GPU Programming Concepts

GPU Introduction

Prof. Dr. Matthias S. Müller
Dr. Christian Terboven
Dr. Sandra Wienke
Julian Miller

i12 High Performance Computing
RWTH AACHEN UNIVERSITY

What is This Chapter About?

- Important concepts of programming GPUs
  - Why use accelerators?
  - Power efficiency of accelerators
  - Trends in HPC
  - Comparison of CPU and GPU architectures

Why Use Accelerators?

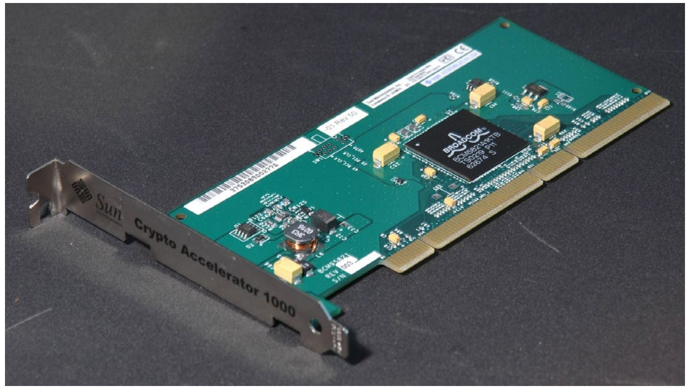
Hardware Accelerators

- Definition: A hardware component to speed up some aspect of the computing workload.

Computation: Intel 80386DX CPU with 80387DX Math Coprocessor

Generic FPGA: A Stratix IV FPGA from Altera

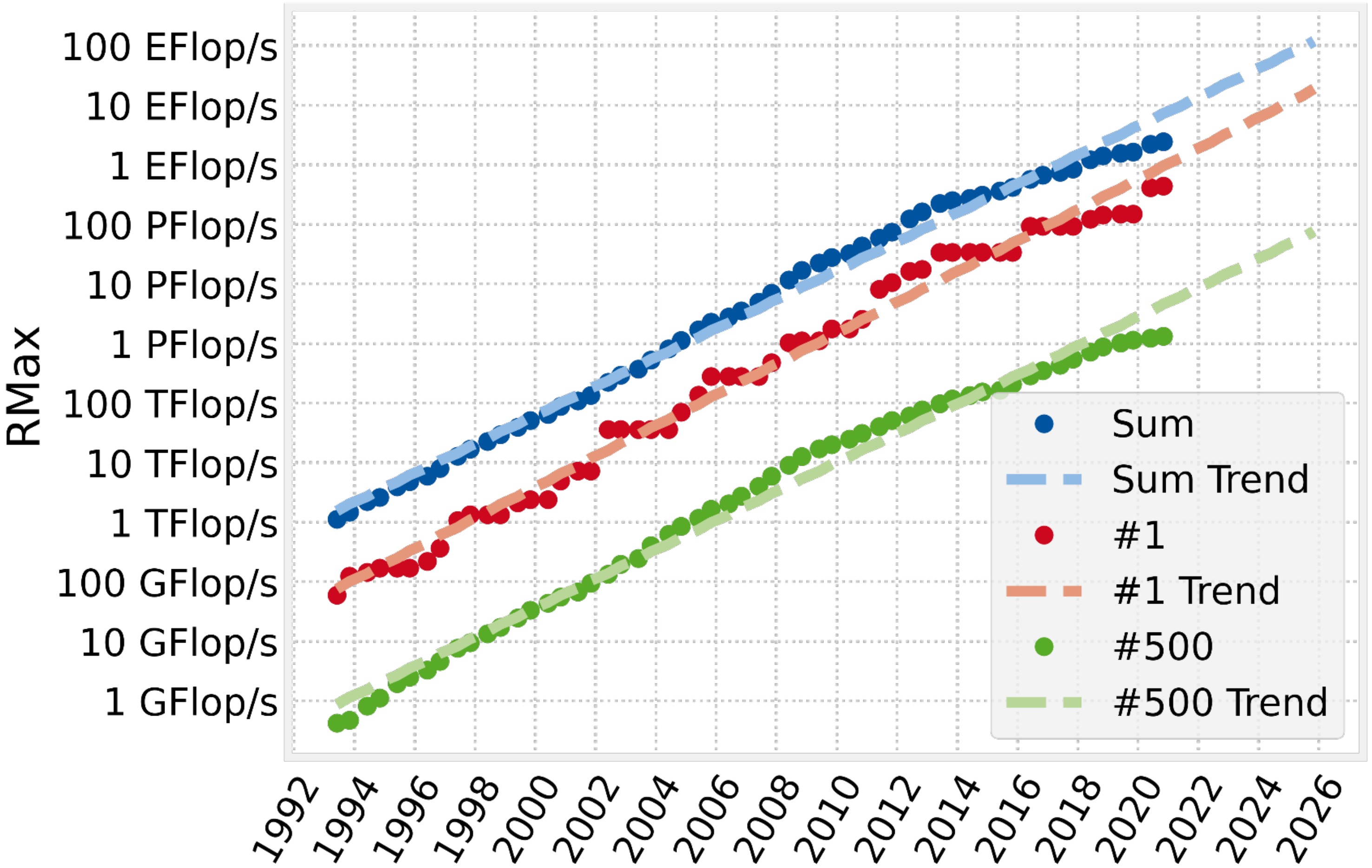
Digital signal processor (DSP), e.g. in music instruments



