Lecture 21:

Heterogeneous Parallelism and Hardware Specialization

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2014

Tunes

Kanye West

Amazing

"Have you seen you fast Kim's iPhone battery dies if she tries to play back video in a software decoder? Amazing."

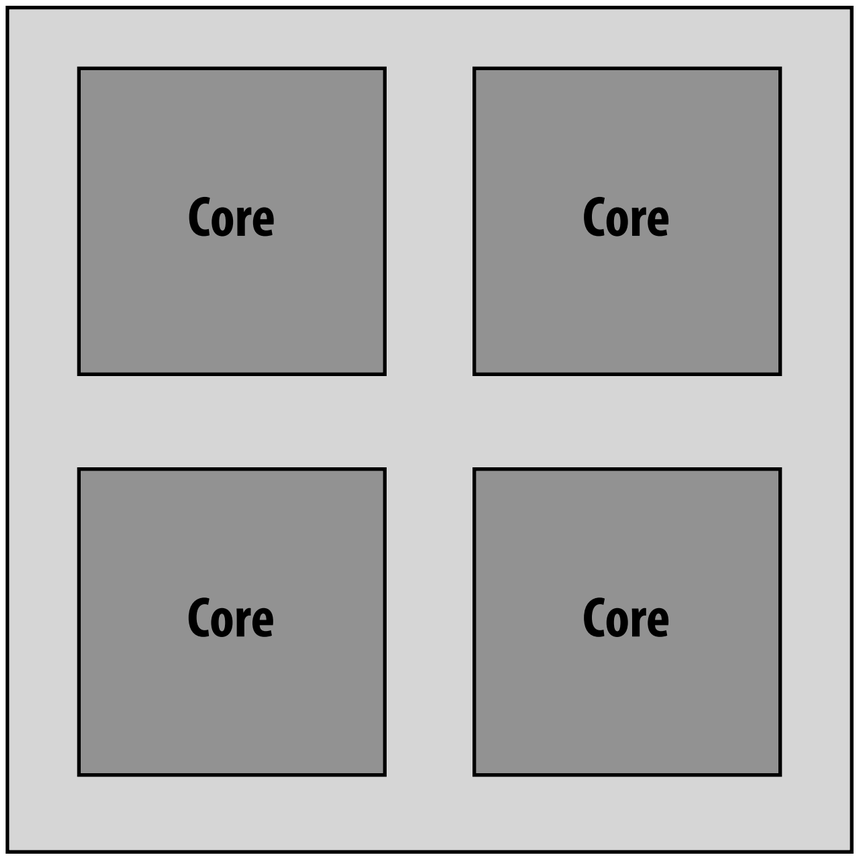
- Kanye

Logistics

- Feedback coming throughout the week ASAP
  - Send us a note if it is critical and we'll prioritize

- Kayvon's Office Hours
  - Wed: 4-5pm
  - Thurs: 1:30-2:30pm
  - Fri: 11:30-12:30pm

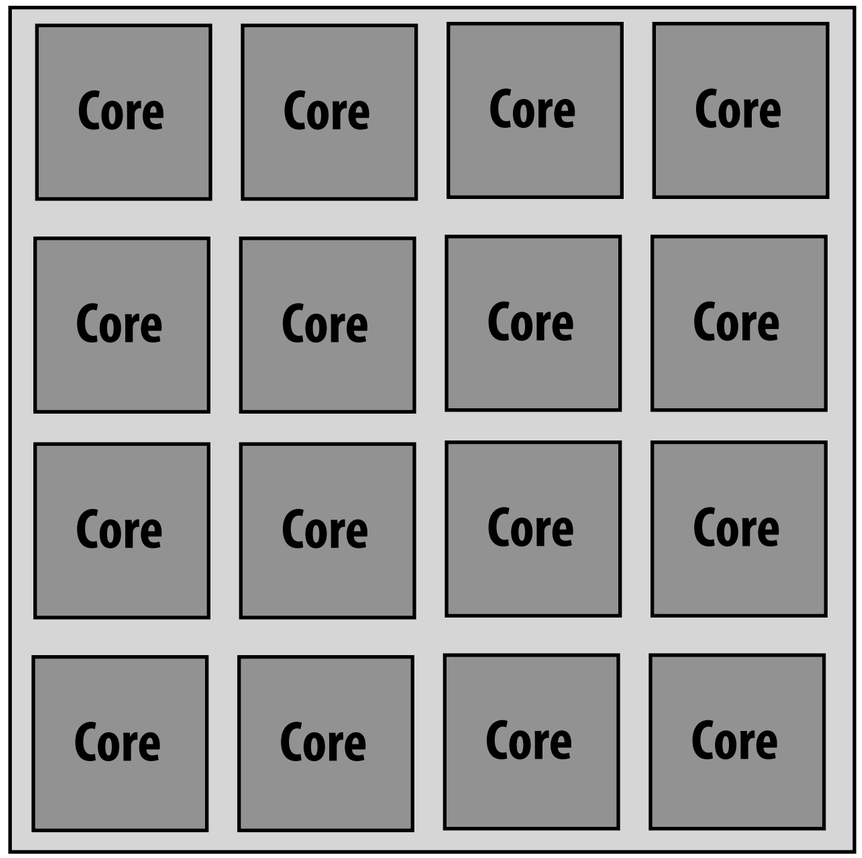
You need to buy a computer system



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

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

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

$n =$ **parallel processors**

Assumptions:

Parallelizable work distributes perfectly onto n processors of equal capability

Account for resource limits

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

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

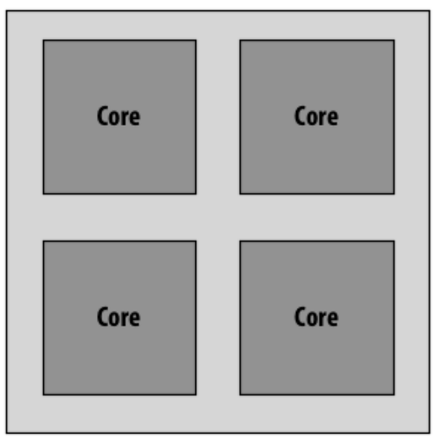
f = fraction of program that is parallelizable

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



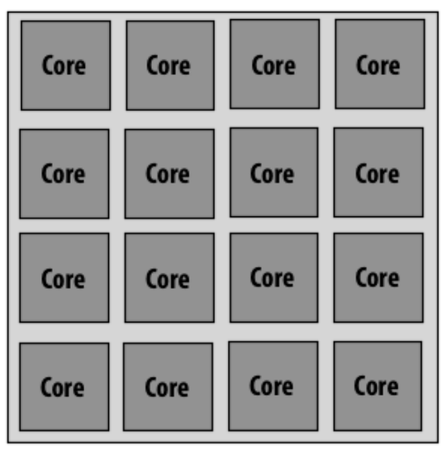
Processor A

Example:

Let n=16

$r_A = 4$

$r_B = 1$

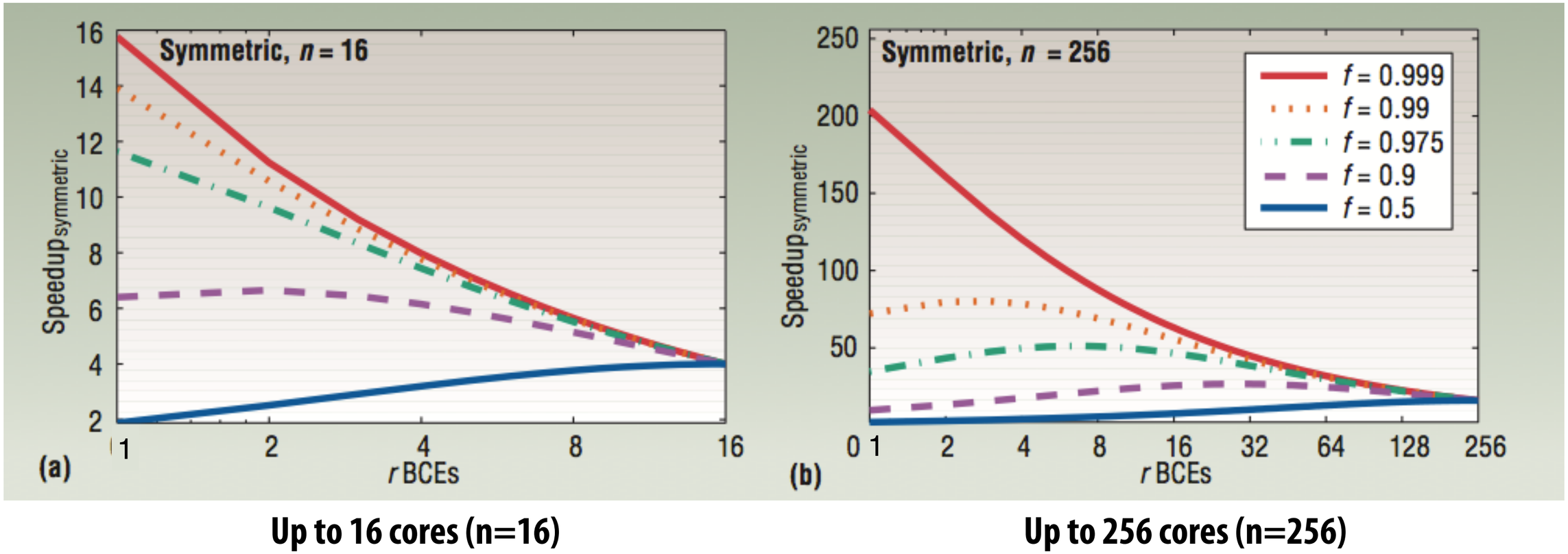


Processor B

[Hill and Marty 08]

Speedup (relative to n=1)

[Source: Hill and Marty 08]



Up to 16 cores (n=16)

Up to 256 cores (n=256)

Each line corresponds to a different chip configuration

All lines on the same graph correspond to same number of resources (same n)

X-axis = r (many small cores to left, fewer "fatter" cores to right)

