

**Lecture 19:**

# **Heterogeneous Parallelism and Hardware Specialization**

---

**Parallel Computer Architecture and Programming  
CMU 15-418/15-618, Spring 2018**

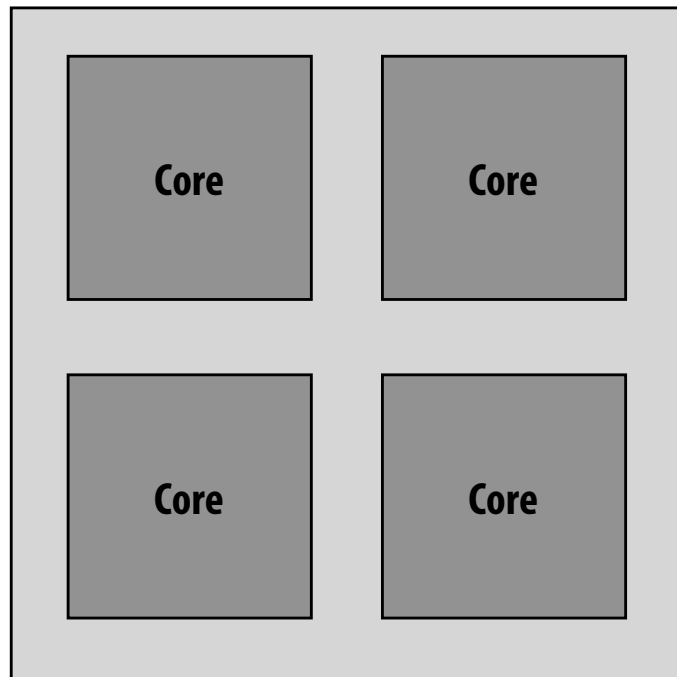
**Let's begin this lecture by reminding you...**

**That we observed in assignment 1 that a well-optimized parallel implementation of a compute-bound application was about 44 times faster on than the output of single-threaded C code compiled with gcc -O3.**



**You need to buy a  
new computer...**

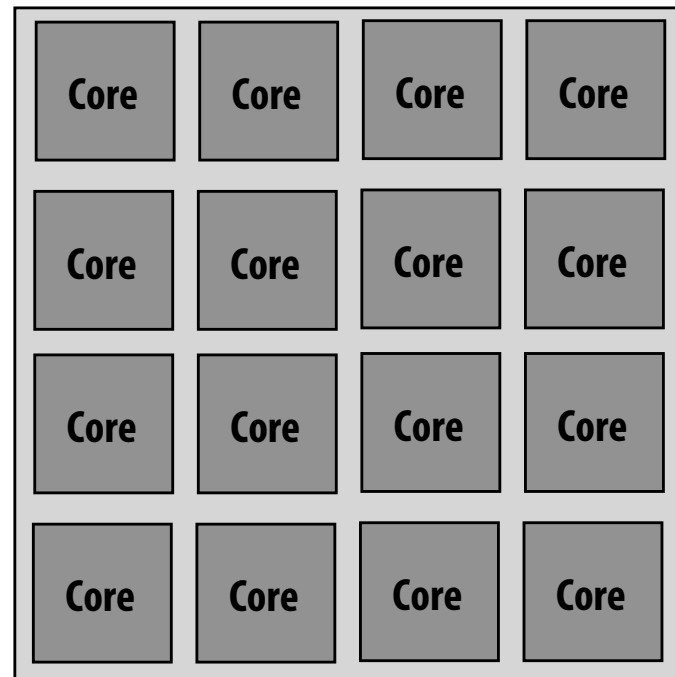
# You need to buy a computer system



**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  $\text{perf}(1) = 1$

$f$  = fraction of program that is parallelizable

$n$  = total processing resources (e.g., transistors on a chip)

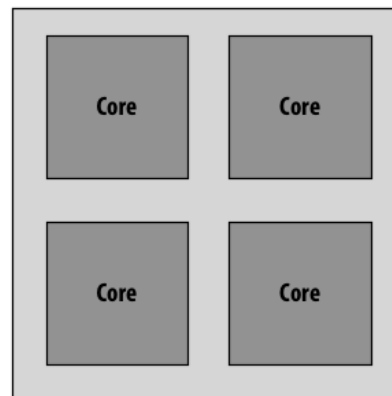
$r$  = resources dedicated to each processing core,  
each of the  $n/r$  cores has sequential performance  $\text{perf}(r)$

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

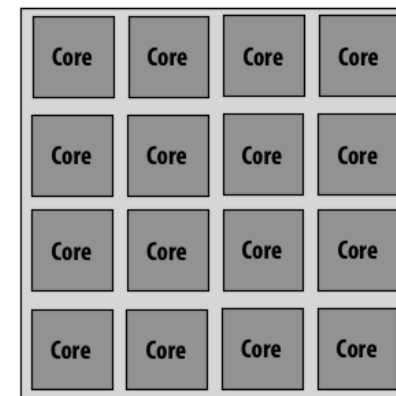
Two examples where  $n=16$

$r_A = 4$

$r_B = 1$

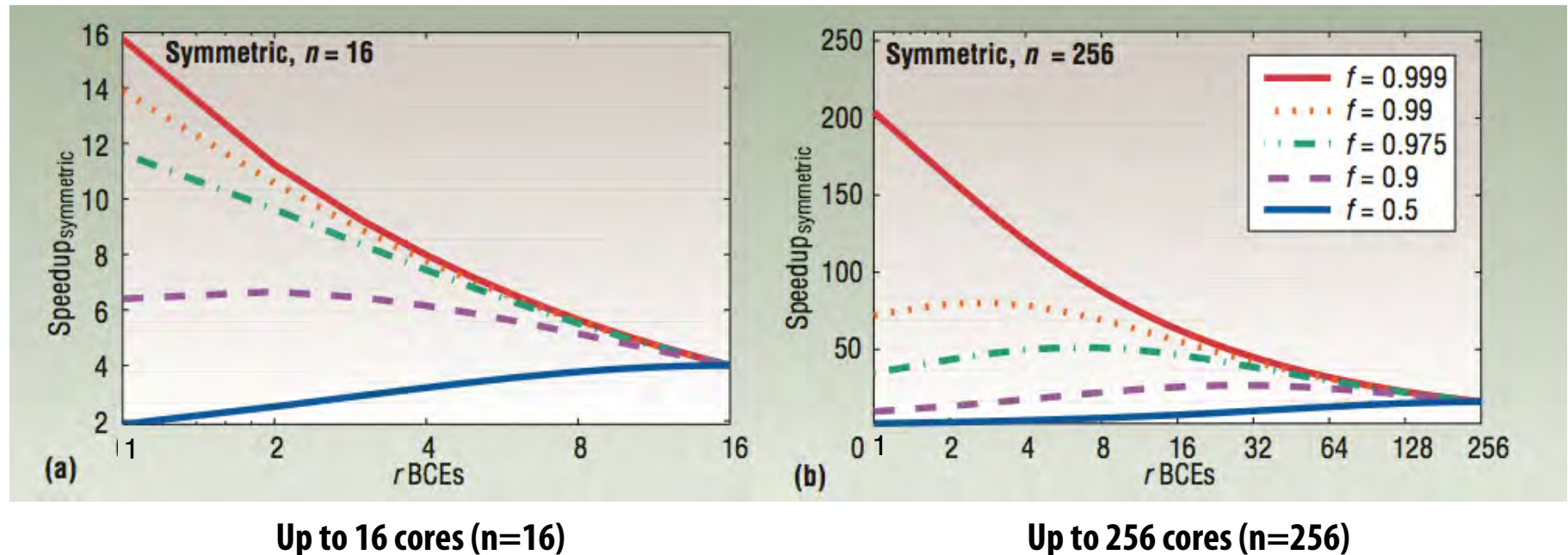


Processor A



Processor B

# Speedup (relative to n=1)



**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 kept the same (constant  $n$  per graph)**

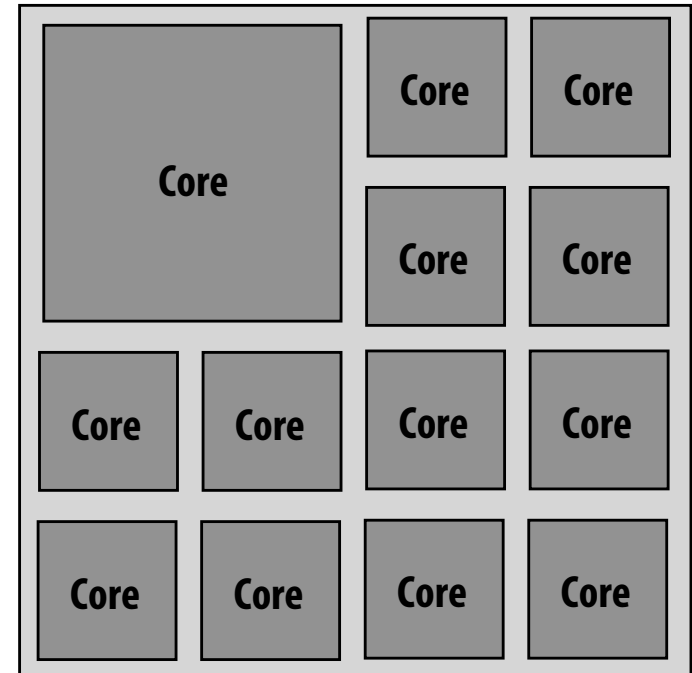
*perf*( $r$ ) modeled as  $\sqrt{r}$

# Asymmetric set of processing cores

Example:  $n=16$

One core:  $r = 4$

Other 12 cores:  $r = 1$



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

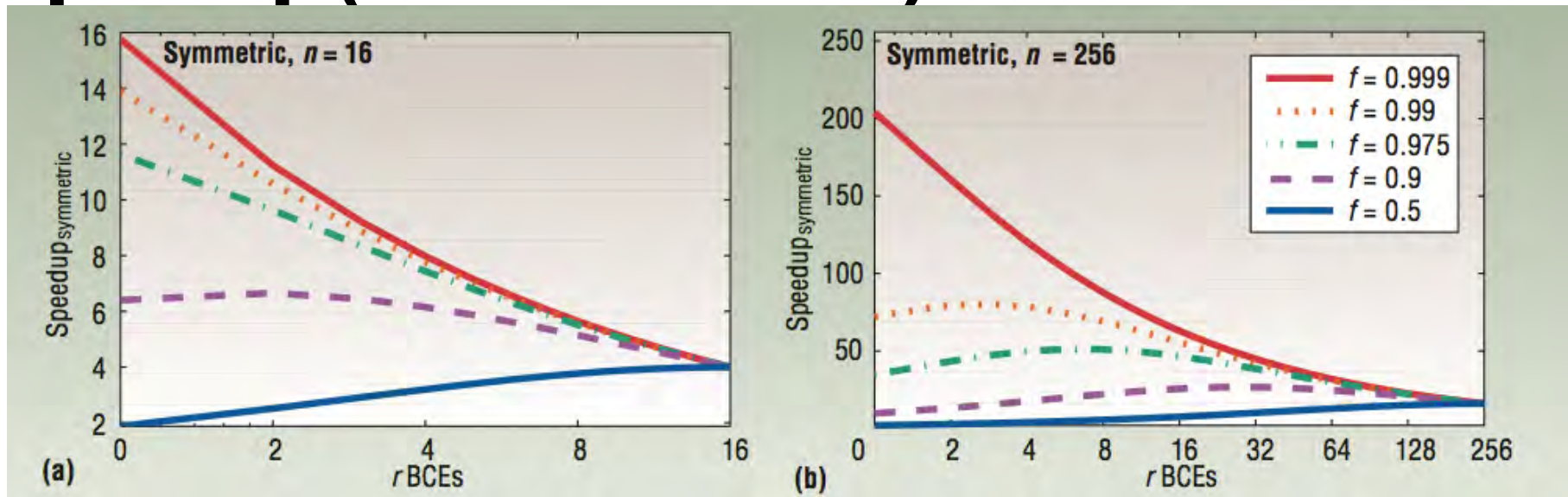
(of heterogeneous processor with  $n$  resources, relative to uniprocessor with one unit worth of resources,  $n=1$ )