Encryption: PCI-X Crypto Accelerator

Why Use Accelerators?

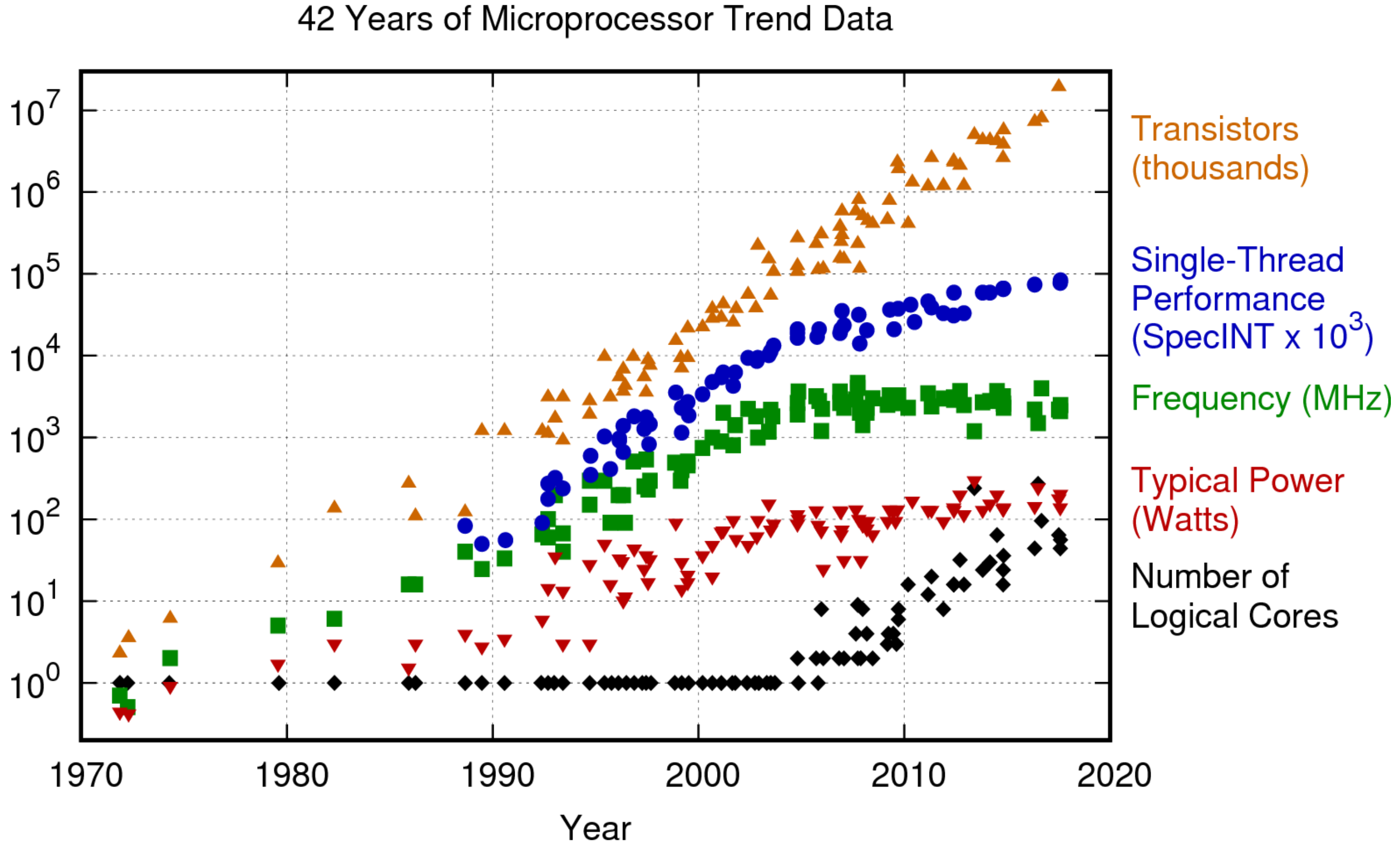
- Demand for compute capacity increases steadily while power is constrained (~20 MW)



