Heterogeneous Parallel Computing

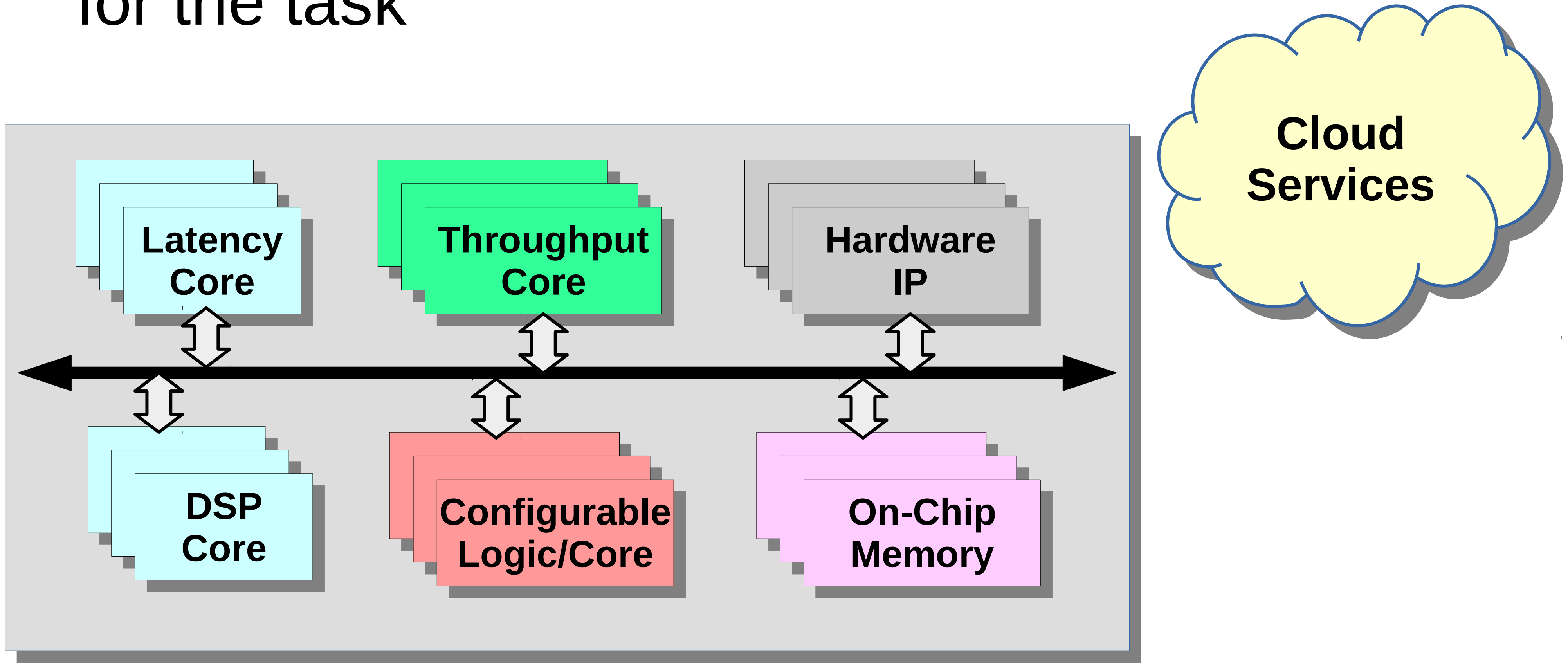
Introduction

Outline

- System-on-Chip
- Latency oriented cores – CPU
- Throughput oriented cores – GPU
- Scalability and Portability