$\xleftarrow{\hspace{10em}}$ 
  
 one  $\text{perf}(r)$  processor +  $(n-r)$   $\text{perf}(1)=1$  processors

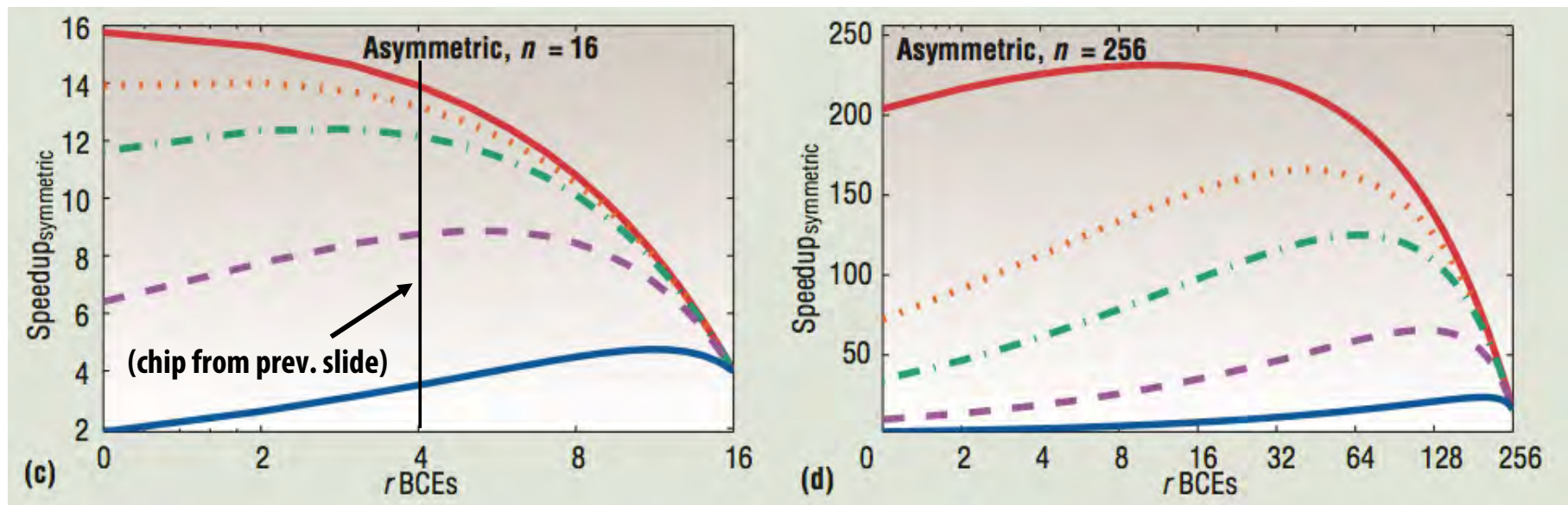


# Speedup (relative to $n=1$ )

[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**

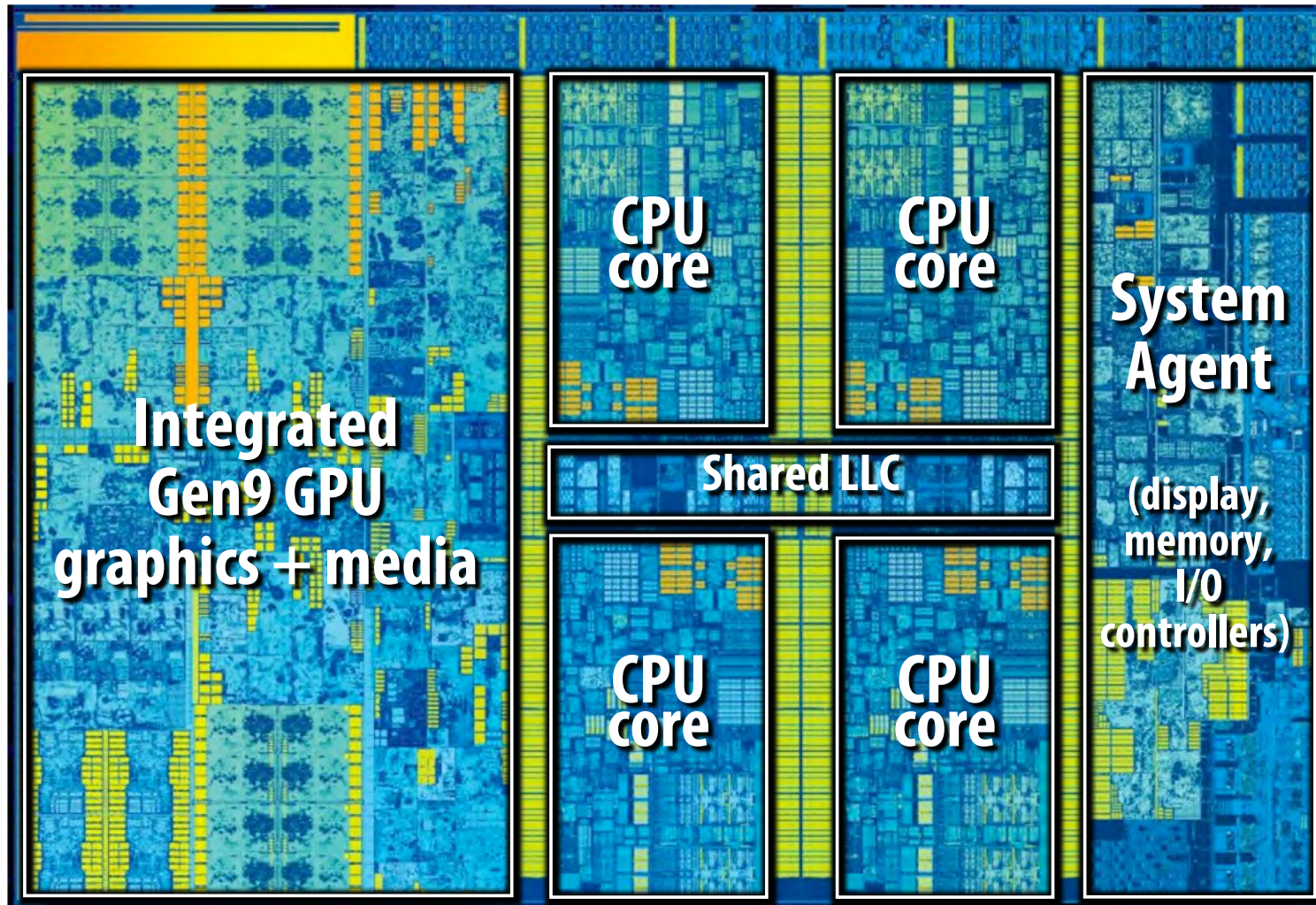
**And components with unpredictable access, but those accesses might cache well.**

**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**

# Example: Intel "Skylake" (2015)

(6th Generation Core i7 architecture)

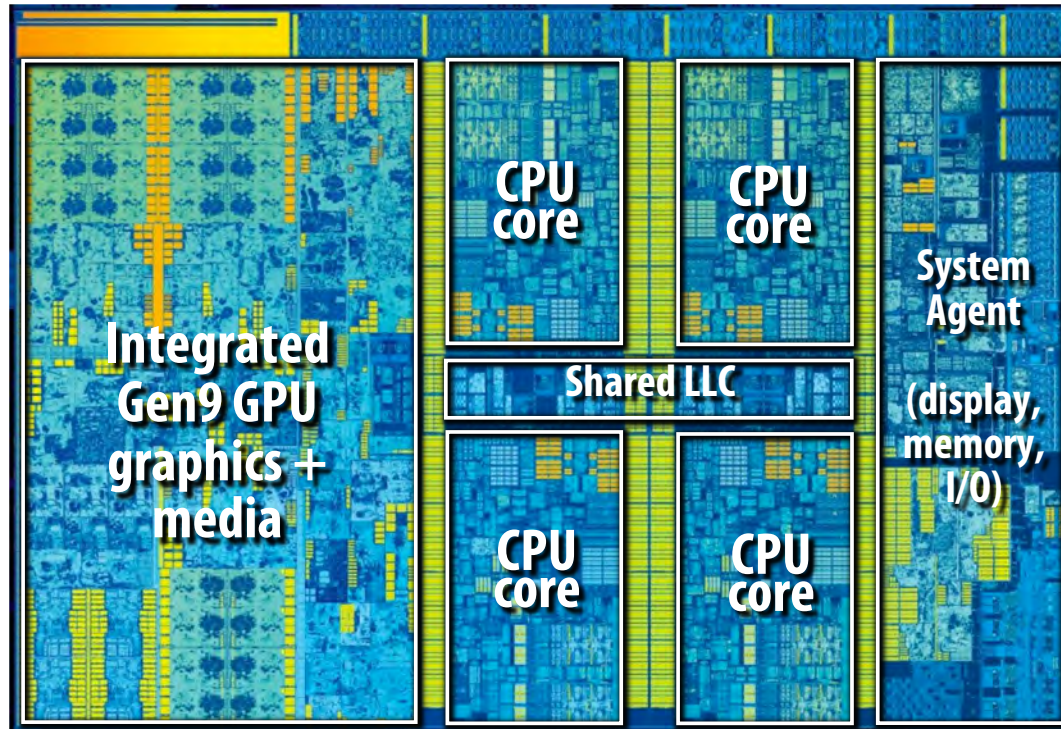


**4 CPU cores + graphics cores + media accelerators**



# Example: Intel "Skylake" (2015)

(6th Generation Core i7 architecture)

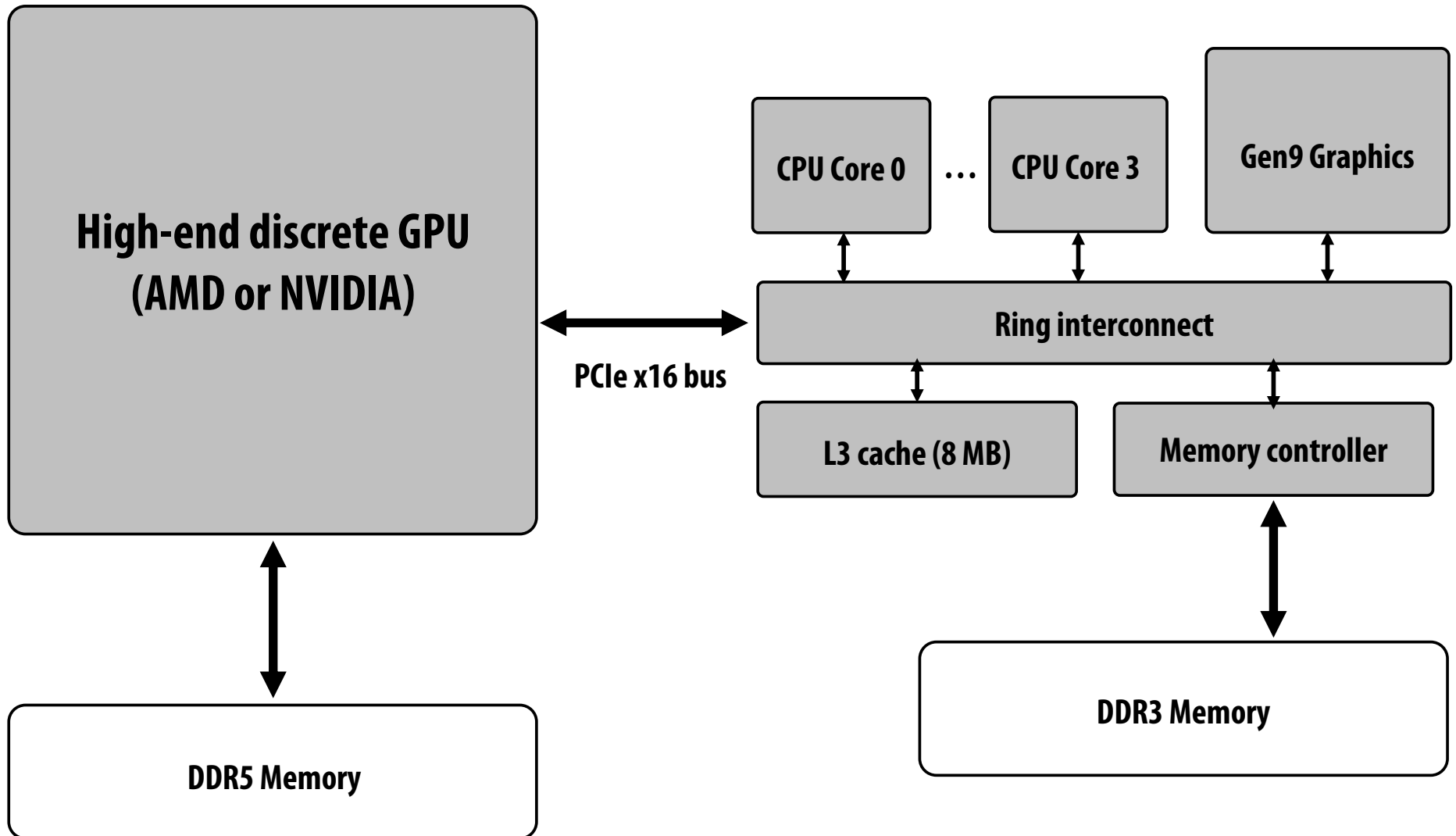


- CPU cores and graphics cores share same memory system
- Also share LLC (L3 cache)
  - Enables, low-latency, high-bandwidth communication between CPU and integrated GPU
- Graphics cores cache coherent with CPU