Source:Top500 11/2020

Hardware Accelerators

- CPU single core performance increases only slowly since ~2005
  - Total performance of chip increases through raising number of logical cores (~power law)
- Frequency stagnates
  - Power and cooling constraints



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Trade-off Clock Frequency for Power

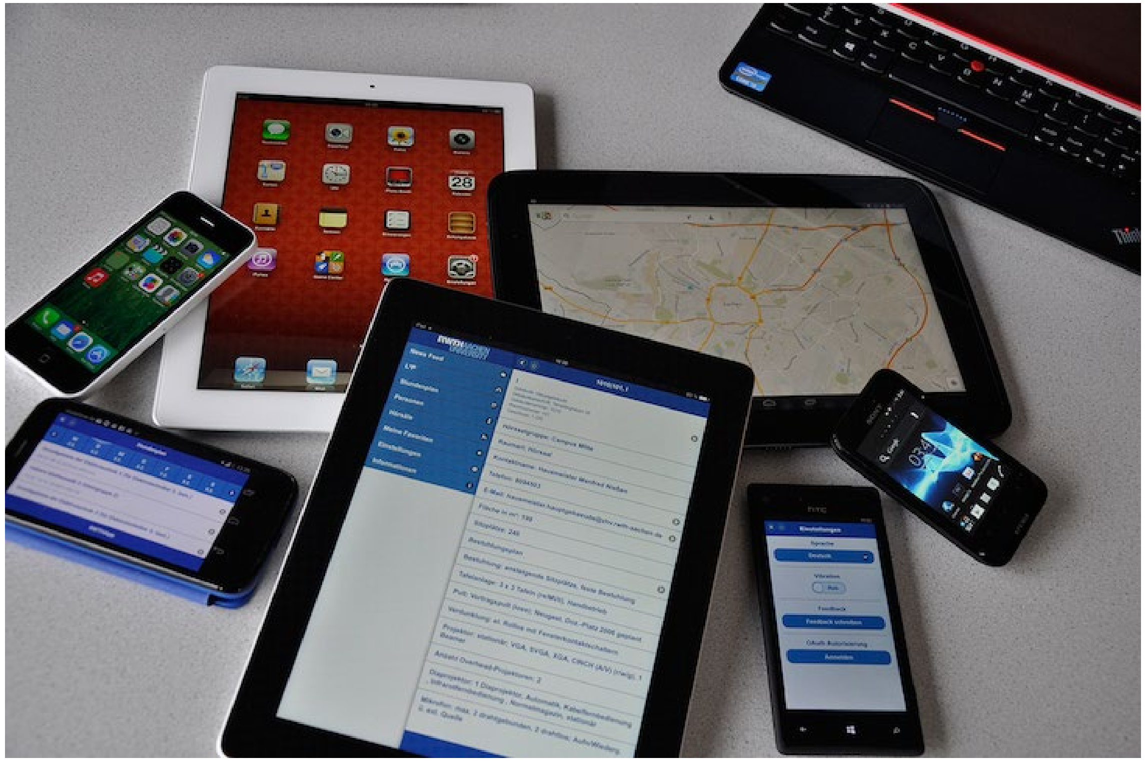
- Power consumption P [W] of a processor:

$$\boldsymbol{P \sim V^2 f}$$

- Rule of thumb: Reduction of 1% voltage and 1% frequency reduces the power consumption by 3% and the performance by 0.66%.

- Example: ARM Cortex A57
  - 4-8 cores @ 1-2 GHz
  - LPDDR, Modem, GPU, DSP,…
  - TDP 2-3 W

P: power consumption [W]
V: voltage [V]
f: clock frequency [Hz]



Source: IT Center, RWTH

Trade-off Parallelism for Power

- Vectorization: single instruction defines n operations (instruction-level parallelism)

- Larger vectors lead to more operations per cycle within a similar power envelop
  - “The most efficient way to execute a vectorizable applications is a vector processor” (Jim Smith)

- 4 DP operations per cycle

- 8 DP operations per cycle
  - doubled performance
  - roughly same power

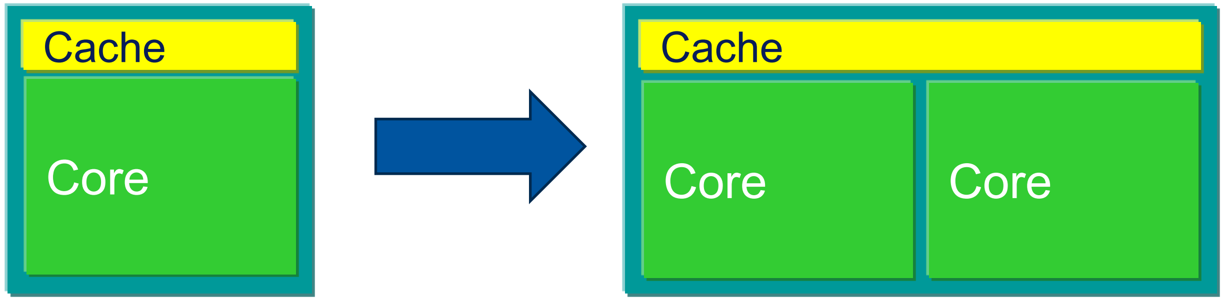
Trade-off Parallelism for Power

- Power consumption P [W] of a processor:

$$P \sim V^2 f$$

P: power consumption [W]
V: voltage [V]
f: clock frequency [Hz]

