Lecture 22:
Heterogeneous Parallelism and Hardware Specialization

CMU 15-418: Parallel Computer Architecture and Programming (Spring 2012)

Announcements

- List of class final projects

  `http://www.cs.cmu.edu/~15418/projectlist.html`

- You are encouraged to keep a log of activities, rants, thinking, findings, on your project web page
  - It will be interesting for us to read
  - It will come in handy when it comes time to do your writeup
  - Writing clarifies thinking

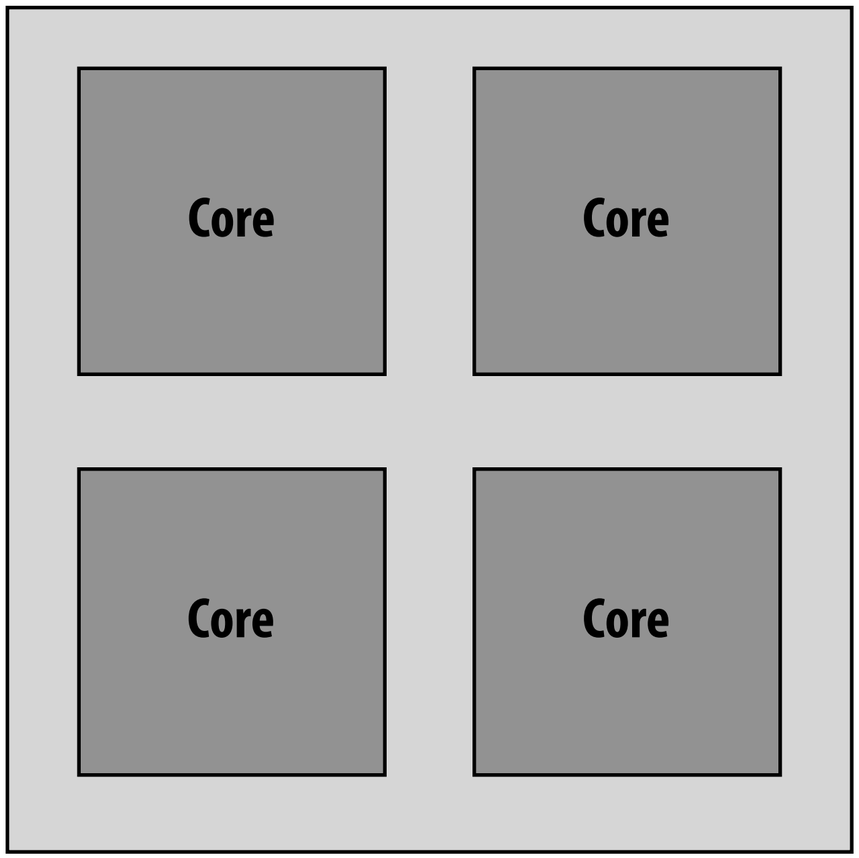
What you should know

- Trade-offs between latency-optimized, throughput-optimized, and fixed-function processing resources

- Advantage of heterogeneous processing: efficiency!

- Disadvantages of heterogeneous processing?

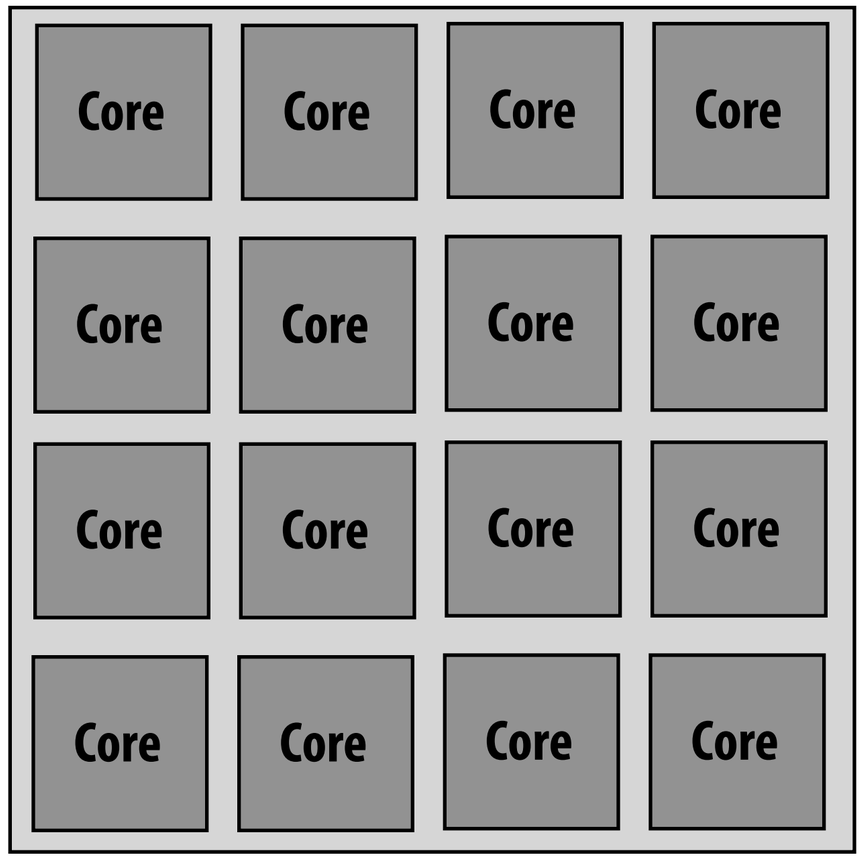
You need to buy a computer system



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

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

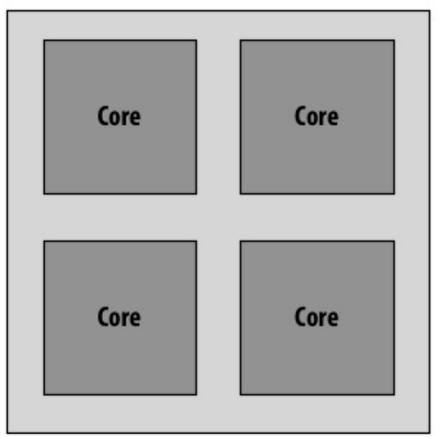
(relative to processor with 1 unit worth of resources, n=1)

