Lecture 20:

Heterogeneous Parallelism and Hardware Specialization

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2025

Logistics

We've heard feedback on grading & communication

Please share your feedback!

Poll currently live:
https://docs.google.com/forms/d/e/1FAIpQLSf8QuPJL20ZX6EFCsxsaWmJ5UVBtXgNYDC8SNoKf3GWfqv7zQ/viewform?usp=dialog

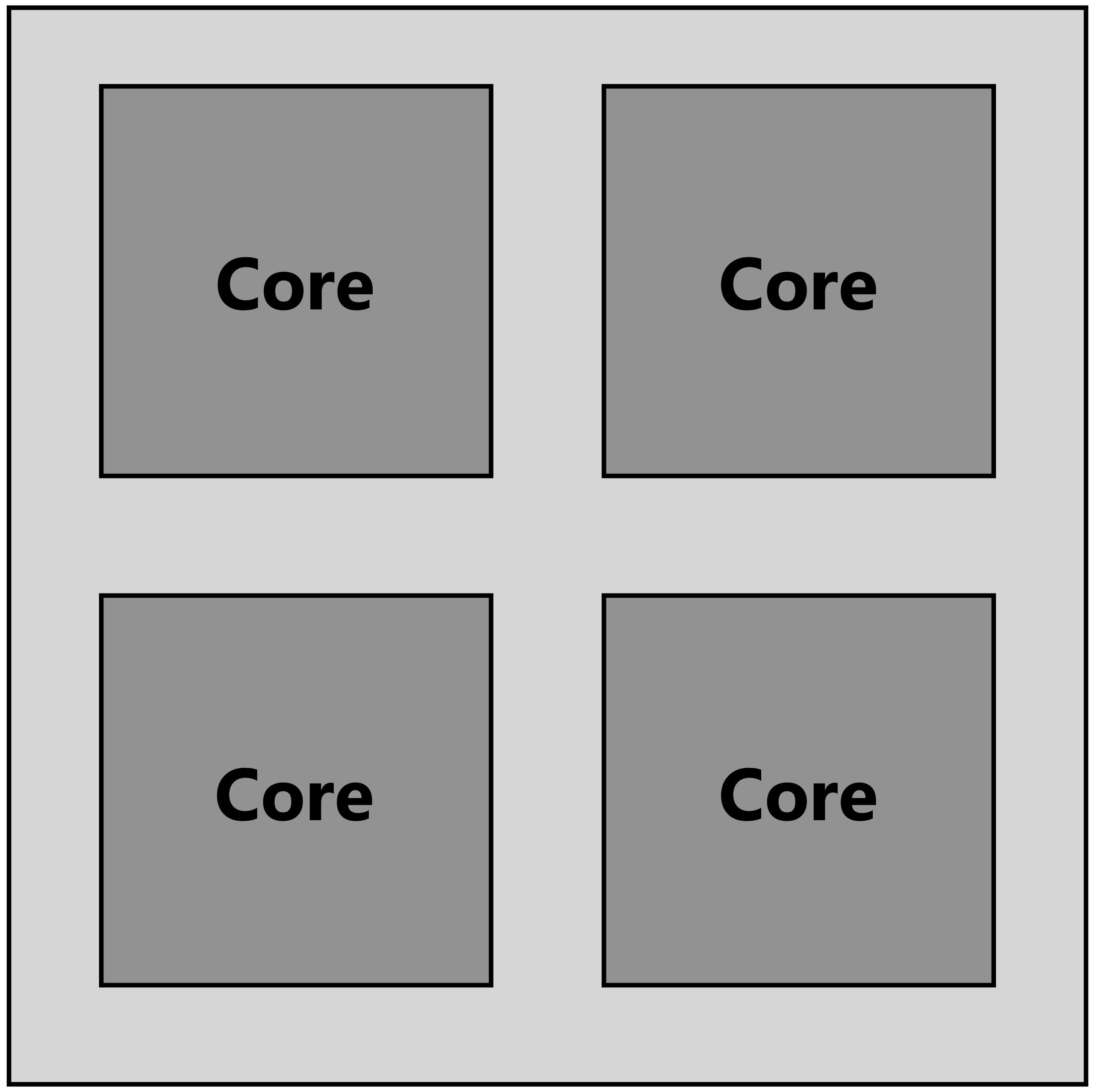
(see Piazza)

Let’s begin this lecture by reminding you…

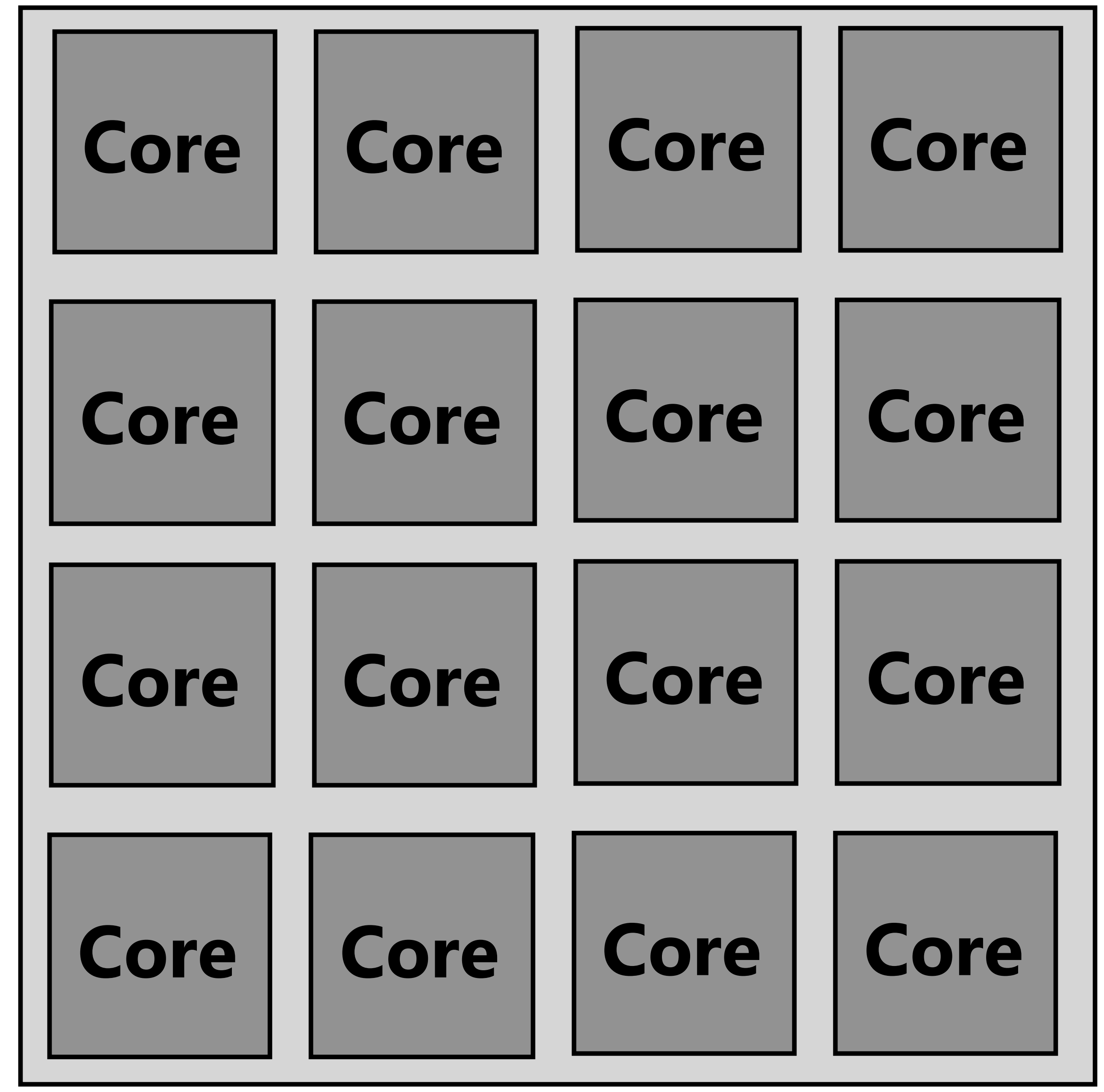
That we observed in assignment 1 that a well-optimized parallel implementation of a <u>compute-bound</u> application was about 44 times faster than single-threaded C code compiled with gcc -O3, running on the same processor


YINZER
PROCESSORS
You need to buy a new computer...

You need to buy a computer system



Processor A
4 cores
Each core has sequential performance P



Processor B
16 cores
Each core has sequential performance P/2

All other components of the system are equal.

Which do you pick?

Amdahl's law revisited

$$\text{speedup}(f, n) = \frac{1}{(1 - f) + \frac{f}{n}}$$

f = **fraction of program that is parallelizable**

n = **parallel processors**

Assumptions:
Parallelizable work distributes perfectly onto n **processors of equal capability**

Rewrite Amdahl's law in terms of resource limits

$$\text{speedup}(f, n, r) = \frac{1}{\frac{1-f}{\text{perf}(r)} + \frac{f}{\text{perf}(r)\cdot \frac{n}{r}}}$$

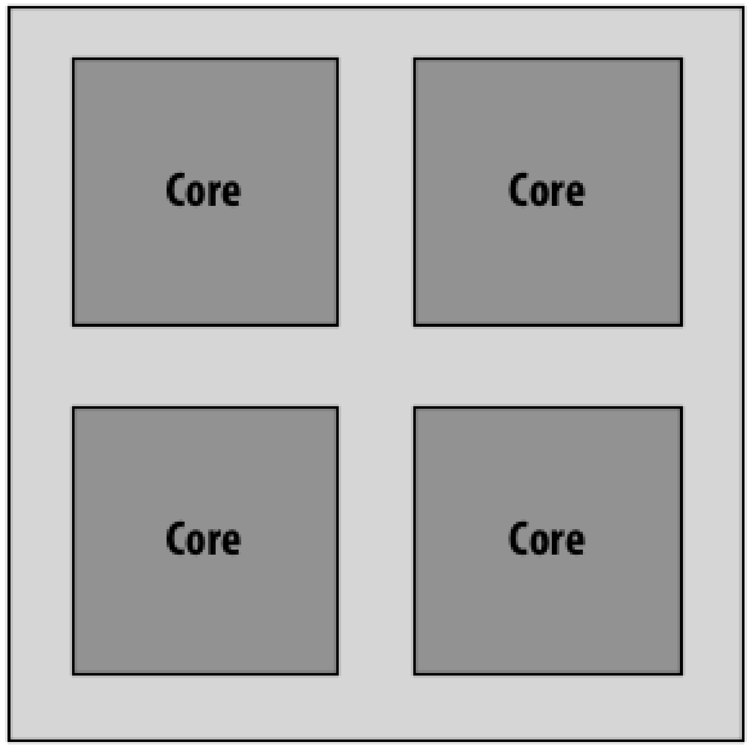
Relative to processor with 1 unit of resources, n=1. Assume perf(1) = 1

f = **fraction of program that is parallelizable**
n = **total processing resources (e.g., transistors on a chip)**
r = **resources dedicated to each processing core,**

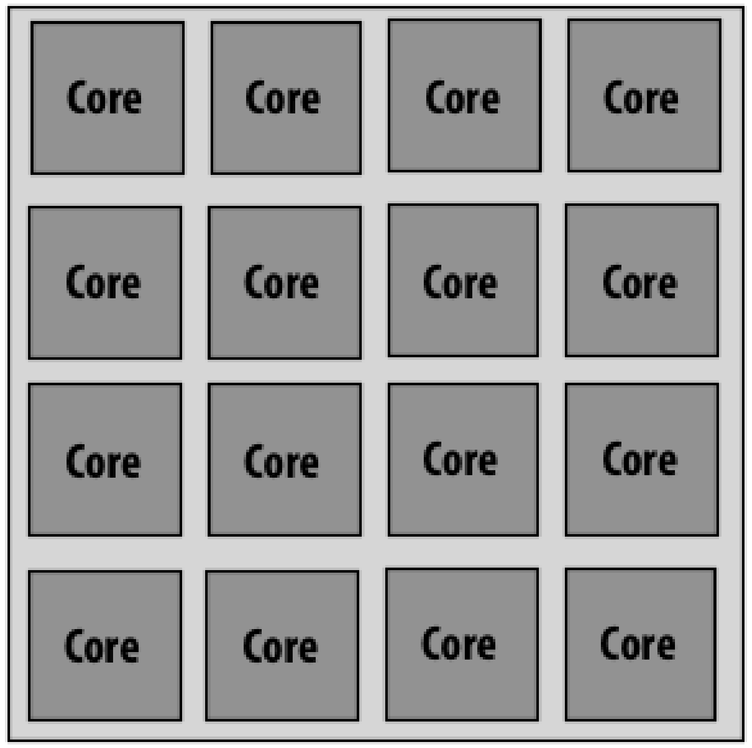
➔ **each of the** n/r **cores has sequential performance** $perf(r)$

More general form of Amdahl's Law in terms of f, n, r

Two examples where
n **= 16**
r_A **= 4**
r_B **= 1**



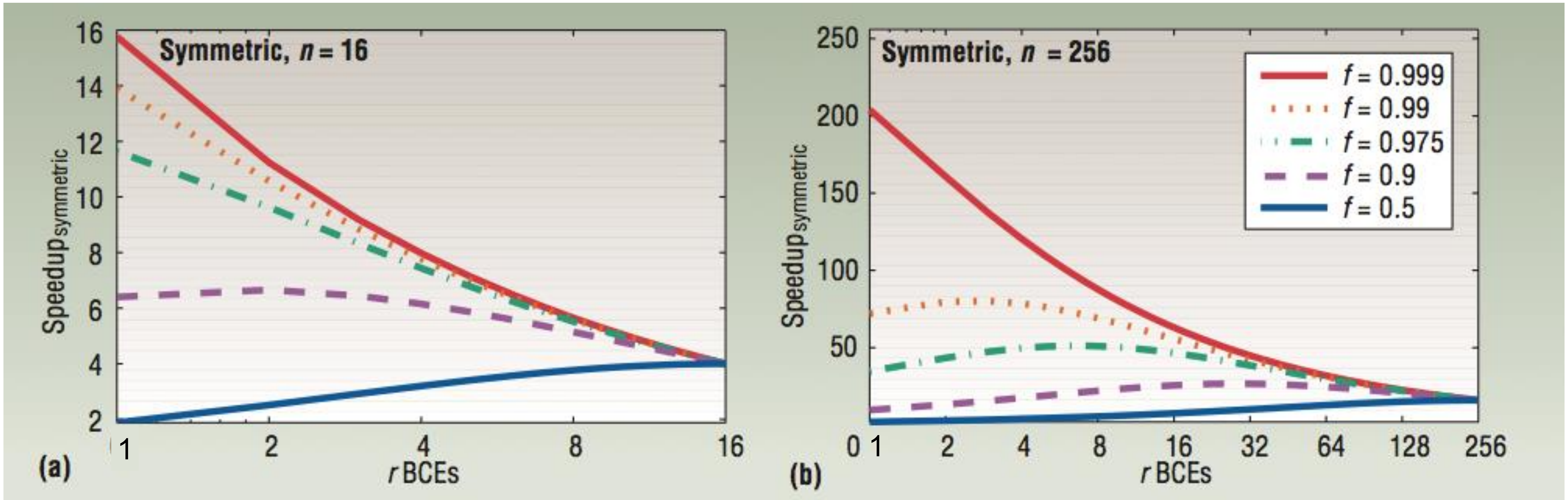
Processor A



Processor B

[Hill and Marty 08] ***Amdahl's Law in the Multicore Era***

Speedup (relative to n=1)