- Rule of thumb: Reduction of 1% voltage and 1% frequency reduces the power consumption by 3% and the performance by 0.66%.




```
Voltage = 1
Freq    = 1
Area    = 1
Power   = 1
Perf    = 1
```

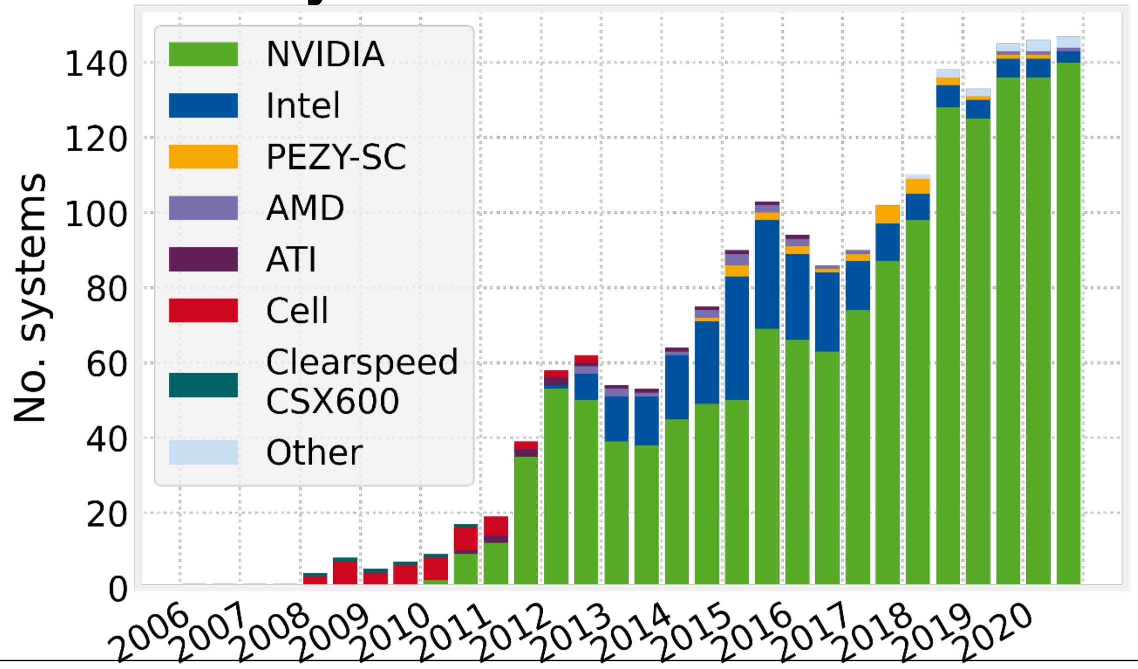
```
Voltage = -15%
Freq    = -15%
Area    = 2
Power   = ~1
Perf    = ~1.8
```

(Based on slides from Shekhar Borkar, Intel Corp.)