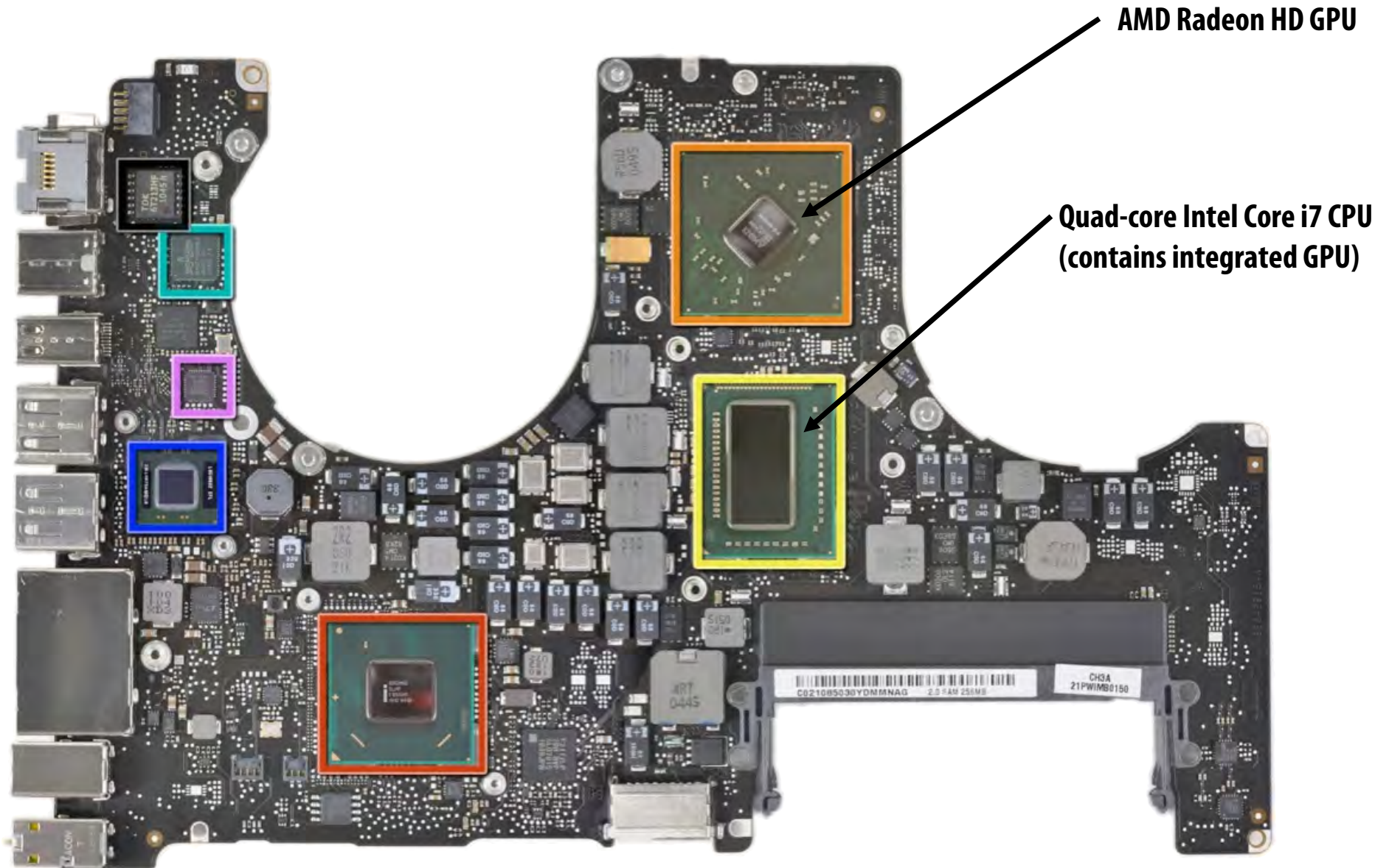
# More heterogeneity: add discrete GPU

Keep discrete (power hungry) GPU unless needed for graphics-intensive applications

Use integrated, low power graphics for basic graphics/window manager/UI

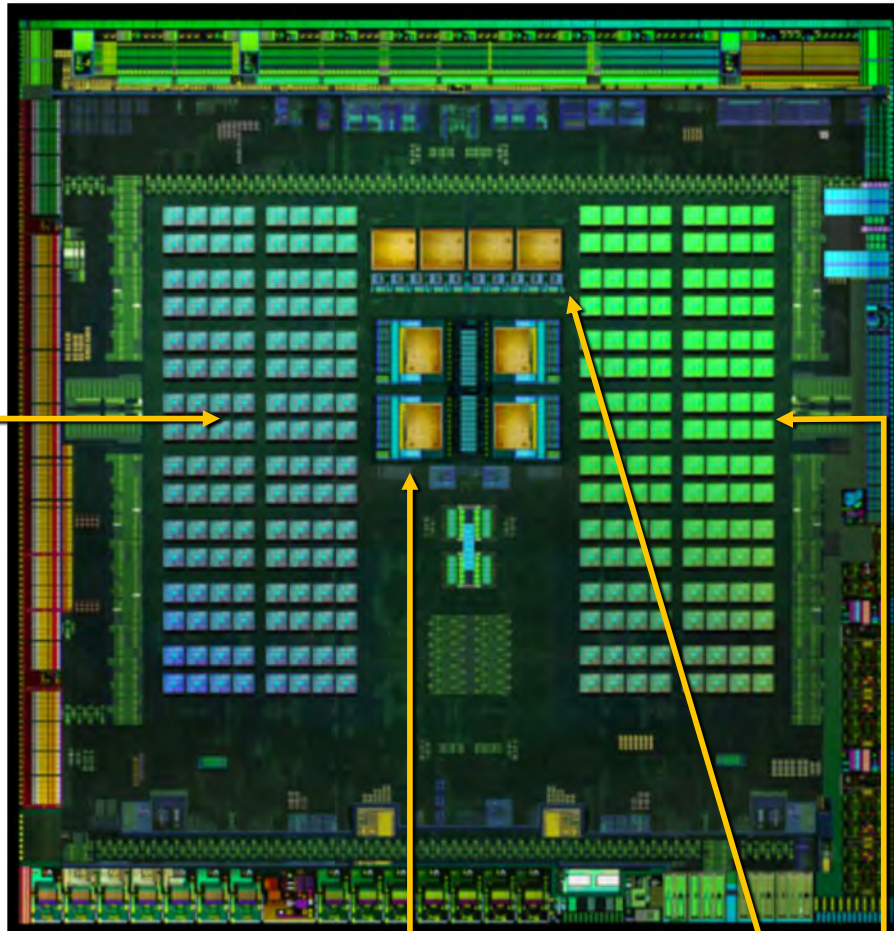


# 15in Macbook Pro 2011 (two GPUs)



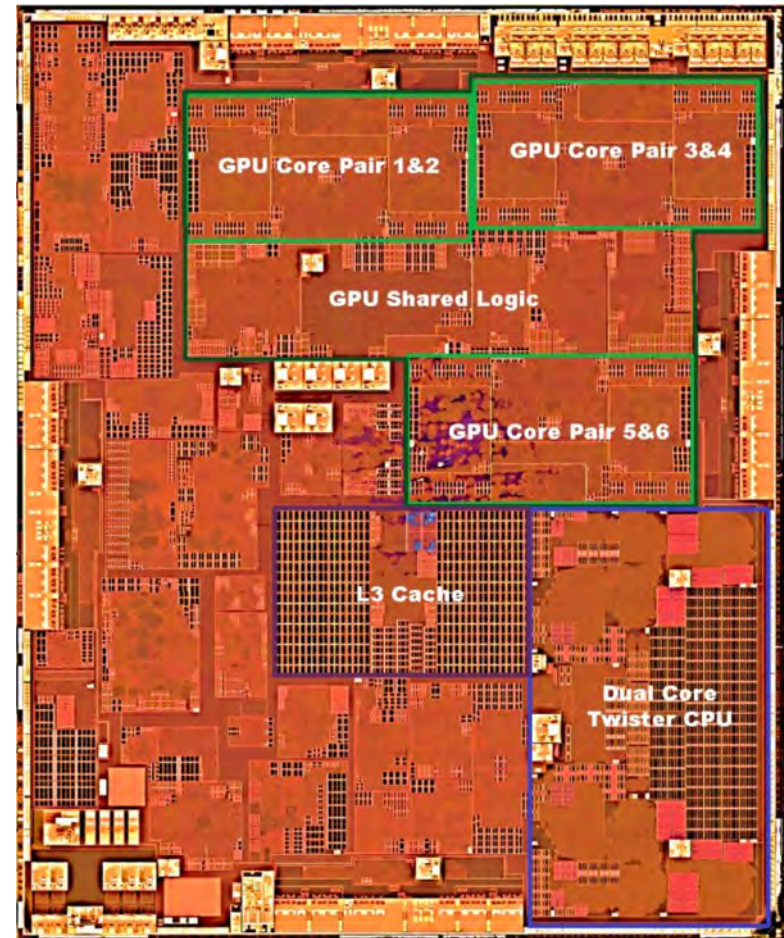


# Mobile heterogeneous processors



## NVIDIA Tegra X1

Four ARM Cortex A57 CPU cores for applications  
Four low performance (low power) ARM A53 CPU cores  
One Maxwell SMM (256 "CUDA" cores)



## Apple A9

Dual Core 64 bit CPU  
GPU PowerVR GT6700 (6 "core") GPU

A9 image credit Chipworks, obtained from

<http://www.anandtech.com/show/9686/the-apple-iphone-6s-and-iphone-6s-plus-review/3>

# 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 #5)
- 18,688 AMD Opteron 16-core CPUs
- 18,688 NVIDIA Tesla K20X GPUs
- 710 TB RAM