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

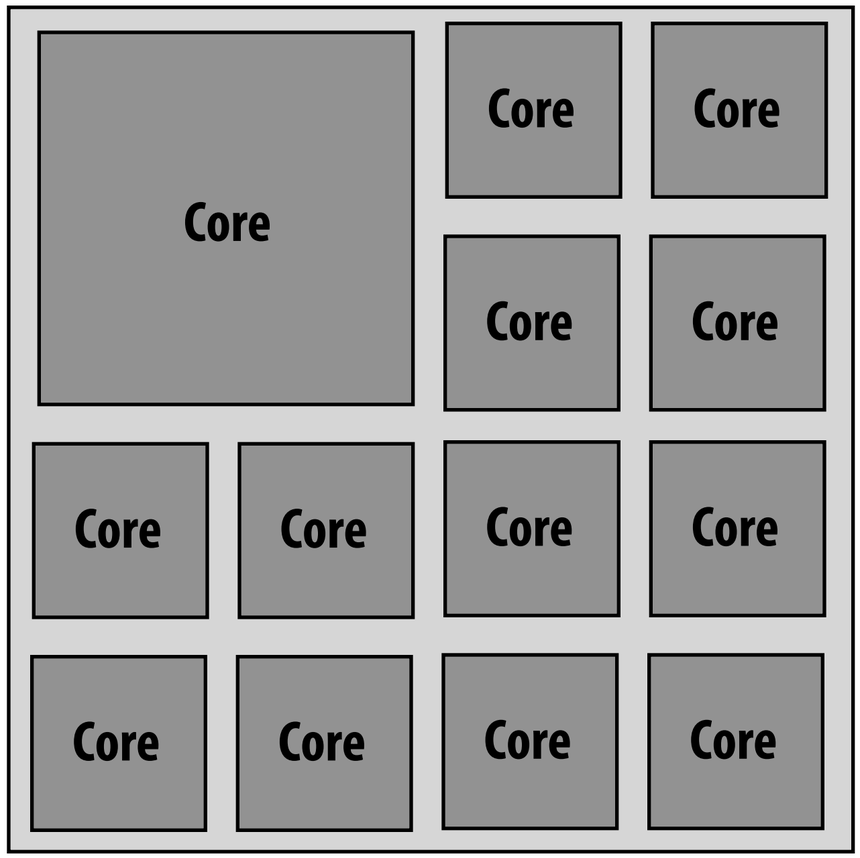
Asymmetric set of processing cores

Example:

Let n=16

One core: $r = 4$

12 cores: $r = 1$



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

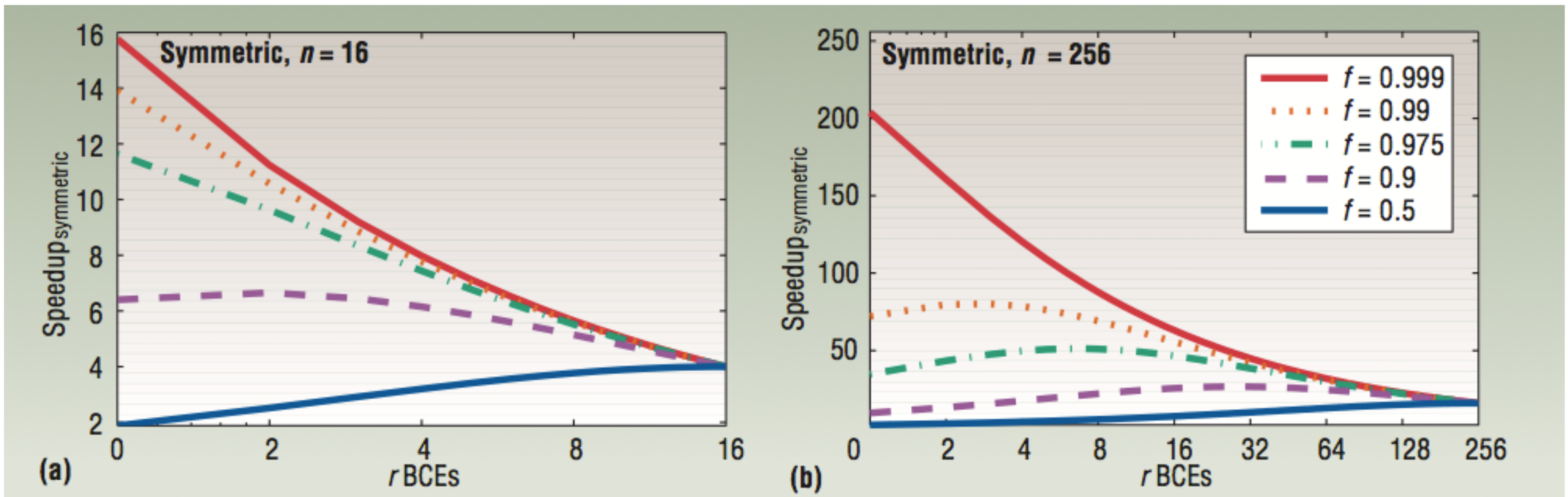
(heterogeneous processor with n recourses relative to uniprocessor with one unit worth of resources, n=1)

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

X-axis for asymmetric architectures gives r for the single "fat" core (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

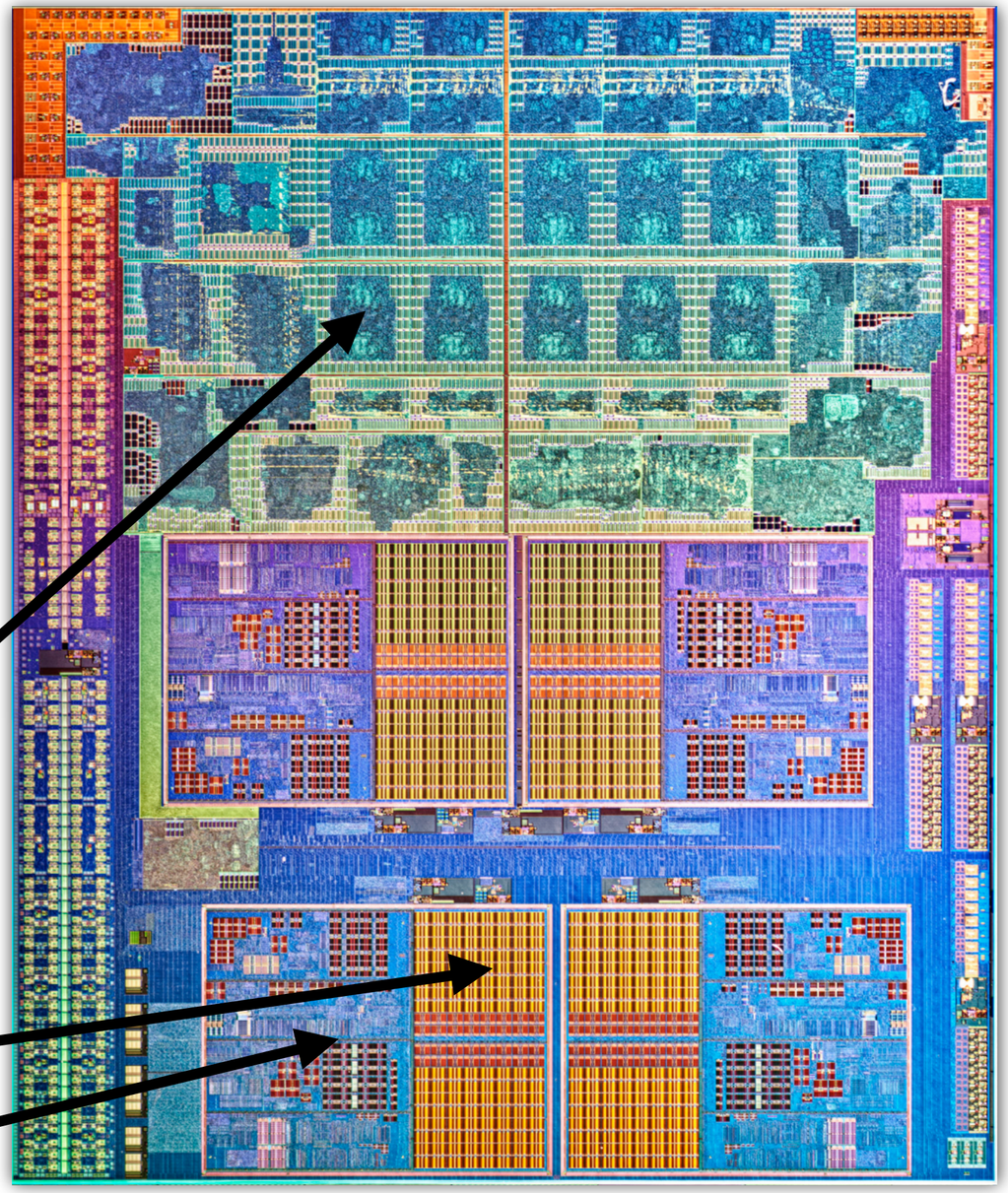
And components with unpredictable access, but those accesses might cache well.

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

Example: AMD Fusion

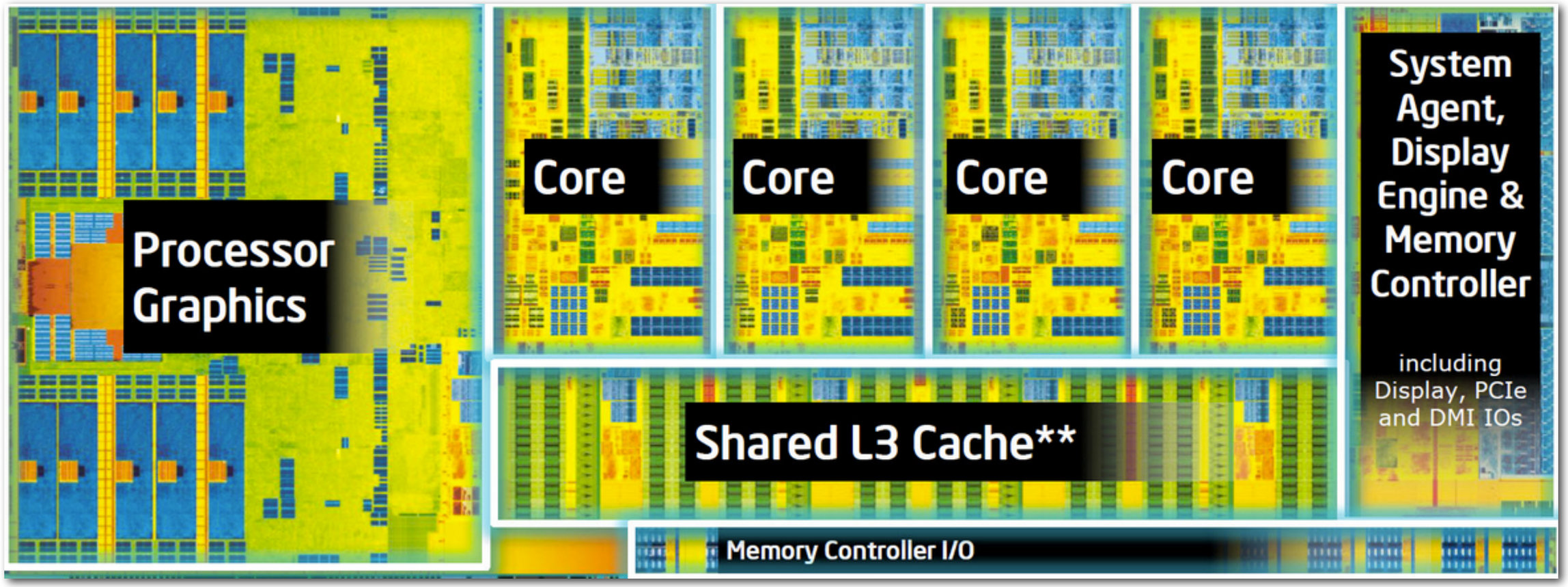
- "APU": accelerated processing unit
- Integrate CPU cores and GPU-style cores on same chip
- Share memory system
  - Regions of physical memory reserved for graphics (not x86 coherent)
  - Rest of memory is x86 coherent
  - CPU and graphics memory spaces are not coherent, but at least there is no need to copy data in physical memory (or over PCIe bus) to communicate between CPU and graphics