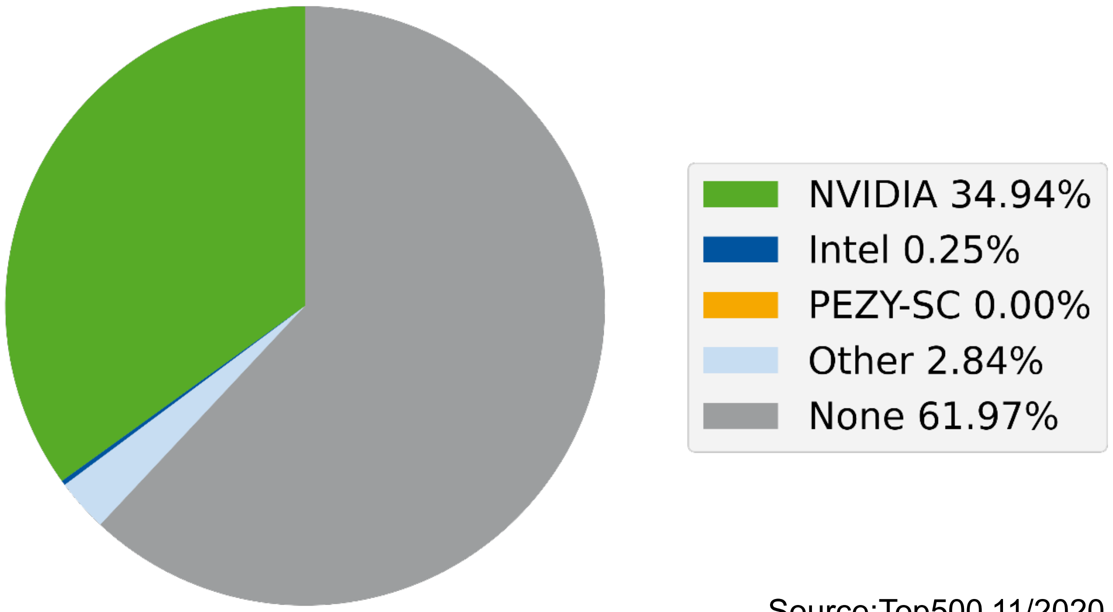
Trends in HPC

- Typical Hardware Accelerators in HPC Systems
  - GPGPUs (e.g. NVIDIA, AMD)
  - Intel Many Integrated Core (MIC) Architecture (Intel Xeon Phi)
  - FPGAs (e.g. Altera), DSPs (e.g. TI), PEZY-SC, …

System share over time



Performance share (Nov 2020)



Source:Top500 11/2020

Power Efficiency – Green500

performance per watt

Rank	Name	Site*	GFLOPS/W	Total Power (kW)	Proc/ Accelerator
1	NVIDIA DGX SuperPOD	NVIDIA Corporation, United States	26.20	90	NVIDIA A100
2	MN-3	Preferred Networks, Japan	26.04	65	MN-Core
3	JUWELS Booster Module	Forschungszentrum Juelich (FZJ), Germany	25.01	1764	NVIDIA A100
4	Spartan2	Atos, France	24.26	106	NVIDIA A100
5	Selene	Nvidia Corporation, United States	23.98	2646	NVIDIA A100
6	A64FX prototype	Fujitsu Numazu Plant, Japan	16.88	118	Fujitsu A64FX
7	AiMOS	Rensselaer Polytechnic Institute Center for Computational Innovations (CCI), United States	16.28	512	NVIDIA Volta GV100
8	HPC5	Eni S.p.A., Italy	15.74	2252	NVIDIA V100
9	Satori	MIT/MGHPCC Holyoke, MA, United States	15.57	94	NVIDIA Tesla V100 SXM2
10	Supercomputer Fugaku	RIKEN Center for Computational Science, Japan	15.42	30	Fujitsu A64FX

Source:Green500 11/2020

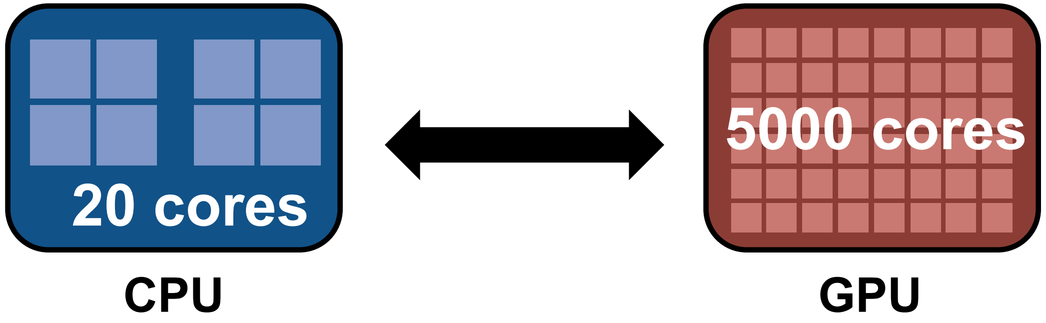
Overview GPUs

Overview

- GPGPUs = General Purpose Graphics Processing Units

- History – a very brief overview
  - ‘80s - ‘90s: Development is mainly driven by games
    Fixed-function 3D graphics pipeline
    Graphics APIs like OpenGL, DirectX popular
  - Since 2001: Programmable pixel and vertex shader in graphics pipeline (adjustments in OpenGL, DirectX)
    Researchers take notice of performance growth of GPUs: Tasks must be cast into native graphics operations
  - Since 2006: Vertex/pixel shader are replaced by a single processor unit
    Support of programming language C, synchronization,…
    → “General purpose”