Up to 16 cores (n=16) **Up to 256 cores (n=256)**

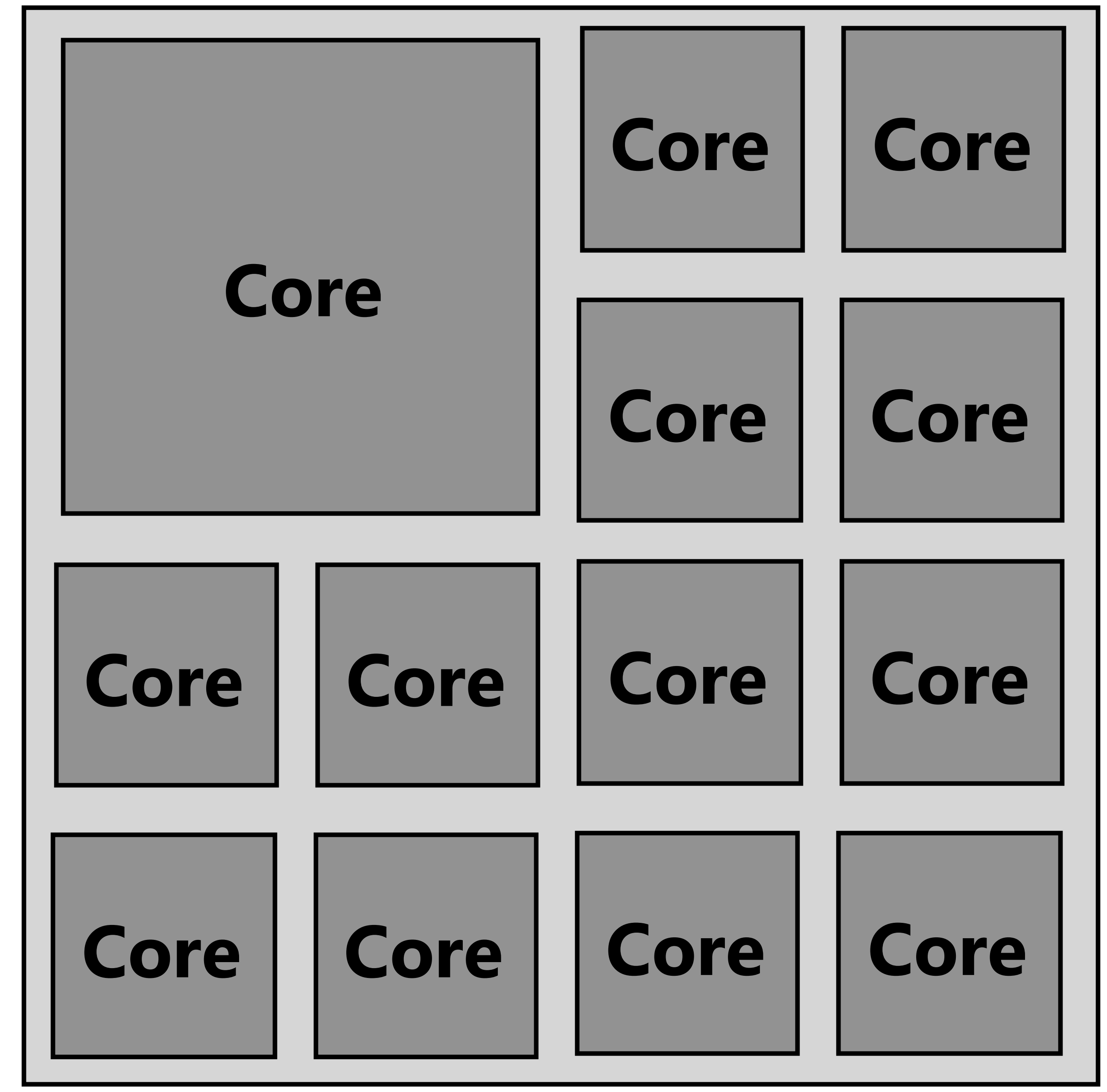
X-axis = r (chip with many small cores to left, fewer "fatter" cores to right)
Each line corresponds to a different workload
Each graph plots performance as resource allocation changes, but total chip resources resources kept the same (constant n per graph)

perf(r) **modeled as** $\sqrt{r}$

[Figure credit: Hill and Marty 08]

Asymmetric set of processing cores

Example: n=16
One core: $r = 4$
Other 12 cores: $r = 1$



$$\text{speedup}(f,n,r) = \frac{1}{\frac{1-f}{\text{perf}(r)} + \underbrace{\frac{f}{\text{perf}(r)+(n-r)}}}$$

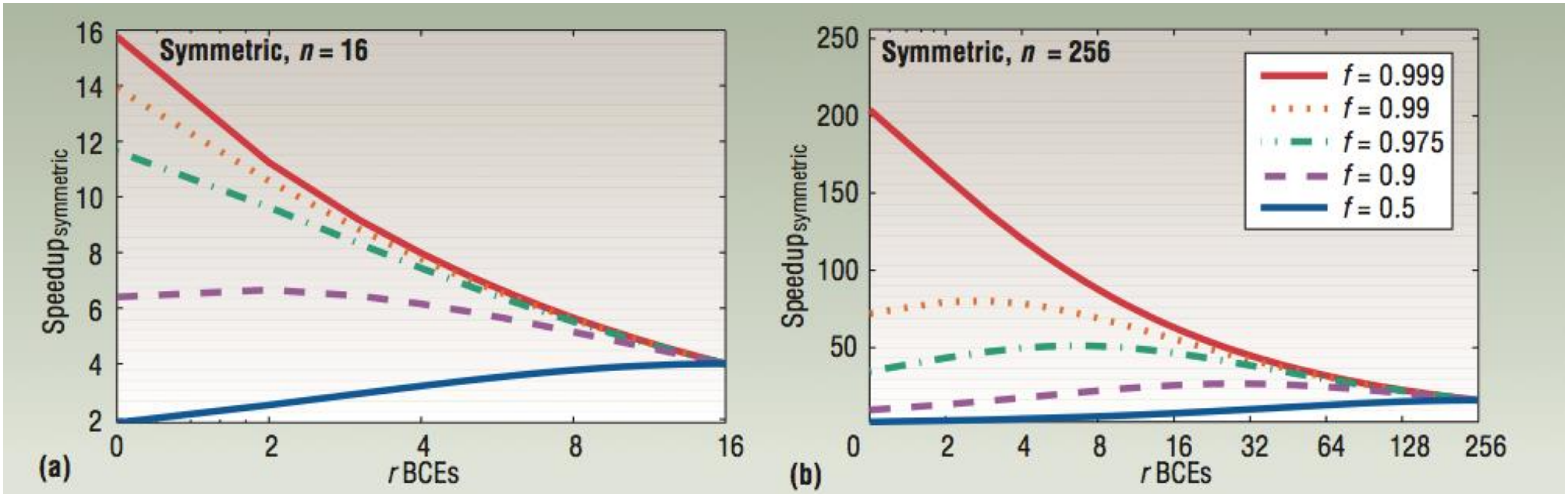
(of heterogeneous processor with *n* resources, relative to uniprocessor with one unit worth of resources, n=1)

one perf(r) processor + (n-r) perf(1)=1 processors

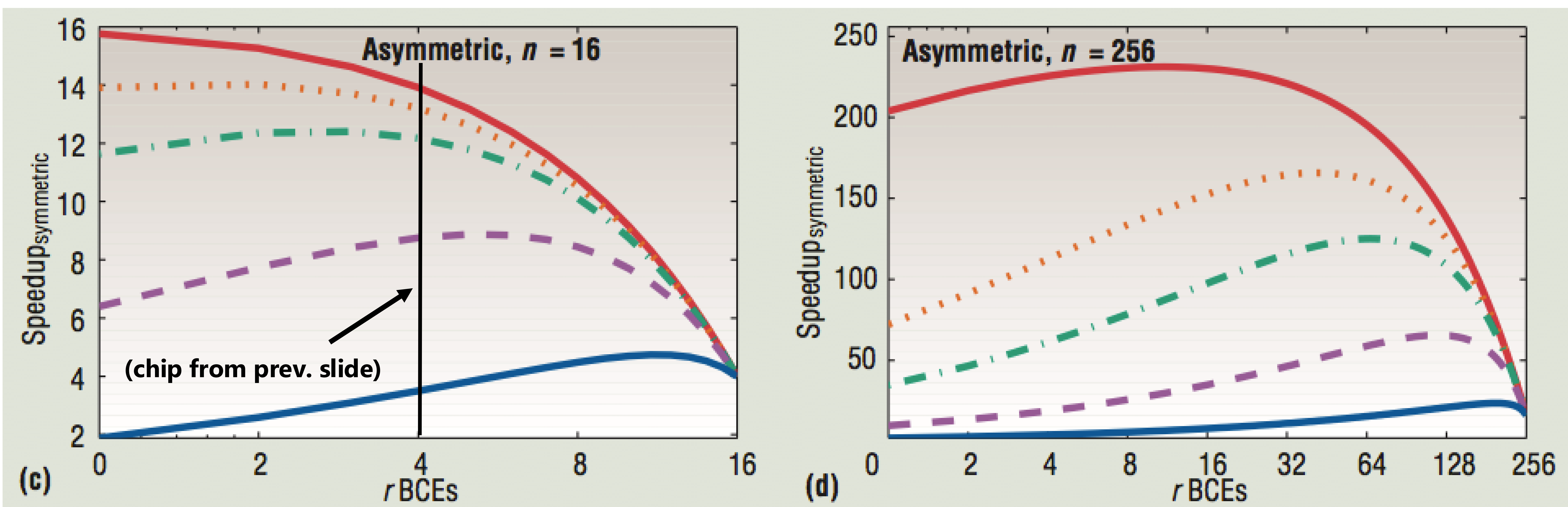
[Hill and Marty 08]

Speedup (relative to n=1)

[Source: Hill and Marty 08]



X-axis for symmetric architectures gives *r* for all cores (many small cores to left, few “fat” cores to right)



X-axis for asymmetric architectures gives *r* for the single “fat” core (assume rest of cores are *r* = 1)

Heterogeneous processing

Observation: most "real world" applications have complex workload characteristics *

They have components that can be widely parallelized.	And components that are difficult to parallelize.
They have components that are amenable to wide SIMD execution.	And components that are not. (divergent control flow)
They have components with predictable data access	And components with unpredictable access (but those accesses might cache well).

Idea: the most efficient processor is a heterogeneous mixture of resources ("use the most efficient tool for the job")

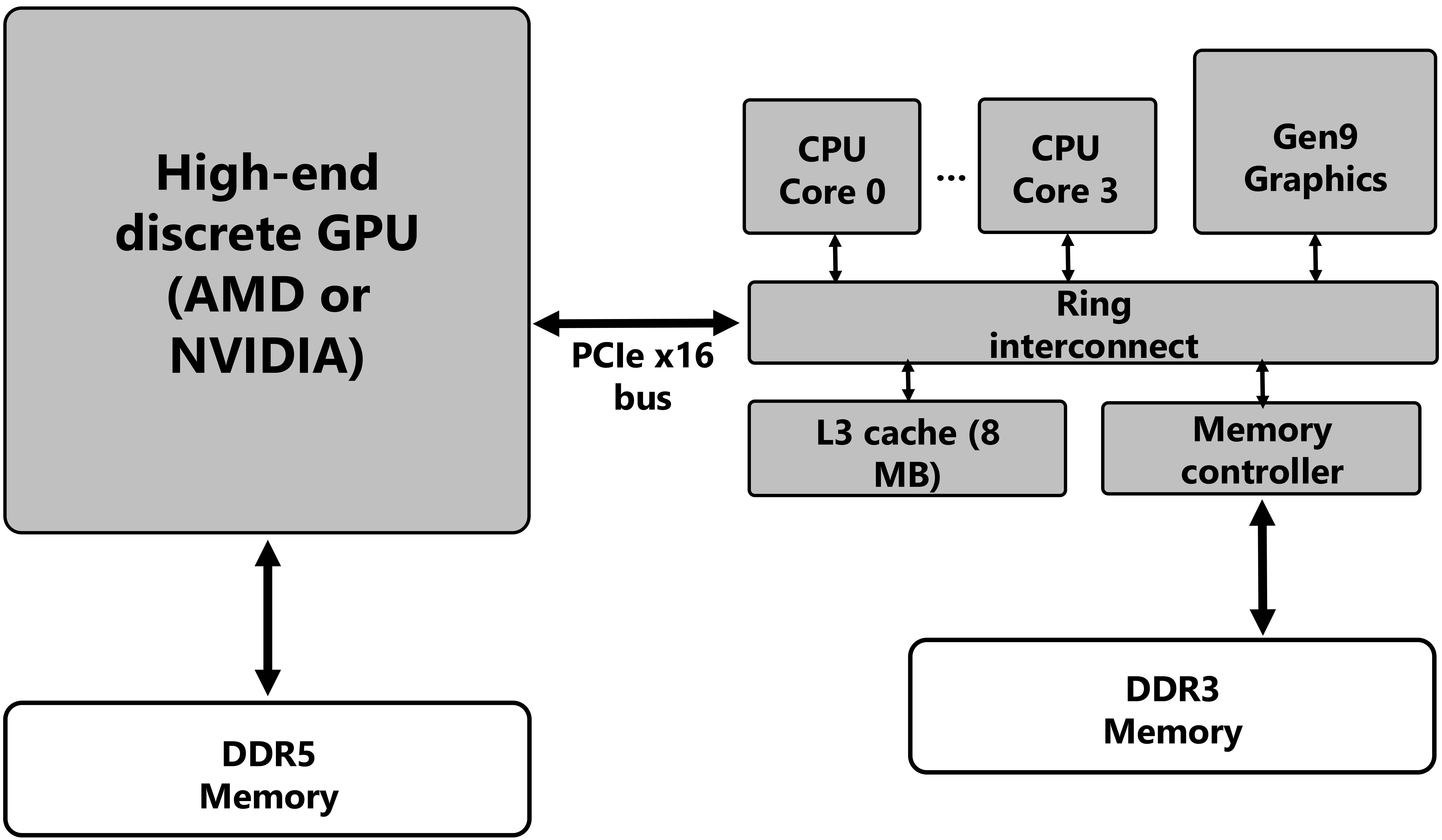
* You will likely make a similar observation during your projects

Intel Alder Lake Core i9 (2022)

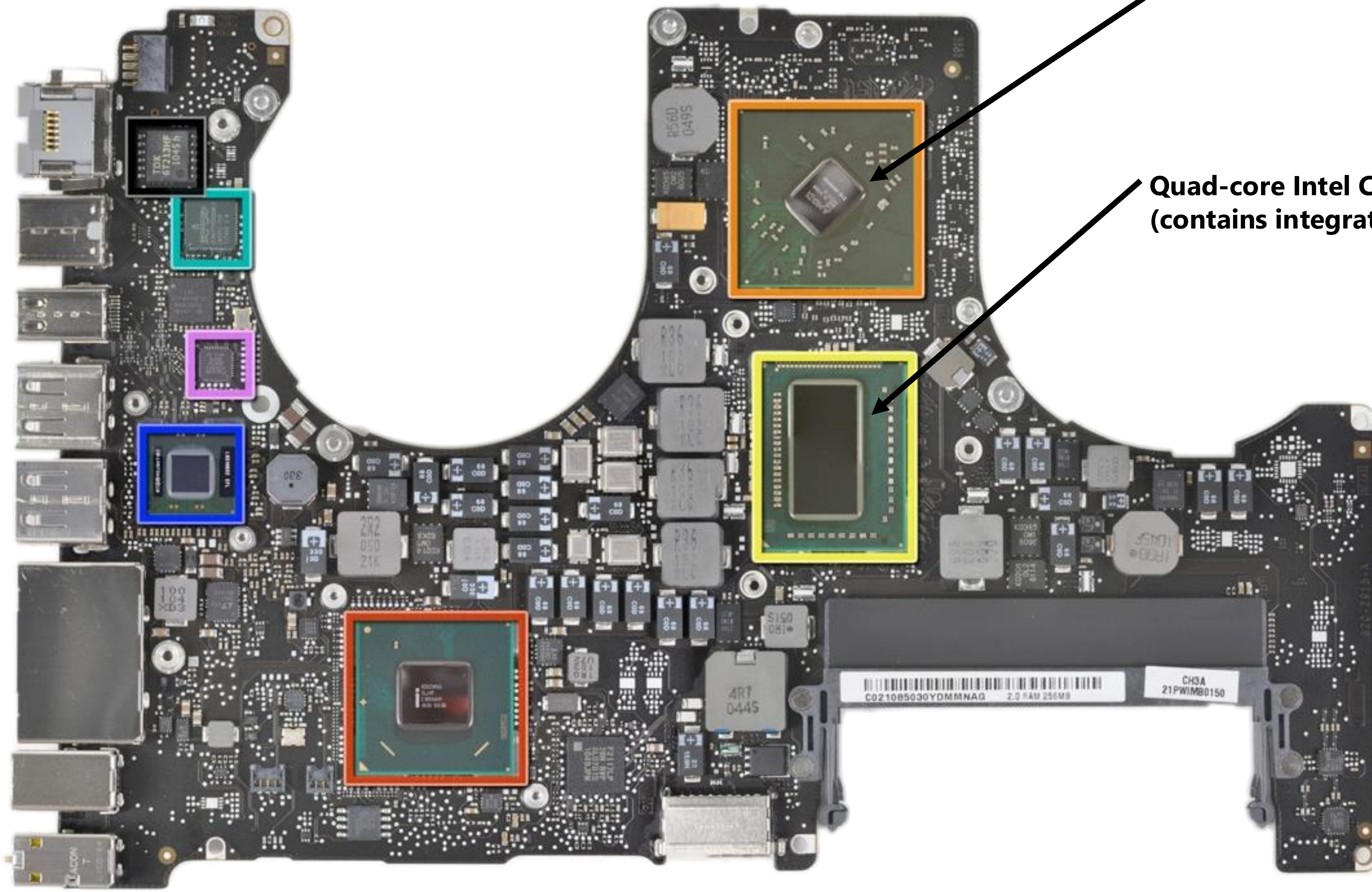
8 fast CPU cores + 8 efficient CPU cores + GPU integrated on one chip