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

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

r = **resources dedicated to each processing cores,**

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



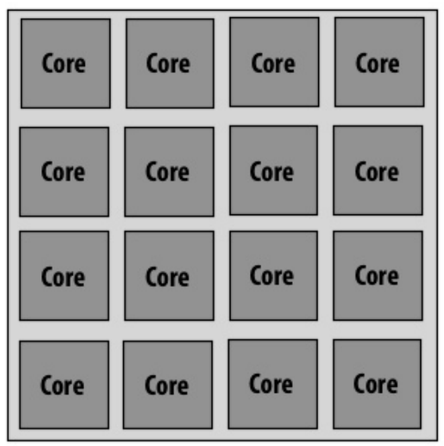
Processor A

Example:

Let n=16

$r_A = 4$

$r_B = 1$

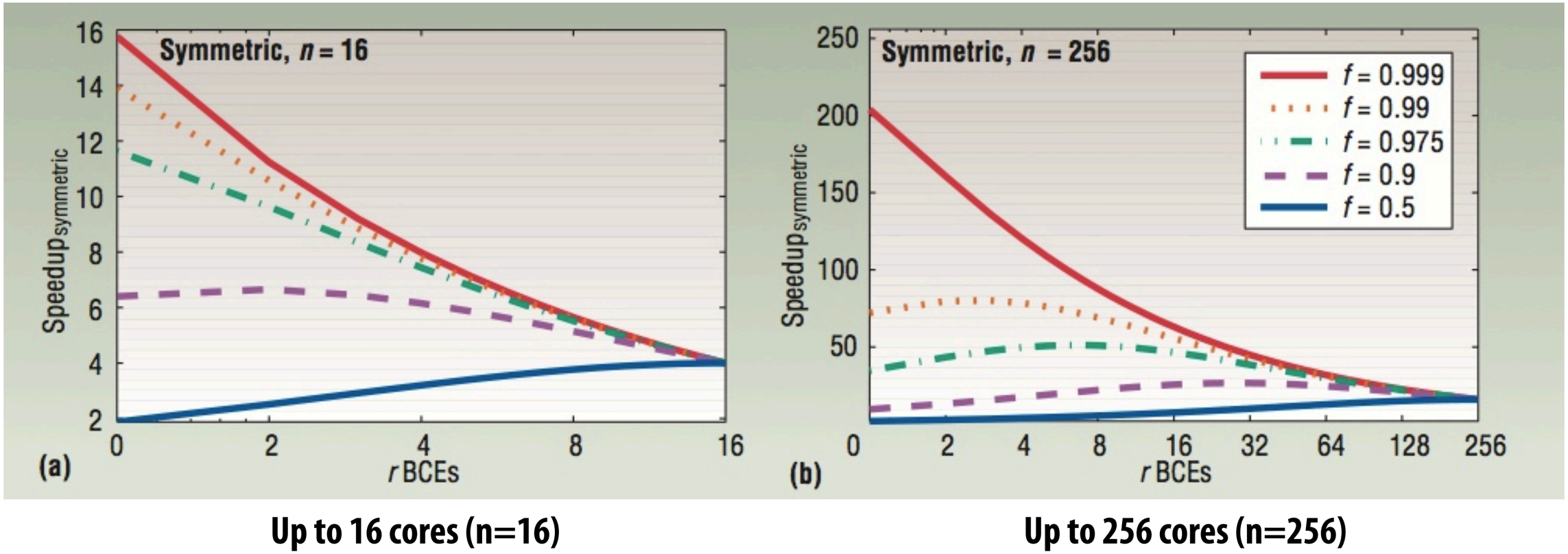


Processor B

[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 graph is iso-resources

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

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

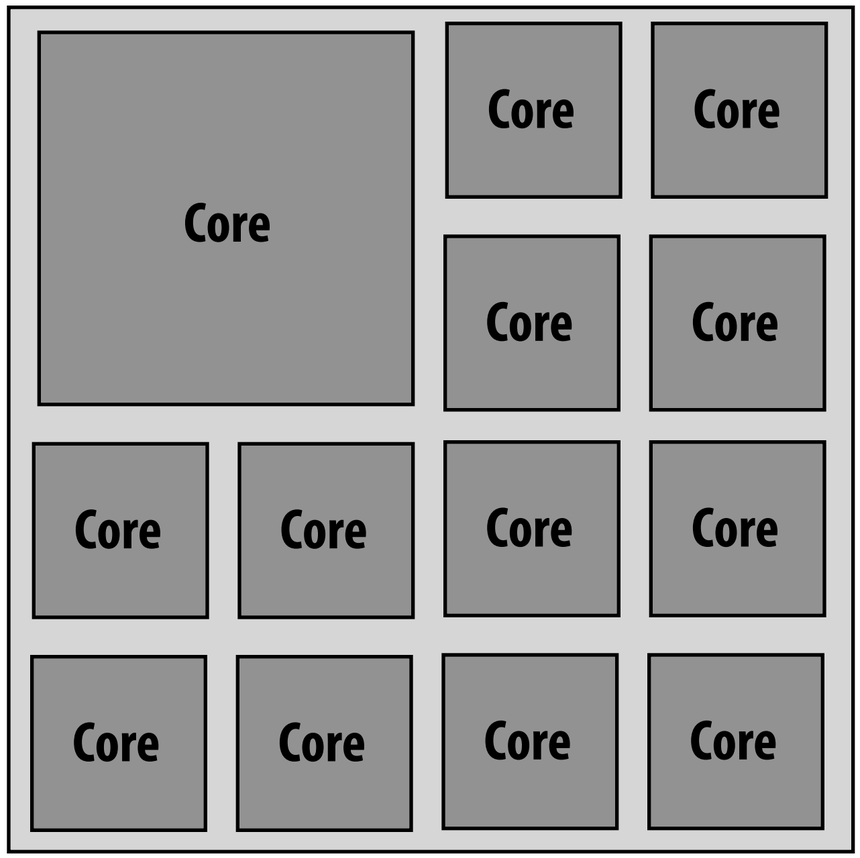
Asymmetric processing cores

Example:

Let n=16

One core: $r = 4$

12 cores: $r = 1$



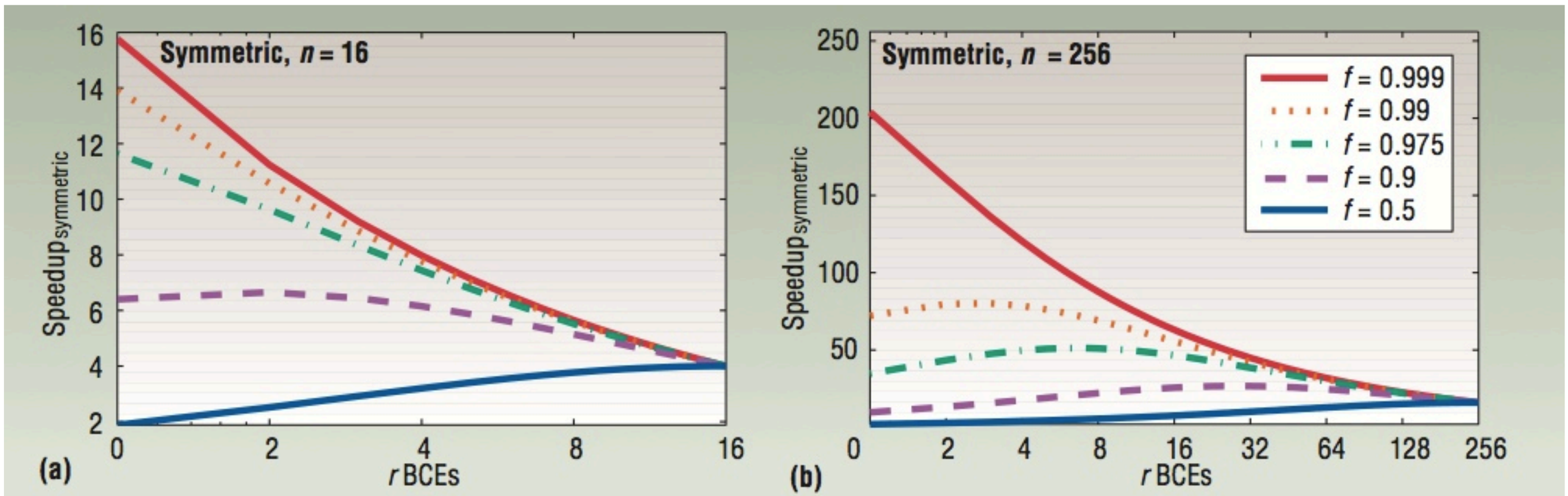
$$speedup(f,n,r) = \frac{1}{\frac{(1-f)}{perf(r)} + \frac{f}{perf(r)+(n-r)}}$$

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

[Hill and Marty 08]

Speedup (relative to n=1)

[Source: Hill and Marty 08]



X-axis for symmetric architectures = r all cores (many small cores to left)

X-axis for asymmetric architectures = r for the single "fat" core (rest of cores are $r = 1$)

Heterogeneous processing

Observation: most real applications are complex **

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

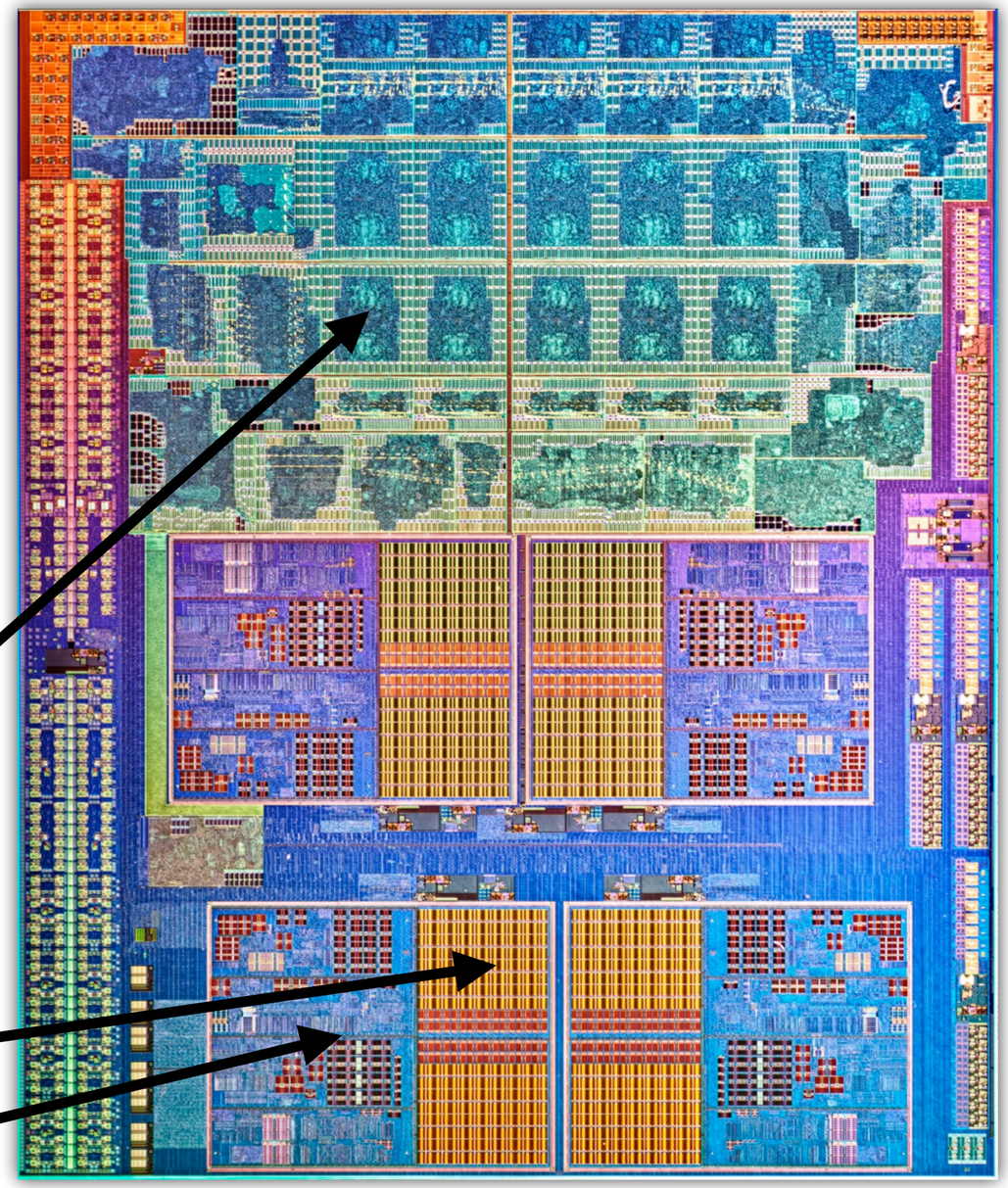
And components with unpredictable access, but those accesses might cache well.