AMD Llano
(4 CPU cores + integrated GPU cores)

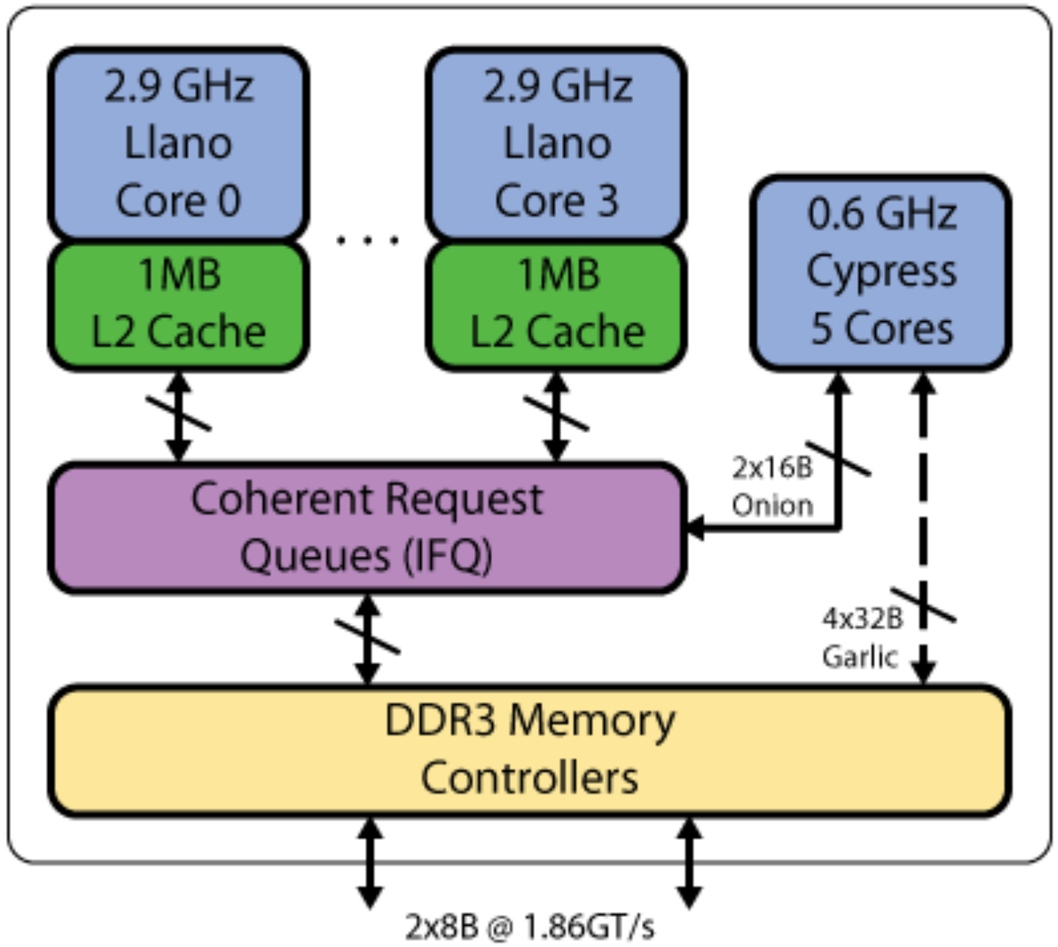
Example: Intel Haswell (2013)

- Four CPU cores + GPU (which itself has multiple cores)

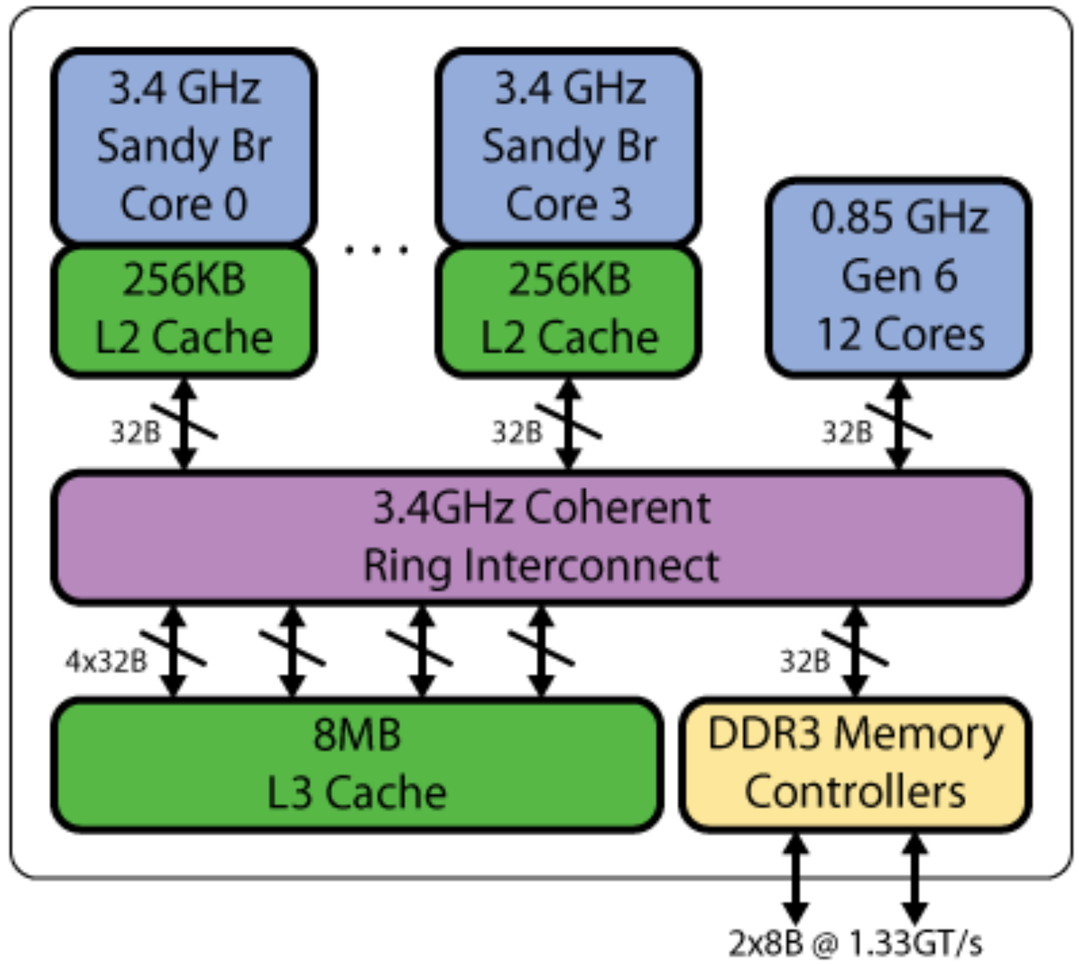


Multi-core CPU + integrated graphics

AMD Llano



Intel Sandy Bridge



More heterogeneity: add discrete GPU

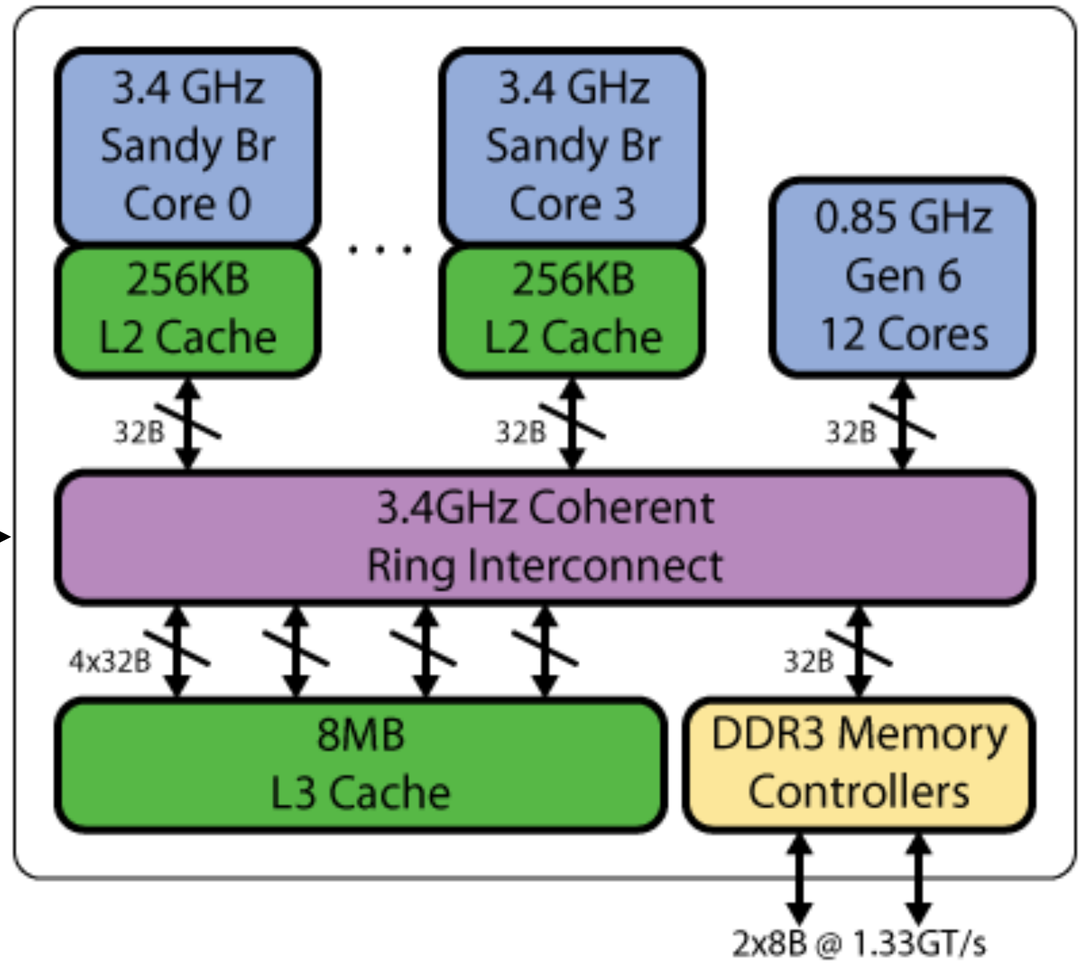
Keep discrete (power hungry) GPU unless needed for graphics-intensive applications
Use integrated, low power graphics for basic graphics/window manager/UI

Discrete high-end GPU (AMD or NVIDIA)