More heterogeneity: add discrete GPU

Keep discrete (power hungry) GPU turned off unless needed for graphics-intensive applications
Use integrated, low power graphics for basic graphics/window manager/UI

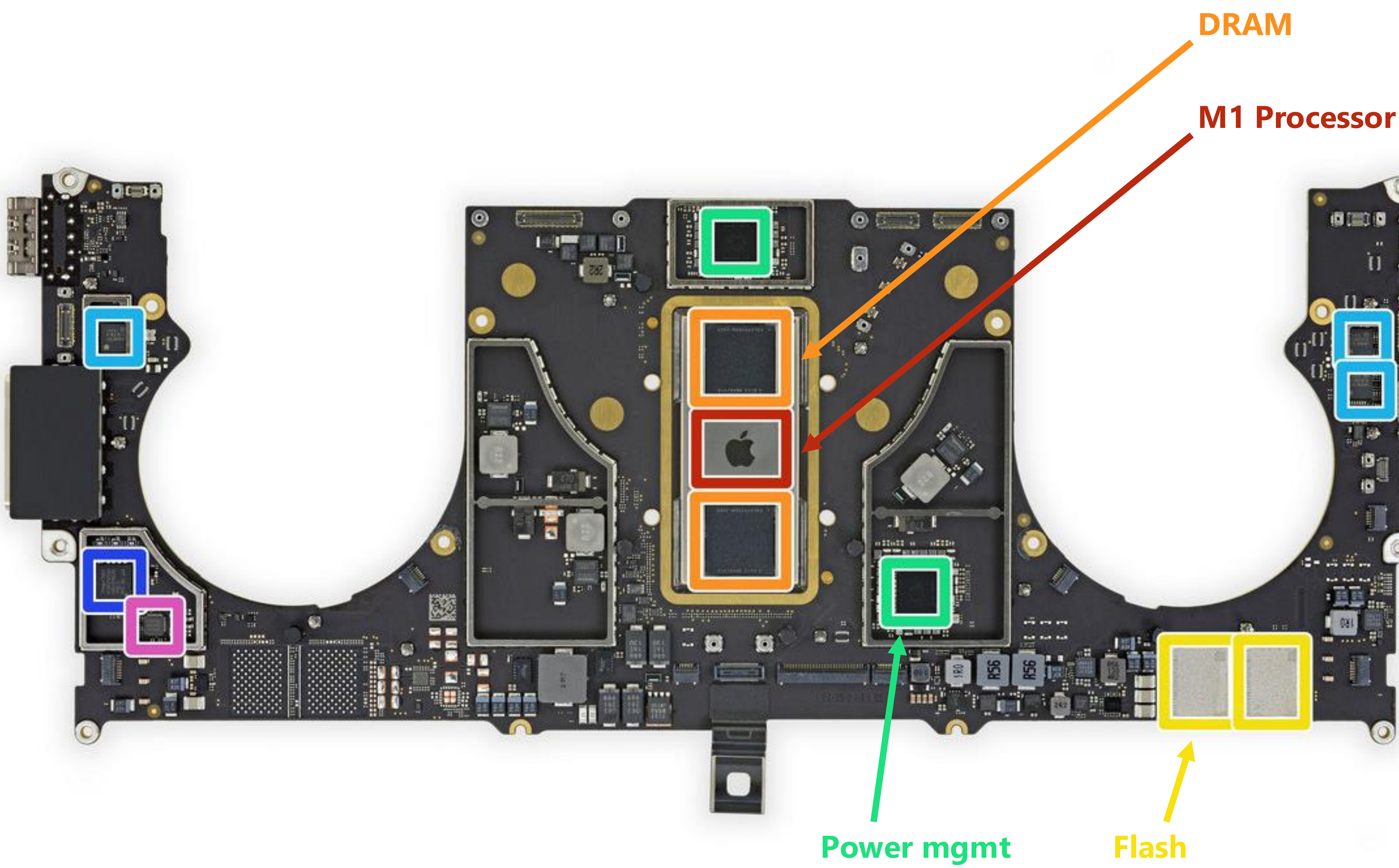


15in Macbook Pro 2011 (two GPUs)



From ifixit.com teardown

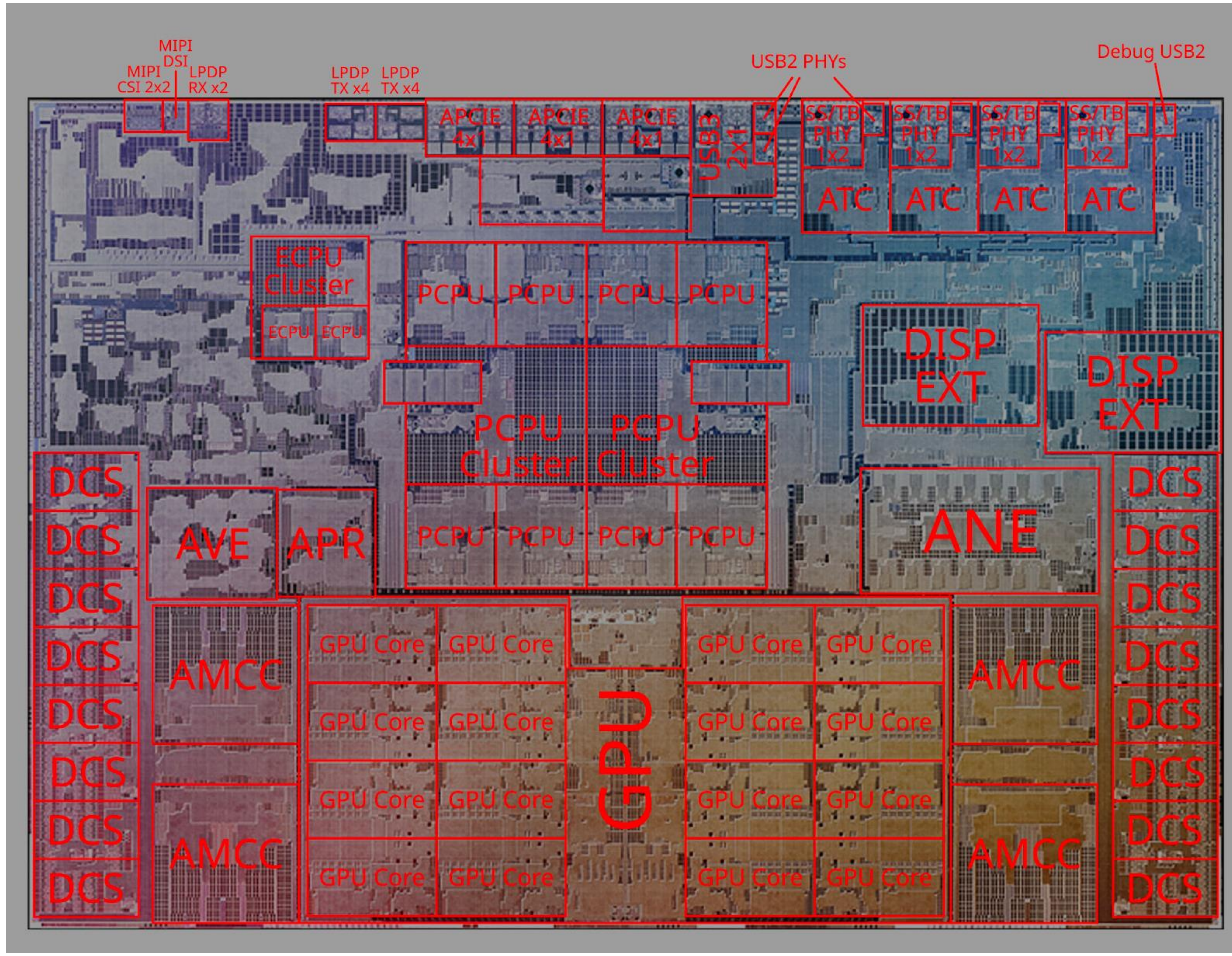
M1 Macbook Pro 2021



From ifixit.com teardown

Apple M1 processor

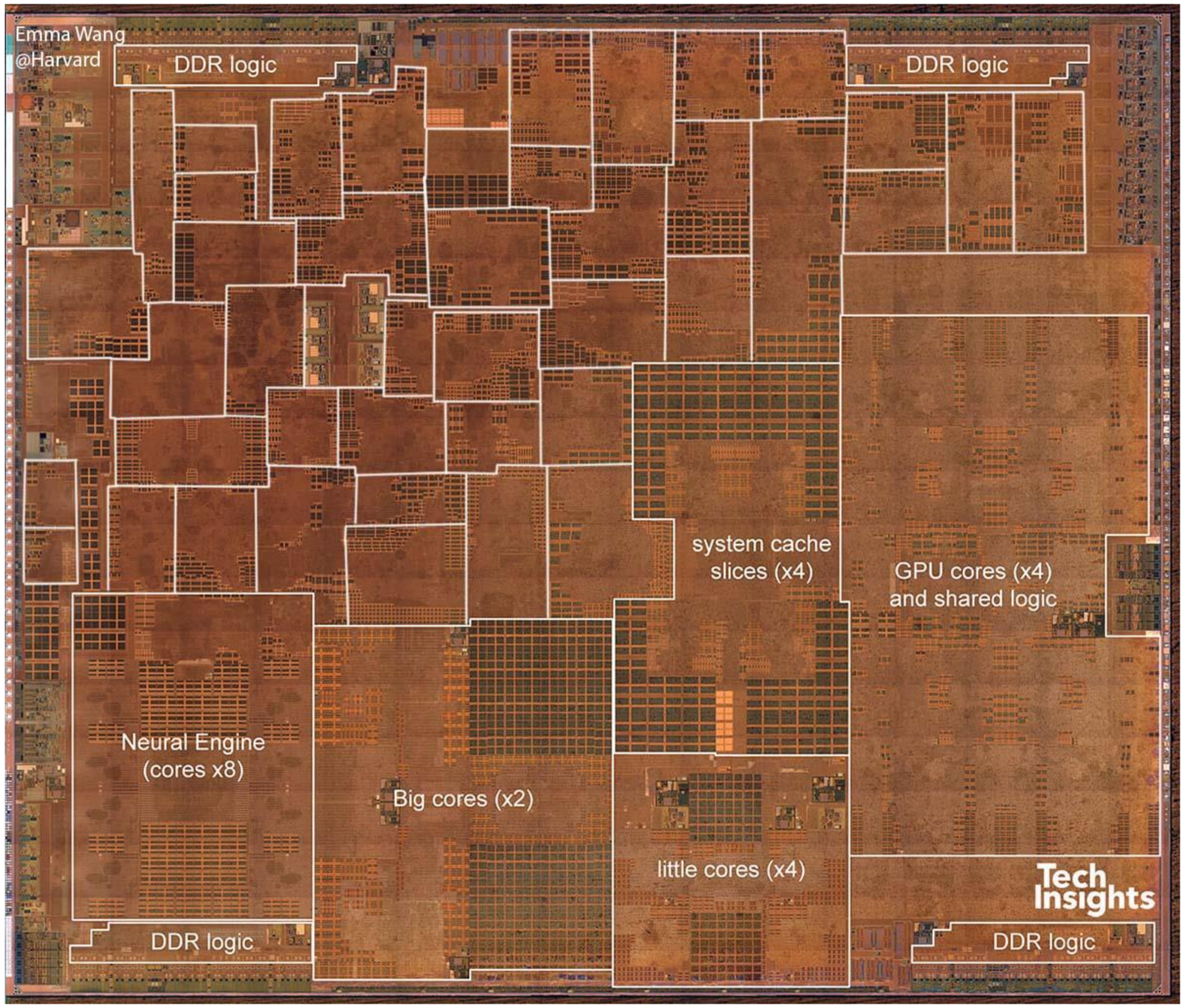
- Big CPUs
- Little CPUs
- GPU
- Neural Engine
- + much more



Credit: Hector Martin

https://twitter.com/marcan42/

Smartphone Processor



- Neural Engine
  - Fixed sequence of arithmetic operations
  - 8-bit FP
  - 8-wide parallelism
  - 5×10^{12} ops/second

- Apple A12, 2018
  - 6.9 billion transistors
- Processors
  - 2 high-power CPUs
    - 7-wide issue
  - 4 low-power CPUs
    - 3-wide issue
  - 4-core GPU
  - Neural engine
    - for deep neural network evaluation
- Specialized hardware
  - Video encode/decode
  - GPS
  - Encryption/Decryption
  - …

Supercomputers use heterogeneous processing

Los Alamos National Laboratory: Roadrunner

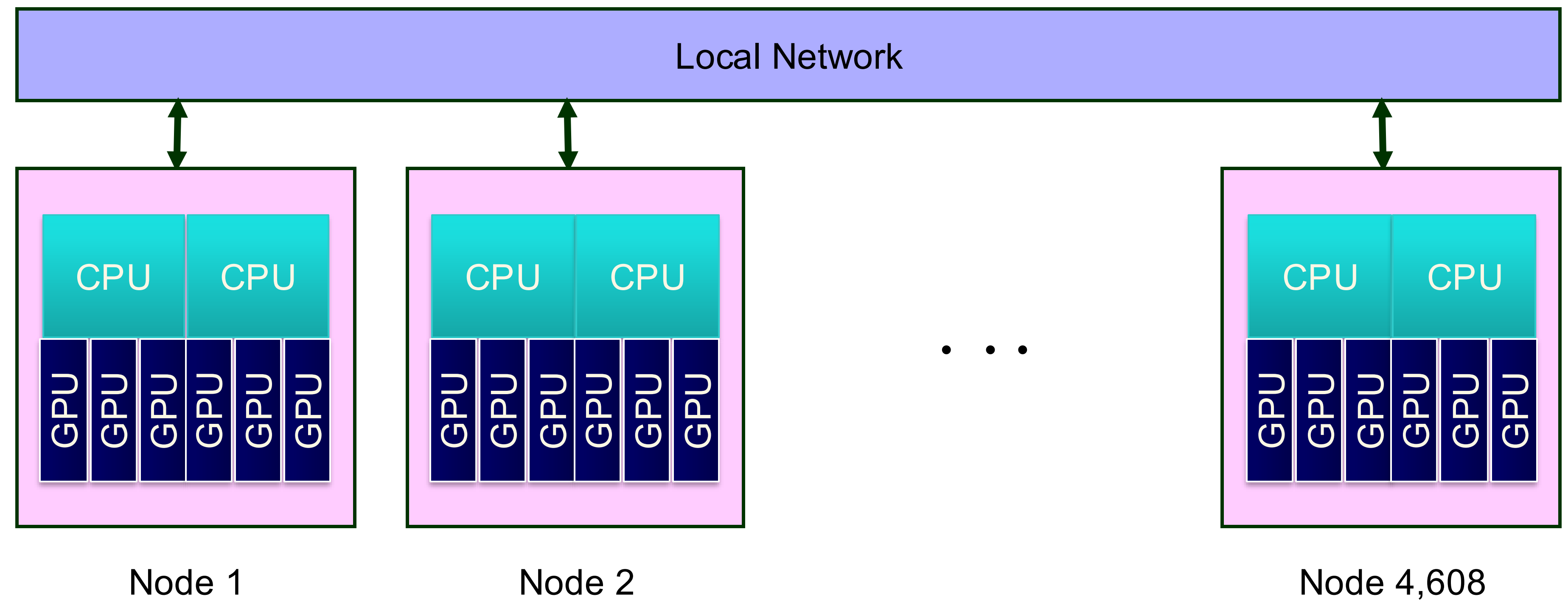
Fastest US supercomputer in 2008, first to break Petaflop barrier: 1.7 PFLOPS