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

** You will likely make this observation during your projects

Example: AMD Fusion

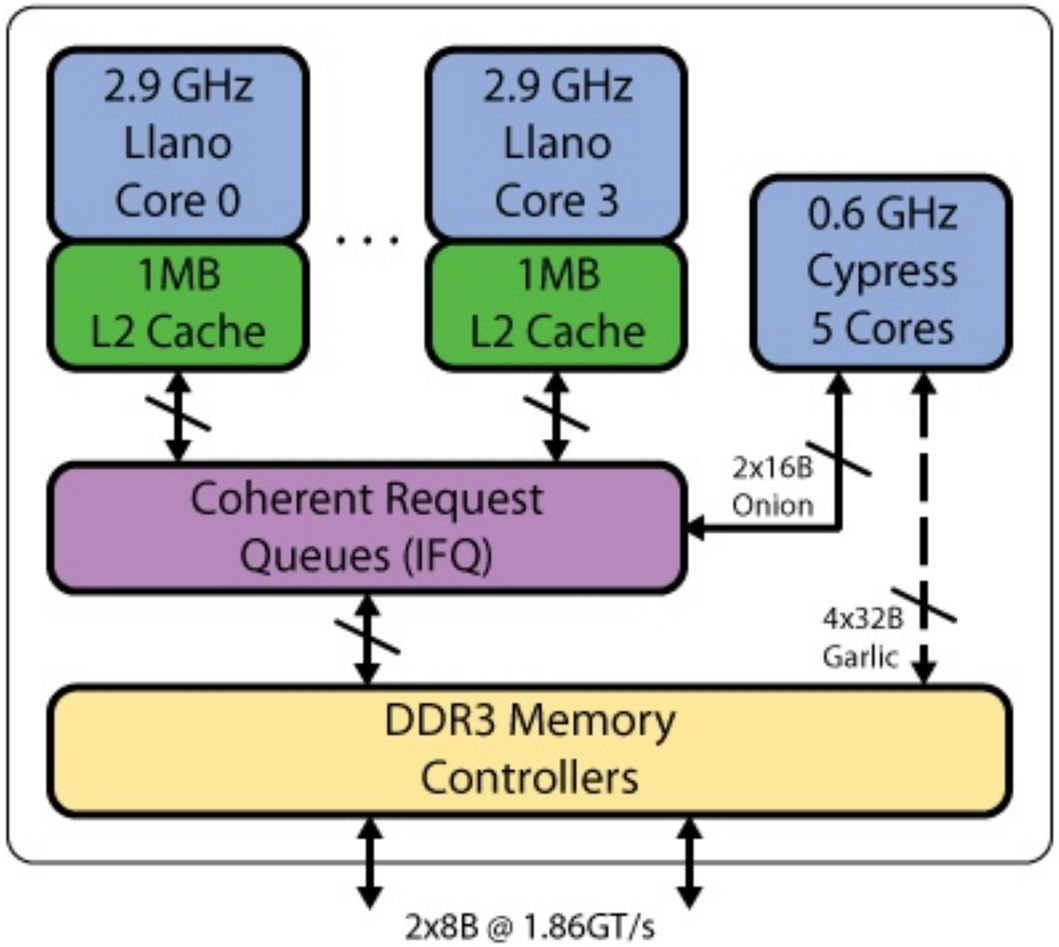
- "APU": accelerated processing unit
- Integrate CPU cores and GPU-style cores on same chip
- Share memory system
  - Regions of physical memory reserved for graphics (not x86 coherent)
  - Rest of memory is x86 coherent
  - CPU and graphics memory spaces are not coherent, but at least there is no need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