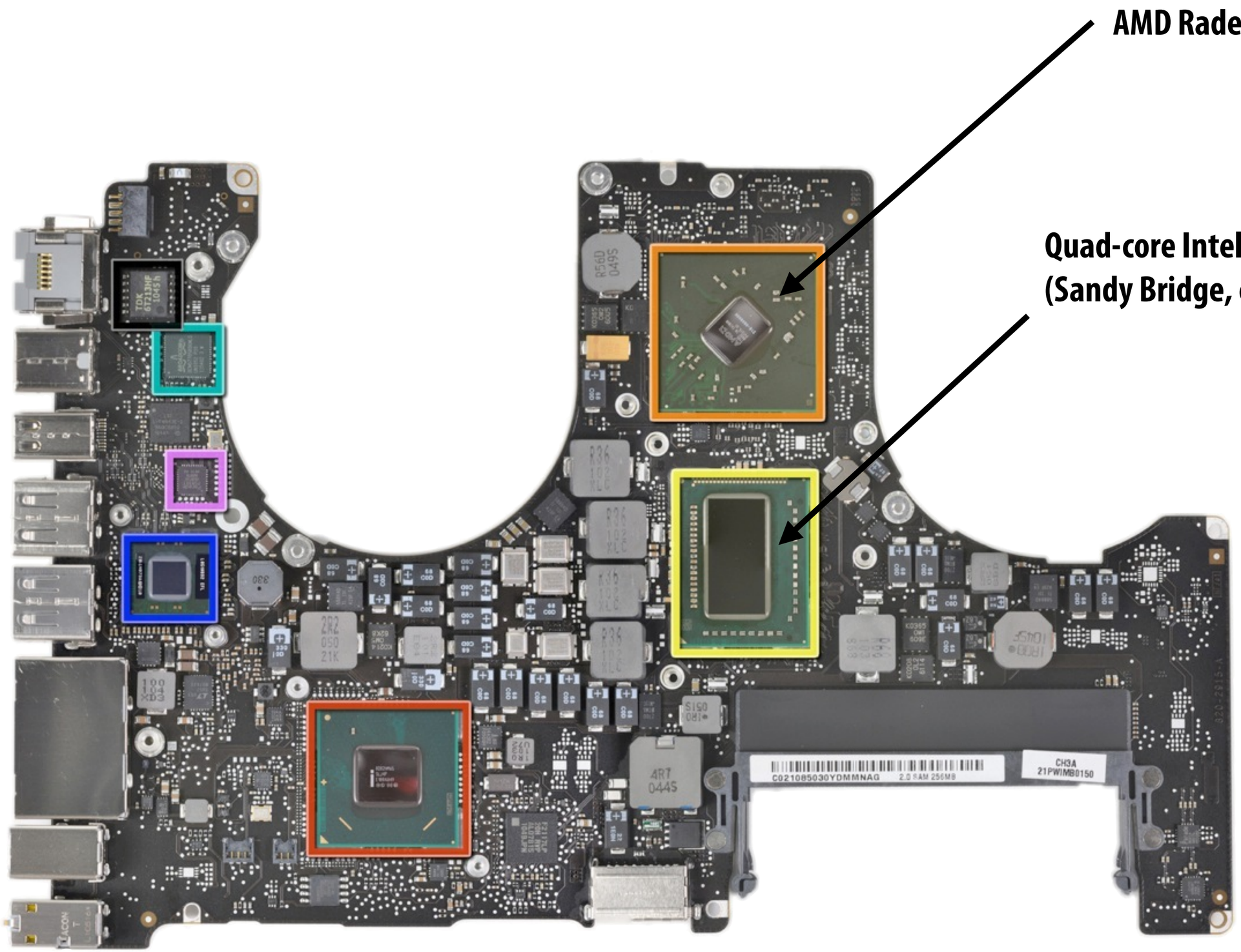
PCIe bus

DDR5 Memory

Intel Sandy Bridge



Macbook Pro 2011 (two GPUs)



From ifixit.com teardown

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 Titan (world's #2)

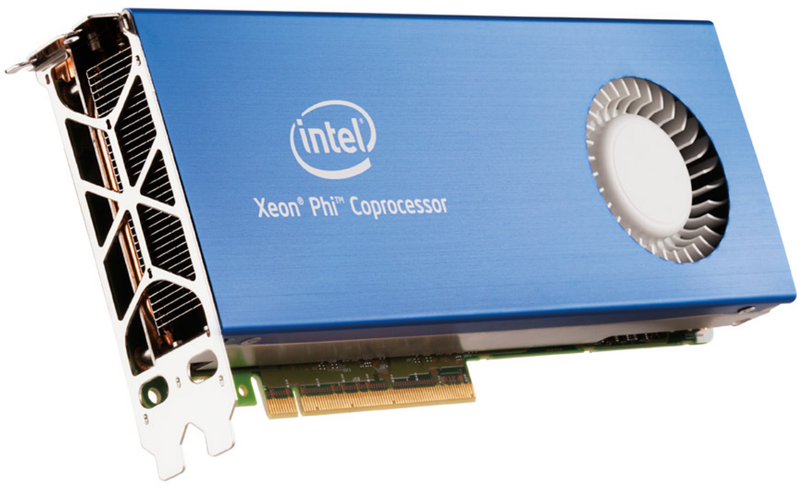
- 18,688 AMD Opteron 16-core CPUs
- 18,688 NVIDIA Tesla K20X GPUs
- 710 TB RAM

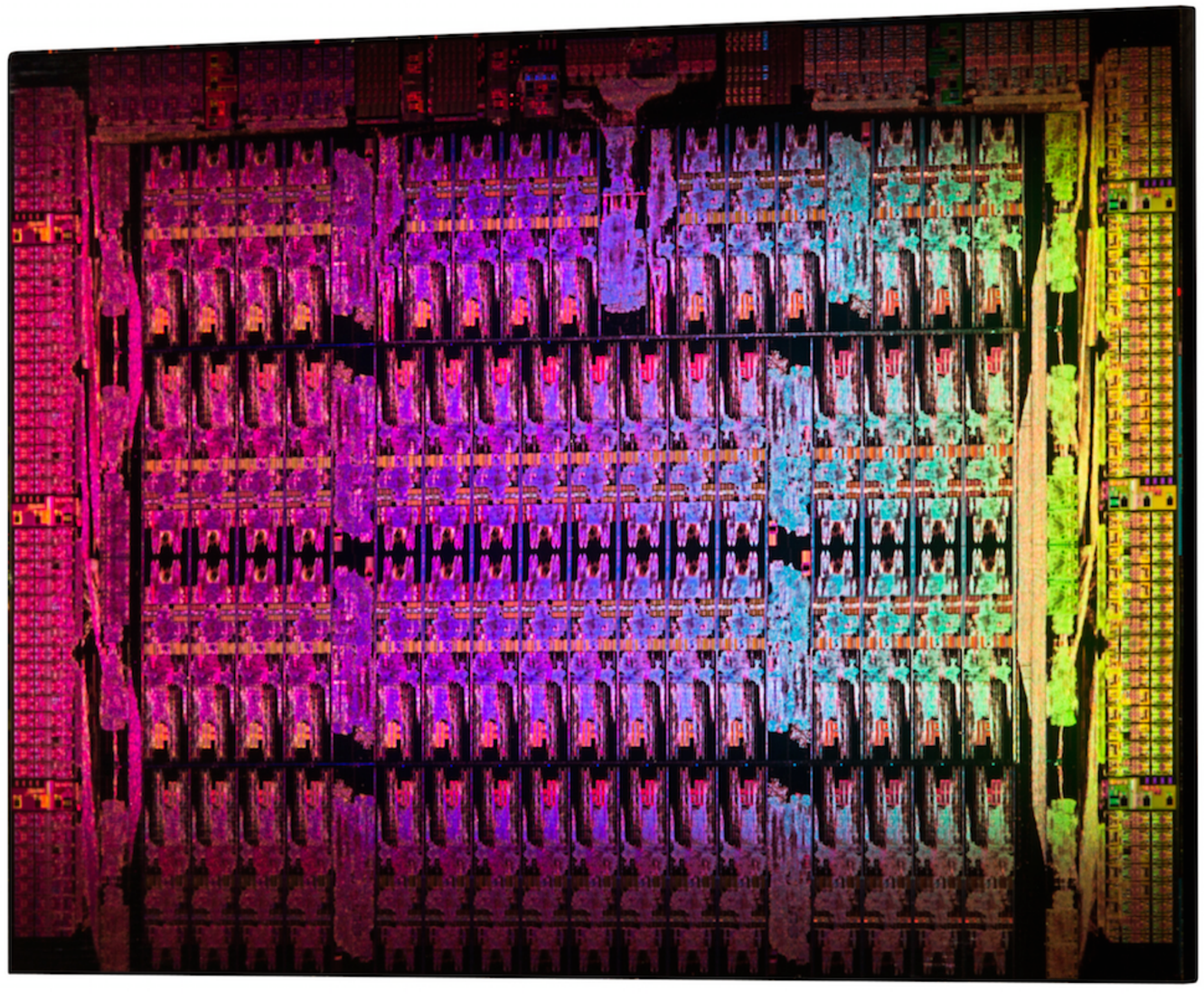
- Estimated machine cost $97M
- Estimated annual power/operating cost: ~ $9M *

[Source: NPR]

Intel Xeon Phi

- 61 "simple" x86 cores (1.1 Ghz, derived from Pentium)
- 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 2013 rankings

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT **two types of CPU**	3120000	33862.7	54902.4	17808 **54 PFLOPS, 17.8 MWatt**
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc. **GPU**	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc. **GPU**	115984	6271.0	7788.9	2325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301

Green500: most energy efficient

Source: Green500 Fall 2013 rankings

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	4,503.17	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.100GHz, Infiniband FDR, NVIDIA K20x	27.78
2	3,631.86	Cambridge University	Wilkes - Dell T620 Cluster, Intel Xeon E5-2630v2 6C 2.600GHz, Infiniband FDR, NVIDIA K20	52.62
3	3,517.84	Center for Computational Sciences, University of Tsukuba	HA-PACS TCA - Cray 3623G4-SM Cluster, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband QDR, NVIDIA K20x	78.77
4	3,185.91	Swiss National Supercomputing Centre (CSCS)	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Level 3 measurement data available	1,753.66
5	3,130.95	ROMEO HPC Center - Champagne-Ardenne	romeo - Bull R421-E3 Cluster, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR, NVIDIA K20x	81.41
6	3,068.71	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.5 - Cluster Platform SL390s G7, Xeon X5670 6C 2.930GHz, Infiniband QDR, NVIDIA K20x	922.54
7	2,702.16	University of Arizona	iDataPlex DX360M4, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR14, NVIDIA K20x	53.62
8	2,629.10	Max-Planck-Gesellschaft MPI/IPP	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband, NVIDIA K20x	269.94
9	2,629.10	Financial Institution	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband, NVIDIA K20x	55.62
10	2,358.69	CSIRO	CSIRO GPU Cluster - Nitro G16 3GPU, Xeon E5-2650 8C 2.000GHz, Infiniband FDR, Nvidia K20m	71.01