Unique at the time due to use of two types of processing elements

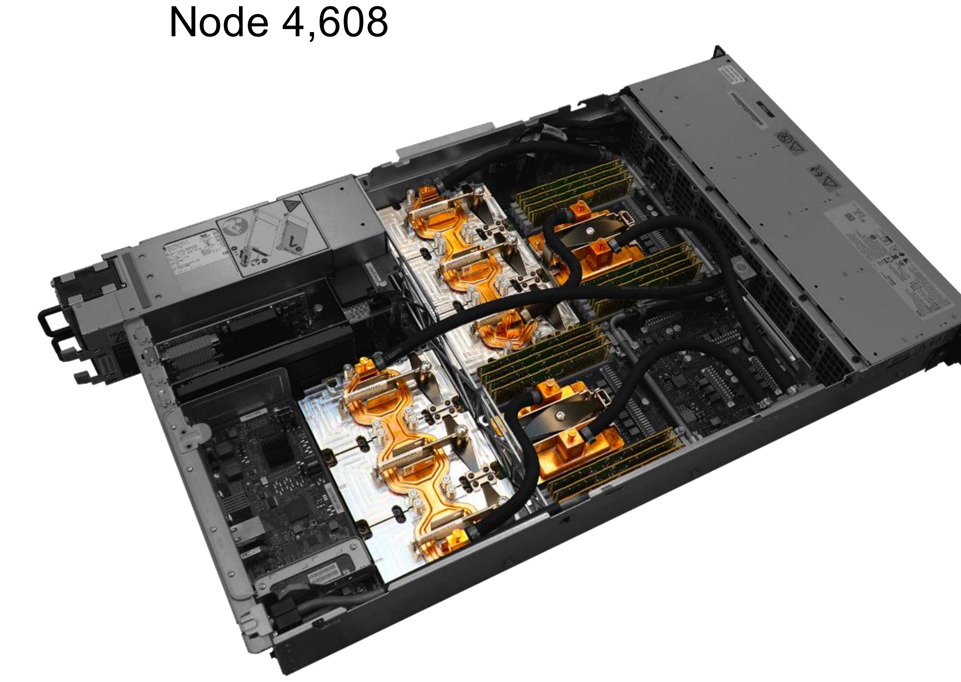
(IBM's Cell processor served as "accelerator" to achieve desired compute density)

- **6,480 AMD Opteron dual-core CPUs (12,960 cores)**
- **12,970 IBM Cell Processors (1 CPU + 8 accelerator cores per Cell = 116,640 cores)**
- **2.4 MWatt (about 2,400 average US homes)**

GPU-accelerated supercomputing



- Oak Ridge Summit
  - World's #2 powerful computer
- Each Node
  - 2 IBM 22-core POWER9 processors
  - 6 nVidia Graphics Processing Units
  - 608 GB DRAM
  - 1600 GB Flash
- Overall
  - 10MW water cooled
  - $325 M for two machi

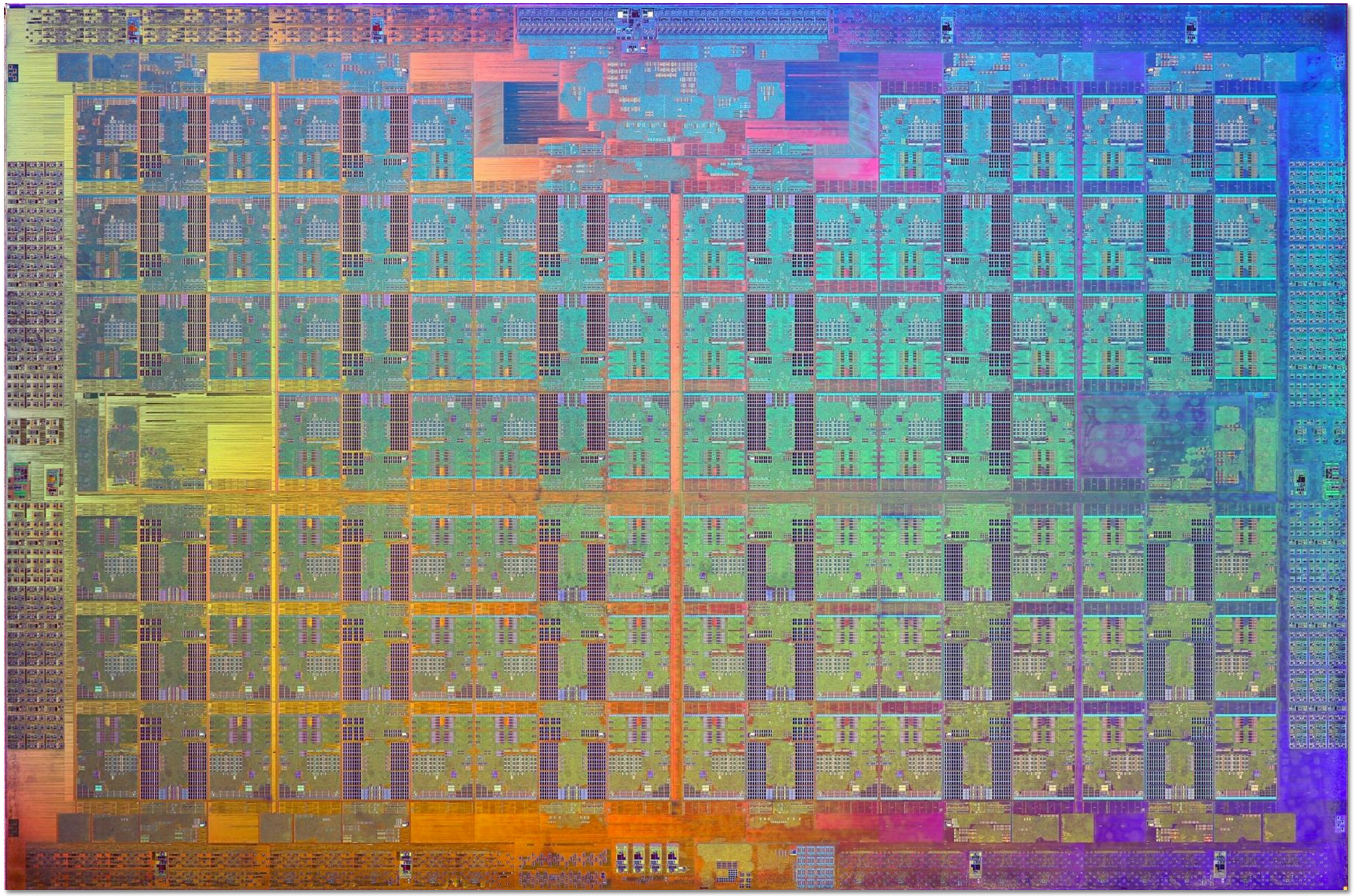


Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096

* Source: NPR

Intel Xeon Phi (Knights Landing)

- 72 “simple” x86 cores (1.1 Ghz, derived from Intel Atom)
- 16-wide vector instructions (AVX-512), four threads per core
- Targeted as an accelerator for supercomputing applications



Heterogeneous architectures for supercomputing

Source: Top500.org Fall 2021 rankings

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70,870.0	93,750.0	2,589
6	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646
7	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
8	**JUWELS Booster Module** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	70,980.0	1,764

A64FX – 48-core ARM w 'scalable vectors'

NVIDIA GPU

NVIDIA GPU

Sunway – 256-core manycore

NVIDIA GPU

NVIDIA GPU

Intel Xeon Phi

NVIDIA GPU

Green500: most energy efficient supercomputers

Efficiency metric: MFLOPS per Watt

Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)
1	301	**MN-3** - MN-Core Server, Xeon Platinum 8260M 24C 2.4GHz, Preferred Networks MN-Core, MN-Core DirectConnect, Preferred Networks Preferred Networks Japan	1,664	2,181.2	55	39.379
2	291	**SSC-21 Scalable Module** - Apollo 6500 Gen10 plus, AMD EPYC 7543 32C 2.8GHz, NVIDIA A100 80GB, Infiniband HDR200, HPE Samsung Electronics South Korea	16,704	2,274.1	103	33.983
3	295	**Tethys** - NVIDIA DGX A100 Liquid Cooled Prototype, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100 80GB, Infiniband HDR, Nvidia NVIDIA Corporation United States	19,840	2,255.0	72	31.538
4	280	**Wilkes-3** - PowerEdge XE8545, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 80GB, Infiniband HDR200 dual rail, DELL EMC University of Cambridge United Kingdom	26,880	2,287.0	74	30.797
5	30	**HiPerGator AI** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Infiniband HDR, Nvidia University of Florida United States	138,880	17,200.0	583	29.521
6	403	**Snellius Phase 1 GPU** - ThinkSystem SD650-N V2, Xeon Platinum 8360Y 36C 2.4GHz, NVIDIA A100 SXM4 40 GB, Infiniband HDR, Lenovo SURF Netherlands	6,480	1,818.0	63	29.046
7	5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70,870.0	2,589	27.374

DNN training accelerator

NVIDIA GPU

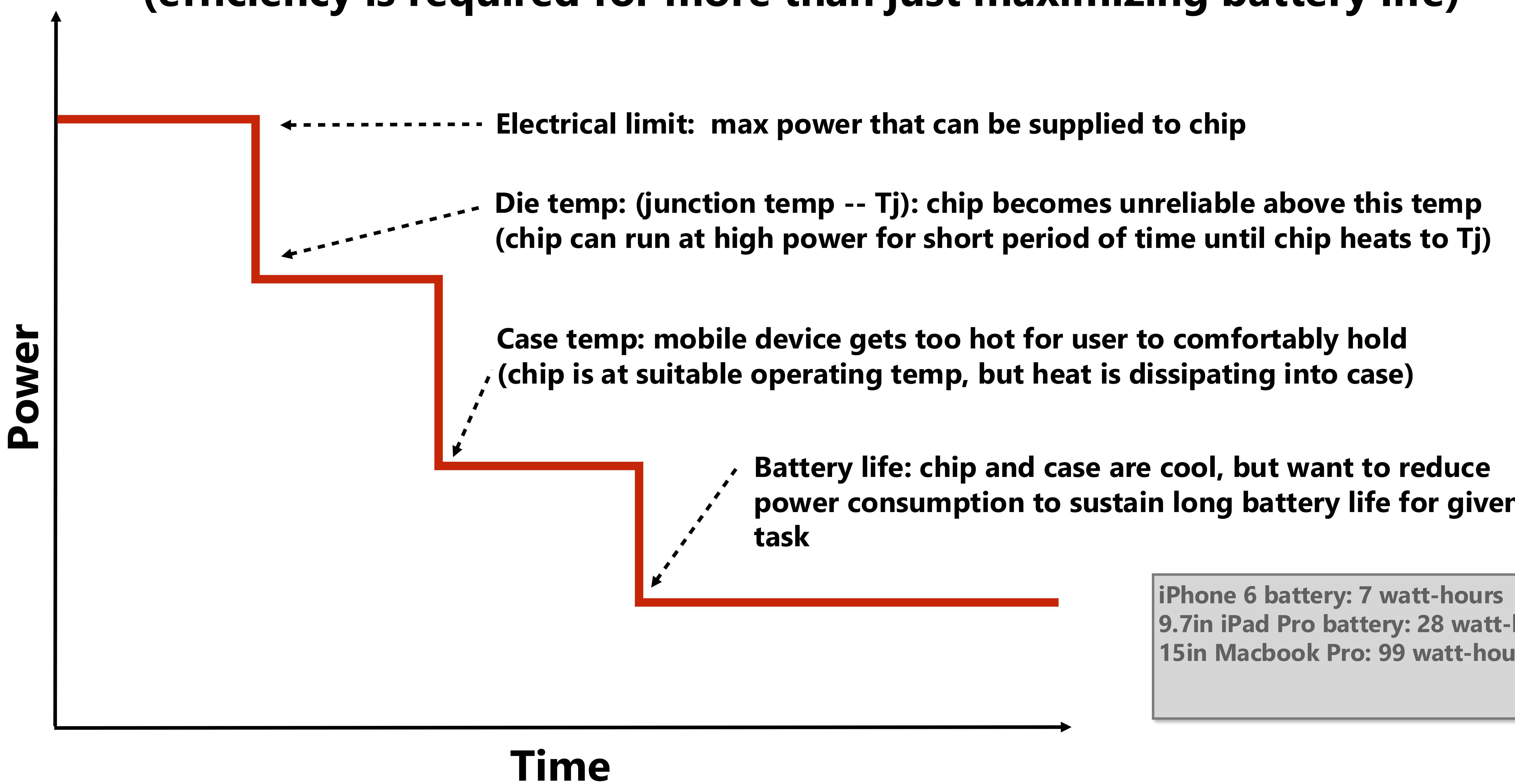
Source: Green500
Fall 2021 rankings

Energy-constrained computing

- **Supercomputers are energy constrained**
  - **Due to shear scale**
  - **Overall cost to operate (power for machine and for cooling)**
- **Datacenters are energy constrained**
  - **Reduce cost of cooling**
  - **Reduce physical space requirements**
- **Mobile devices are energy constrained**
  - **Limited battery life**
  - **Heat dissipation**

Limits on chip power consumption

- **General mobile processing rule: the longer a task runs the less power it can use**
  - **Processor's power consumption is limited by heat generated (efficiency is required for more than just maximizing battery life)**



iPhone 6 battery: 7 watt-hours
9.7in iPad Pro battery: 28 watt-hours
15in Macbook Pro: 99 watt-hours

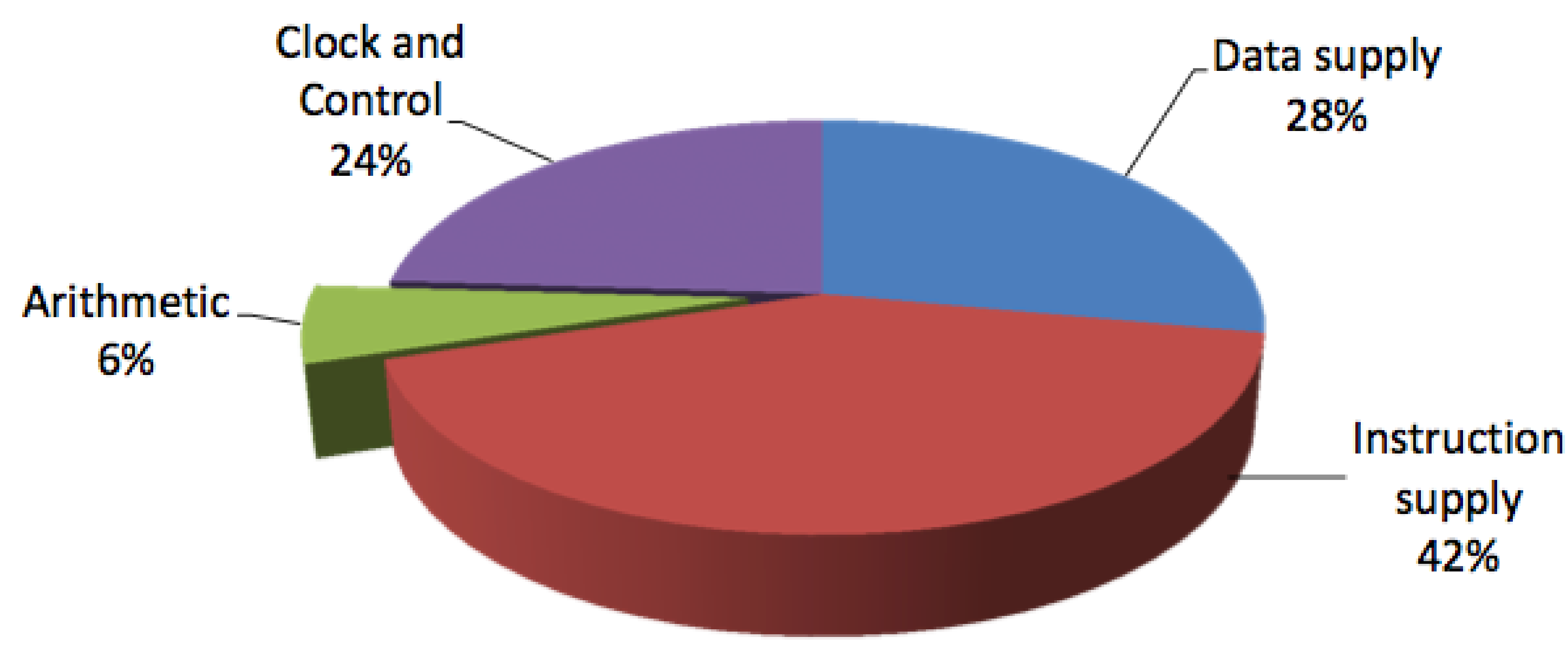
Slide credit: adopted from original slide from M. Shebanow: HPG 2013 keynote