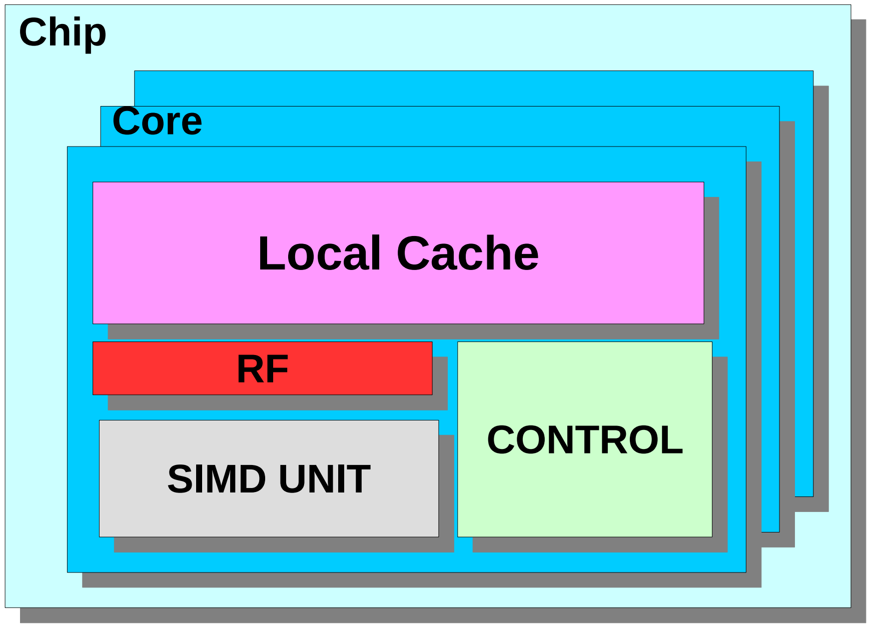
Heterogeneous Parallel Computing

- Heterogeneity – use the best hardware block for the task



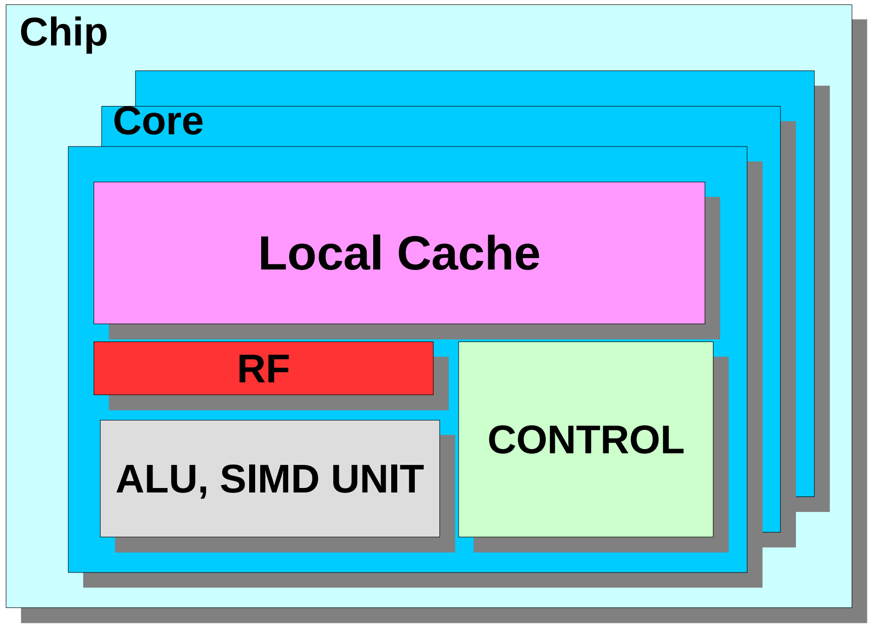
