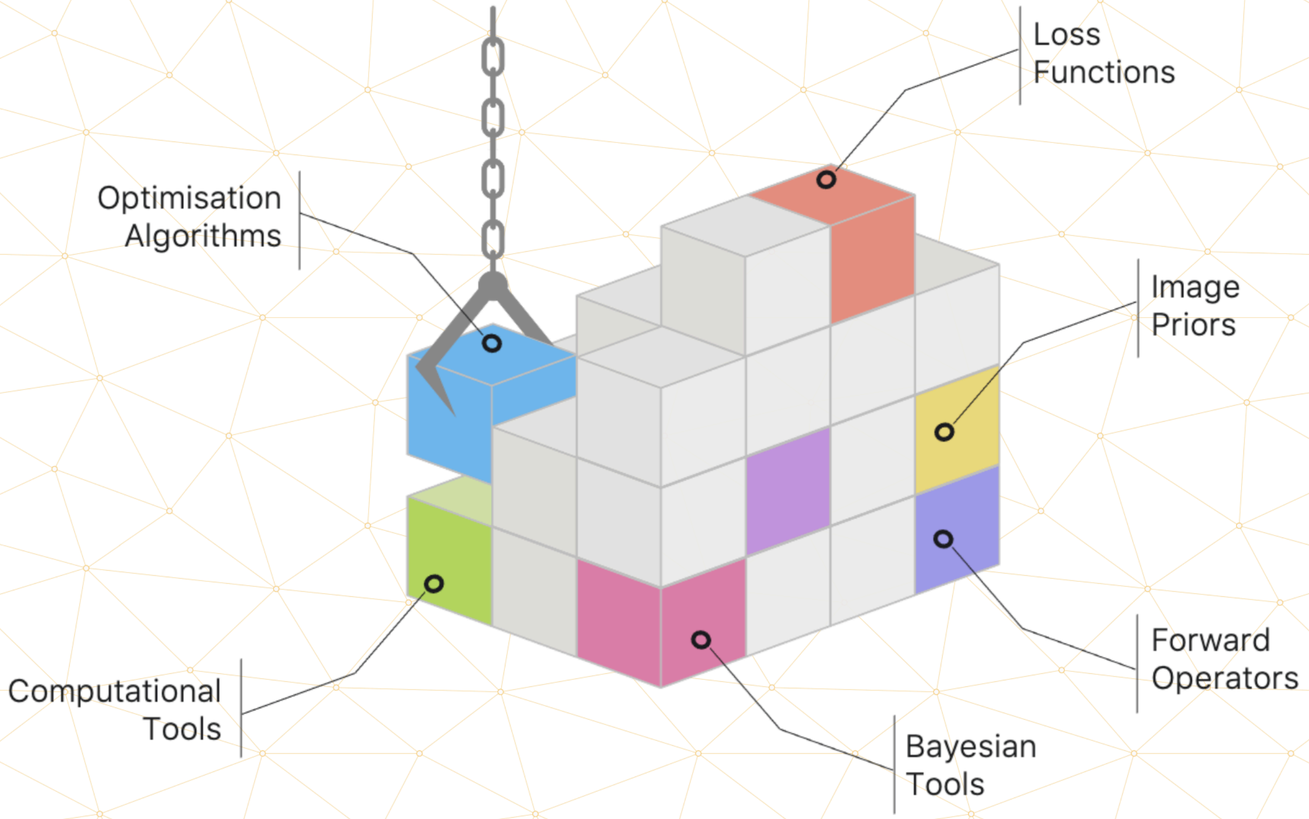
Pyxu

Modular & Scalable Computational Imaging

Pyxu (pronounced [piksu], formerly known as Pycsou) is an open-source Python framework allowing scientists at any level to quickly prototype/deploy *hardware accelerated and out-of-core* computational imaging pipelines at scale. Thanks to its hardware-agnostic **microservice architecture** and its tight integration with the PyData ecosystem, Pyxu supports a wide range of imaging applications, scales, and computation architectures.

Get Started | See Examples

See API Reference →



Feedback

EPFL Center for Imaging

Join at menti.com | use code 5483 3539