Efficiency benefits of compute specialization

- **Rules of thumb: compared to high-quality C code on CPU...**

- **Throughput-maximized processor architectures: e.g., GPU cores**
  - **Approximately 10x improvement in perf / watt**
  - **Assuming code maps well to wide data-parallel execution and is compute bound**

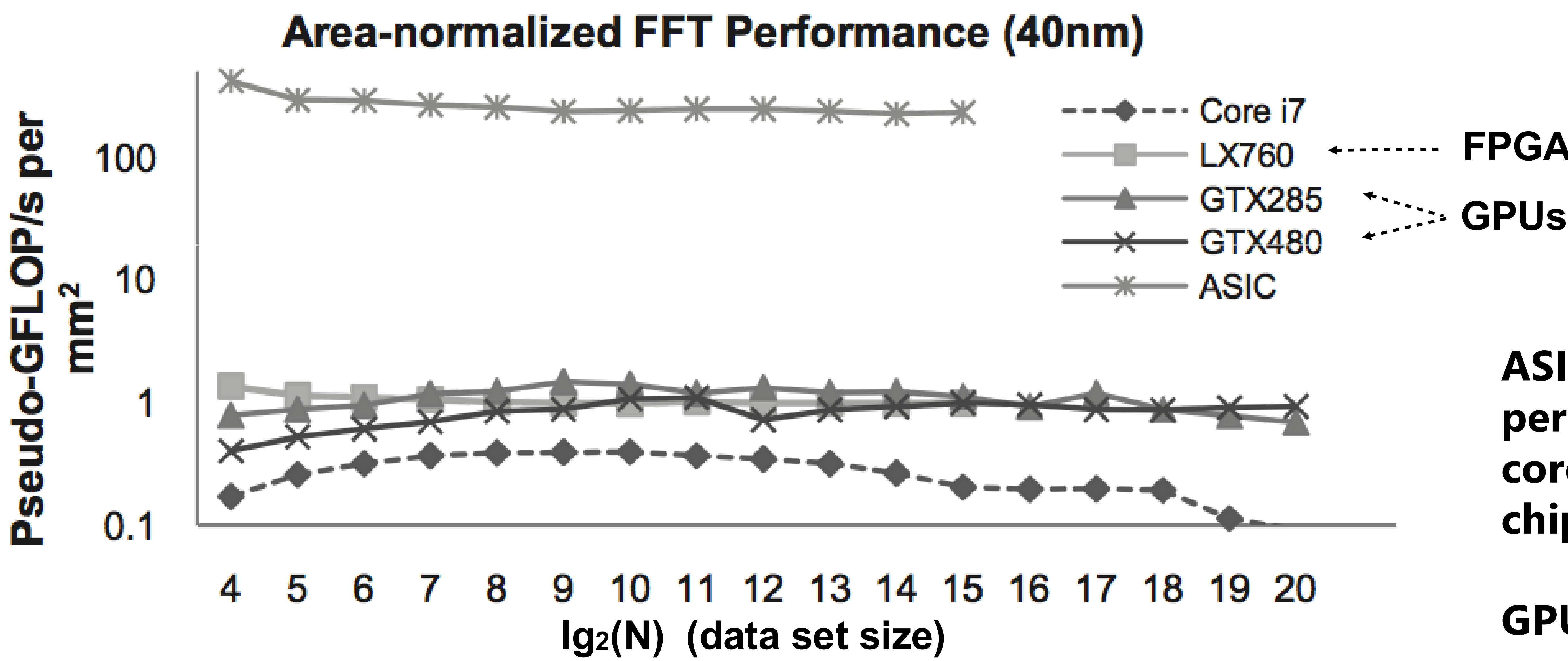
- **Fixed-function ASIC ("application-specific integrated circuit")**
  - **Can approach 100-1000x or greater improvement in perf/watt**



Efficient Embedded Computing [Dally et al. 08]

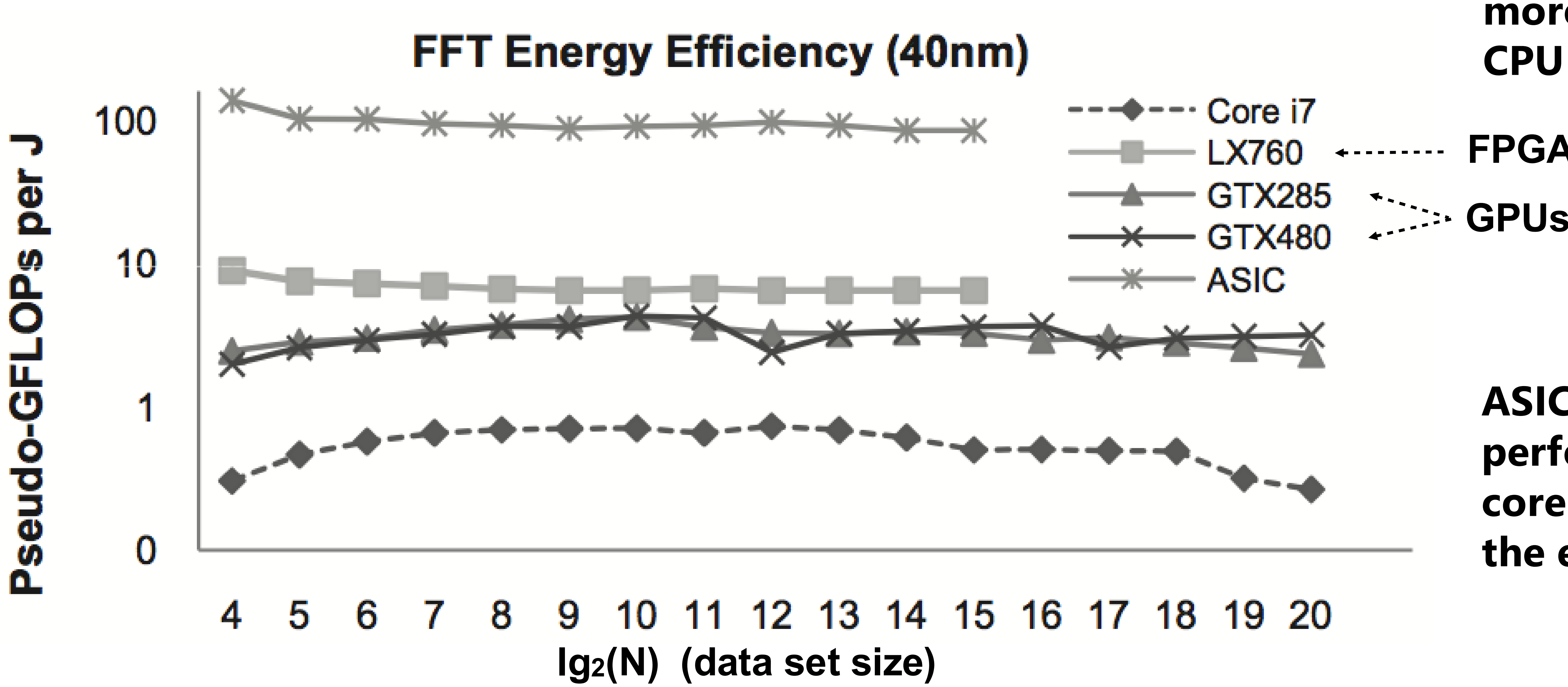
[Source: Chung et al. 2010 , Dally 08]

Hardware specialization increases efficiency



ASIC delivers same performance as one CPU core with ~ 1/1000th the chip area.

GPU cores: ~ 5-7 times more area efficient than CPU cores.



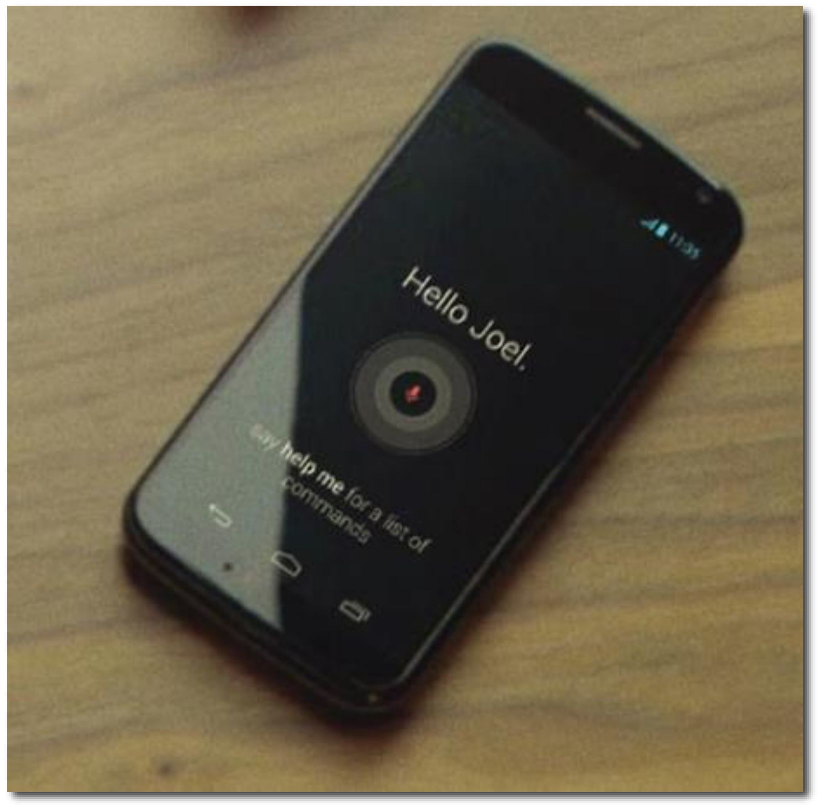
ASIC delivers same performance as one CPU core with only ~ 1/100th the energy.

[Chung et al. MICRO 2010]

Benefits of increasing efficiency

- Run faster for a fixed period of time
  - Run at higher clock, use more cores (reduce latency of critical task)
  - Do more at once
- Run at a fixed level of performance for longer
  - e.g., video playback
  - Achieve "always-on" functionality that was previously impossible

Siri activated by button press or holding phone up to ear



Always listening for "ok, google now"
Device contains ASIC for detecting this audio pattern.

Recording triggered by motion (continuous would drain battery in ~hours)

Original iPhone touchscreen controller

Separate digital signal processor to interpret raw signal from capacitive touch sensor (do not burden main CPU)