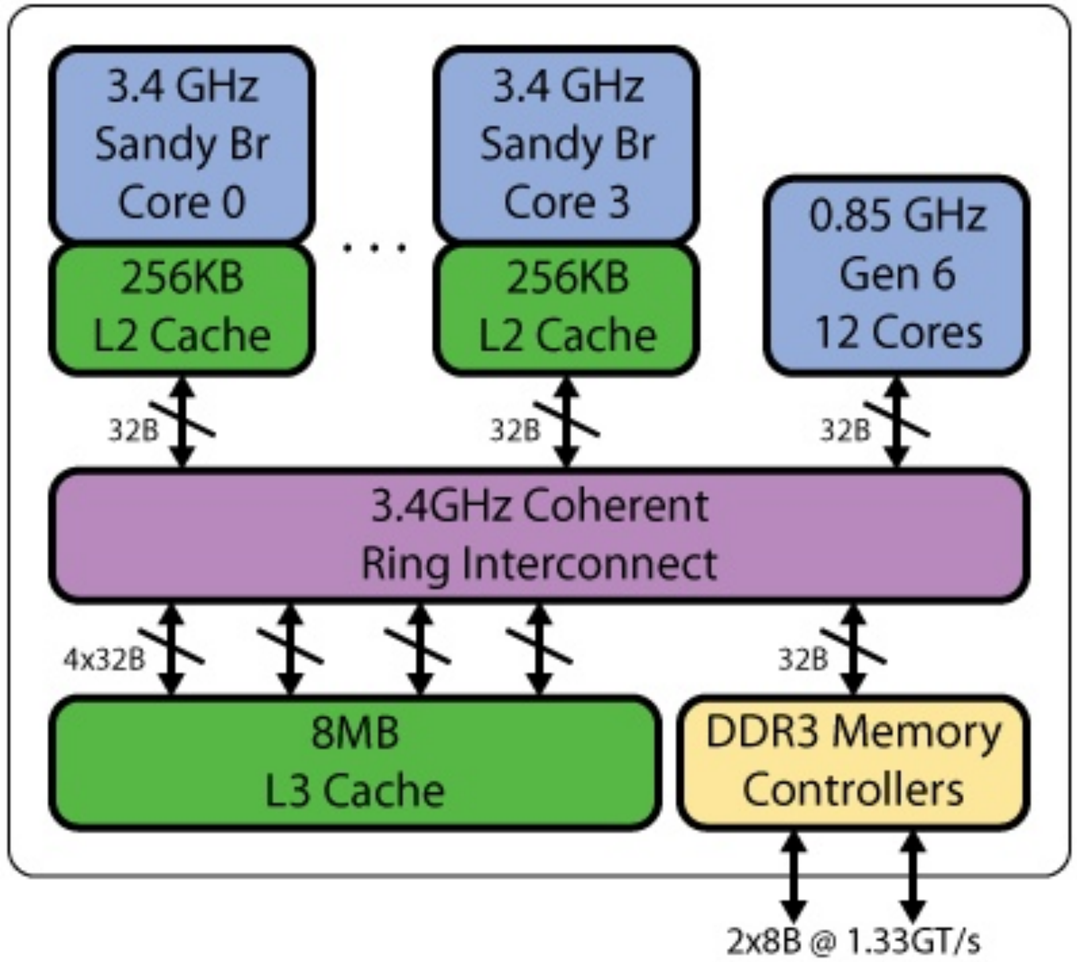
AMD Llano
(4 CPU cores + integrated GPU cores)