CPU vs. GPU

CPU – Latency Oriented Cores

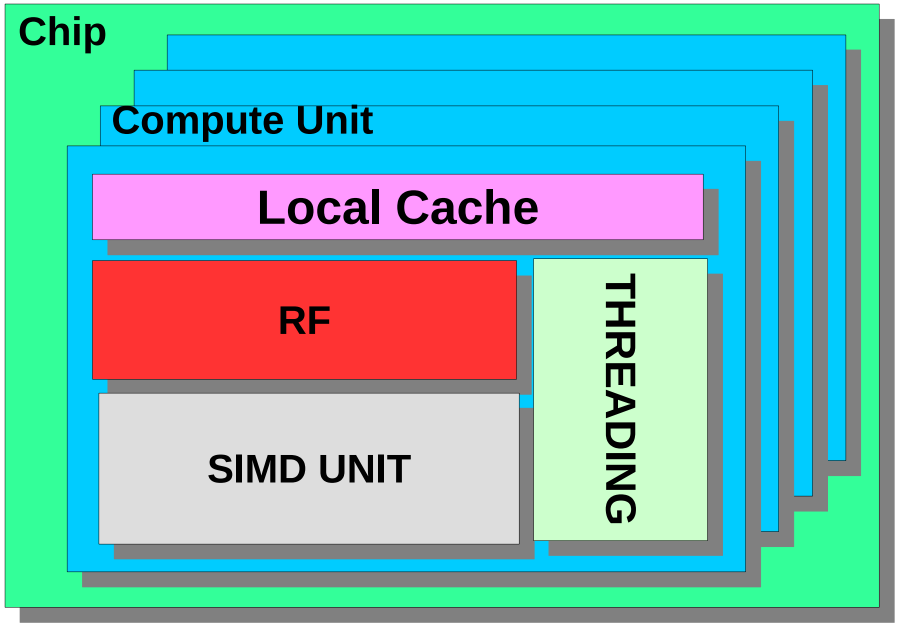


CPU vs. GPU

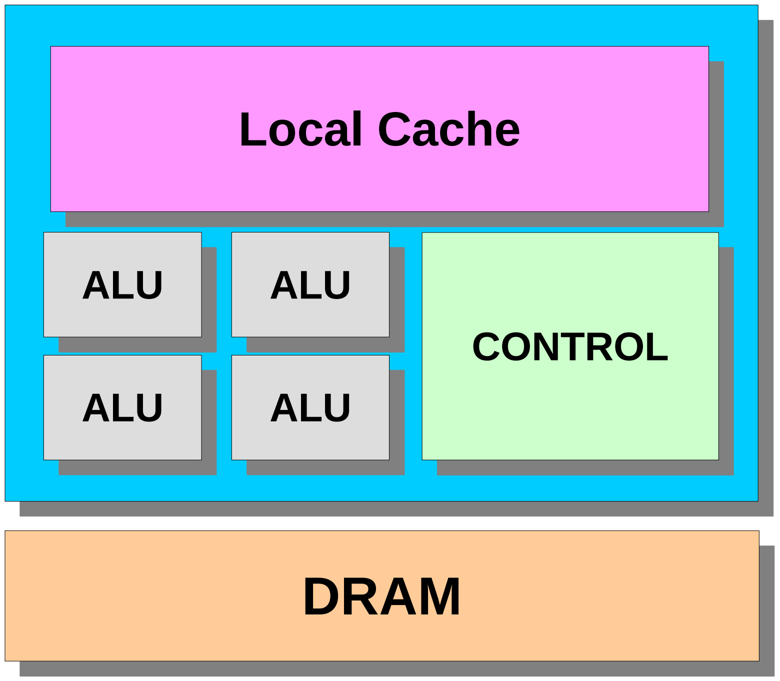
CPU – Latency Oriented Cores