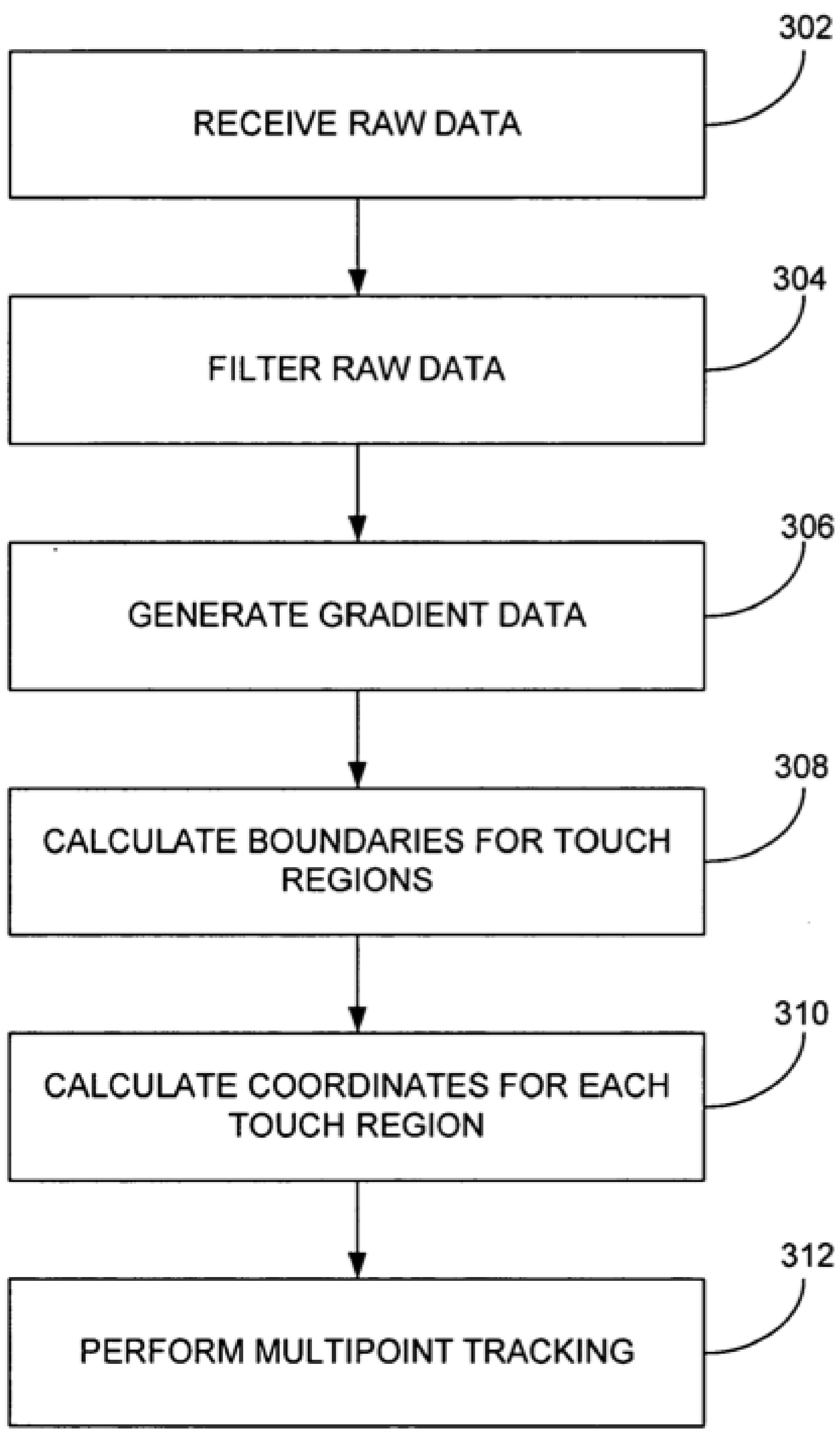


FIG. 16

RAW DATA INCLUDING NOISE

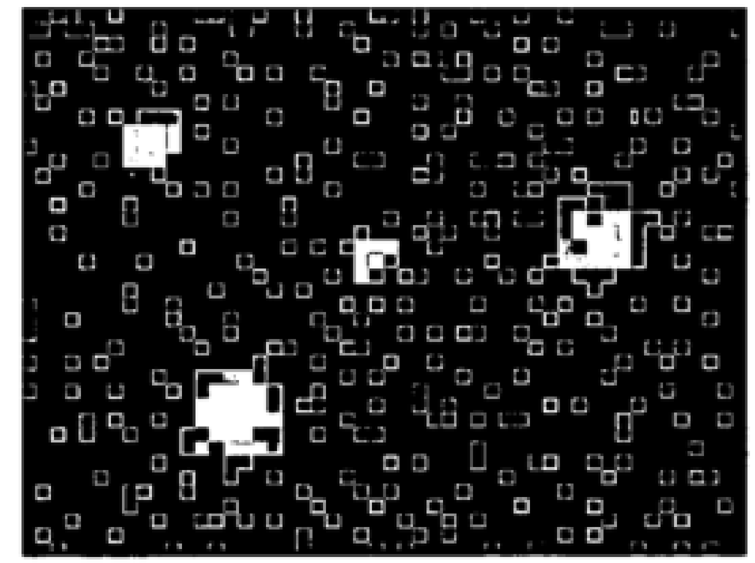


FIG. 17A

FILTERED DATA

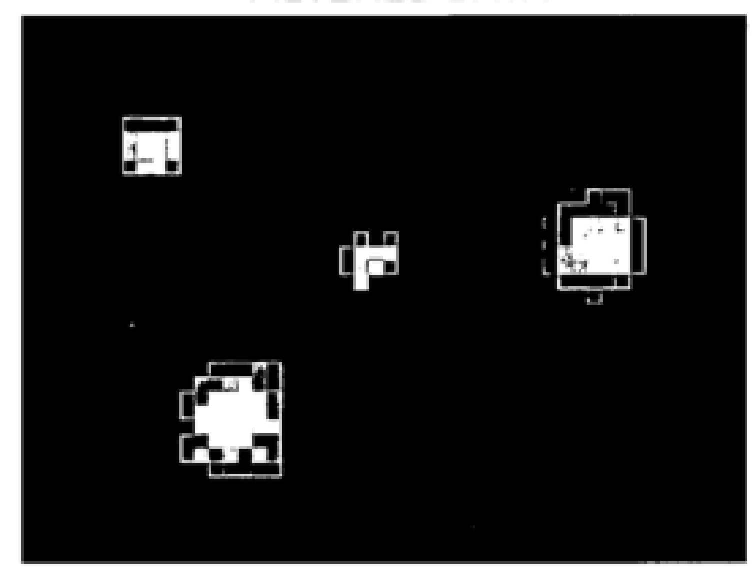


FIG. 17B

GRADIENT DATA

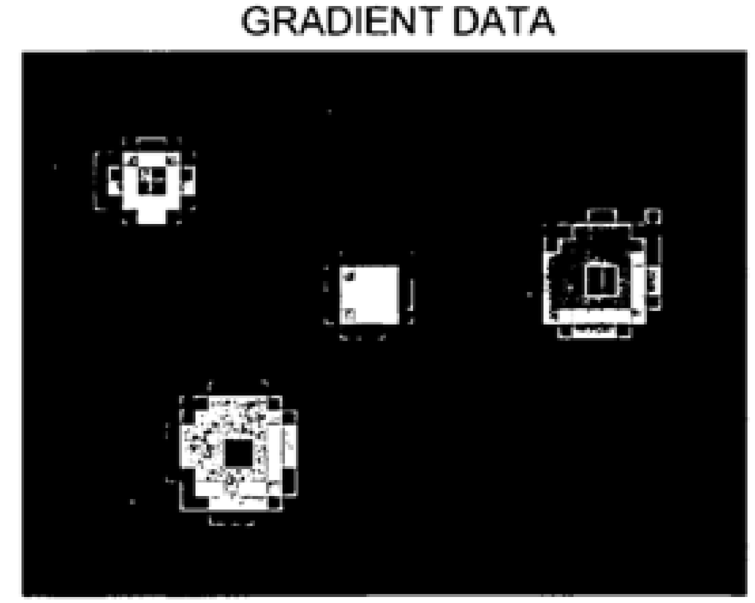


FIG. 17C

TOUCH REGIONS

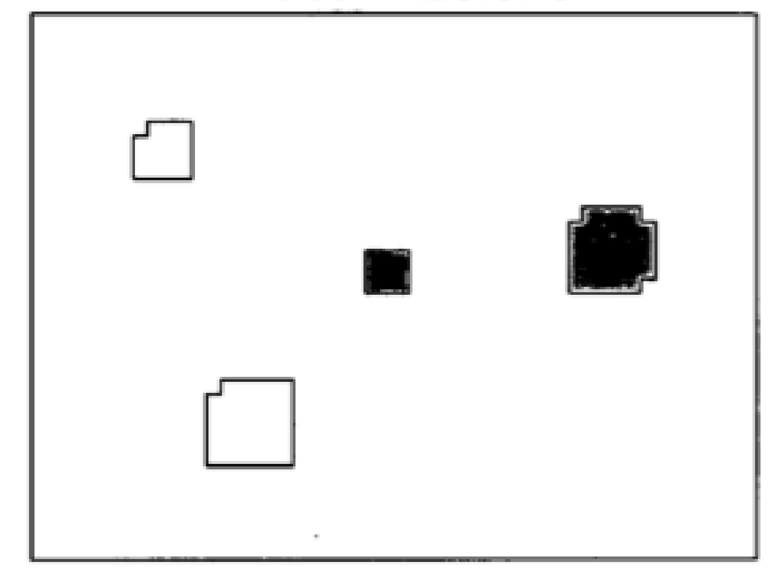


FIG. 17D

COORDINATES OF TOUCH REGIONS

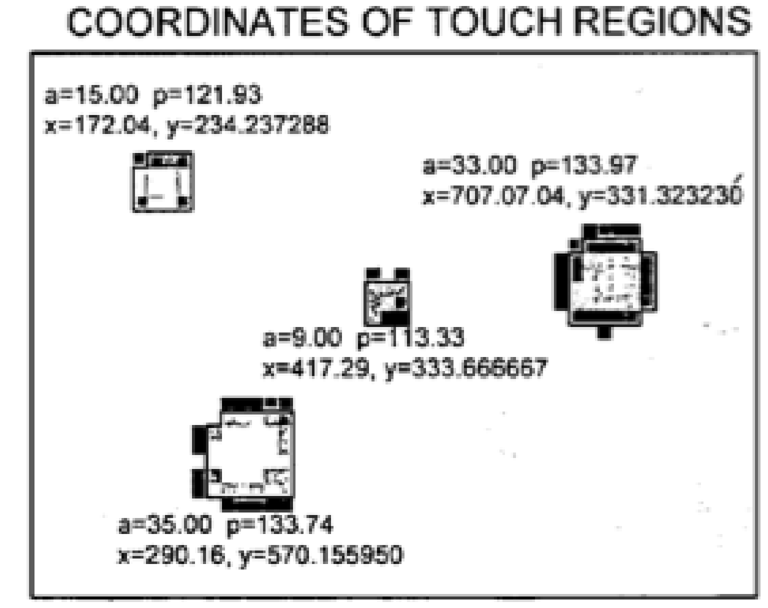


FIG. 17E

From US Patent Application 2006/0097991

Modern computing: efficiency often matters more than in the past, not less

Fourth, there's battery life.

To achieve long battery life when playing video, mobile devices must decode the video in hardware; decoding it in software uses too much power. Many of the chips used in modern mobile devices contain a decoder called H.264 – an industry standard that is used in every Blu-ray DVD player and has been adopted by Apple, Google (YouTube), Vimeo, Netflix and many other companies.

Although Flash has recently added support for H.264, the video on almost all Flash websites currently requires an older generation decoder that is not implemented in mobile chips and must be run in software. The difference is striking: on an iPhone, for example, H.264 videos play for up to 10 hours, while videos decoded in software play for less than 5 hours before the battery is fully drained.

When websites re-encode their videos using H.264, they can offer them without using Flash at all. They play perfectly in browsers like Apple's Safari and Google's Chrome without any plugins whatsoever, and look great on iPhones, iPods and iPads.

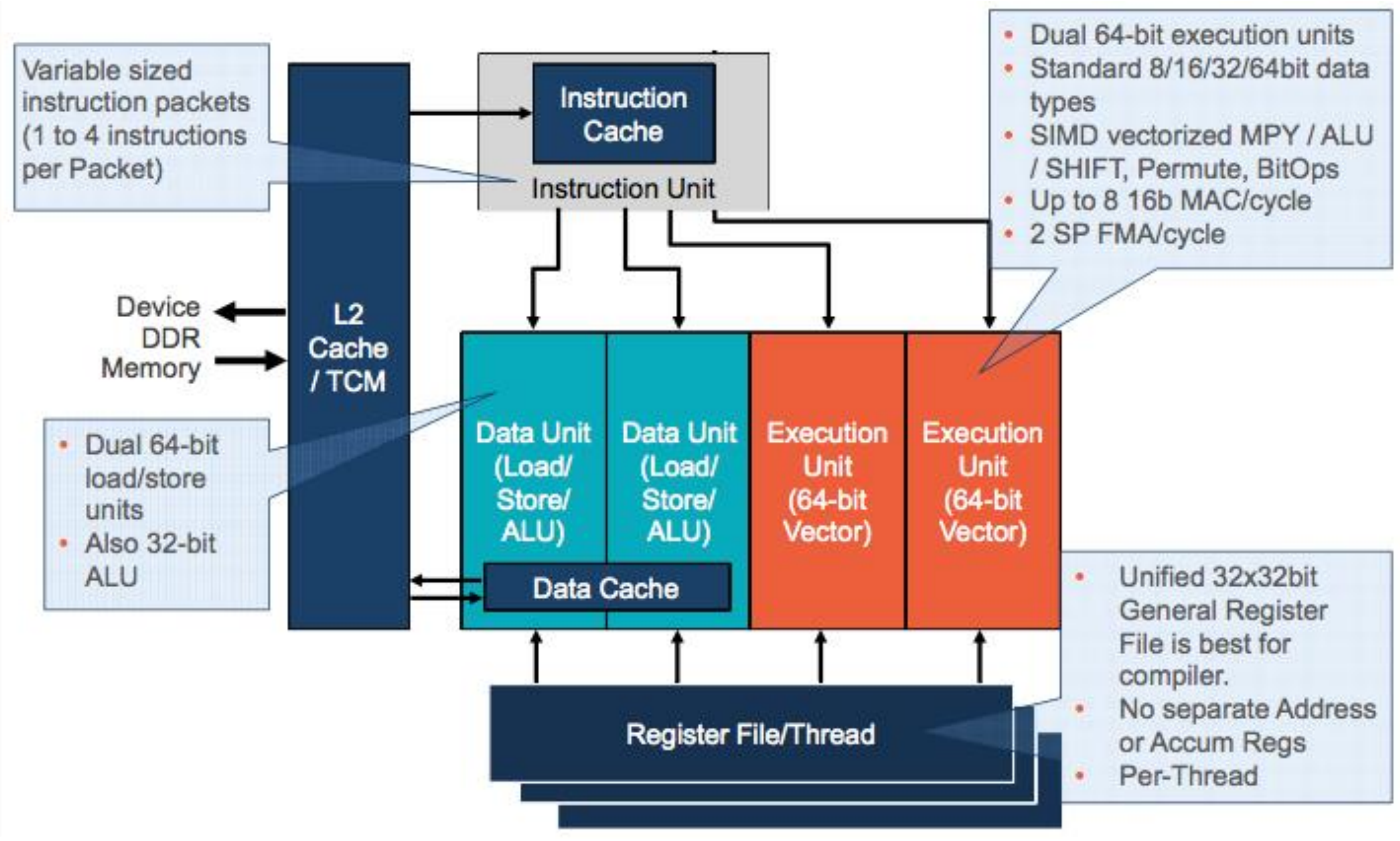
Steve Jobs' "Thoughts on Flash", 2010

http://www.apple.com/hotnews/thoughts-on-flash/

(Justification for why Apple won't support Adobe Flash)

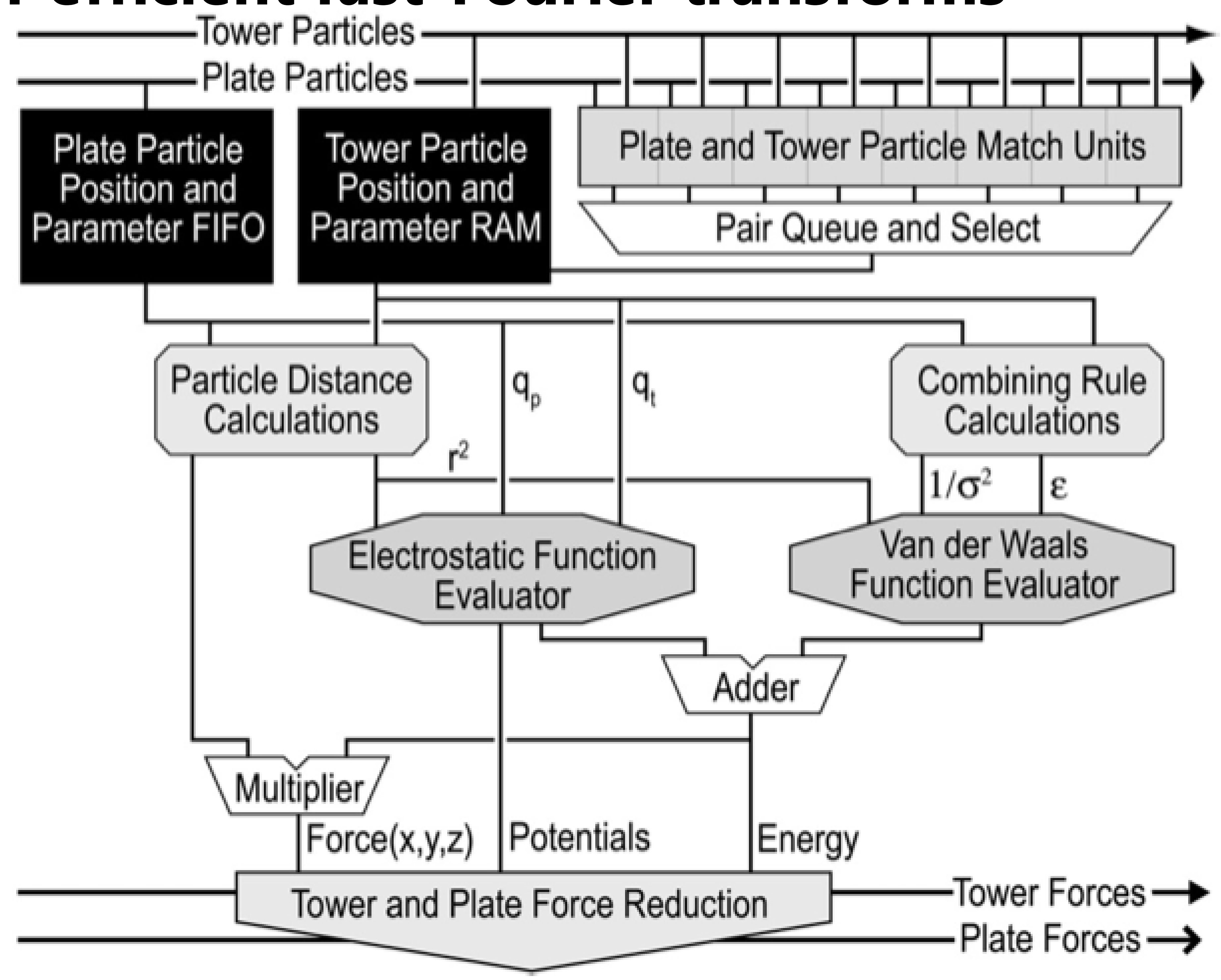
Qualcomm Hexagon Digital Signal Processor

- Originally used for audio/LTE support on Qualcomm SoCs
- Multi-threaded, VLIW DSP
- Third major programmable unit on Qualcomm SoCs
  - Multi-core CPU
  - Multi-core GPU (Adreno)
  - Hexagon DSP

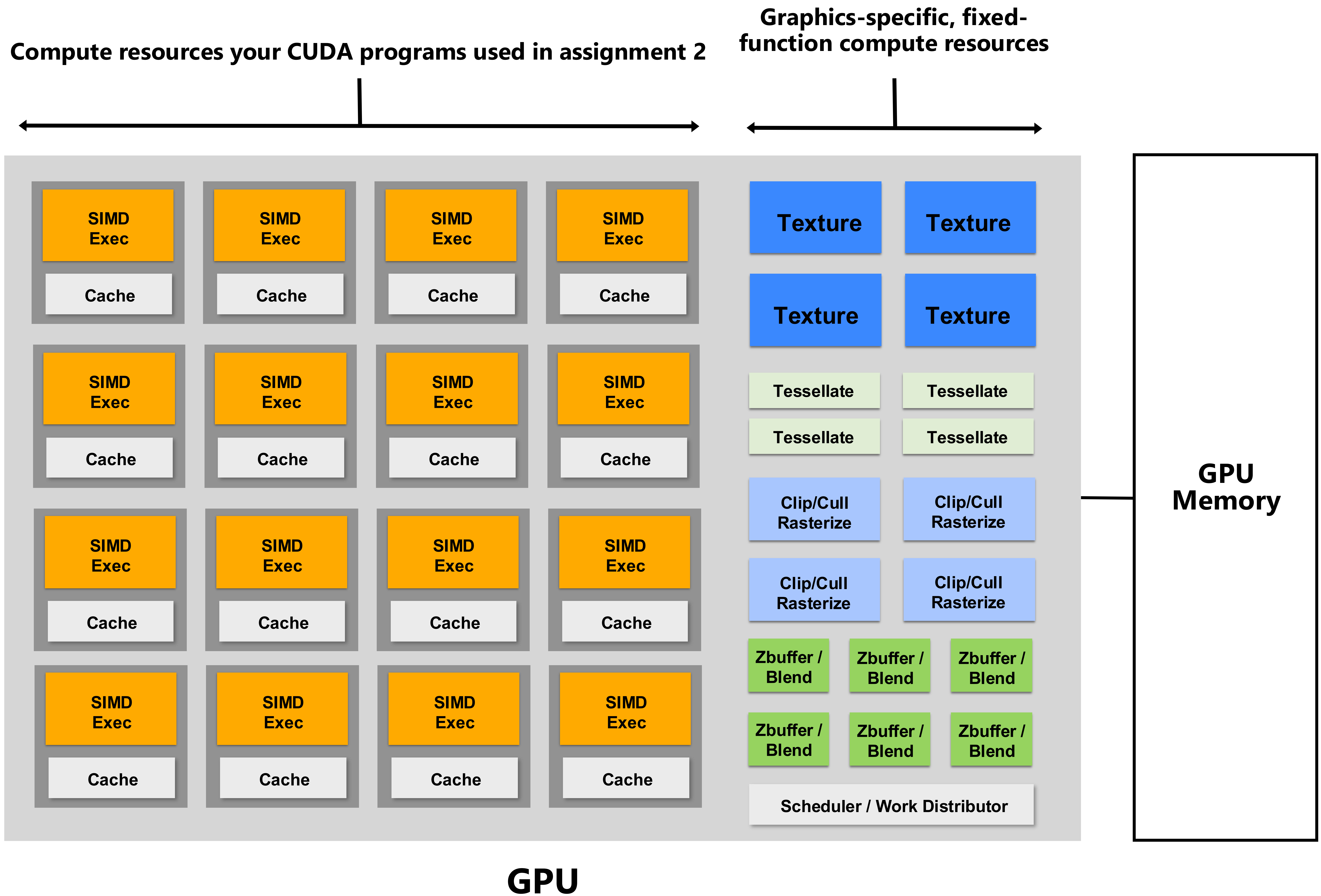


Anton supercomputer

- Supercomputer highly specialized for molecular dynamics
  - Simulates time evolution of proteins
- ASIC for computing particle-particle interactions (512 of them in machine)
- Throughput-oriented subsystem for efficient fast-Fourier transforms
- Custom, low-latency communication network designed for communication patterns of N-body simulations