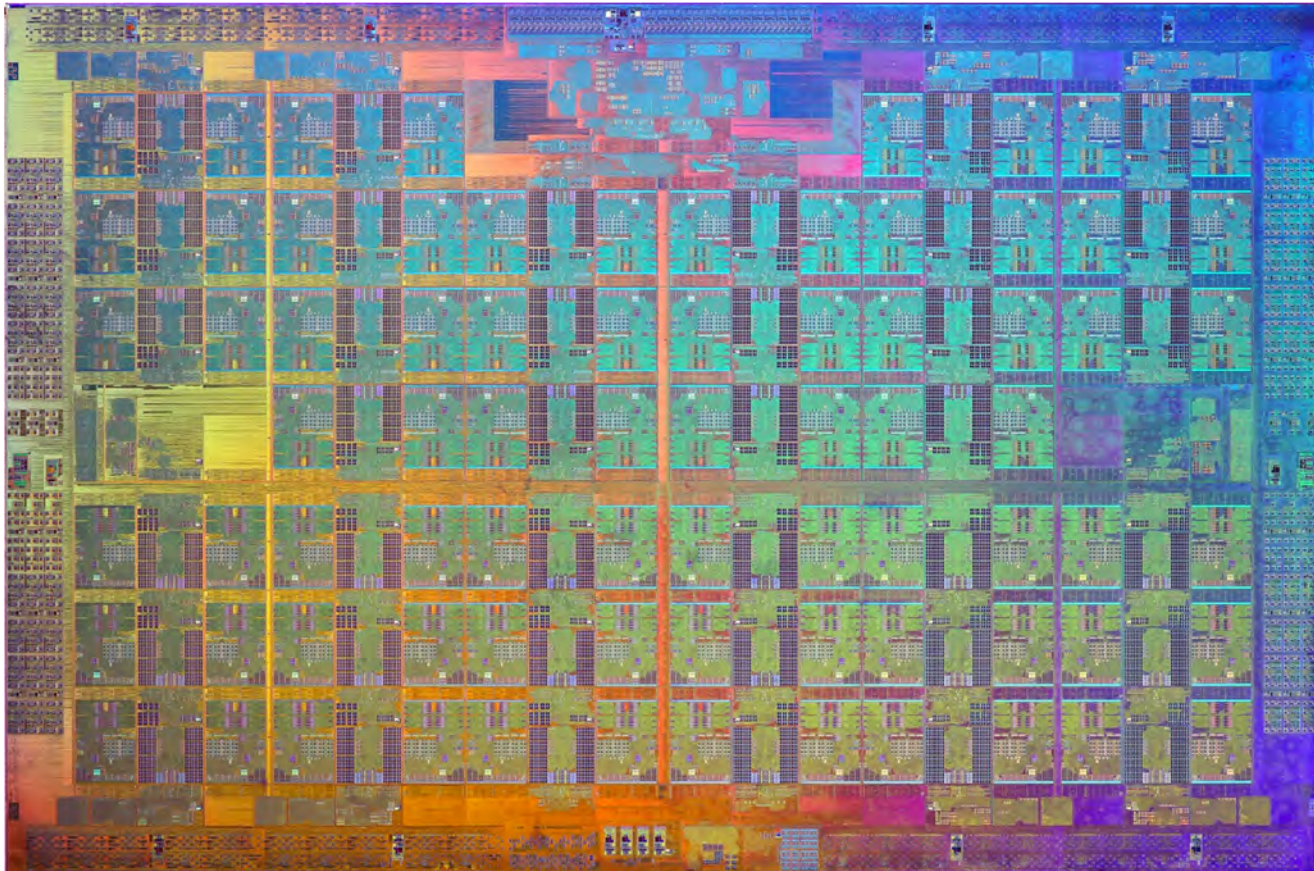


- Estimated machine cost \$97M
- Estimated annual power/operating cost: ~ \$9M \*

\* Source: NPR

# Intel Xeon Phi (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 2017 rankings

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCP	10,649,600	93,014.6	125,435.9	15,371
2	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	3,120,000	33,862.7	54,902.4	17,808
3	Swiss National Supercomputing Centre (CSCS) Switzerland	Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100 Cray Inc.	361,760	19,590.0	25,326.3	2,272
4	Japan Agency for Marine-Earth Science and Technology Japan	Gyokkou - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz ExaScaler	19,860,000	19,135.8	28,192.0	1,350
5	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
6	DOE/NNSA/LLNL United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
7	DOE/NNSA/LANL/SNL United States	Trinity - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	979,968	14,137.3	43,902.6	3,844
8	DOE/SC/LBNL/NERSC United States	Cori - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939

Xeon Phi

NVIDIA GPU

NVIDIA GPU

Xeon Phi

Xeon Phi



# Green500: most energy efficient supercomputers

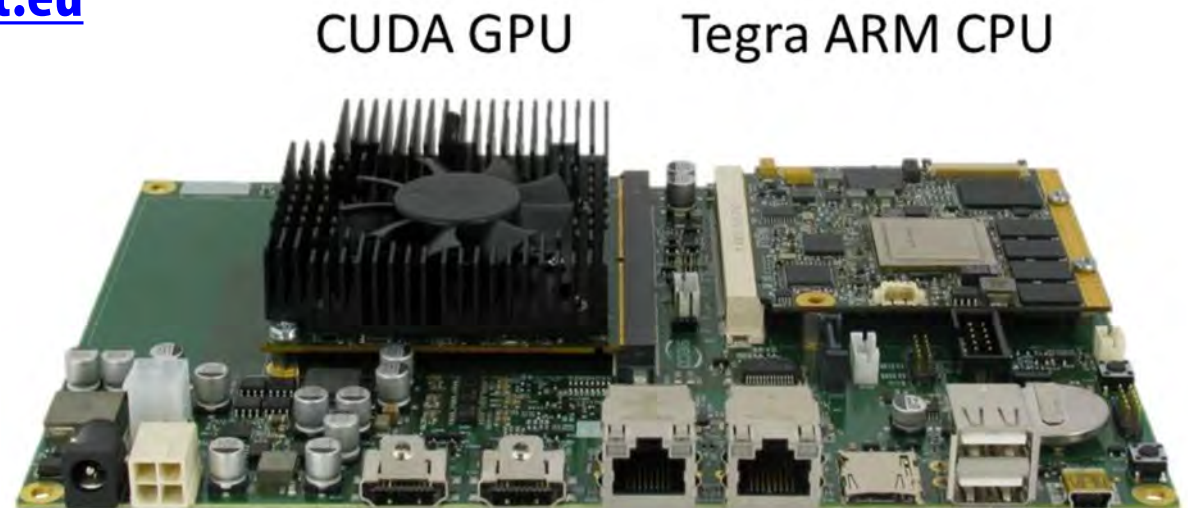
Efficiency metric: MFLOPS per Watt

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	7,031.58	Institute of Physical and Chemical Research (RIKEN)	Shoubu - ExaScaler-1.4 80Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, PEZY-SC	50.32
2	5,331.79	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC/DL - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.1GHz, Infiniband FDR, NVIDIA Tesla K80	51.13
3	5,271.81	GSI Helmholtz Center	ASUS ESC4000 FDR/G2S, Intel Xeon E5-2690v2 10C 3GHz, Infiniband FDR, AMD FirePro S9150	57.15
4	4,778.46	Institute of Modern Physics (IMP), Chinese Academy of Sciences	Sugon Cluster W780I, Xeon E5-2640v3 8C 2.6GHz, Infiniband QDR, NVIDIA Tesla K80	65.00
5	4,112.11	Stanford Research Computing Center	XStream - Cray CS-Storm, Intel Xeon E5-2680v2 10C 2.8GHz, Infiniband FDR, Nvidia K80	190.00
6	3,856.90	IT Company	Inspur TS10000 HPC Server, Xeon E5-2620v3 6C 2.4GHz, 10G Ethernet, NVIDIA Tesla K40	58.00
7	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00
8	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00
9	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00
10	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00

Source: Green500 Fall 2015 rankings

# Research: 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)
- **Project underway at Barcelona Supercomputing Center**
  - [www.montblanc-project.eu](http://www.montblanc-project.eu)



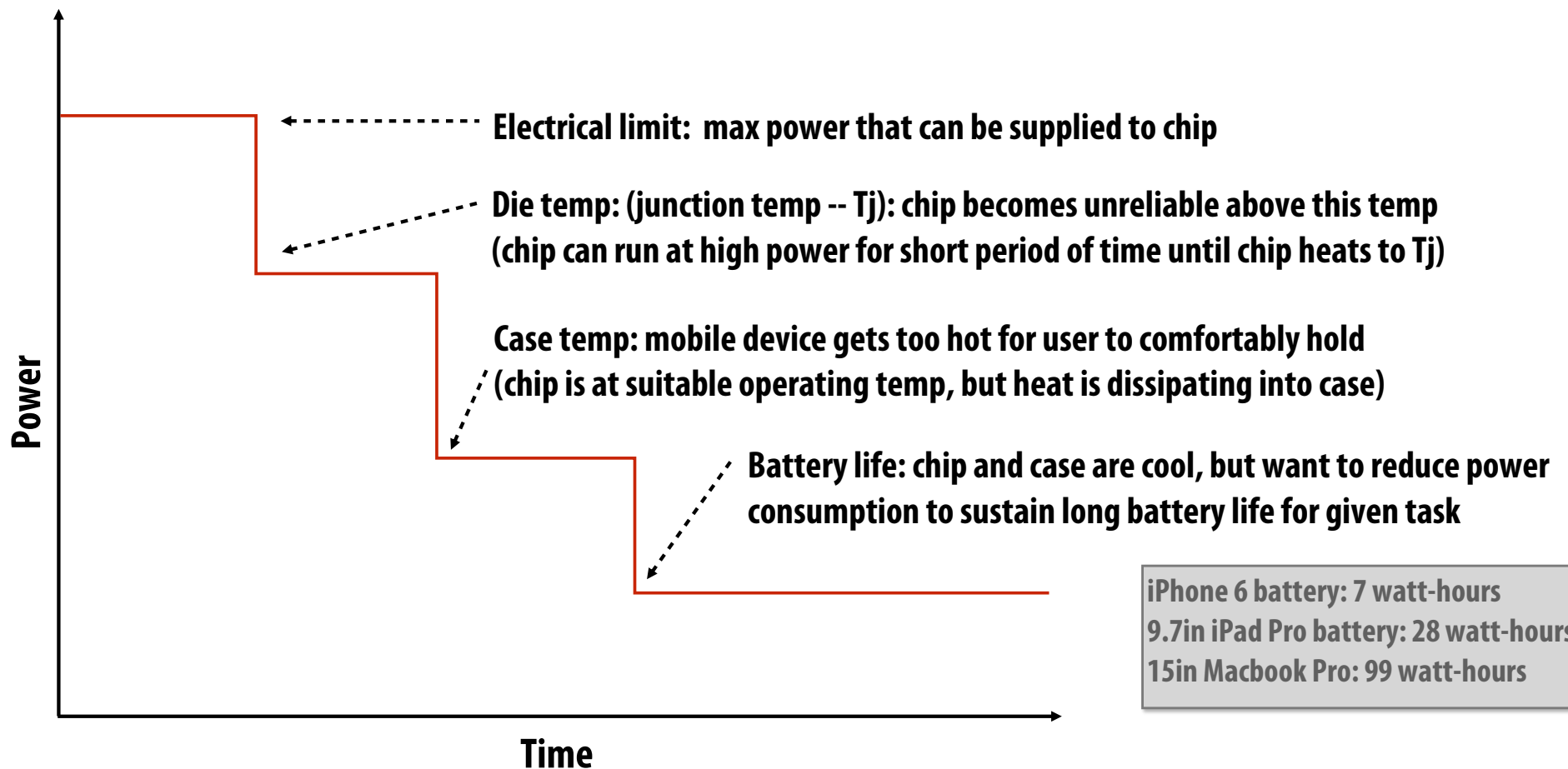
SECO Hardware Development Kit

# Energy-constrained computing

- **Supercomputers are energy constrained**
  - Due to sheer 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