GPU – Througput Oriented Cores

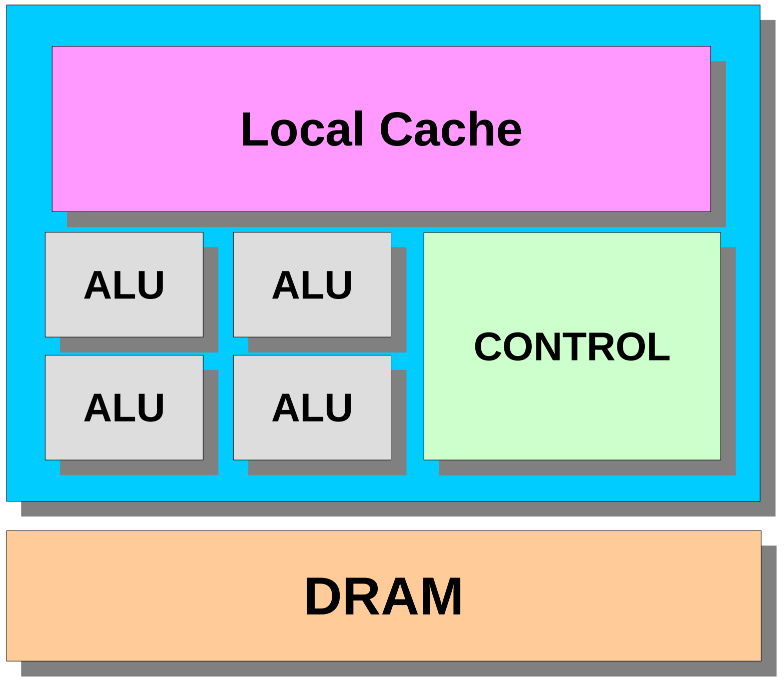


Latency Oriented Design – CPU



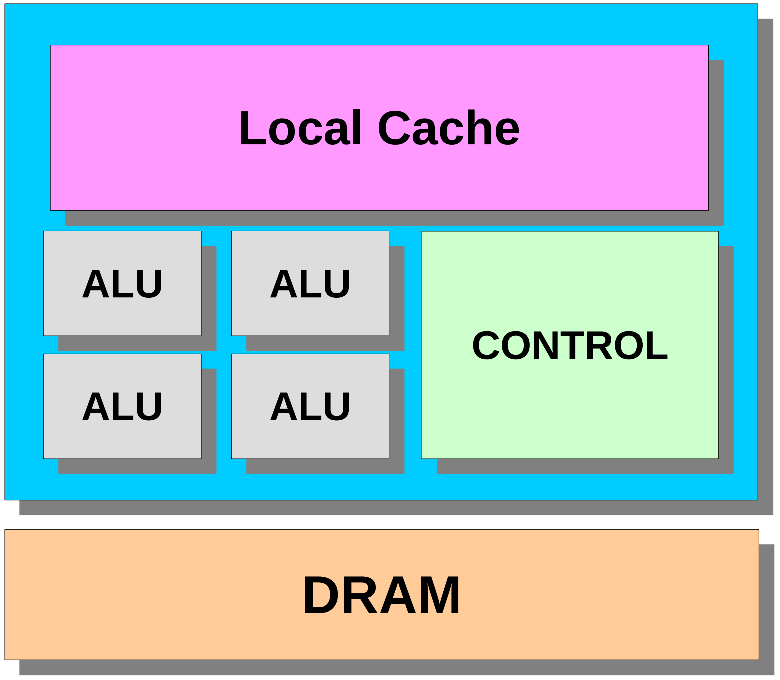
- Powerful ALU
  - Reduced operation latency