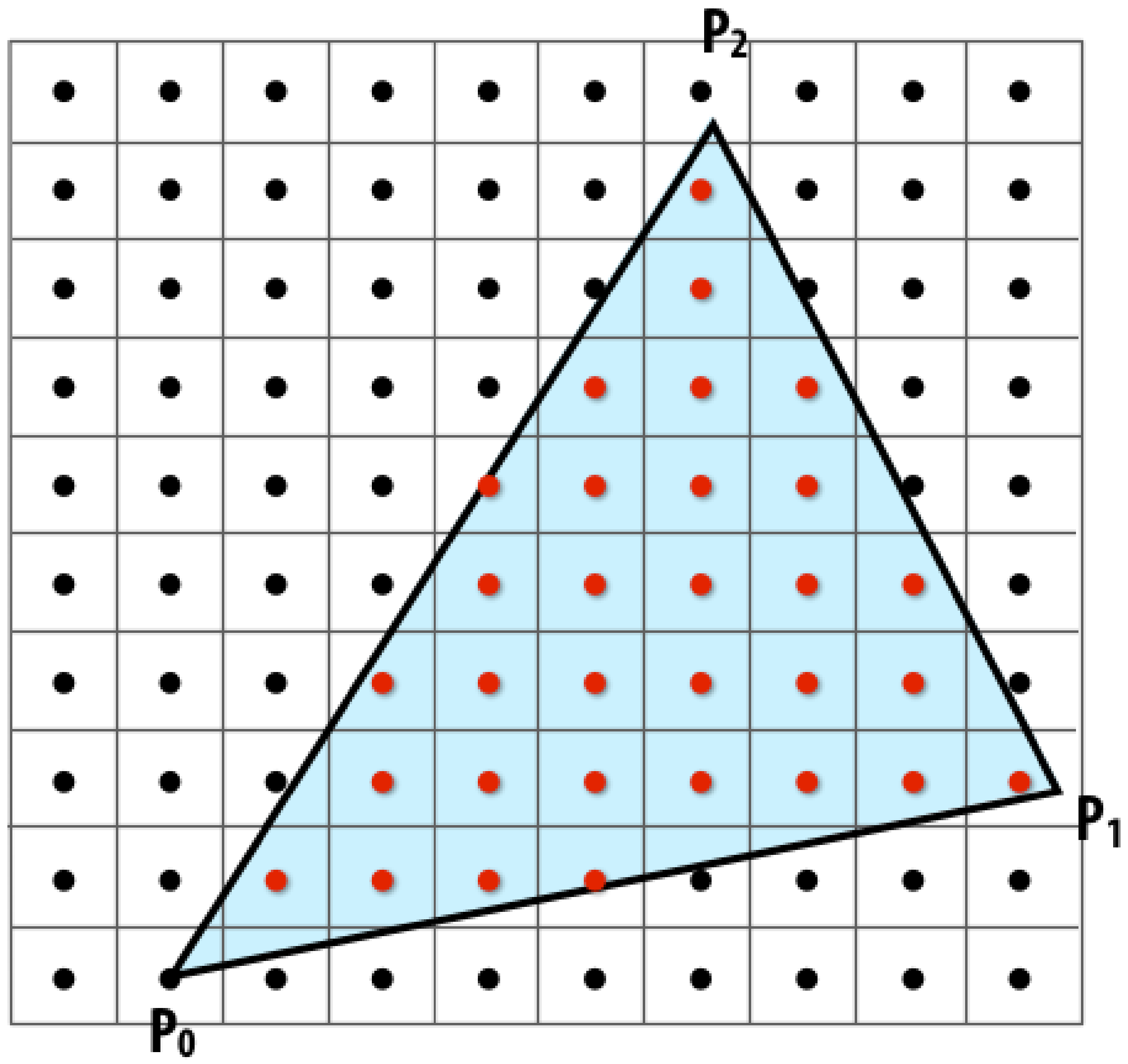


GPUs are heterogeneous multi-core processors

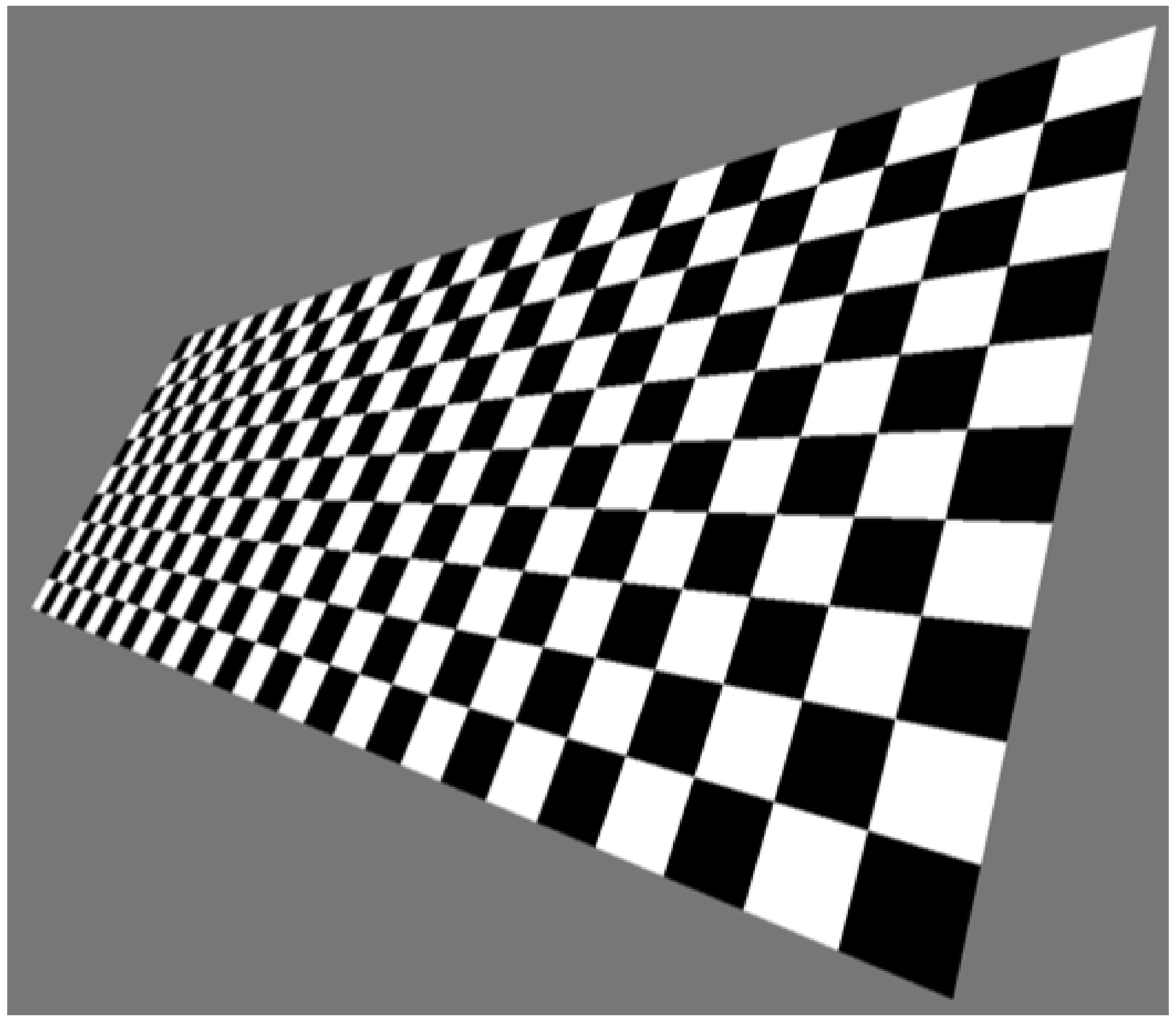


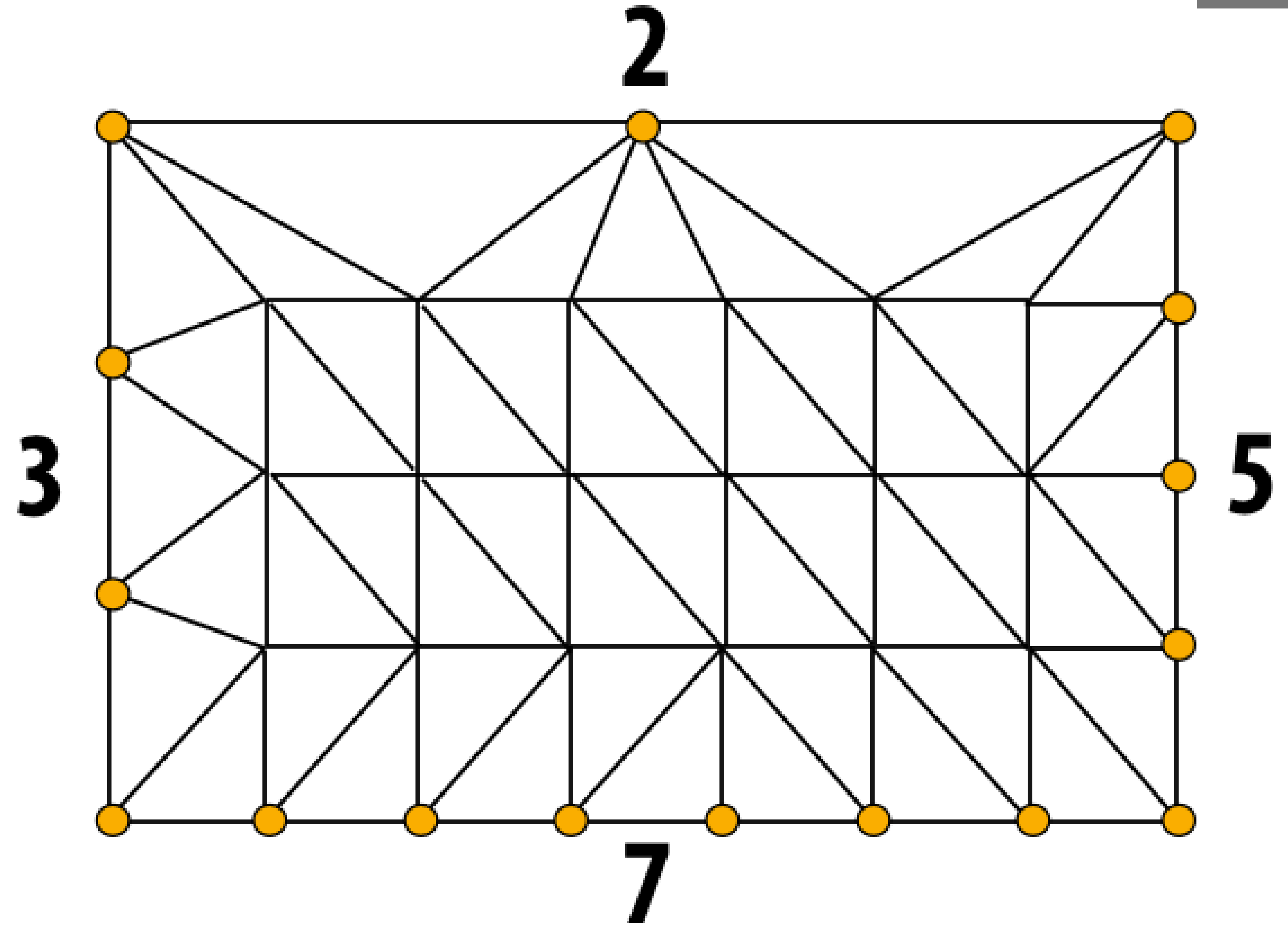
Example graphics tasks performed in fixed-function HW

Rasterization:
Determining what pixels a triangle overlaps



Texture mapping:
Warping/filtering images to apply detail to surfaces

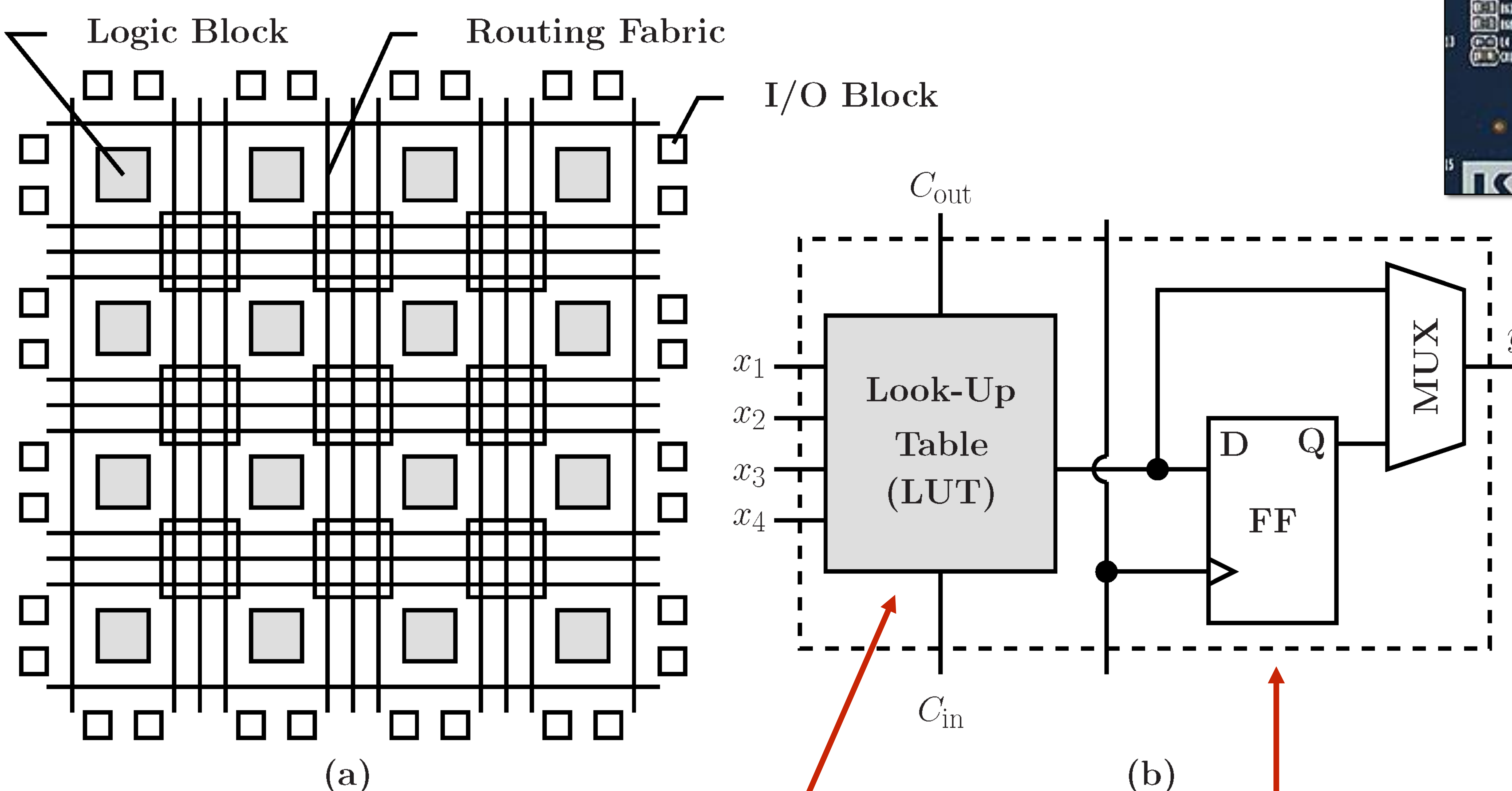




Geometric tessellation: computing fine-scale geometry from coarse geometry

FPGAs (Field Programmable Gate Arrays)