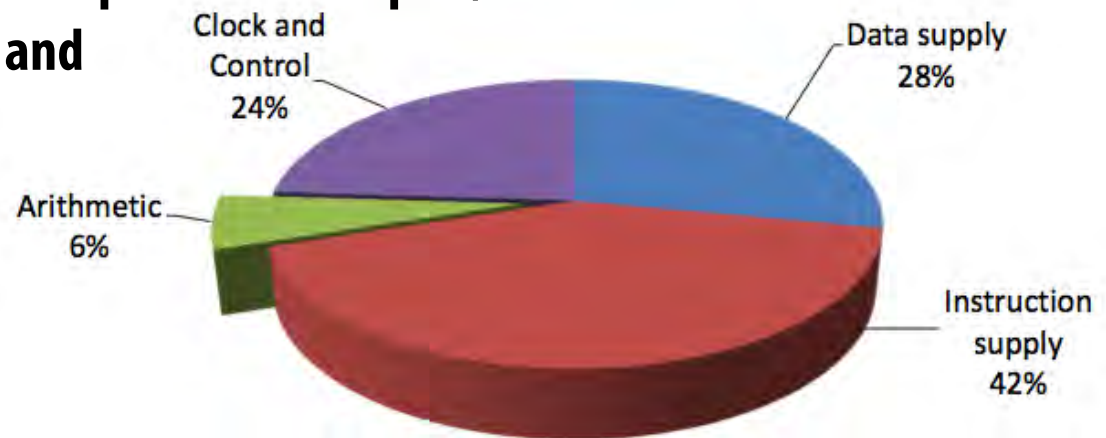
# Limits on chip power consumption

- 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)



# Efficiency benefits of compute specialization

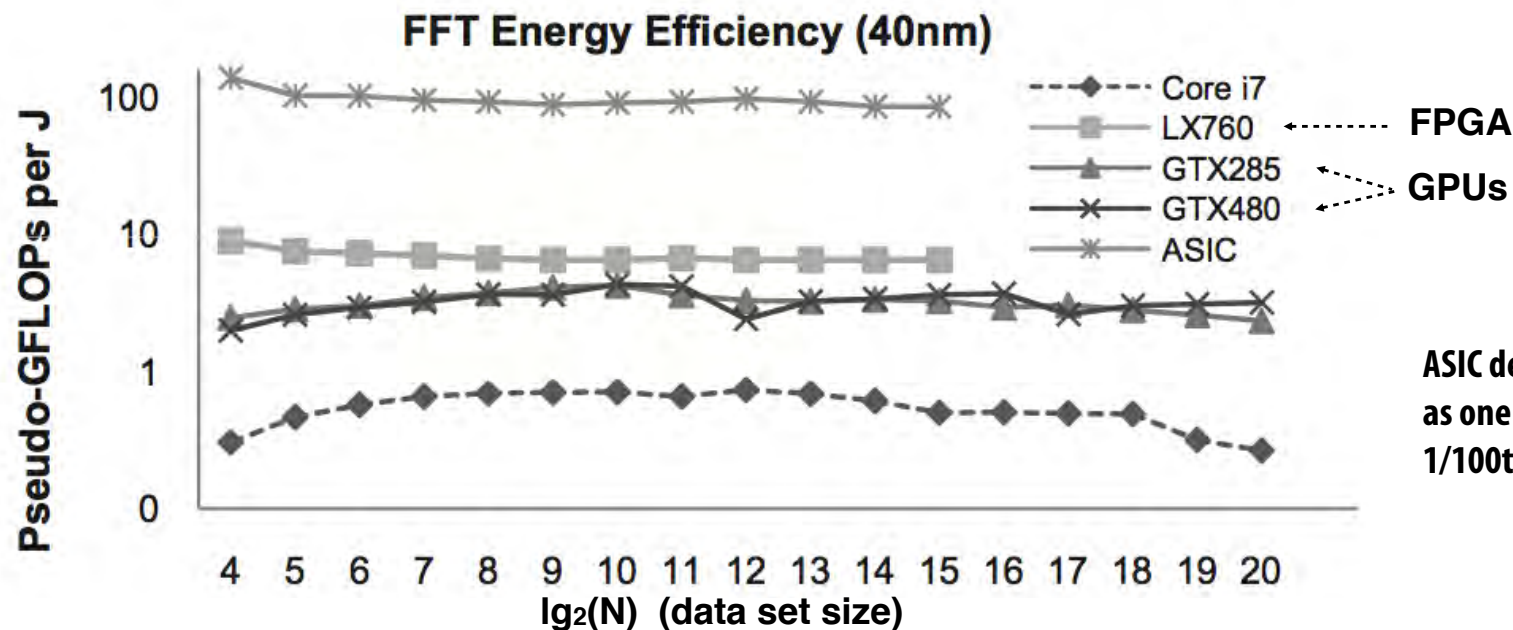
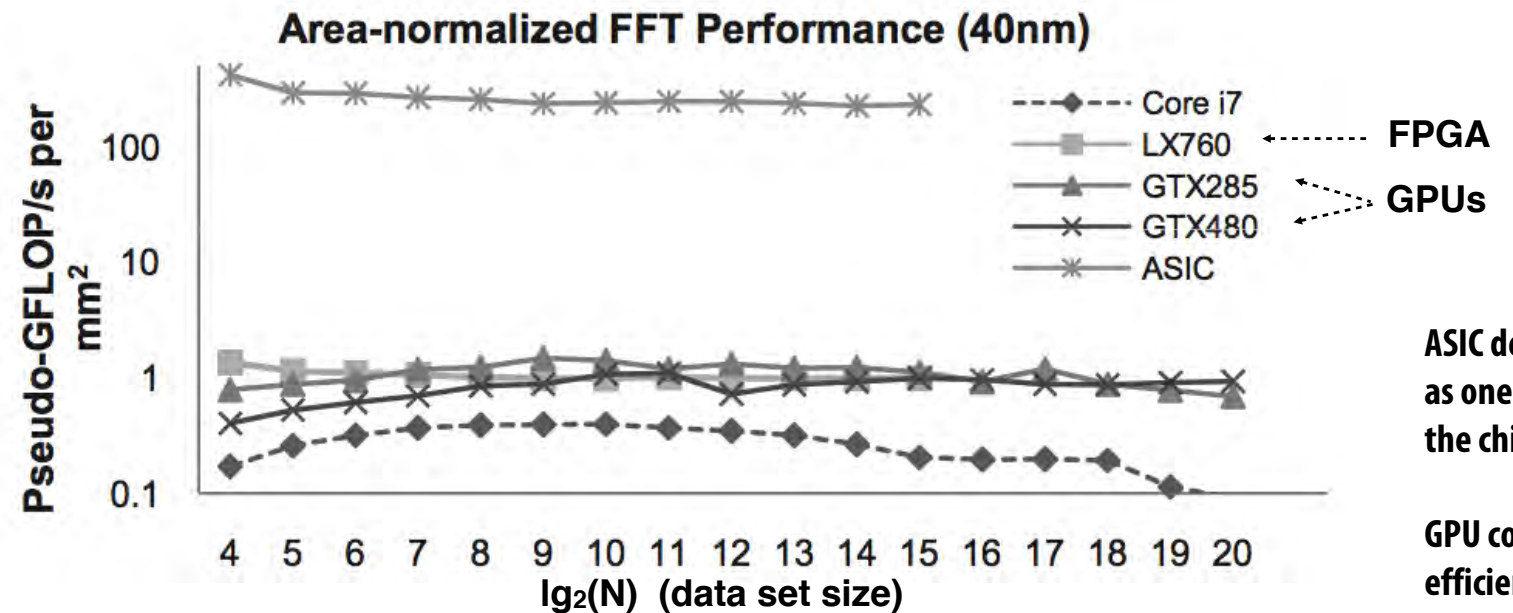
- Rules of thumb: compared to high-quality C code on CPU...
- Throughput-maximized processor architectures: e.g., GPU cores
  - Approximately 10x improvement in perf / watt
  - Assuming code maps well to wide data-parallel execution and is compute bound
- Fixed-function ASIC (“application-specific integrated circuit”)
  - Can approach 100-1000x or greater improvement in perf/watt
  - Assuming code is compute bound and and is not floating-point math