Latency Oriented Design – CPU



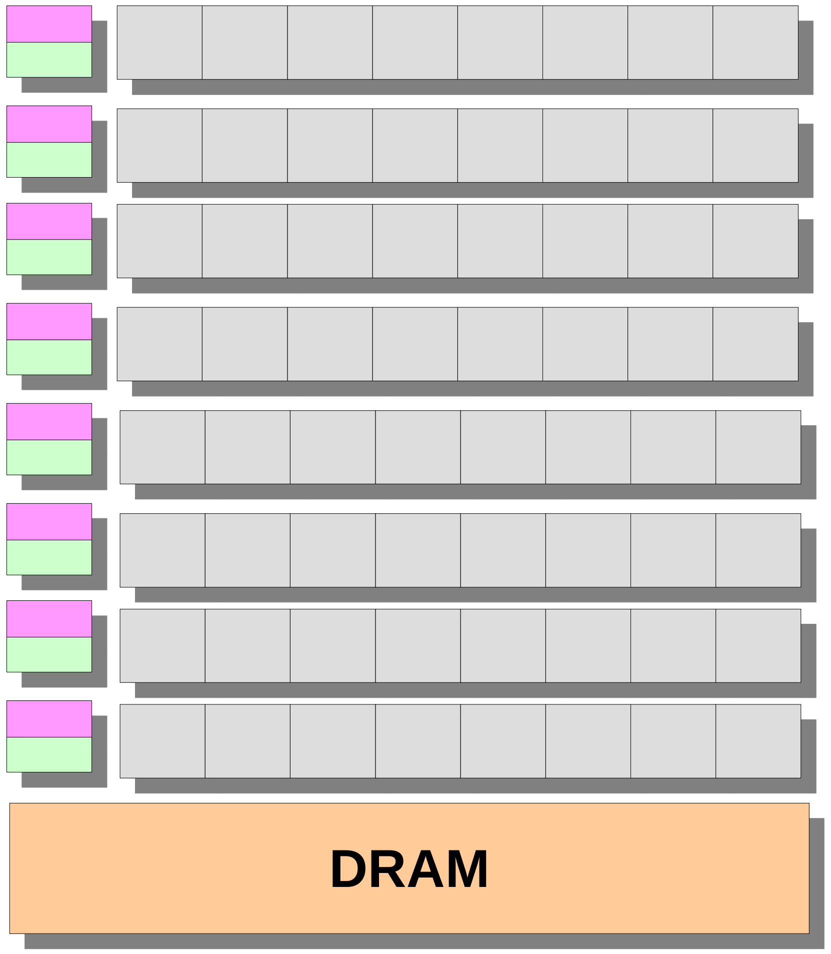
- Powerful ALU
  - Reduced operation latency
- Large caches
  - Convert long latency memory accesses to short latency cache accesses

Latency Oriented Design – CPU



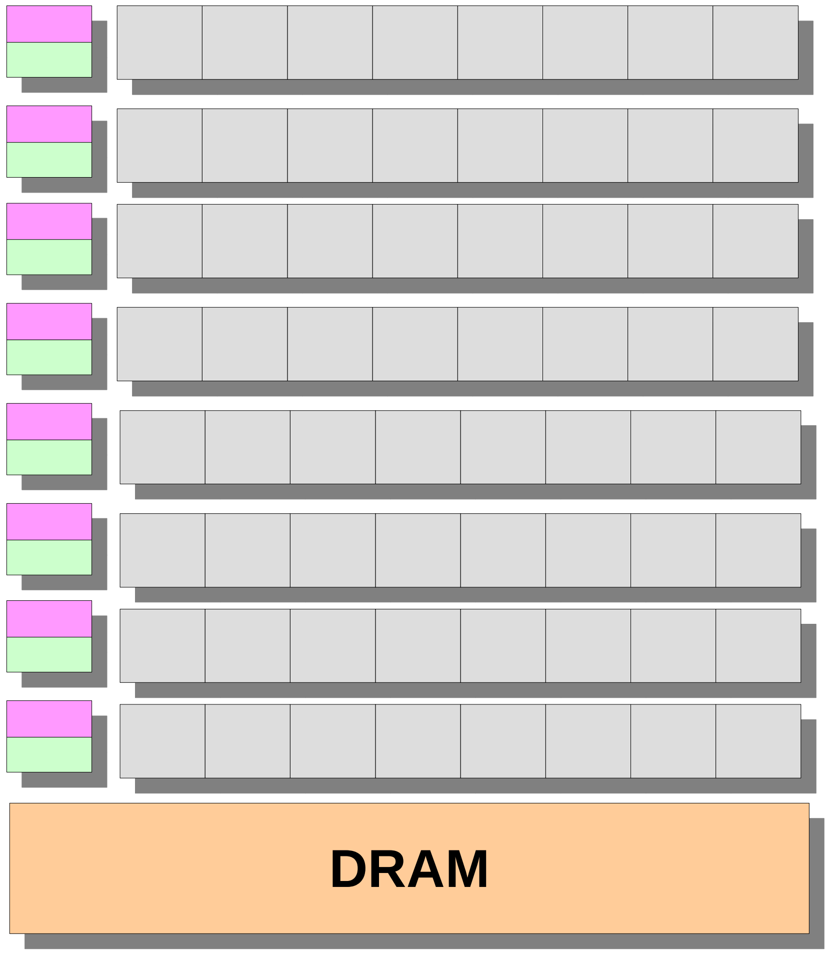
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency

Throughput Oriented Design – GPU



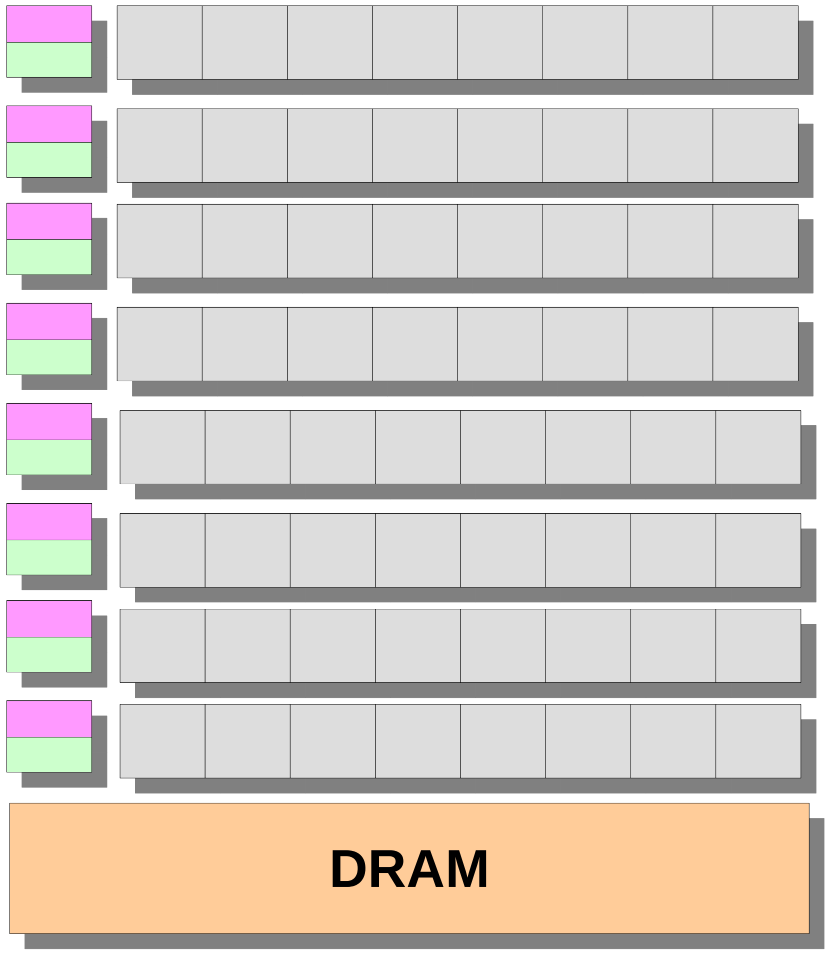
- Small caches
  - To boost memory throughput