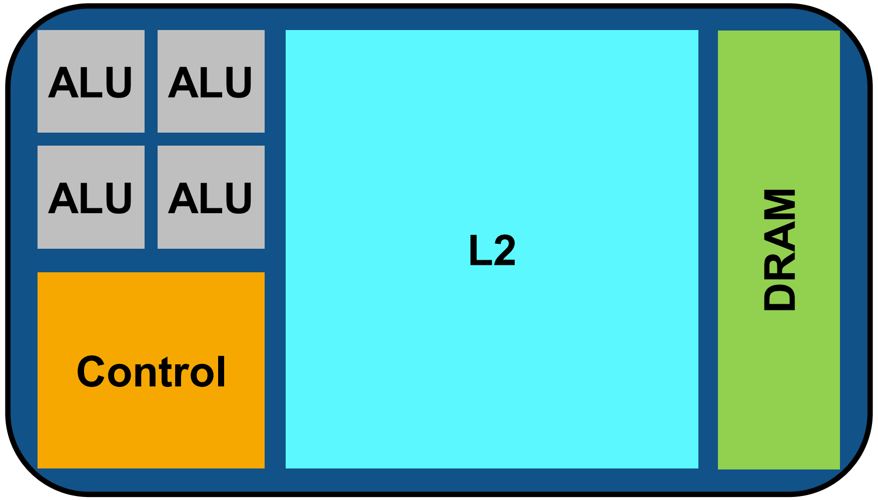
Comparison CPU ⇔ GPU



- GPU-Threads
  - Scheduled chain of instructions running on a CUDA core (basically a pipeline)
  - Light-weight, little creation overhead, fast context switching
  - SMT on CPU: few thread share core to better utilize execution units
  - GPU threads: up to 32 threads per core to hide memory latencies

- Lots of parallelism needed on GPU to get good performance!

Comparison CPU ⇔ GPU – Hardware Design



CPU

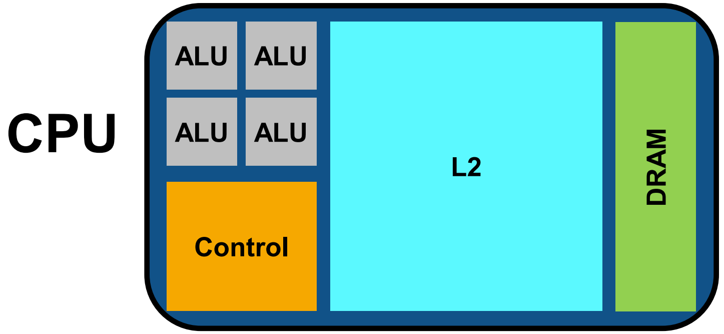
- Optimized for low latencies
- Huge caches
- Control logic for out-of-order and speculative execution
- **Targets on general-purpose applications**

GPU

- Optimized for data-parallel throughput
- Architecture tolerant of memory latency
- More transistors dedicated to computation
- **Suited for special kind of apps**

Why Can Accelerators Deliver Good Performance Watt Ratio?

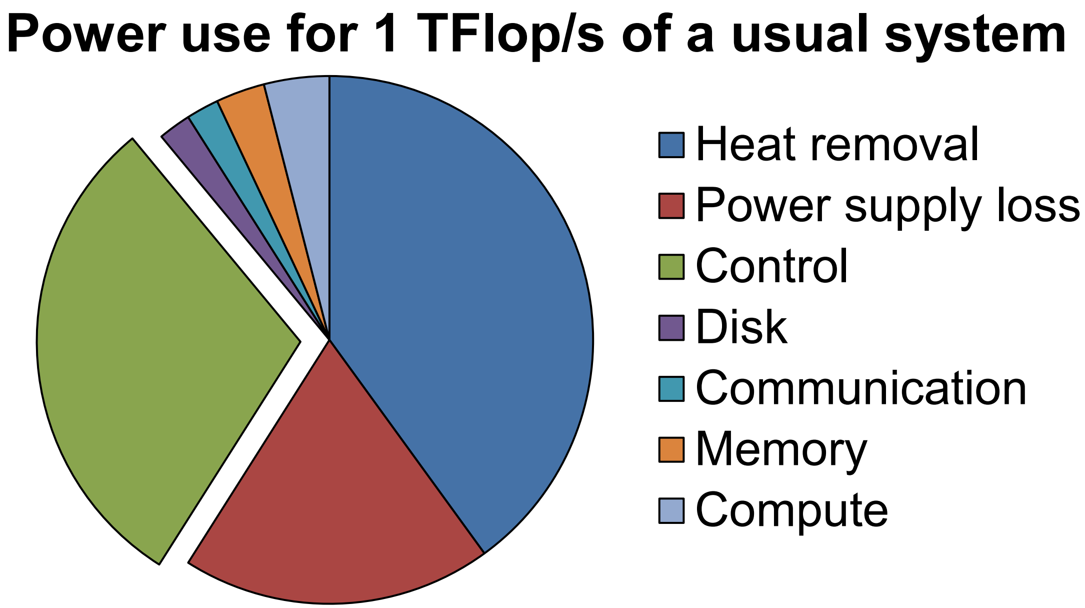
1. High (peak) performance
   - More transistors for computation
     - No control logic
     - Small caches