*Efficient Embedded Computing [Dally et al. 08]*

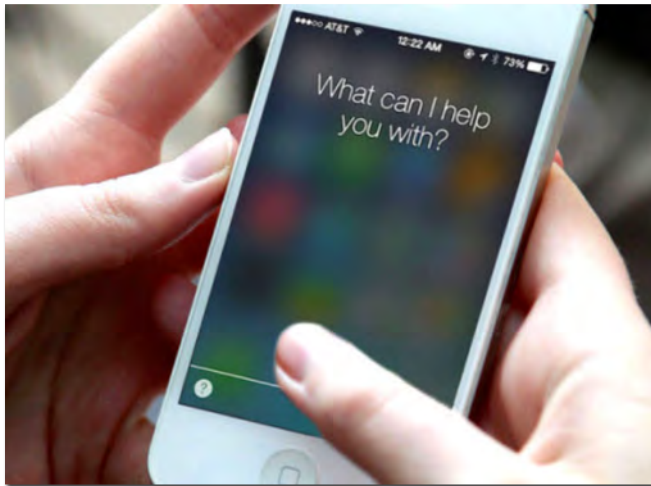


# Hardware specialization increases efficiency

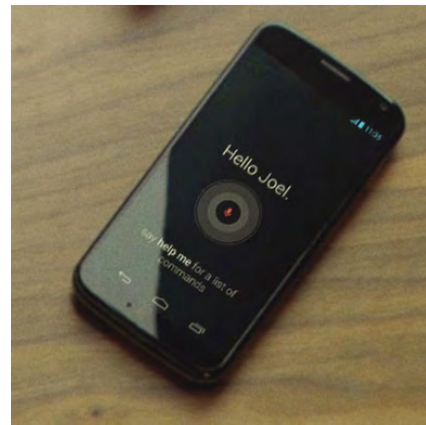


# Benefits of increasing efficiency

- **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



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

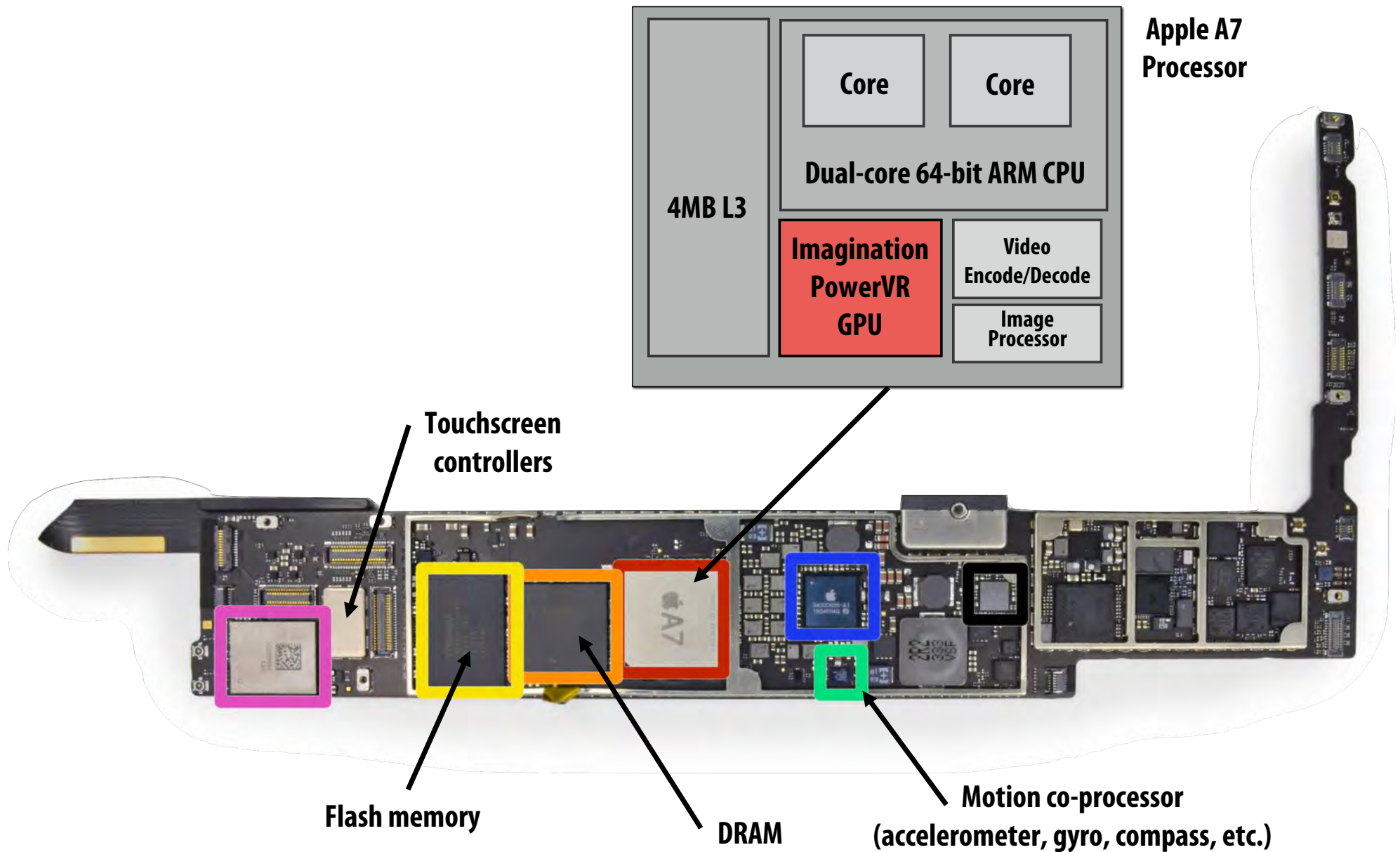


**Moto X:**  
Always listening for “ok, google now”  
Device contains ASIC for detecting this audio pattern.



**Google Glass:** ~40 min  
recording per charge  
(nowhere near “always on”)

## Example: iPad Air (2013)



# Original iPhone touchscreen controller

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

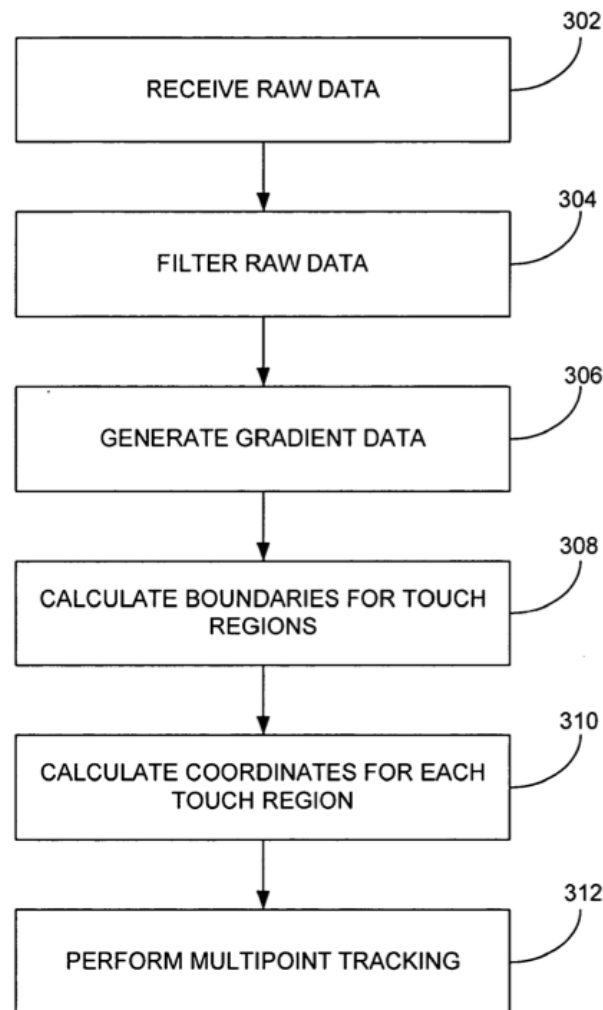


FIG. 16

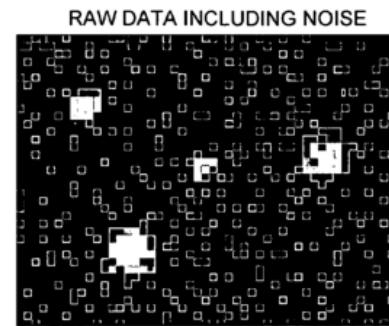


FIG. 17A

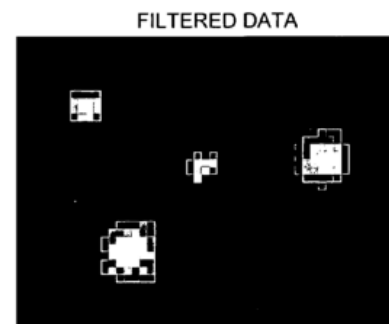


FIG. 17B

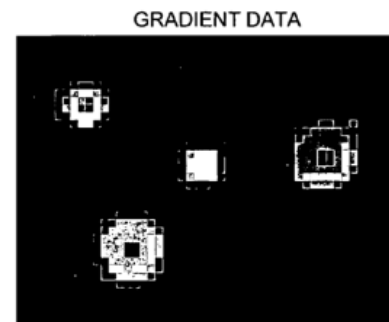


FIG. 17C

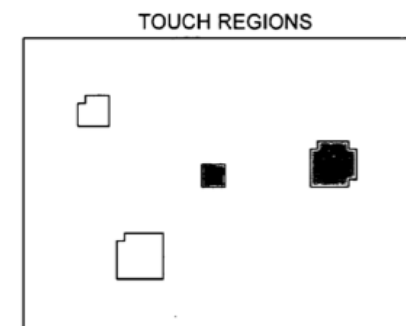


FIG. 17D

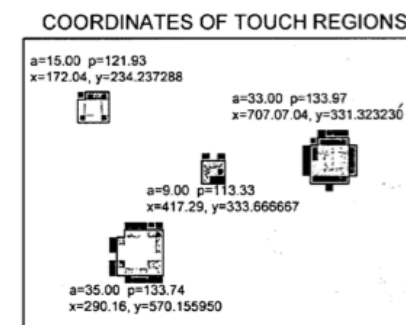


FIG. 17E

# 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)



# Example: image processing on a Nikon D7000



**Process 16 MPixel RAW data from sensor to obtain JPG image:**

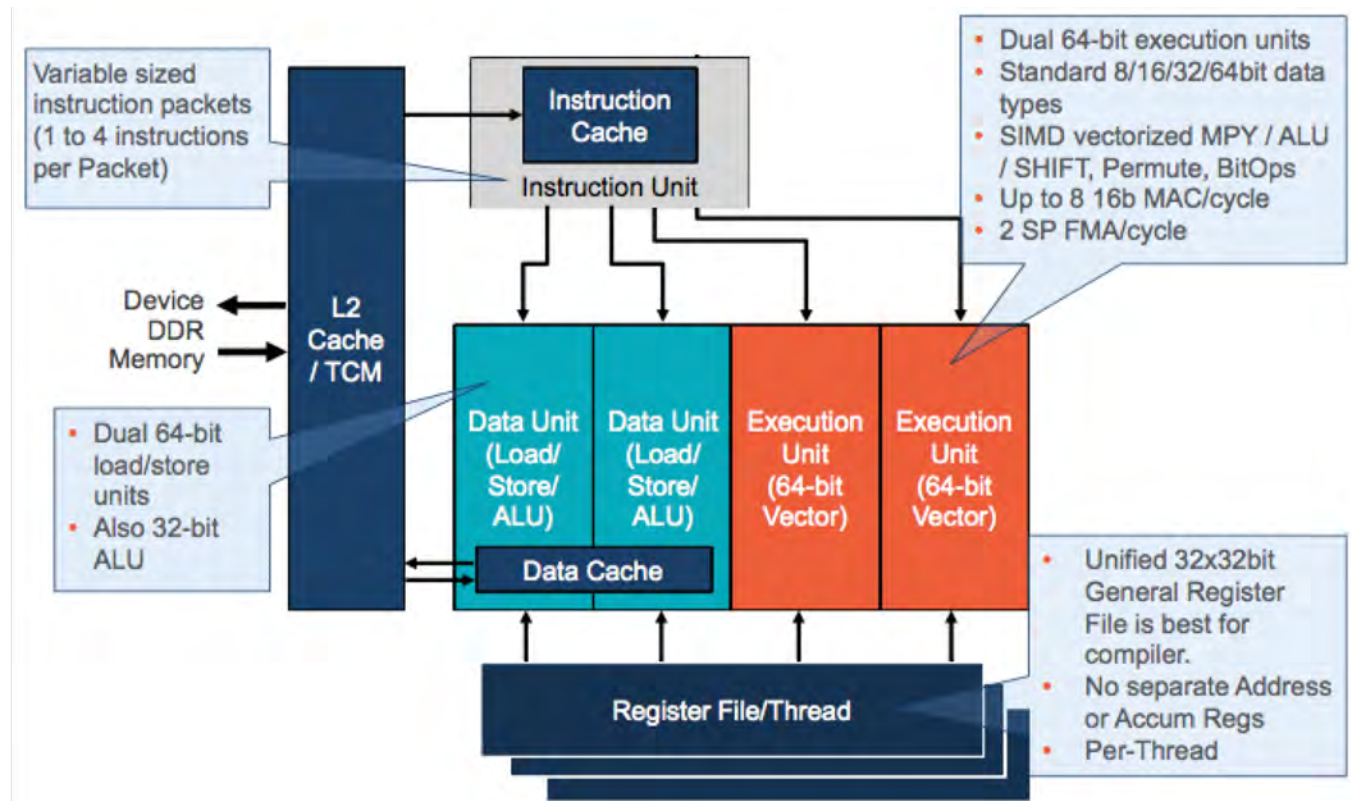
**On camera: ~ 1/6 sec per image**

**Adobe Lightroom a quad-core Macbook Pro laptop: 1-2 sec per image**

**This is a older camera: much, much faster image processing performance on a modern smart phone (burst mode)**

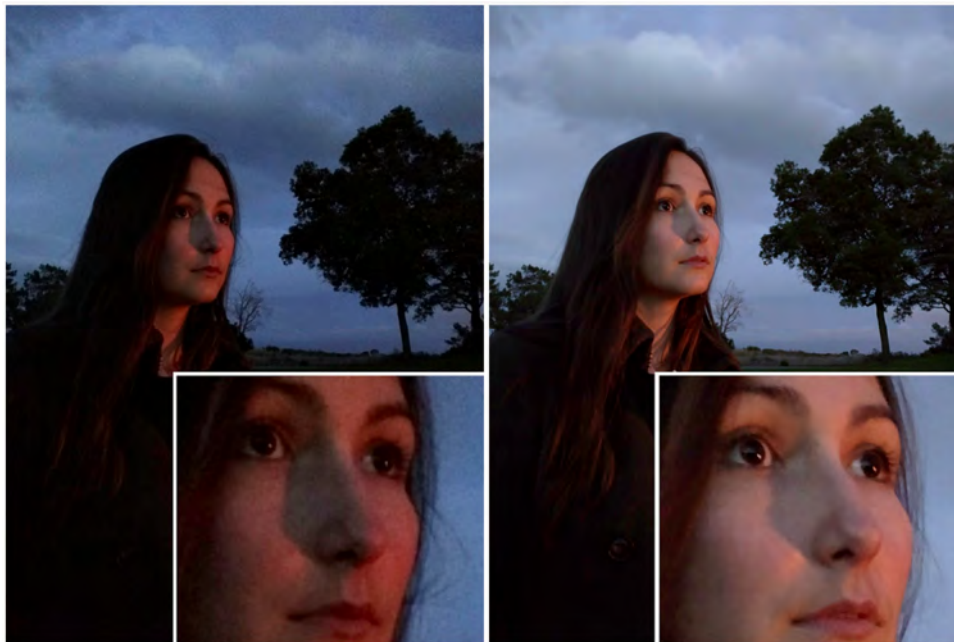
# Qualcomm Hexagon Digital Signal Processor

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



# Up next? application programmable image signal processors

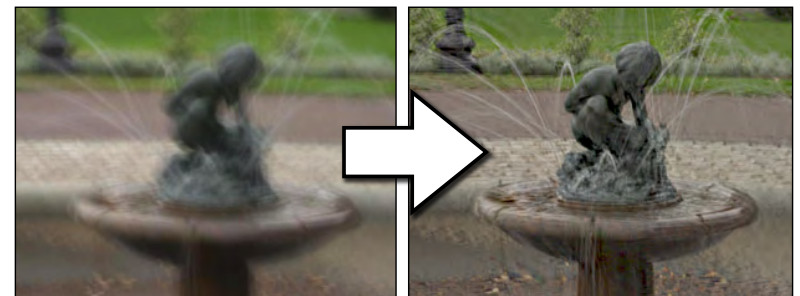
- All modern systems have fixed-function support for common image processing tasks: image/video encode/decode, sensor to image conversion, etc.
- Computational photography: use of advanced algorithms to make better photographs and videos
  - Large space of (rapidly evolving techniques)