Energy-constrained computing

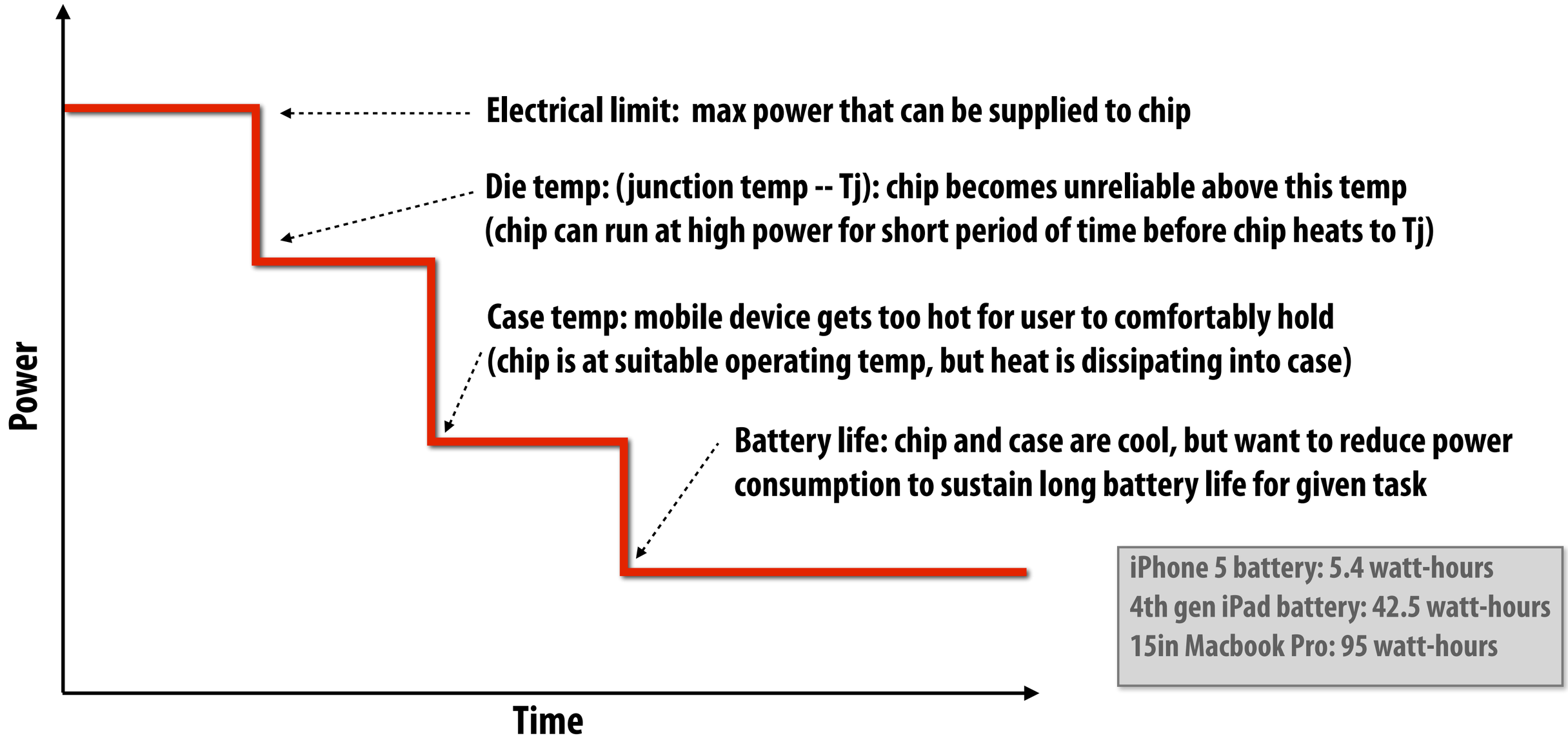
- Supercomputers are energy constrained
  - Due to shear scale
  - Overall cost to operate (power for machine and for cooling)

- Datacenters are energy constrained
  - Reduce cost of cooling
  - Reduce physical space requirements

- Mobile devices are energy constrained
  - Limited battery life
  - Heat dissipation

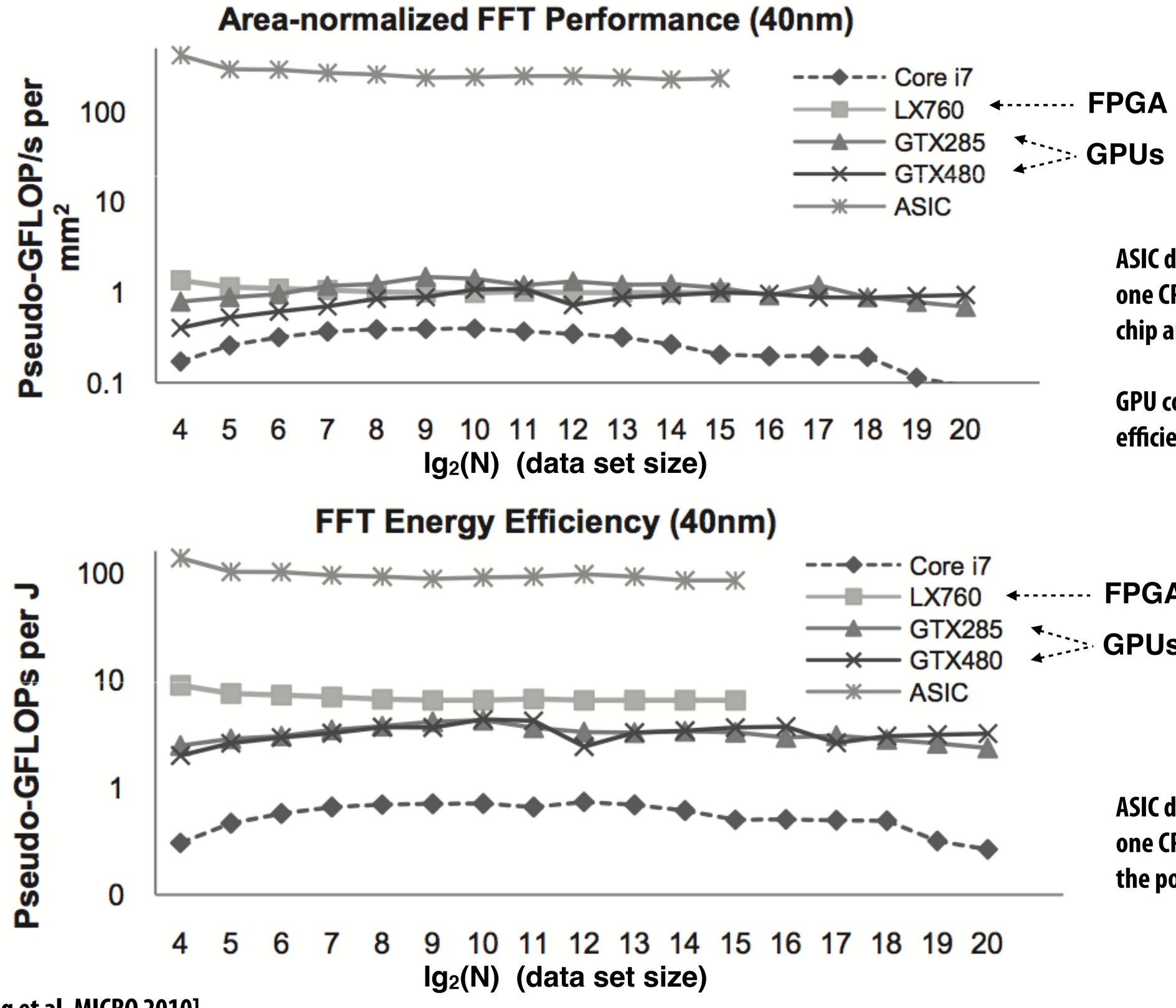
Limits on chip power consumption

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



Slide credit: adopted from original slide from M. Shebanow

Hardware specialization increases



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 power.

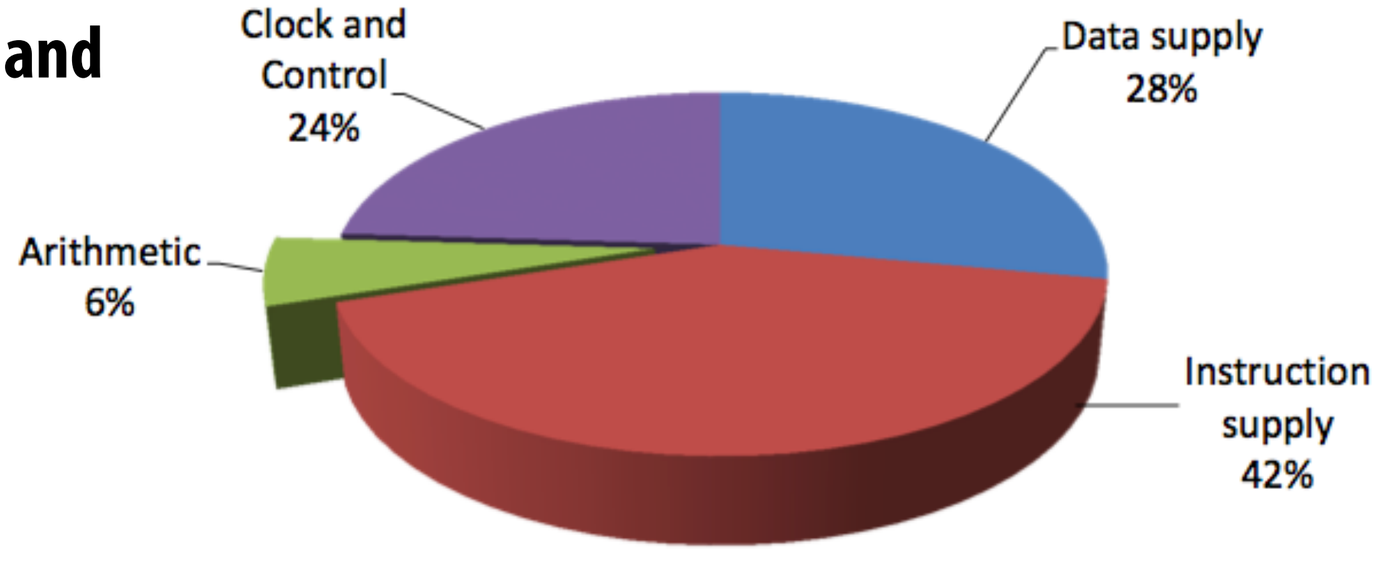
[Chung et al. MICRO 2010]

Efficiency benefits of compute specialization

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

- Throughput-maximized processor architectures: e.g., GPU cores
  - ~ 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")
  - ~ 100x or greater improvement in perf/watt
  - Assuming code is compute bound and and is not floating-point math



Efficient Embedded Computing [Dally et al. 08]

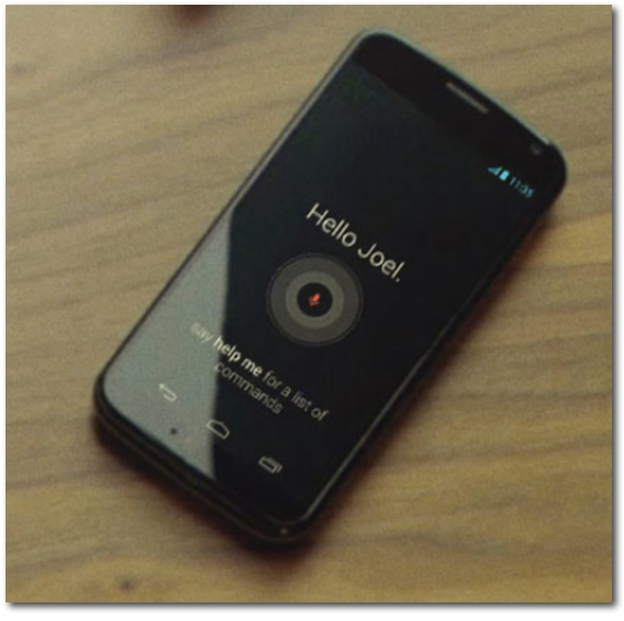
[Source: Chung et al. 2010 , Dally 08]

[Figure credit Eric Chung]

Benefit 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

iPhone 5:
Siri activated by button press or holding phone up to ear



Moto X:
Always listening for "ok, google now"

Device contains special ASIC for detecting this audio pattern.