Integrated CPU+graphics

AMD Llano

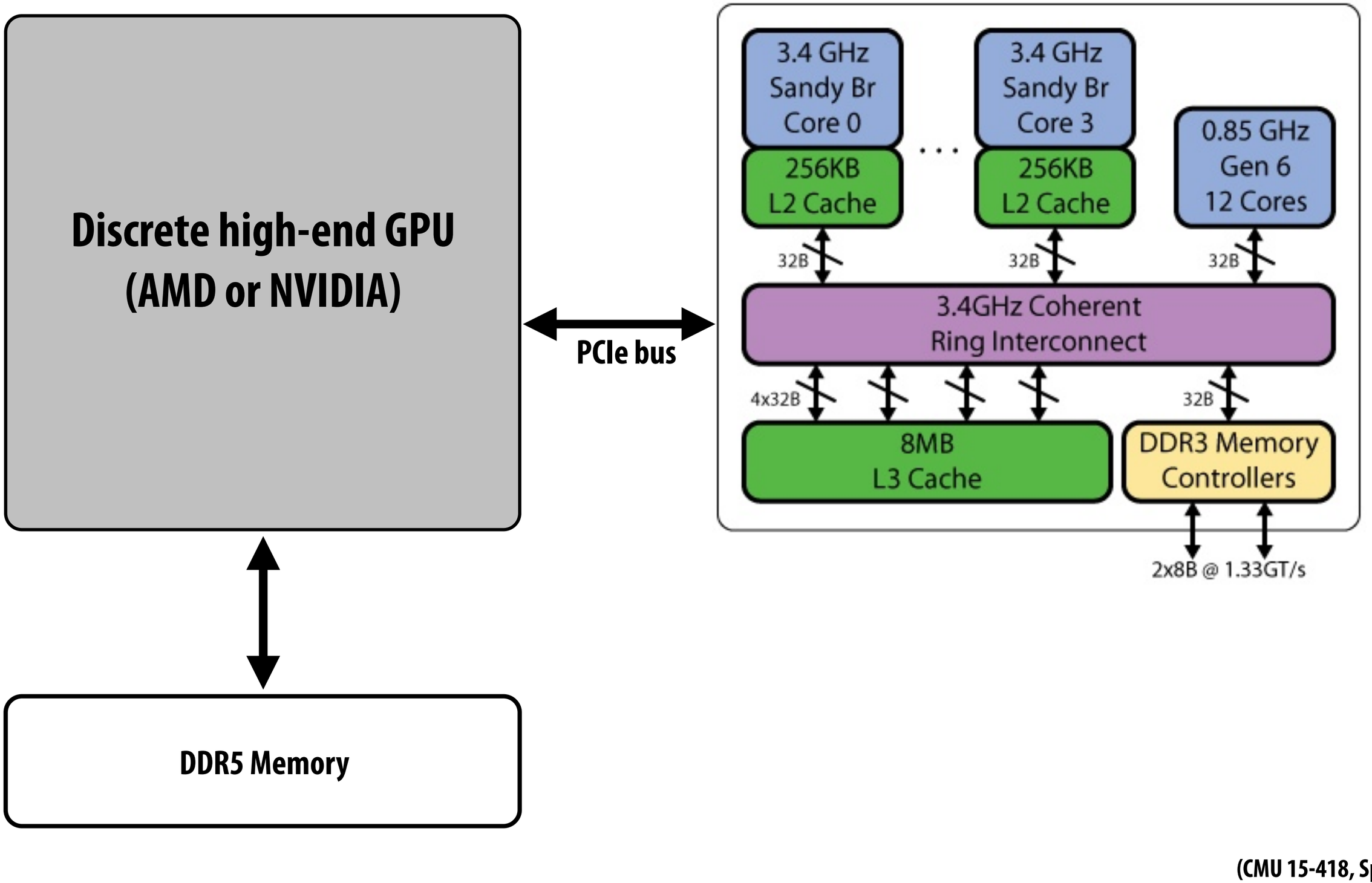


Intel Sandy Bridge

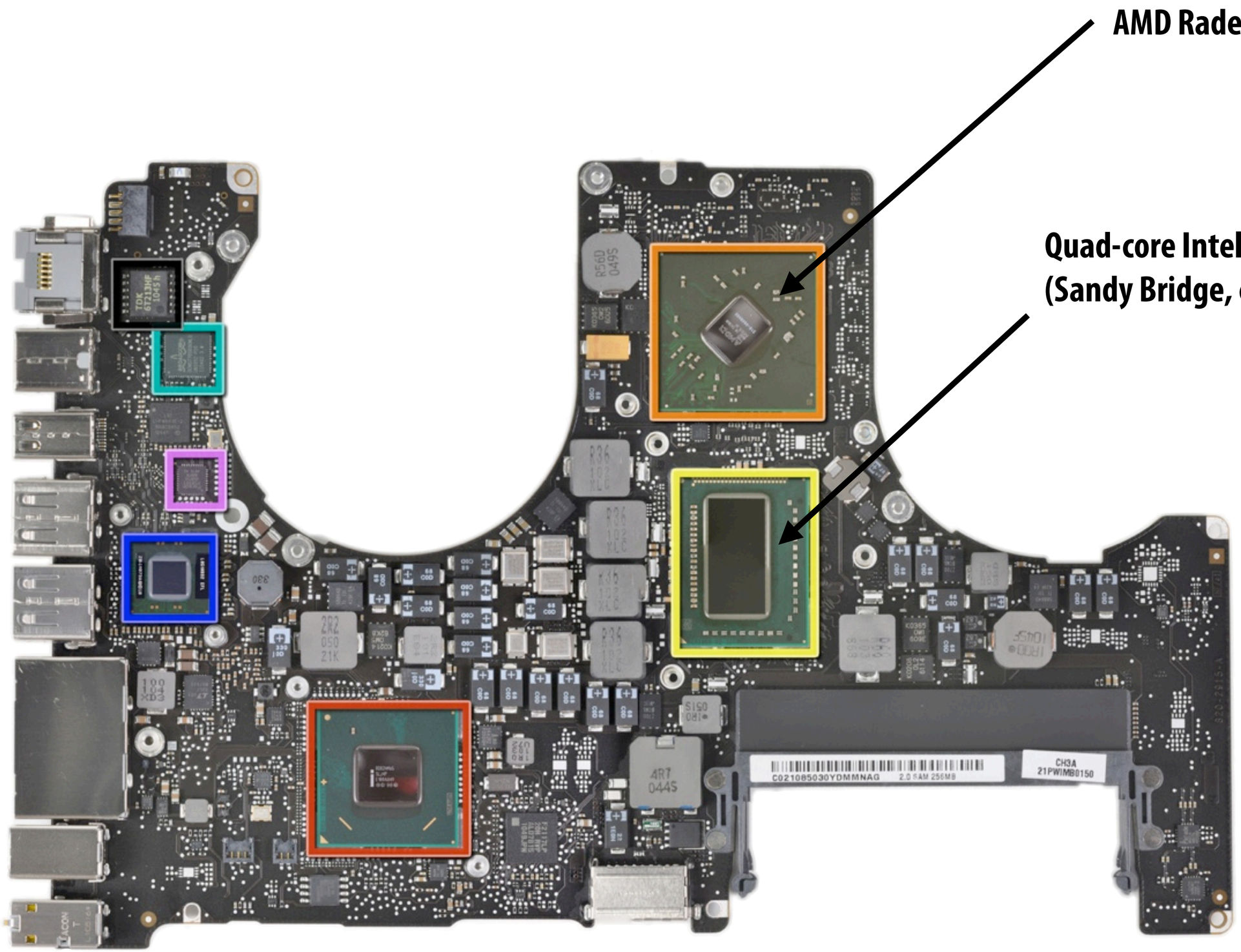


More heterogeneity: add discrete GPU

Keep discrete (power hungry) GPU unless needed for graphics-intensive applications
Use integrated, low power graphics for window manager/UI
(Neat: AMD Fusion can parallelize graphics across integrated and discrete GPU)



My 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 MWatts of power (about 2,400 average US homes)**

Recent supercomputing trend: GPU acceleration

Although #1 uses only 8-core SPARC64 CPUs (128 GFLOPs per CPU)

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.0
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 / 2010 Dawning	120640	1271.00	2984.30	2580.0
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.6
6	DOE/NNSA/LANL/SNL United States	Cray XE6, Opteron 6136 8C 2.40GHz, Custom / 2011 Cray Inc.	142272	1110.00	1365.81	3980.0
7	NASA/Ames Research Center/NAS United States	SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 Ghz, Infiniband / 2011 SGI	111104	1088.00	1315.33	4102.0
8	DOE/SC/LBNL/NERSC United States	Cray XE6, Opteron 6172 12C 2.10GHz, Custom / 2010 Cray Inc.	153408	1054.00	1288.63	2910.0

11 PFLOPS, 12.6 MW

Use GPUs as accelerators!

GPU-accelerated supercomputing

- Tianhe-1A (world's #2)
- 7168 NVIDIA Tesla M2050 GPUs
  (basically what we have in 5205)