2. **Low power consumption**
   - Many low frequency cores

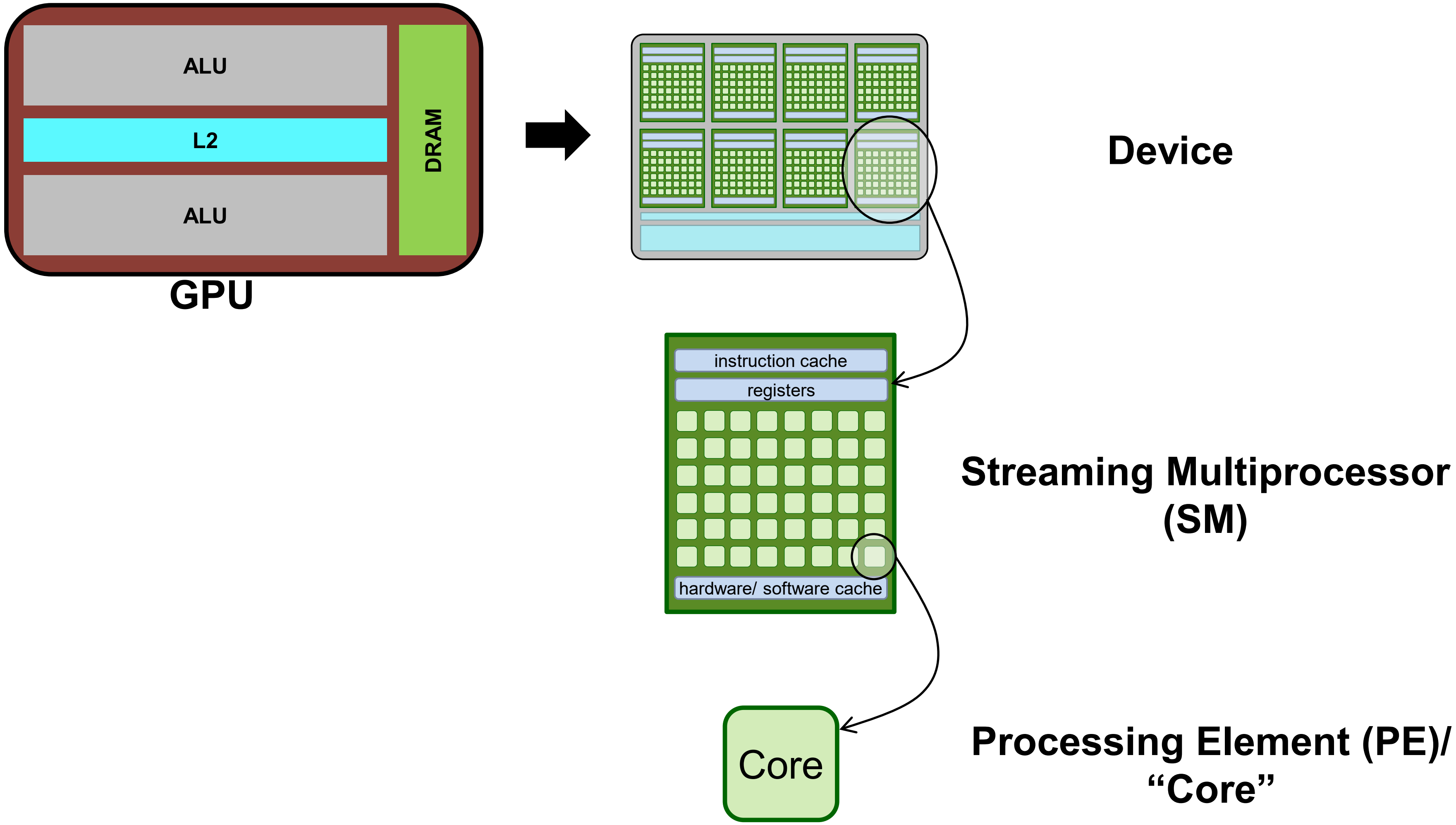
$$P \sim V^2 \cdot f$$

   - No control logic



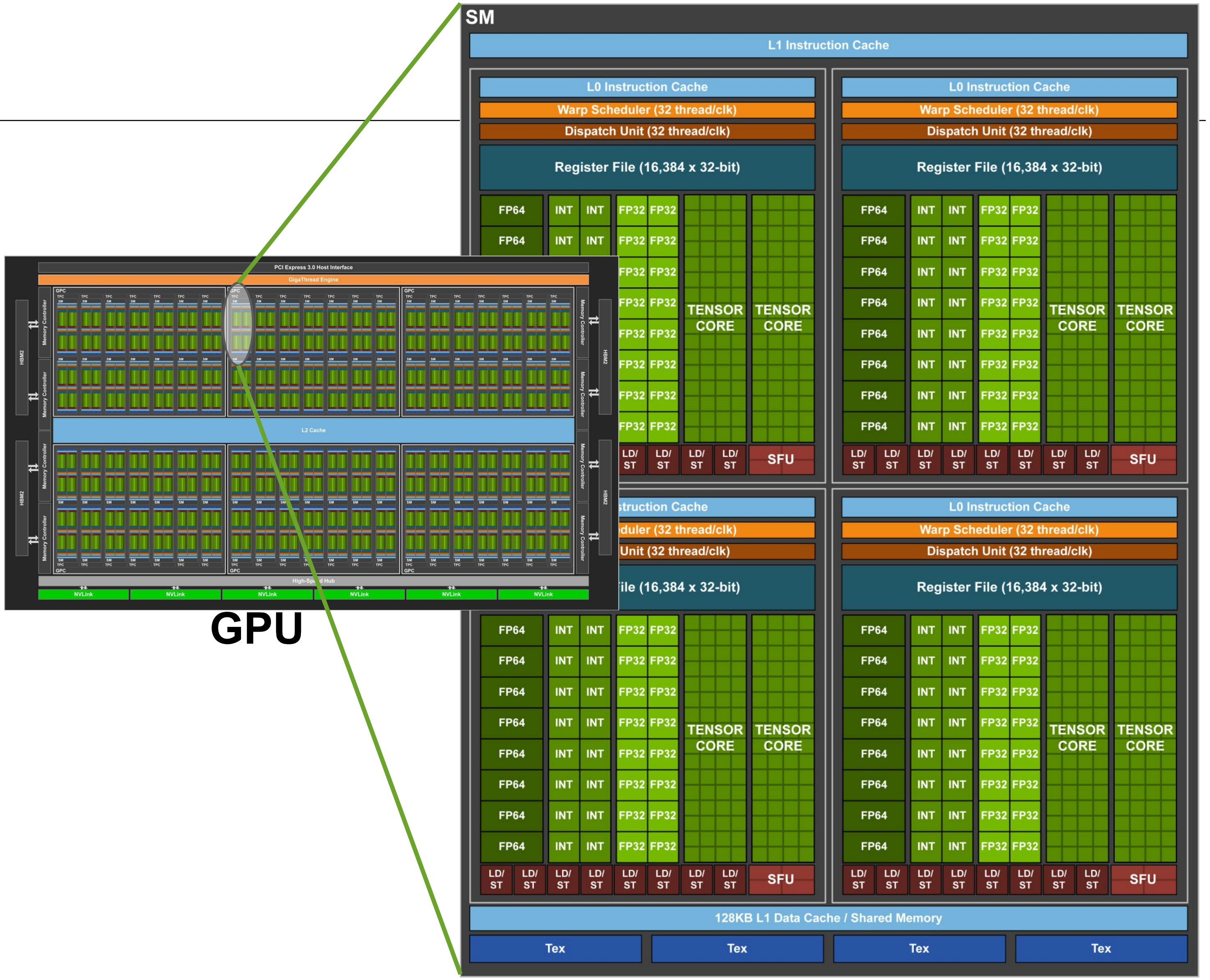
Source: Andrey Semin (Intel): HPC systems energy efficiency optimization thru hardware-software co-design on Intel technologies, EnaHPC 2011; & S. Borkar, J. Gustafson

GPU architecture: Abstraction



GPU architecture: Volta (V100)

- 21.1 billion transistors
- 80 streaming multiprocessors (SM)
  - Each: 64 (SP) cores, 32 (DP) cores, 8 Tensor cores
- Peak performance
  - SP: 15.7 Tflops
  - DP: 7.8 Tflops
  - Tensor: 125 Tflops
- 32 GB / 16 GB HBM2 memory
  - 900 GB/s bandwidth
- 300W thermal design power



Source: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

CUDA Books & Links

- Nvidia CUDA Zone (Toolkit, Profiler, SDK, documentation,…): https://developer.nvidia.com/cuda-zone
  - CUDA Programming Guide
  - CUDA Best Practice Guide

- List of books: https://developer.nvidia.com/cuda-books-archive, e.g.:
  - David Kirk and Wen-Mei W. Hwu: *Programming Massively Parallel Processors – A Hands-on Approach* (2010)
  - Jason Sanders and Edward Kandrot: *CUDA by Example: An Introduction to General-Purpose GPU Programming (2010)*
  - The CUDA Handbook: A Comprehensive Guide to GPU Programming (2nd edition in 2020)