Example: iPad Air (2013)

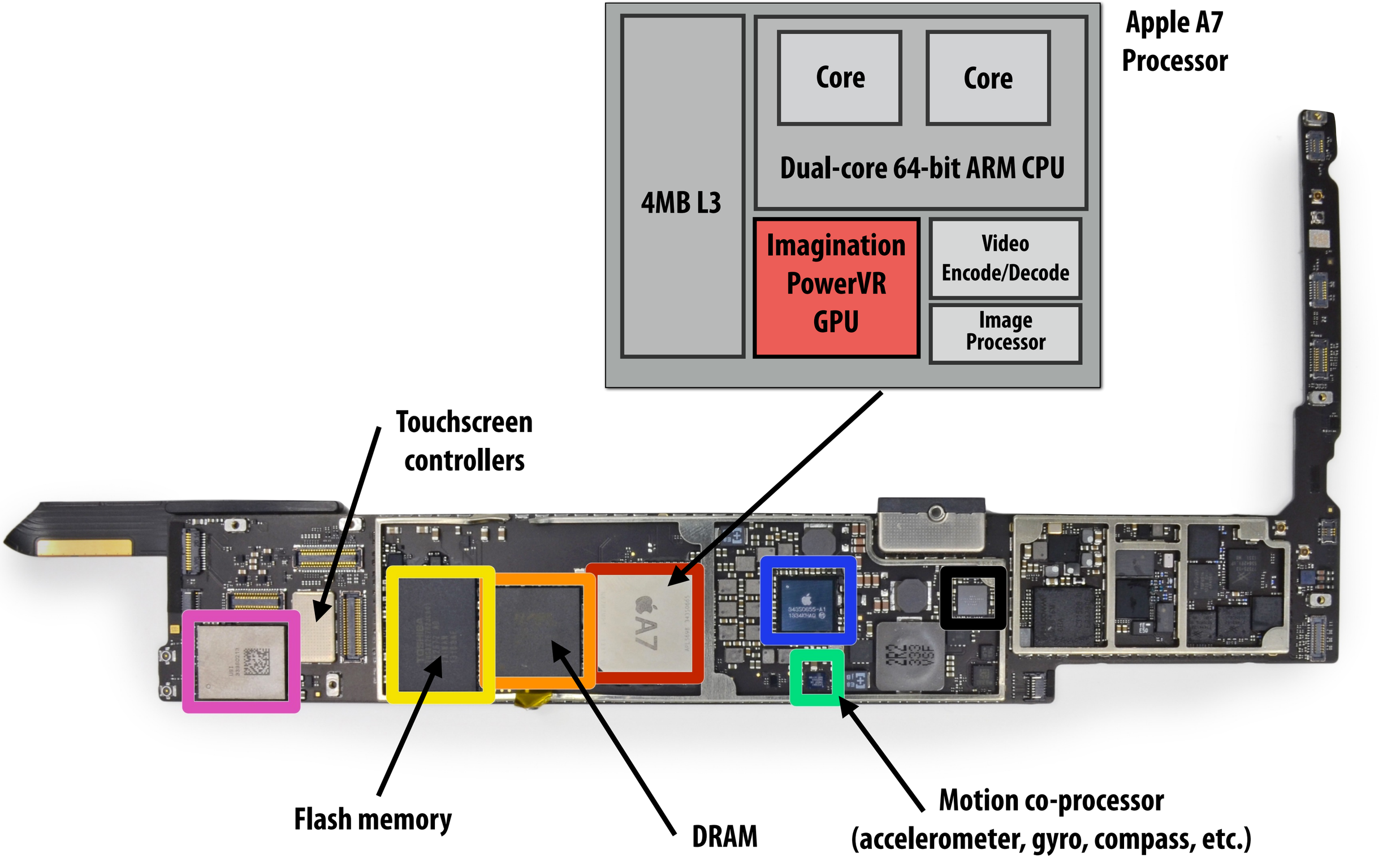


Image Credit: ifixit.com

Original iPhone touchscreen controller

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

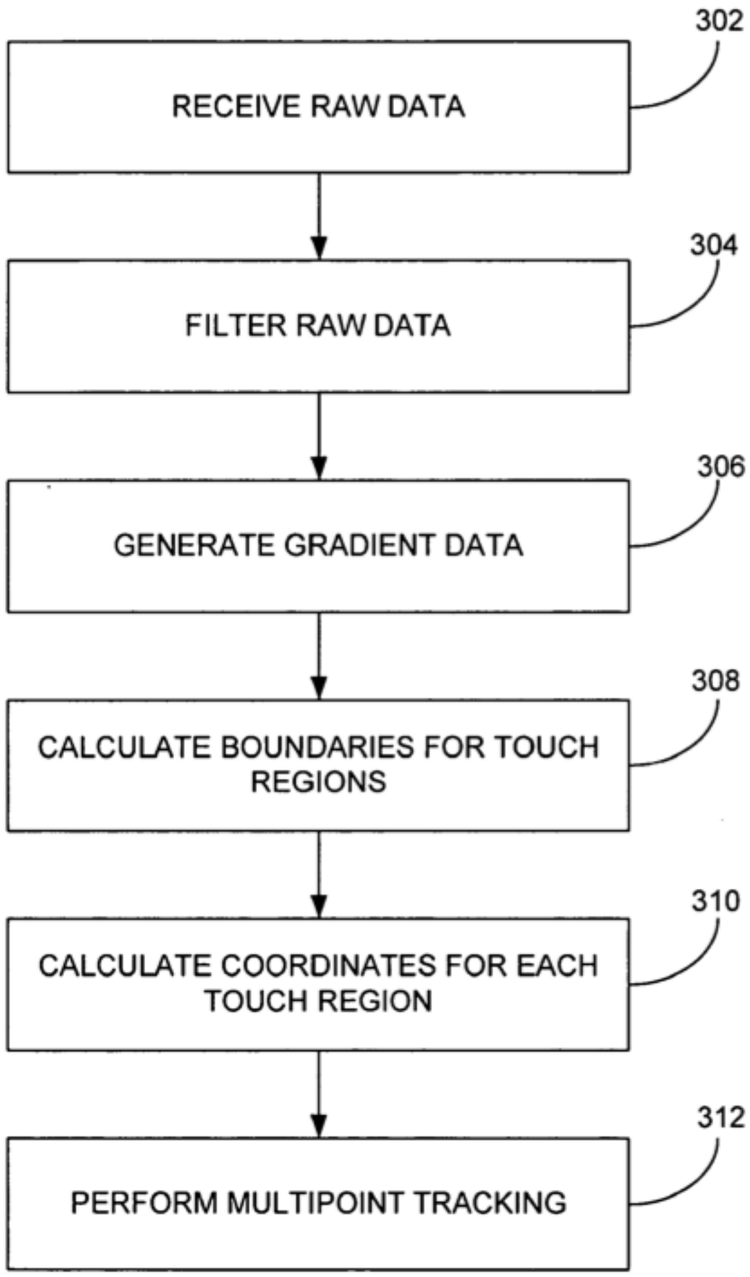


FIG. 16

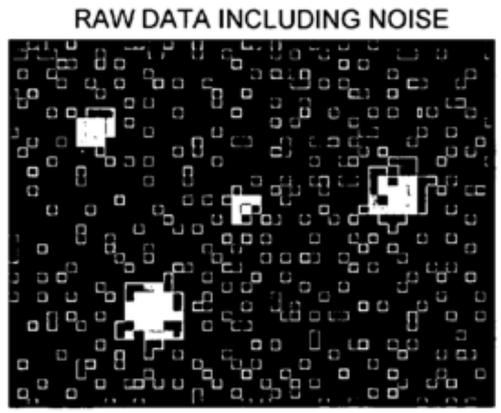


FIG. 17A

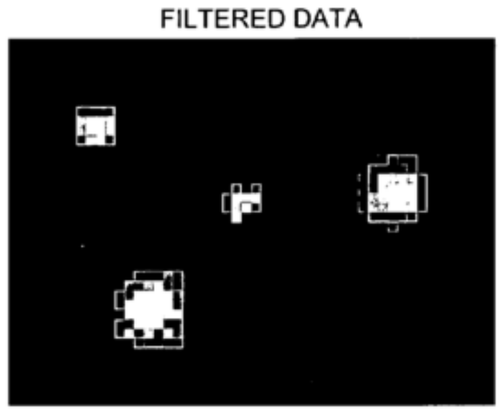


FIG. 17B

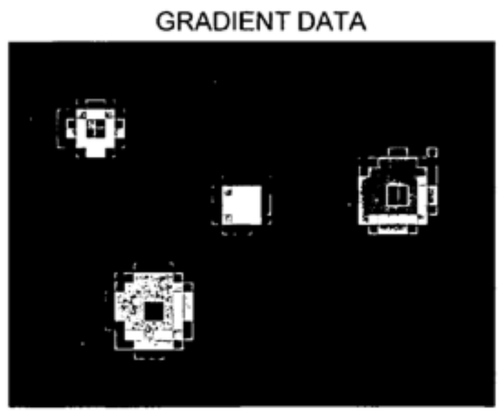


FIG. 17C

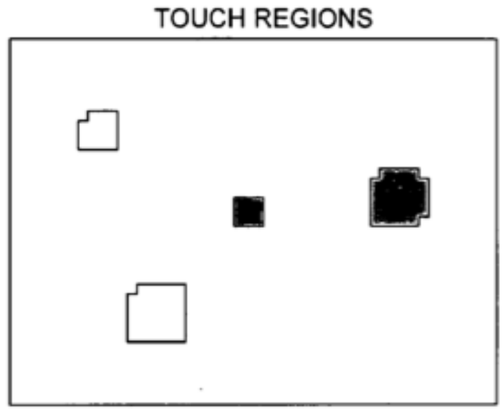


FIG. 17D

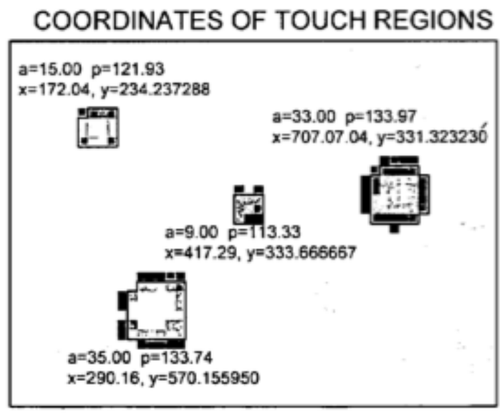


FIG. 17E

From US Patent Application 2006/0097991

Today: performance often matters more, 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/