Tianhe-1A

- Estimated cost $88M
- Estimated <u>annual</u> power/operating cost: $20M

Energy-constrained computing

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

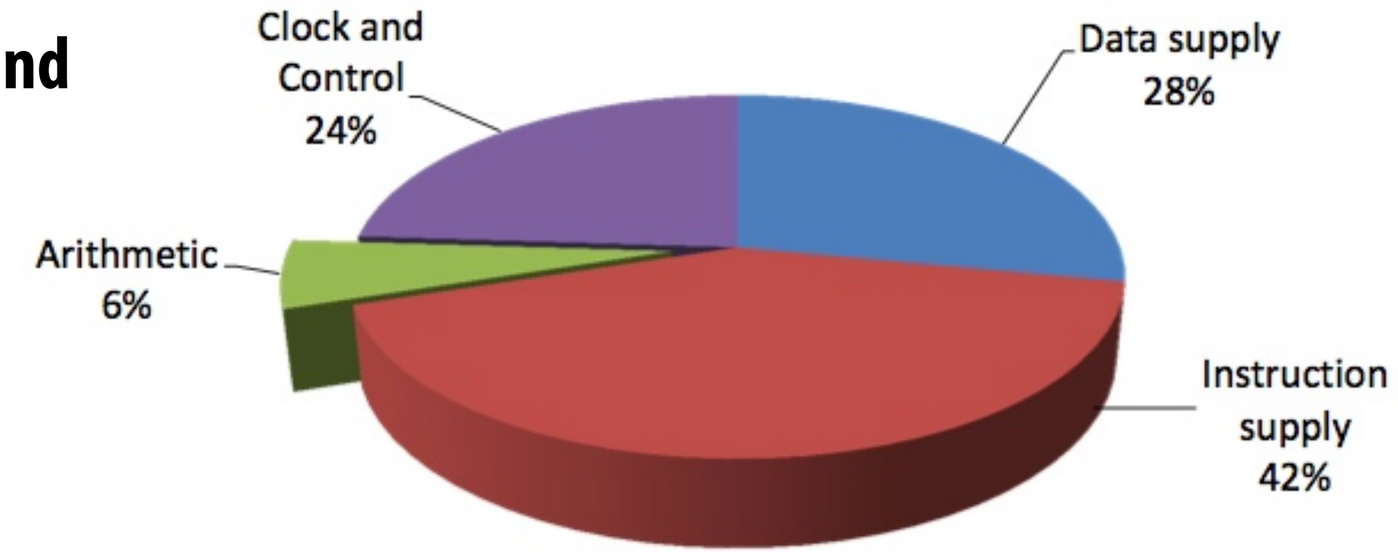
- Mobile devices are energy-constrained
  - Limited battery life

Efficiency benefits of specialization

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

- Throughput-maximized 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

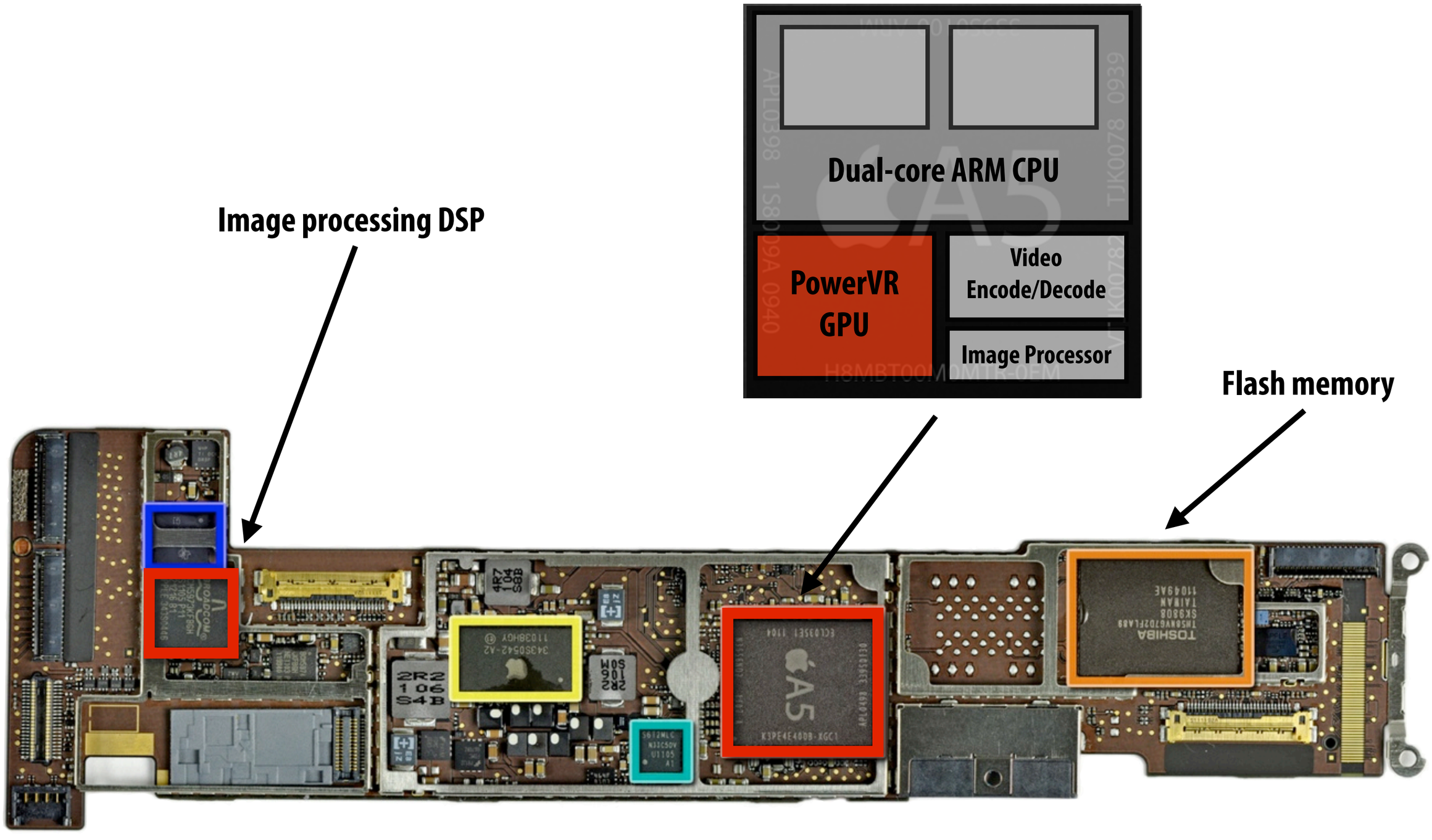


Efficient Embedded Computing [Dally et al. 08]

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

[Figure credit Eric Chung]