Throughput Oriented Design – GPU



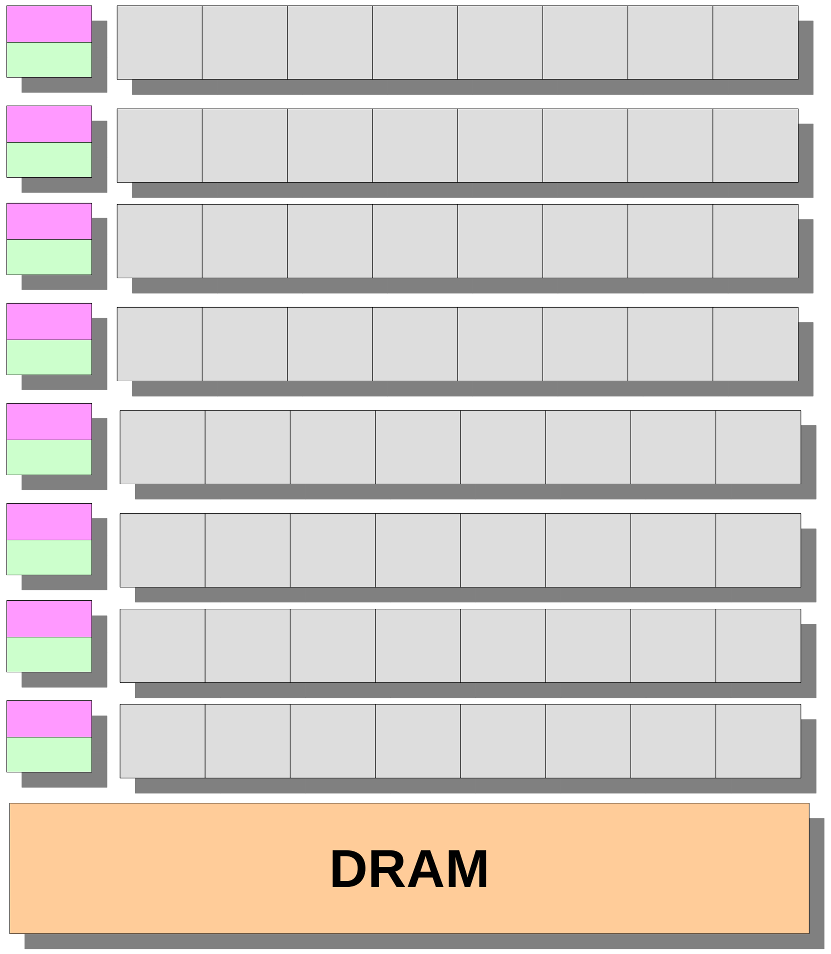
- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding

Throughput Oriented Design – GPU



- Energy efficient ALUs
  - Several
  - Long latency, heavily pipelined for high throughput

Throughput Oriented Design – GPU



- Energy efficient ALUs
  - Several
  - Long latency, heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies

CPU + GPU

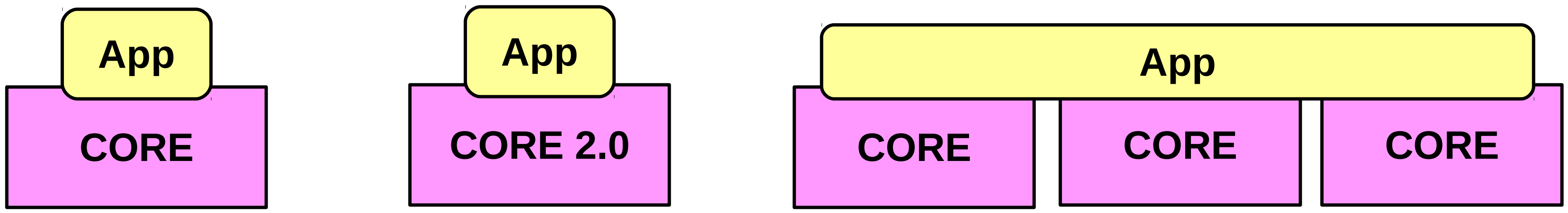
- CPUs for sequential parts where latency matters
  - CPUs can be 10x faster than GPUs for sequential code

CPU + GPU

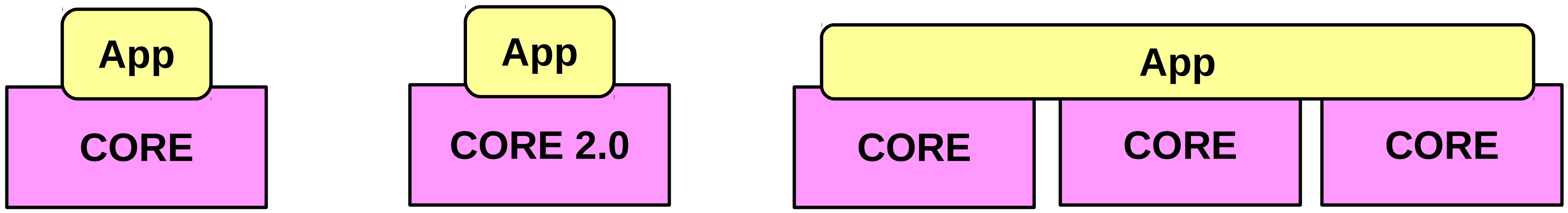
- CPUs for sequential parts where latency matters
  - CPUs can be 10x faster than GPUs for sequential code
- GPUs for parallel parts where throughput wins
  - GPUs can be 10x faster than CPUs for parallel code