- **Middle ground between an ASIC and a processor**
- **FPGA chip provides array of logic blocks, connected by interconnect**
- **Programmer-defined logic implemented directly by FGPA**



Flip flop (a register)

Programmable lookup table (LUT)

Image credit: Bai et al. 2014

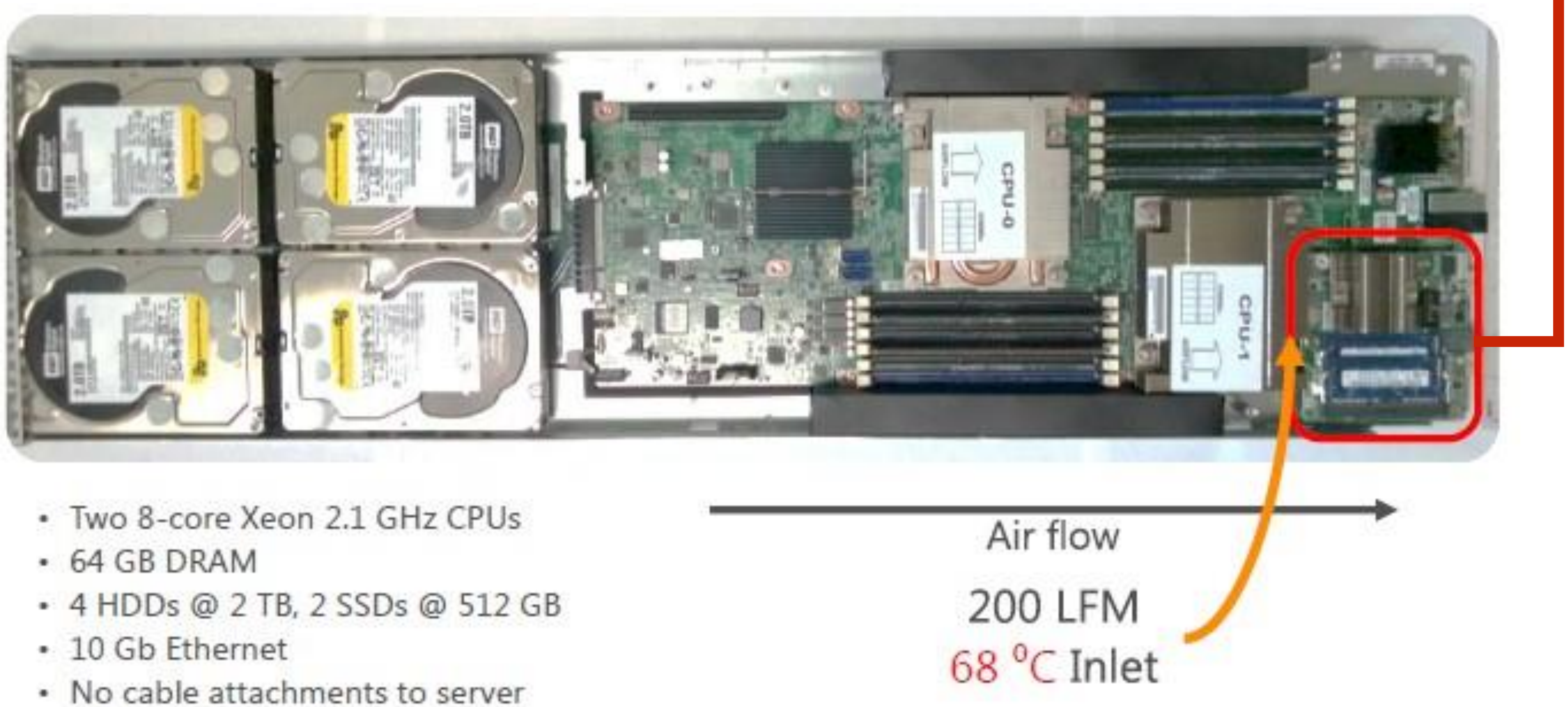
Project Catapult

[Putnam et al. ISCA 2014]

- Microsoft Research using FPGAs to accelerate datacenter workloads
- Demonstrated offload of part of Bing Search's document ranking logic
- Now widely used to accelerate DNNs across Microsoft services & other system infrastructure

FPGA board

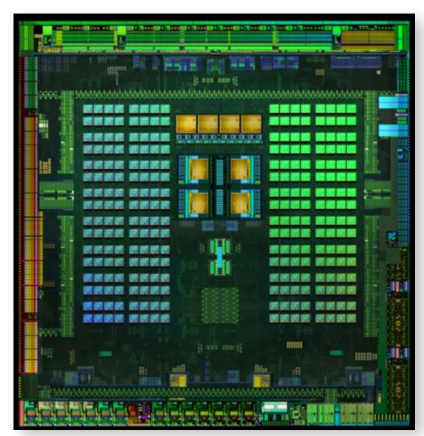
1U server (Dual socket CPU + FPGA connected via PCIe bus)



Summary: choosing the right tool for the job

Energy-optimized CPU

Throughput-oriented processor (GPU)



Programmable DSP

HEXAGON™

FPGA/Future reconfigurable logic

ASIC

Video encode/decode, Audio playback, Camera RAW processing, neural nets (future?)

Easiest to program

~10X more efficient

~100X??? (jury still out)

~100-1000X more efficient

Difficult to program (making it easier is active area of research)

Not programmable + costs 10-100's millions of dollars to design / verify / create

Credit Pat Hanrahan for this taxonomy

Challenges of heterogeneous designs

Challenges of heterogeneity

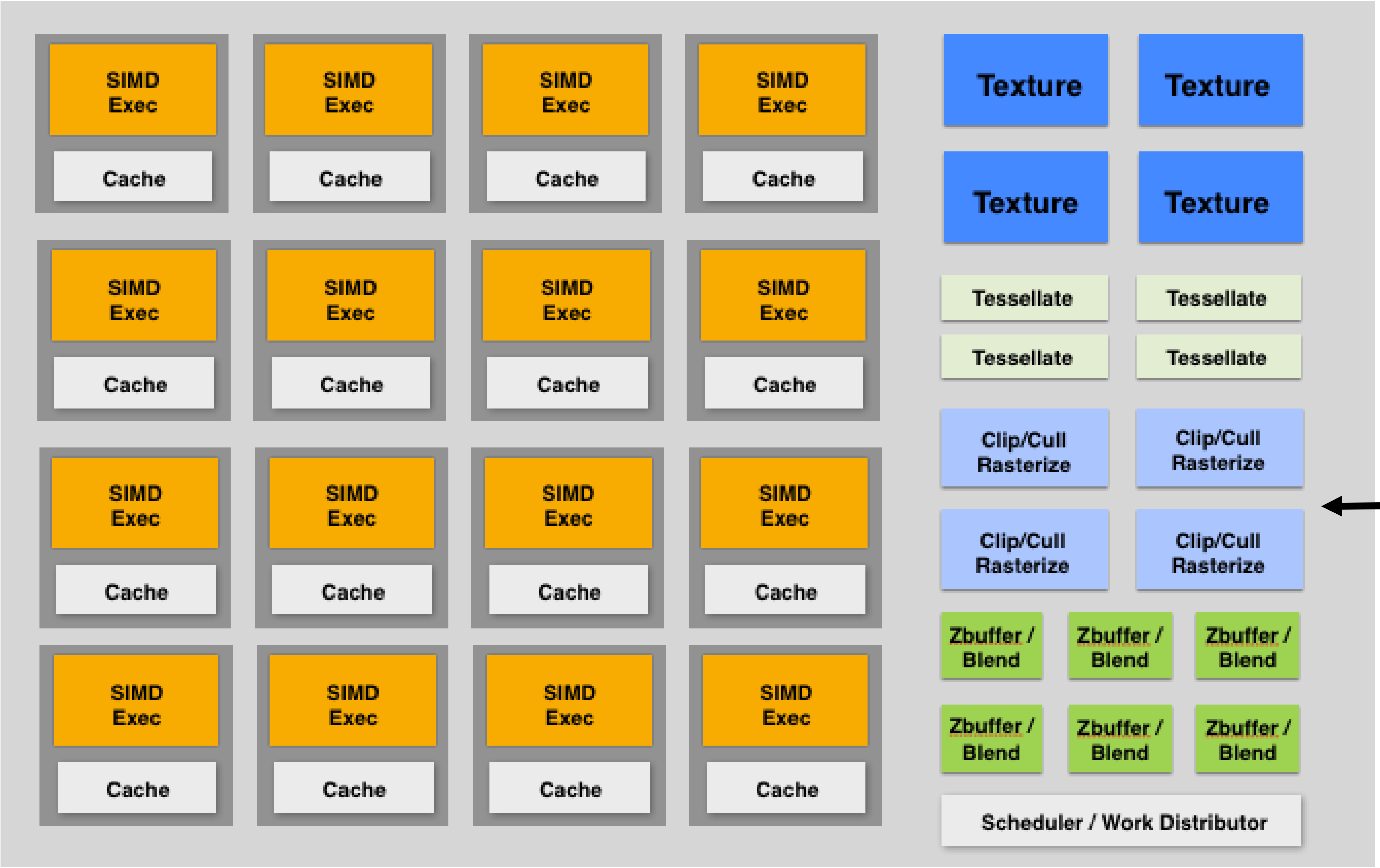
- So far in this course:
  - Homogeneous system: every processor can be used for every task
  - To get best speedup vs. sequential execution, “keep all processors busy all the time” (this is hard to achieve)

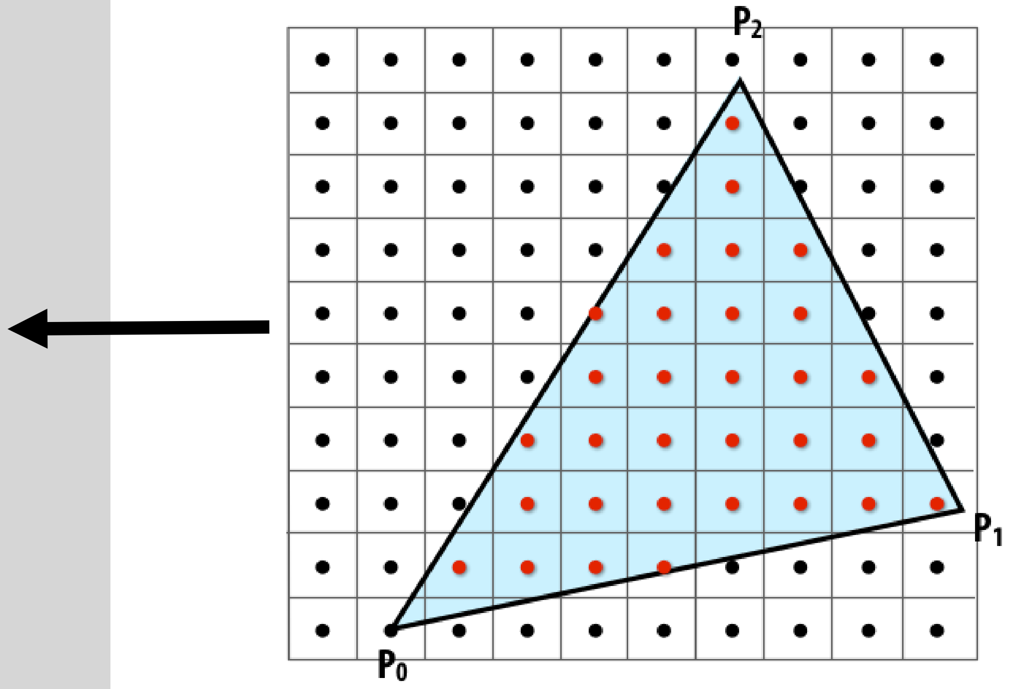
- Heterogeneous system: use preferred processor for each task
  - Challenge for system designer: what is the right mixture of resources to meet performance, cost, and energy goals?
  - Too few throughput-oriented resources (lower peak performance/efficiency for parallel workloads -- should have used resources for more throughput cores)
  - Too few sequential processing resources (get bitten by Amdahl’s Law)
  - How much chip area should be dedicated to a specific function, like video? (Resources taken away from general-purpose processing)
  - How to publish what’s available to software? (What abstractions? What about compilers & operating systems?)

- Implication: increased pressure to understand workloads accurately at chip design time

Pitfalls of heterogeneous designs

[Molnar 2010]





Say 10% of the workload is rasterization
Let's say you under-provision the fixed-function rasterization unit on GPU:
Chose to dedicate 1% of chip area used for rasterizer, really needed 20% more throughput: 1.2% of chip area

Problem: rasterization is now the bottleneck, so the expensive programmable processors (99% of chip) are idle waiting on rasterization. 99% of the chip runs at 80% efficiency!
➔ Tendency is to be conservative, and over-provision fixed-function components (diminishing their advantage)

Challenges of heterogeneity

- Heterogeneous system: preferred processor for each task
  - Challenge for hardware designer: what is the right mixture of resources?
  - Too few throughput oriented resources (lower peak throughput for parallel workloads)
  - Too few sequential processing resources (limited by sequential part of workload)
  - How much chip area should be dedicated to a specific function, like video? (these resources are taken away from general-purpose processing)
  - Work balance must be anticipated at chip design time
    ➔ System cannot adapt to changes in usage over time, new algorithms, etc.

- Challenge to software developer: how to map programs onto a heterogeneous collection of resources?
  - Challenge: "Pick the right tool for the job": design algorithms that decompose well into components that each map well to different processing components of the machine
  - The scheduling problem is more complex on a heterogeneous system
  - Available mixture of resources can dictate choice of algorithm
  - Software portability & maintenance nightmare (we'll revisit this next class)

Reducing energy consumption idea 1: use specialized processing

Reducing energy consumption idea 2: move less data

Data movement has high energy cost

- Rule of thumb in mobile system design: always seek to reduce amount of data transferred from memory
  - Earlier in class we discussed minimizing communication to reduce stalls (poor performance). Now, we wish to reduce communication to reduce energy consumption
- "Ballpark" numbers [Sources: Bill Dally (NVIDIA), Tom Olson (ARM)]
  - Integer op: ~ 1 pJ *
  - Floating point op: ~20 pJ *
  - Reading 64 bits from small local SRAM (1mm away on chip): ~ 26 pJ
  - Reading 64 bits from low power mobile DRAM (LPDDR): ~1200 pJ
- Implications
  - Reading 10 GB/sec from memory: ~1.6 watts
  - Entire power budget for mobile GPU: ~1 watt (remember phone is also running CPU, display, radios, etc.)
  - iPhone 6 battery: ~7 watt-hours (compare: Macbook Pro laptop: 99 watt-hour battery)
  - Exploiting locality matters!!!

Suggests that recomputing values, rather than storing and reloading them, is a better answer when optimizing code for energy efficiency!

* Cost to just perform the logical operation, not counting overhead of instruction decode, load data from registers, etc.

Three trends in energy-optimized computing

- **Compute less!**
  - Computing costs energy: parallel algorithms that do more work than sequential counterparts may not be desirable even if they run faster
  - …But performance matters too, because processors burn energy whenever they are turned on ("static power")
- **Specialize compute units:**
  - Heterogeneous processors: CPU-like cores + throughput-optimized cores (GPU-like cores)
  - Fixed-function units: audio processing, "movement sensor processing" video decode/encode, image processing/computer vision?
  - Specialized instructions: expanding set of AVX vector instructions, new instructions for accelerating AES encryption (AES-NI)
  - Programmable soft logic: FPGAs
- **Reduce bandwidth requirements**
  - Exploit locality (restructure algorithms to reuse on-chip data as much as possible)
  - Aggressive use of compression: perform extra computation to compress application data before transferring to memory (likely to see fixed-function HW to reduce overhead of general data compression/decompression)

Summary

- **Heterogeneous parallel processing: use a mixture of computing resources that each fit with mixture of needs of target applications**
  - Latency-optimized sequential cores, throughput-optimized parallel cores, domain-specialized fixed-function processors
  - Examples exist throughout modern computing: mobile processors, servers, supercomputers
- **Traditional rule of thumb in "good system design" is to design simple, general-purpose components**
  - This is not the case with emerging processing systems (optimized for perf/watt)
  - Today: want collection of components that meet perf requirement AND minimize energy use
- **Challenge of using these resources effectively is pushed up to the programmer**
  - Current CS research challenge: how to write efficient, portable programs for emerging heterogeneous architectures?