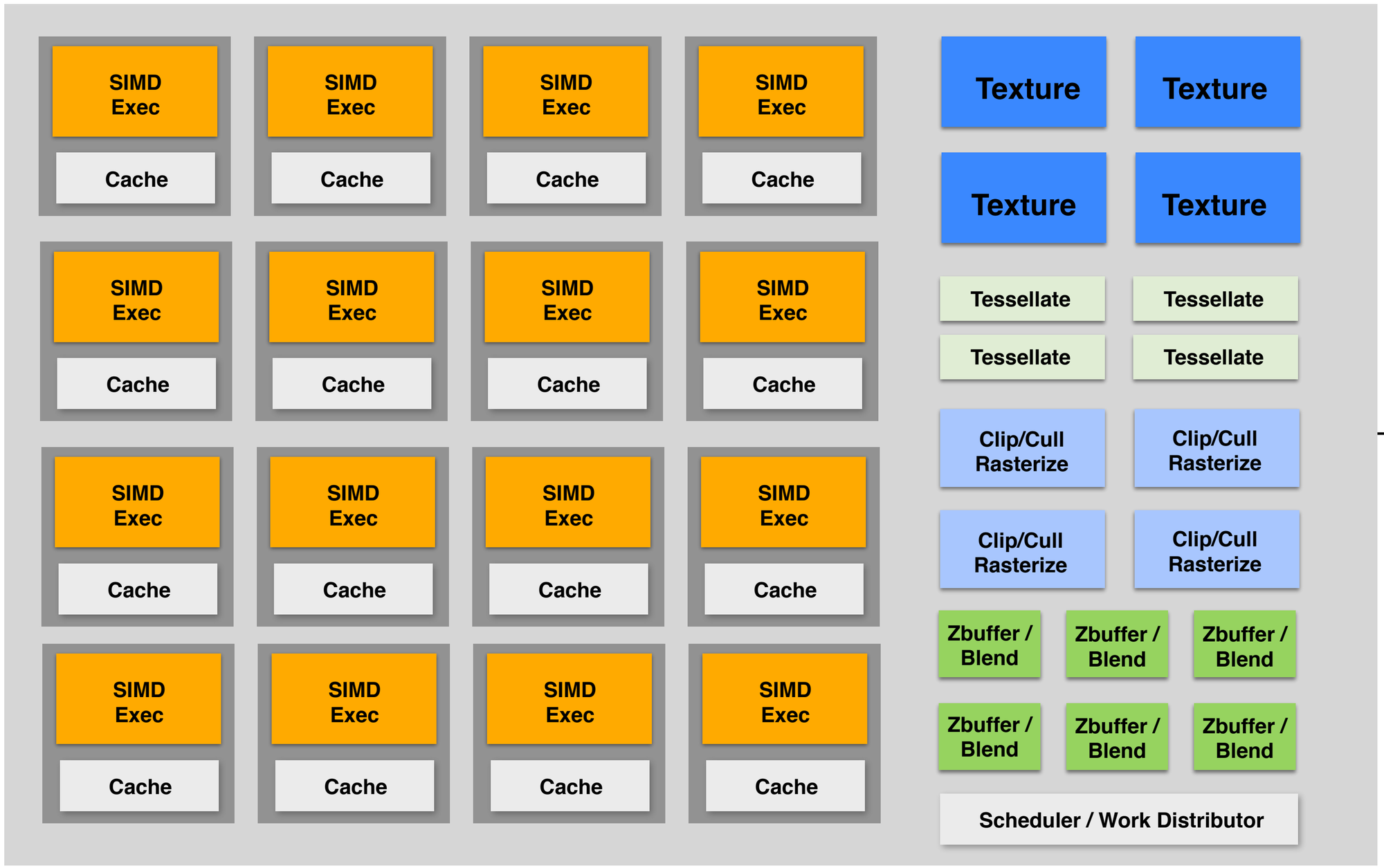
Example: image processing on Nikon D7000

16 MPixel RAW image to JPG image conversion:
Adobe Lightroom on my quad-core Macbook Pro laptop: 1-2 sec
Camera: ~ 1/6 sec

GPU is itself a heterogeneous multi-core processor

Compute resources your CUDA programs used in assignment 2

Graphics-specific, fixed-function compute resources

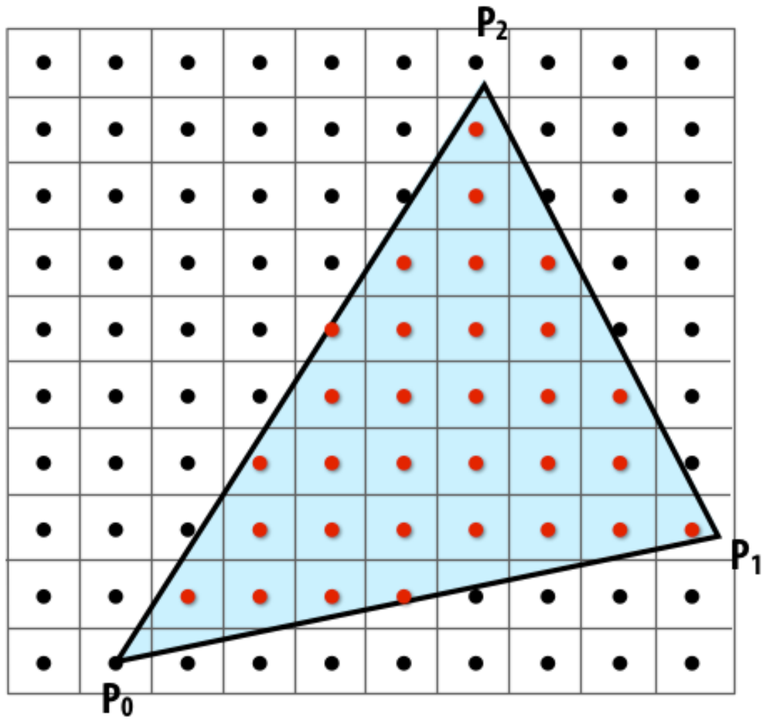


GPU

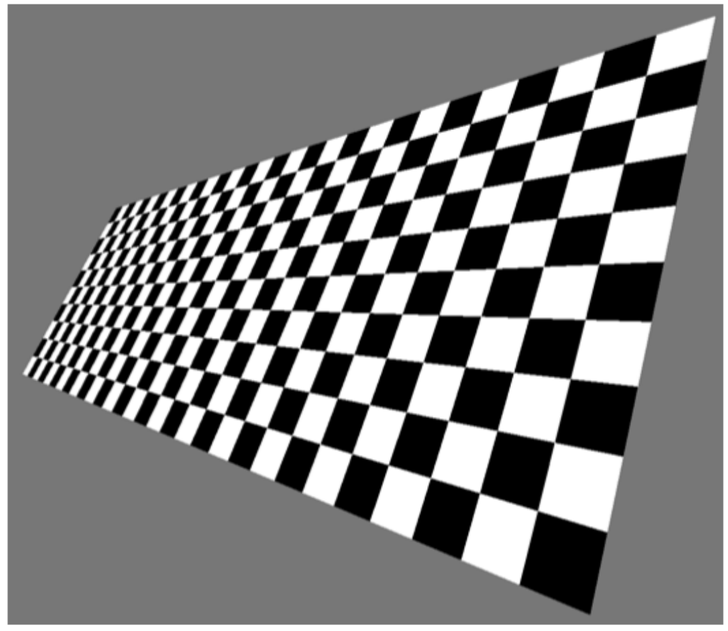
GPU Memory

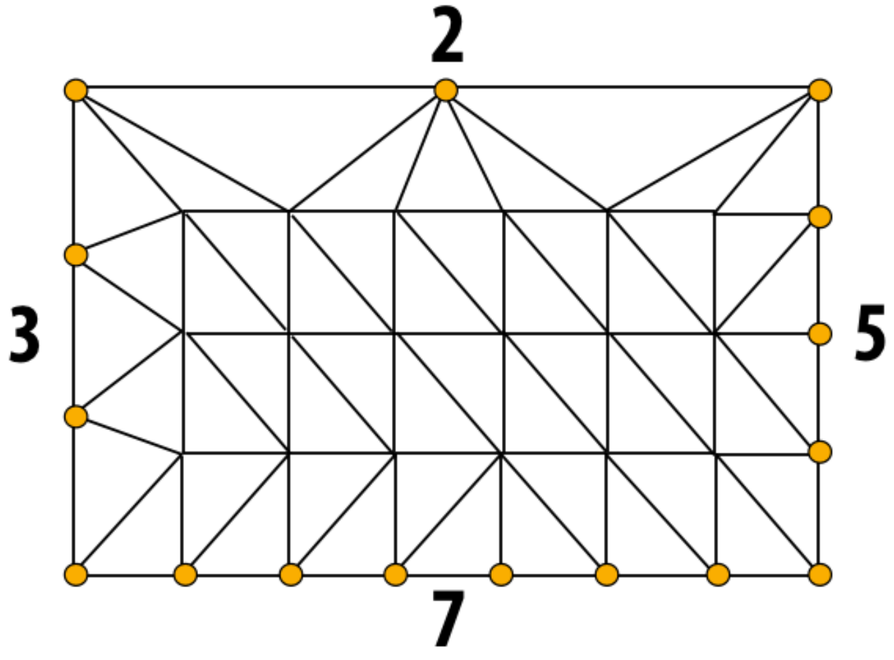
Example graphics tasks performed in fixed-function HW