J. Nickolls and W. J. Dally, "The GPU Computing Era," in IEEE Micro, vol. 30, no. 2, pp. 56-69, March-April 2010.

Scalability of Programs

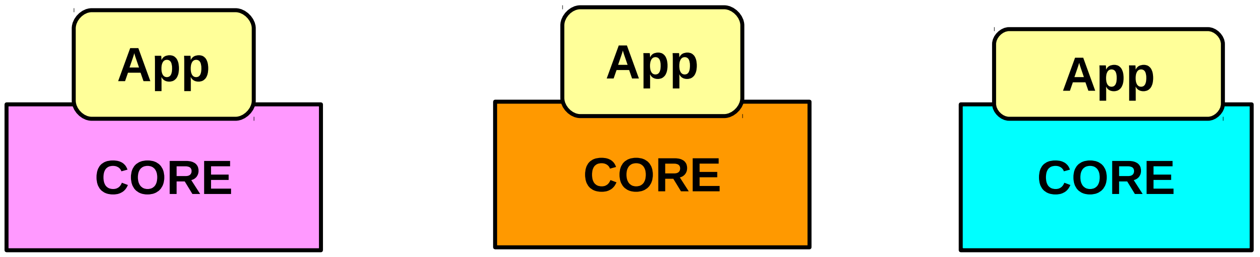


Scalability of Programs



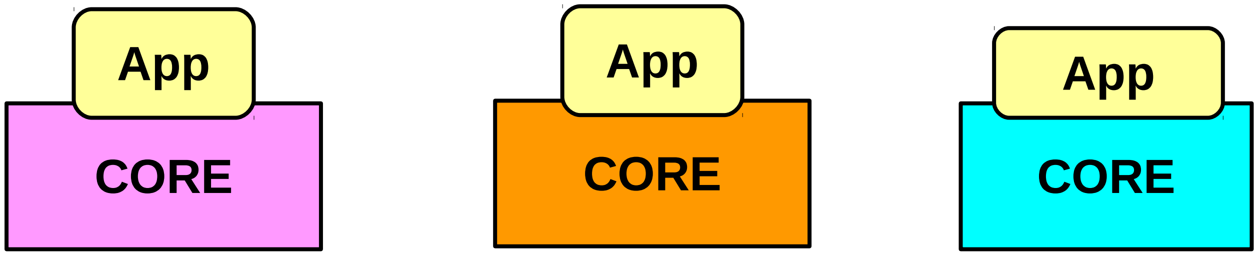
- An application <u>runs efficiently</u> on new generations of cores/on more of the same cores
  - Increasing number of compute units, number of threads, vector length, pipeline depth, DRAM burst size/DRAM channels, data movement latency

Portability of Programs



- An application runs efficiently on systems with different organizations and interfaces
  - x86 vs. ARM, etc.; Latency oriented CPUs vs. throughput oriented GPUs; VLIW vs. SIMD vs. Threading; Shared memory vs. Distributed memory

Portability of Programs



- An application runs efficiently on systems with different organizations and interfaces
  - x86 vs. ARM, etc.; Latency oriented CPUs vs. throughput oriented GPUs; VLIW vs. SIMD vs. Threading; Shared memory vs. Distributed memory
- OpenCL and Heterogeneous System Architecture help address portability

Outline

- SoC
- Latency oriented cores – CPU
- Throughput oriented cores – GPU
- Scalability and Portability

Reference

- D. Kirk and W. Hwu, “Programming Massively Parallel Processors – A Hands-on Approach,” 3e, MK.