High Dynamic Range (HDR) and low light enhancement



Automatic panoramas



Remove camera shake



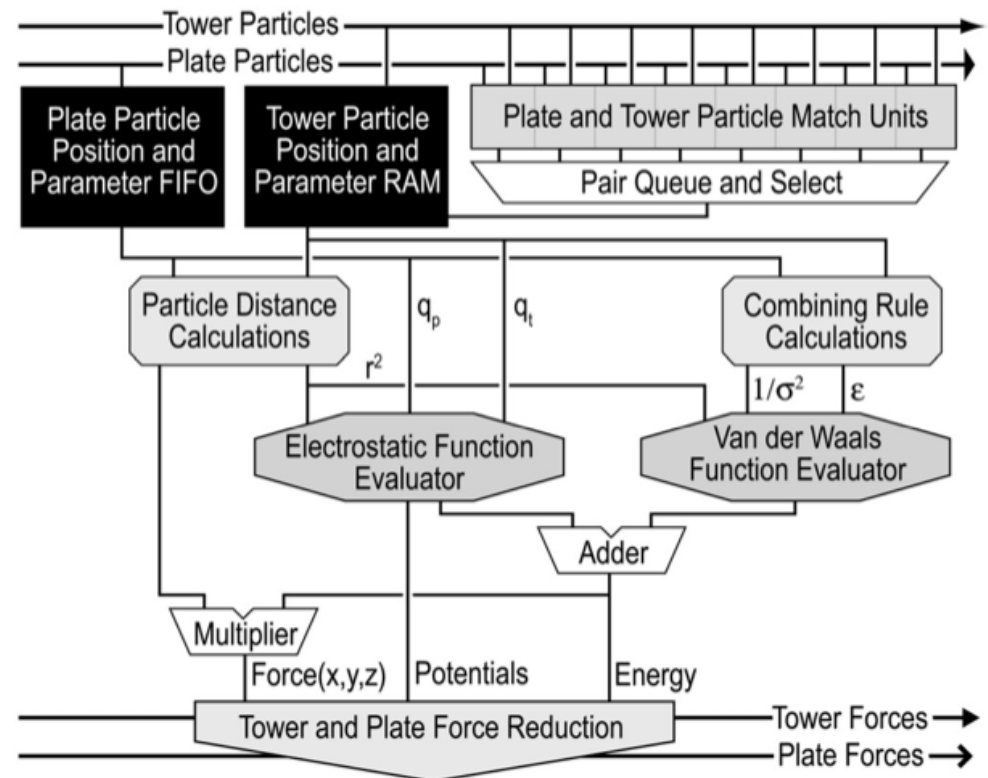
# Anton supercomputer

[Developed by DE Shaw Research]

- 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

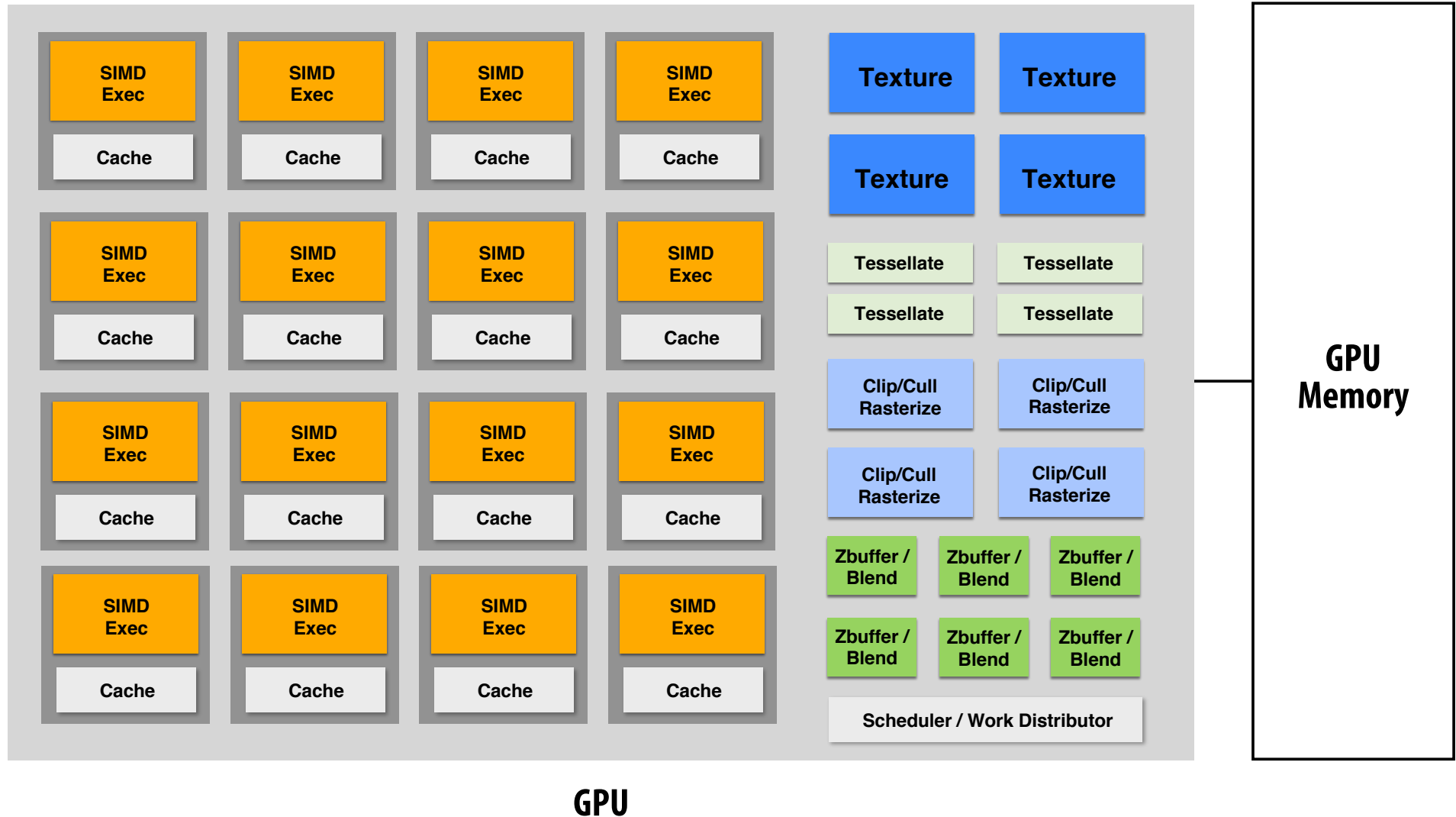
- Custom, low-latency communication network designed for communication patterns of N-body simulations



# GPU's are heterogeneous multi-core processors

Compute resources your CUDA programs used in assignment 2

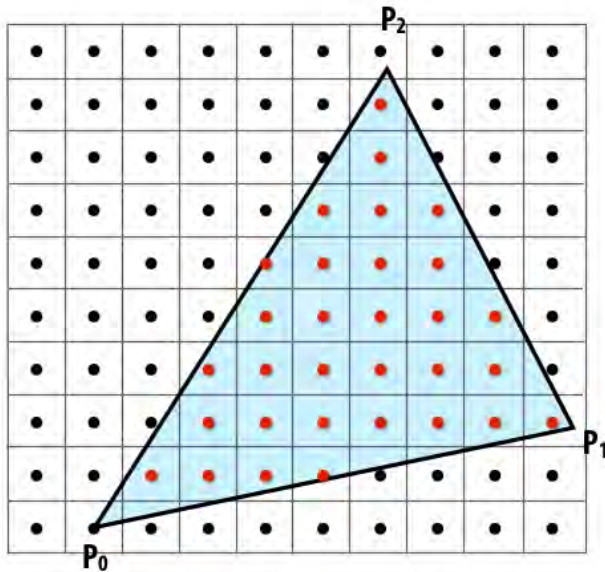
Graphics-specific, fixed-function compute resources



# Example graphics tasks performed in fixed-function HW

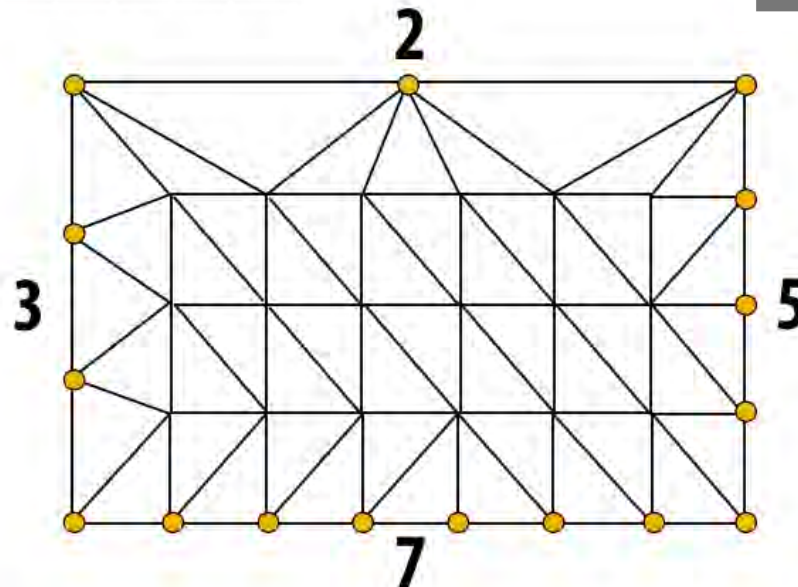
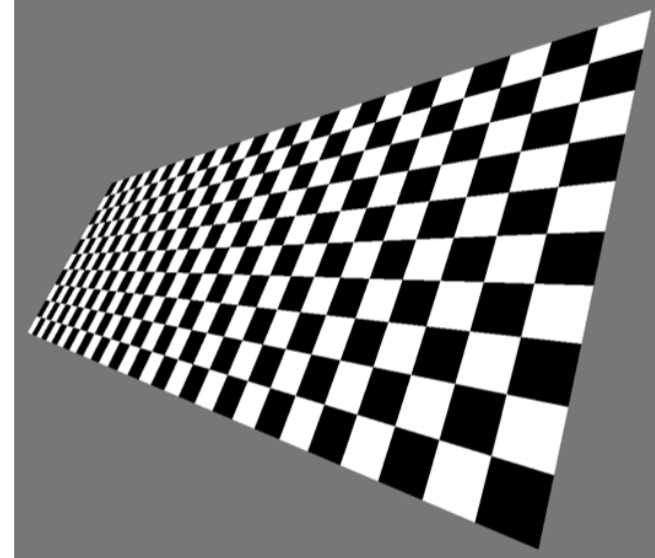
## Rasterization:

Determining what pixels a triangle overlaps



## Texture mapping:

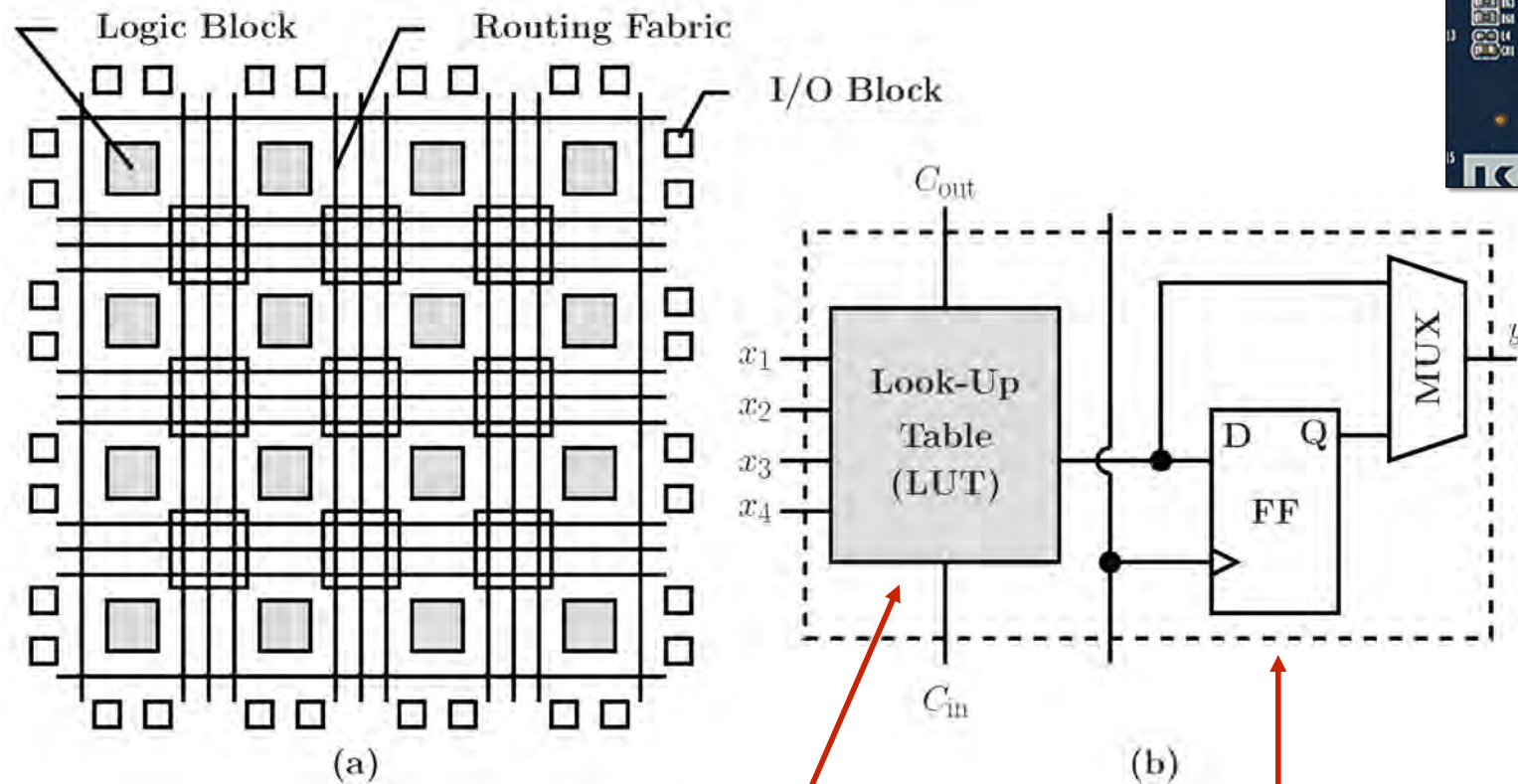
Warping/filtering images to apply detail to surfaces



**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 FPGA



Programmable lookup table (LUT)

Flip flop (a register)



# Project Catapult [Putnam et al. ISCA 2014]

- Microsoft Research investigation of use of FPGAs to accelerate datacenter workloads
- Demonstrated offload of part of Bing Search's document ranking logic

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

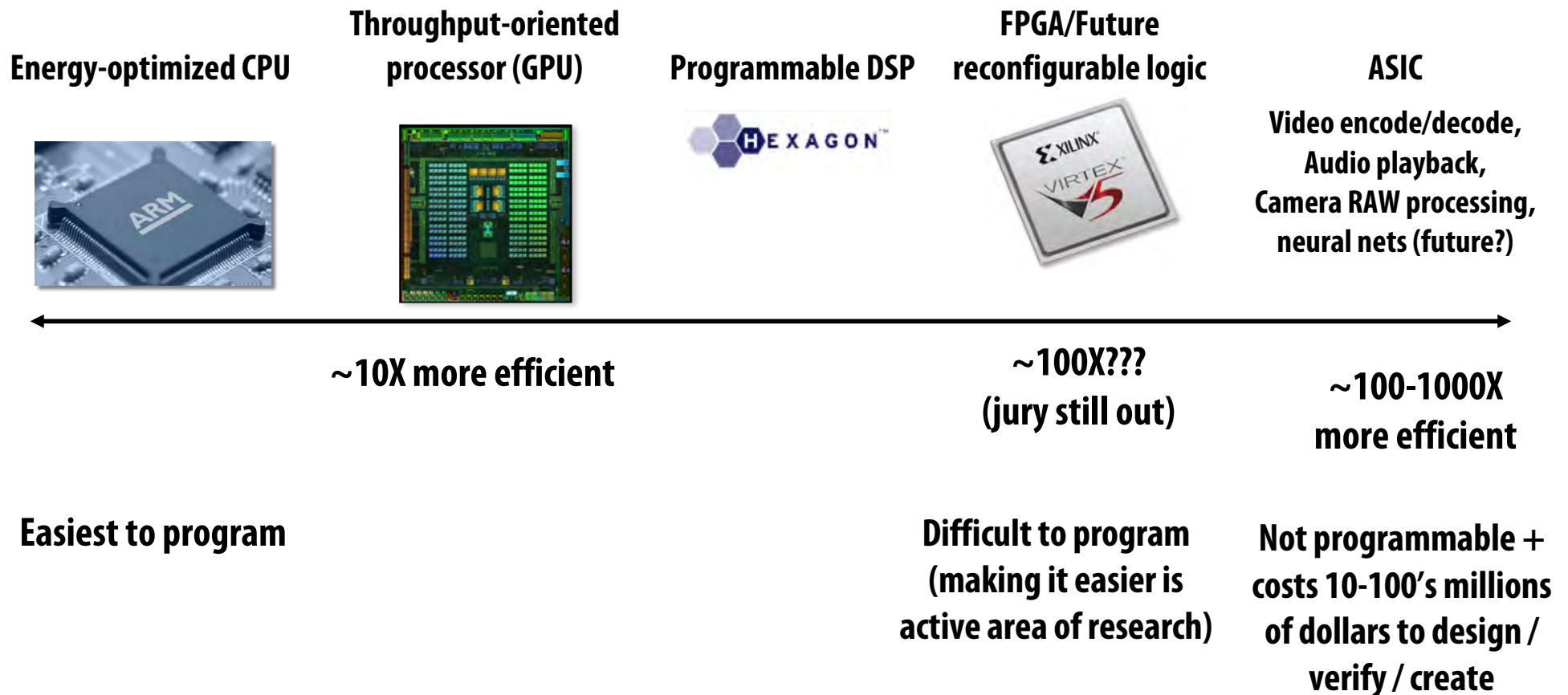
FPGA board



- Two 8-core Xeon 2.1 GHz CPUs
- 64 GB DRAM
- 4 HDDs @ 2 TB, 2 SSDs @ 512 GB
- 10 Gb Ethernet
- No cable attachments to server

Air flow  
200 LFM  
68 °C Inlet

# Summary: choosing the right tool for the job



# **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”**

## ■ 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?

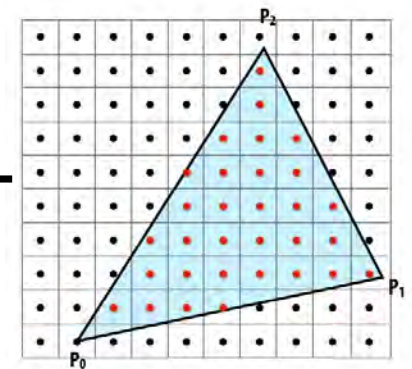
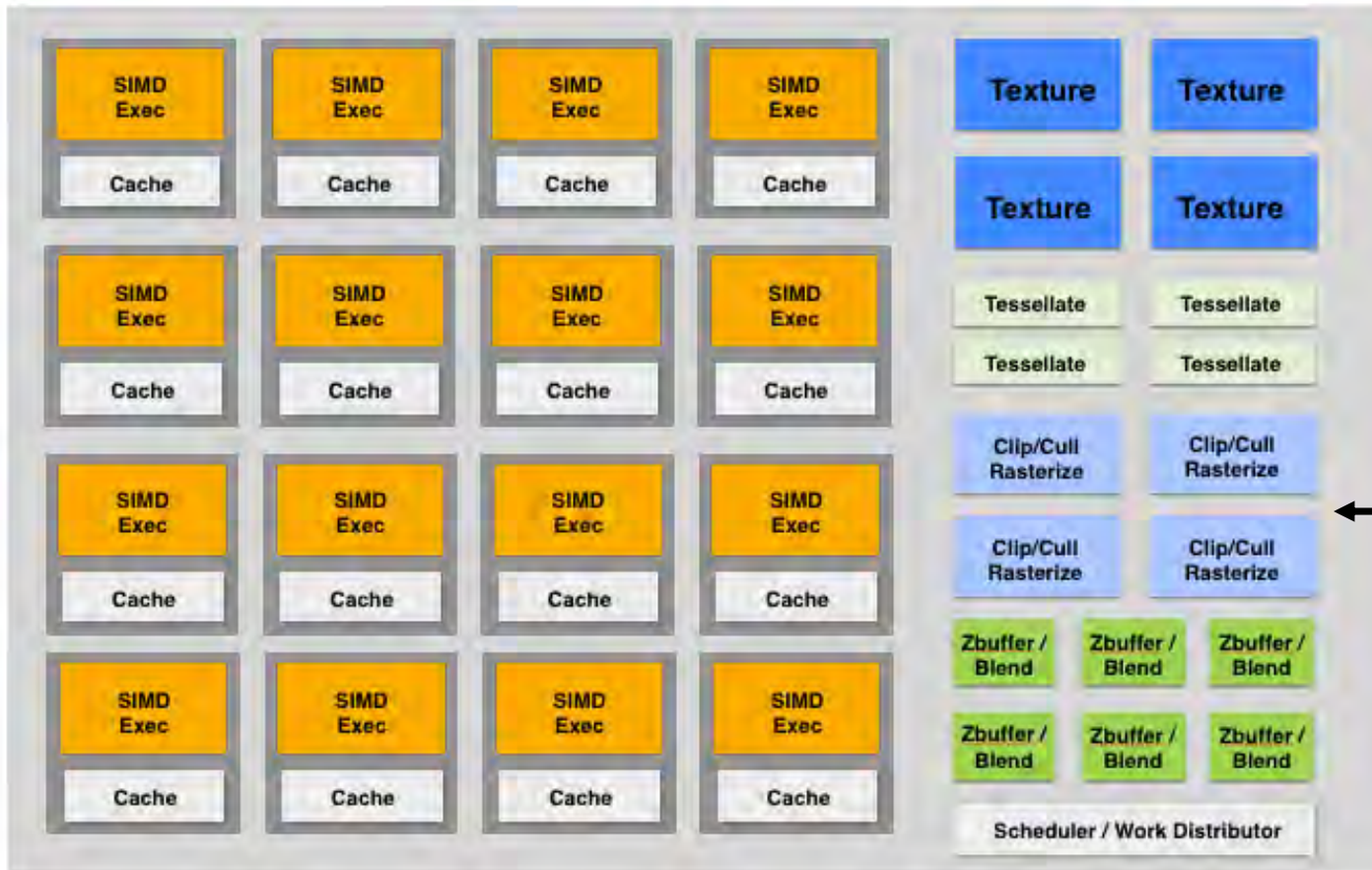
(these resources are taken away from general-purpose processing)

## ■ 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 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 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:  $\sim 1$  pJ \*
  - Floating point op:  $\sim 20$  pJ \*
  - Reading 64 bits from small local SRAM (1mm away on chip):  $\sim 26$  pJ
  - Reading 64 bits from low power mobile DRAM (LPDDR):  $\sim 1200$  pJ
- Implications
  - Reading 10 GB/sec from memory:  $\sim 1.6$  watts
  - Entire power budget for mobile GPU:  $\sim 1$  watt (remember phone is also running CPU, display, radios, etc.)
  - iPhone 6 battery:  $\sim 7$  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

## ■ 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?