Rasterization:
Determining what pixels a triangle overlaps



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

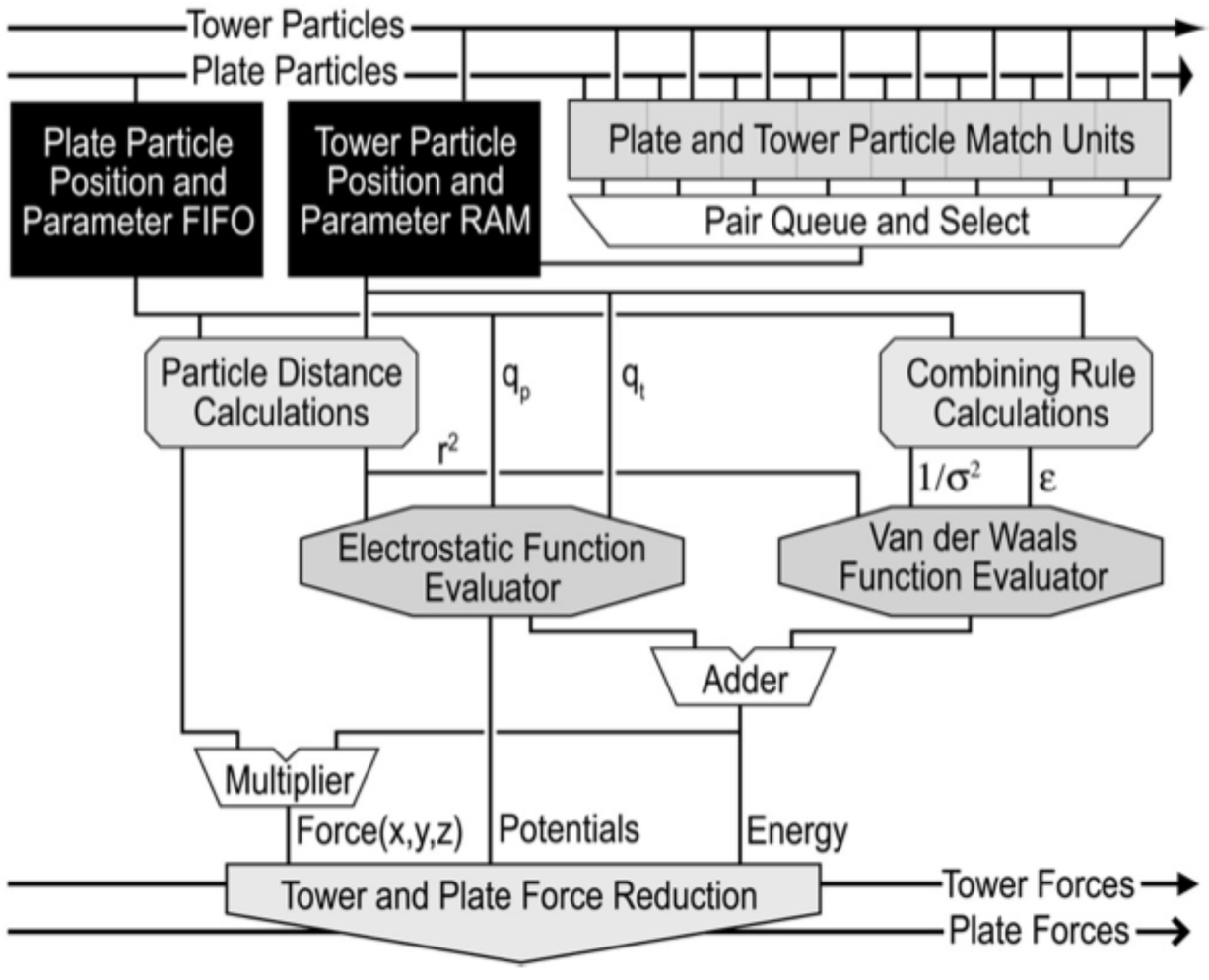




Geometric tessellation:
computing fine-scale geometry from coarse geometry

DESRES Anton supercomputer

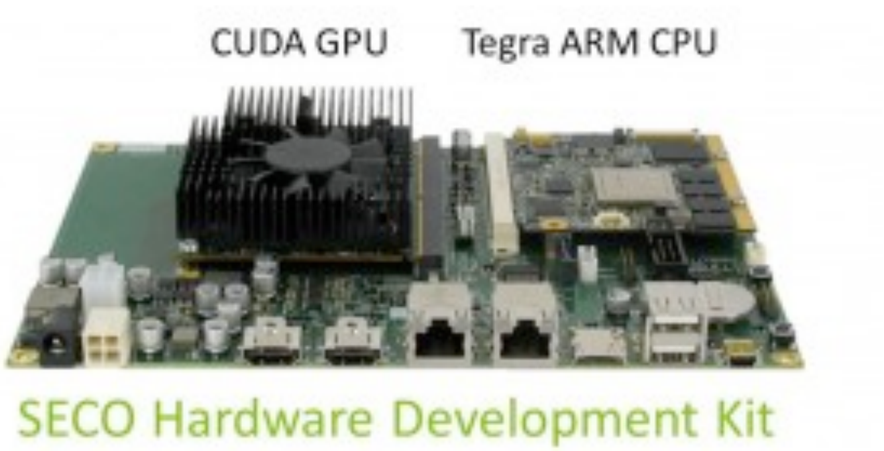
- Supercomputer highly specialized for molecular dynamics
  - Simulate time evolution of proteins
- ASIC for computing particle-particle interactions (512 of them in machine)
- Throughput-oriented subsystem for efficient fast-fourier transforms
- Custom, low-latency communication network designed for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



ARM + GPU Supercomputer

- Observation: the heavy lifting in supercomputing applications is the data-parallel part of workload
  - Less need for "beefy" sequential performance cores
- Idea: build supercomputer out of power-efficient building blocks
  - ARM CPUs (for control/scheduling) + GPU cores (primary compute engine)
- Goal: 7 GFLOPS/Watt efficiency
- Project underway at Barcelona Supercomputing Center

  http://www.montblanc-project.eu



Challenges of heterogeneity

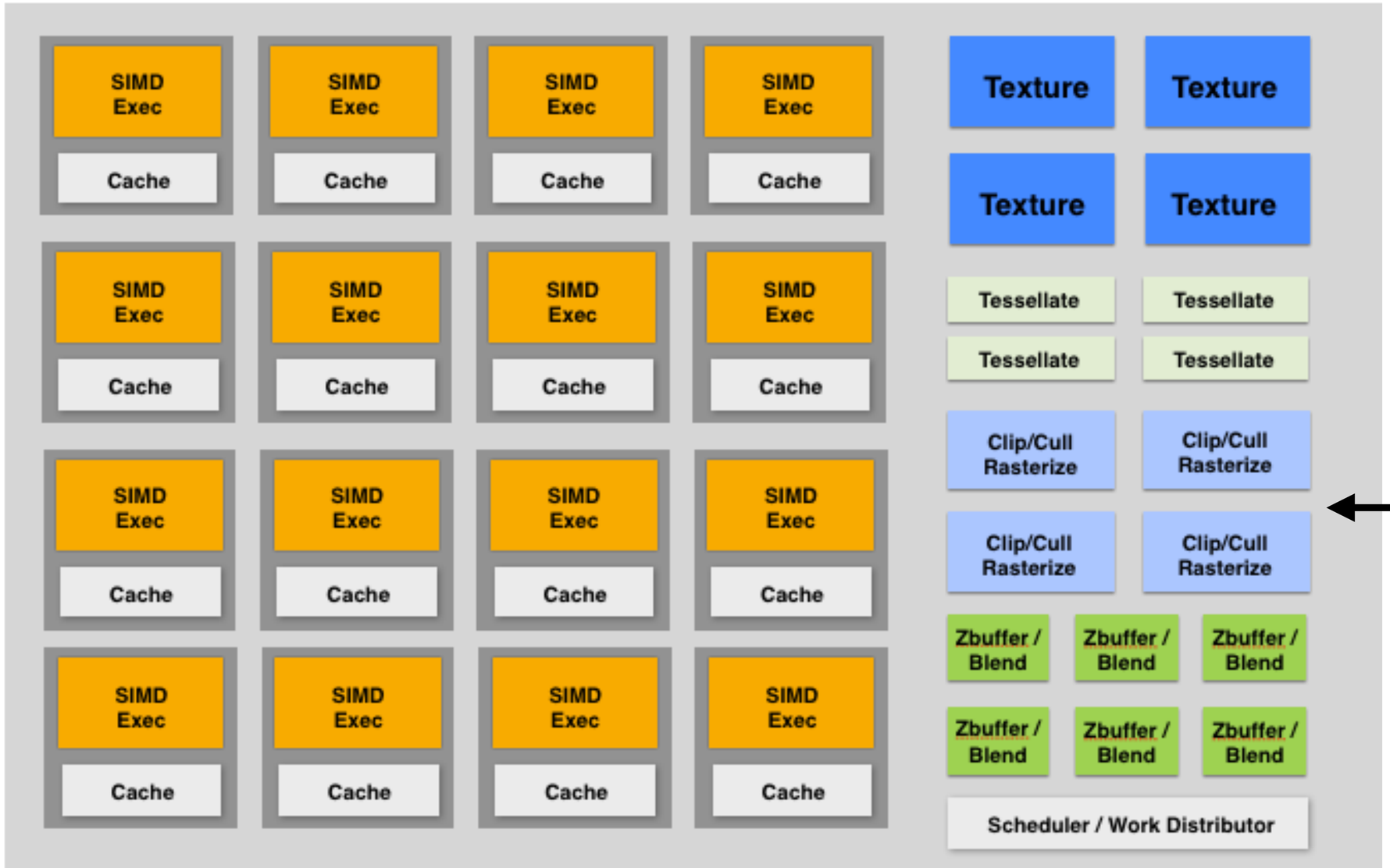
- So far in this course:
  - Homogeneous system: every processor can be used for every task
  - Goal: to get best speedup vs. sequential execution, keep all processors busy all the time

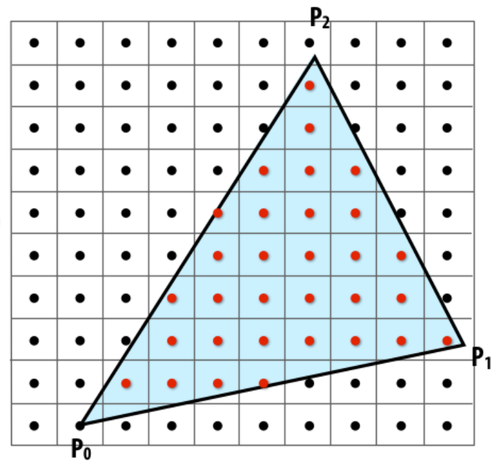
- Heterogeneous system: use preferred processor for each task
  - Challenge for system designer: what is the right mixture of resources to meet performance, cost, and energy goals?
    - Too few throughput-oriented resources (lower peak performance/efficiency for parallel workloads -- should have used 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? (these resources are taken away from general-purpose processing)

- Increased pressure to understand workloads accurately at chip design time

Pitfalls of heterogeneous design

[Molnar 2010]





Say 10% of the computation is rasterization (most of graphics workload is computing color of pixels)
Let's say you under-provision the fixed-function rasterization unit on GPU:
(e.g., 1% of chip area used for rasterizer, really needed 1.2% of chip)

Problem: rasterization is bottleneck, so the expensive programmable processors (99% of chip) are idle waiting on rasterization. So the other 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 system 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: design of new algorithms that decompose well into components that each map well to different processing components of the machine
  - Scheduling problem is more complex on a heterogeneous system
  - Available mixture of resources can dictate choice of algorithm
  - Software portability nightmare

Data movement has high energy cost

- Rule of thumb in mobile system design: 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 5 battery: ~5.5 watt-hours (note: my Macbook Pro laptop: 77 watt-hour battery)
  - Exploiting locality matters!!!

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

Trends in energy-focused computing

- **Compute less!**
  - Computing more costs energy: parallel algorithms that do more work than sequential counterparts may not be desirable even if they run faster
- **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)
- **Specialize compute units:**
  - Heterogeneous programmable processors: CPU-like cores + throughput-optimized cores (GPU-like cores)
  - Fixed-function units: video decode/encode, image processing, sound processing, computer vision?
  - Specialized instructions: expanding set of AVX vector instructions, new instructions for accelerating AES encryption (AES-NI)
  - Increasing use of programmable logic: FPGAs

Fun reads about mobile power concerns

- **Power concerns in ARM Mali GPUs**
  - http://blogs.arm.com/multimedia/780-how-low-can-you-go-building-low-power-low-bandwidth-arm-mali-gpus/

- **Display Backlight measurements**
  - http://www.displaymate.com/iPad_ShootOut_1.htm#Backlight_Power

Summary

- Heterogeneous 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?