Example: iPad 2



Original iPhone touchscreen controller

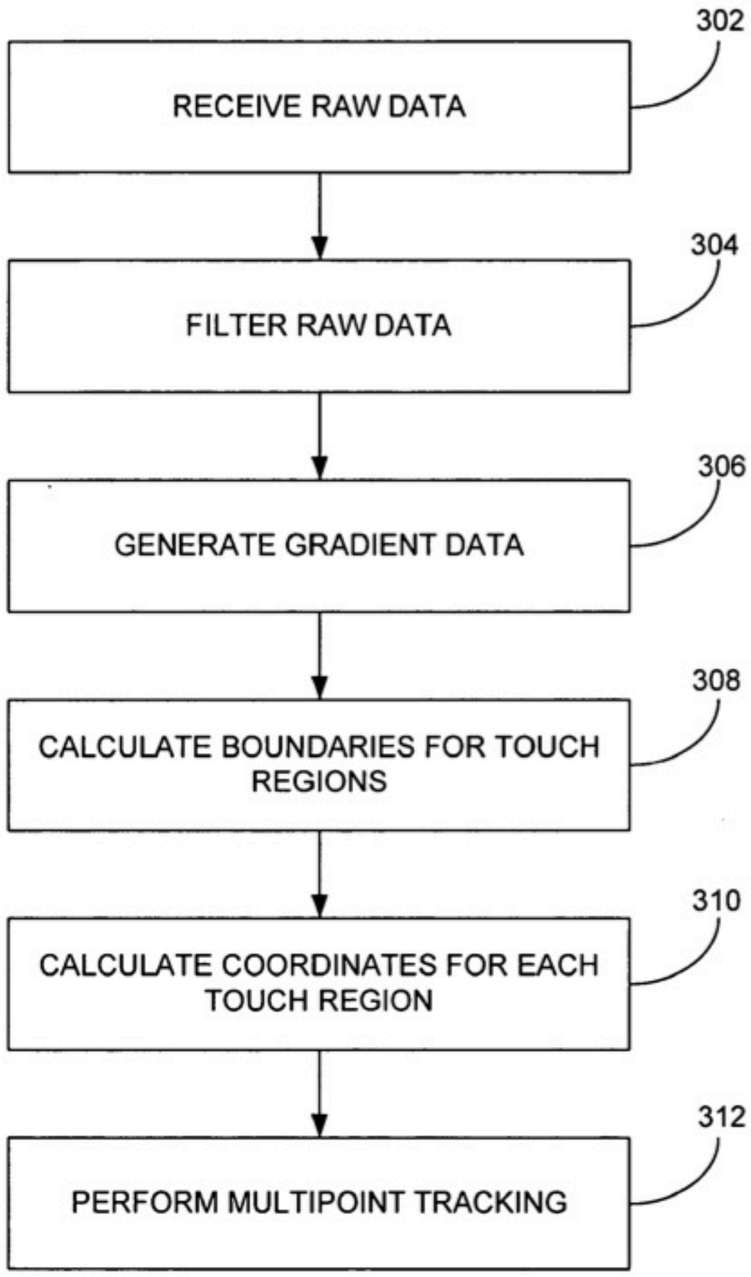


FIG. 16

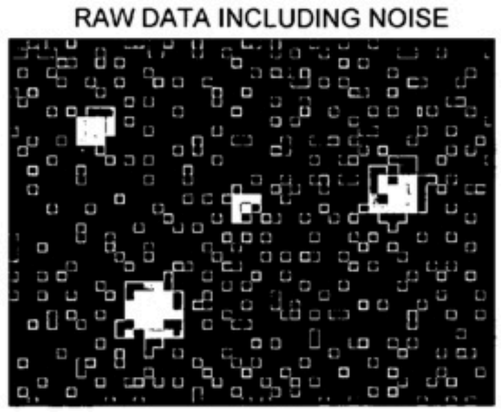


FIG. 17A

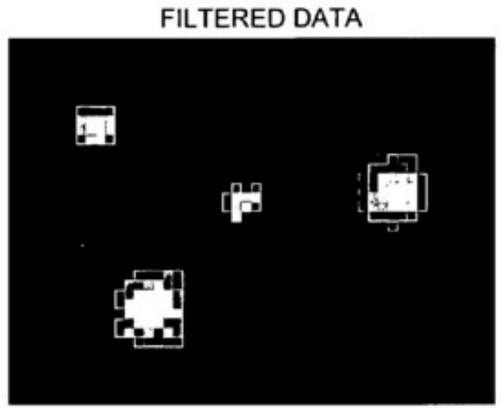


FIG. 17B

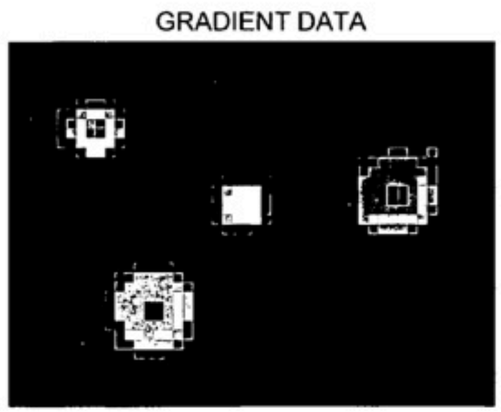


FIG. 17C

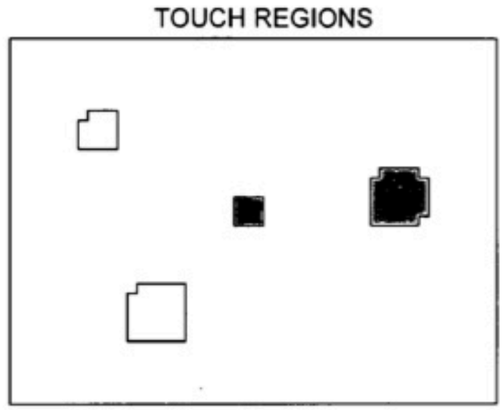


FIG. 17D

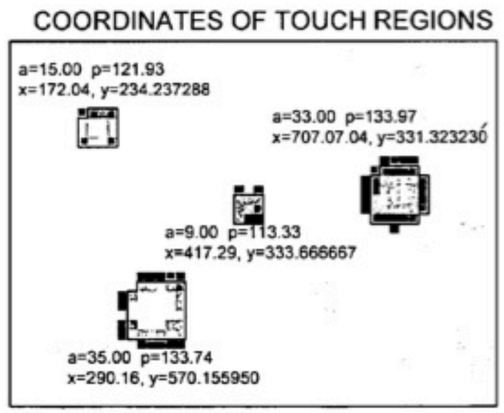


FIG. 17E

From US Patent Application 2006/0097991

NVIDIA Tegra 3 (2011)

Asymmetric CPU-style cores

Higher performance, higher power

Low performance, low power

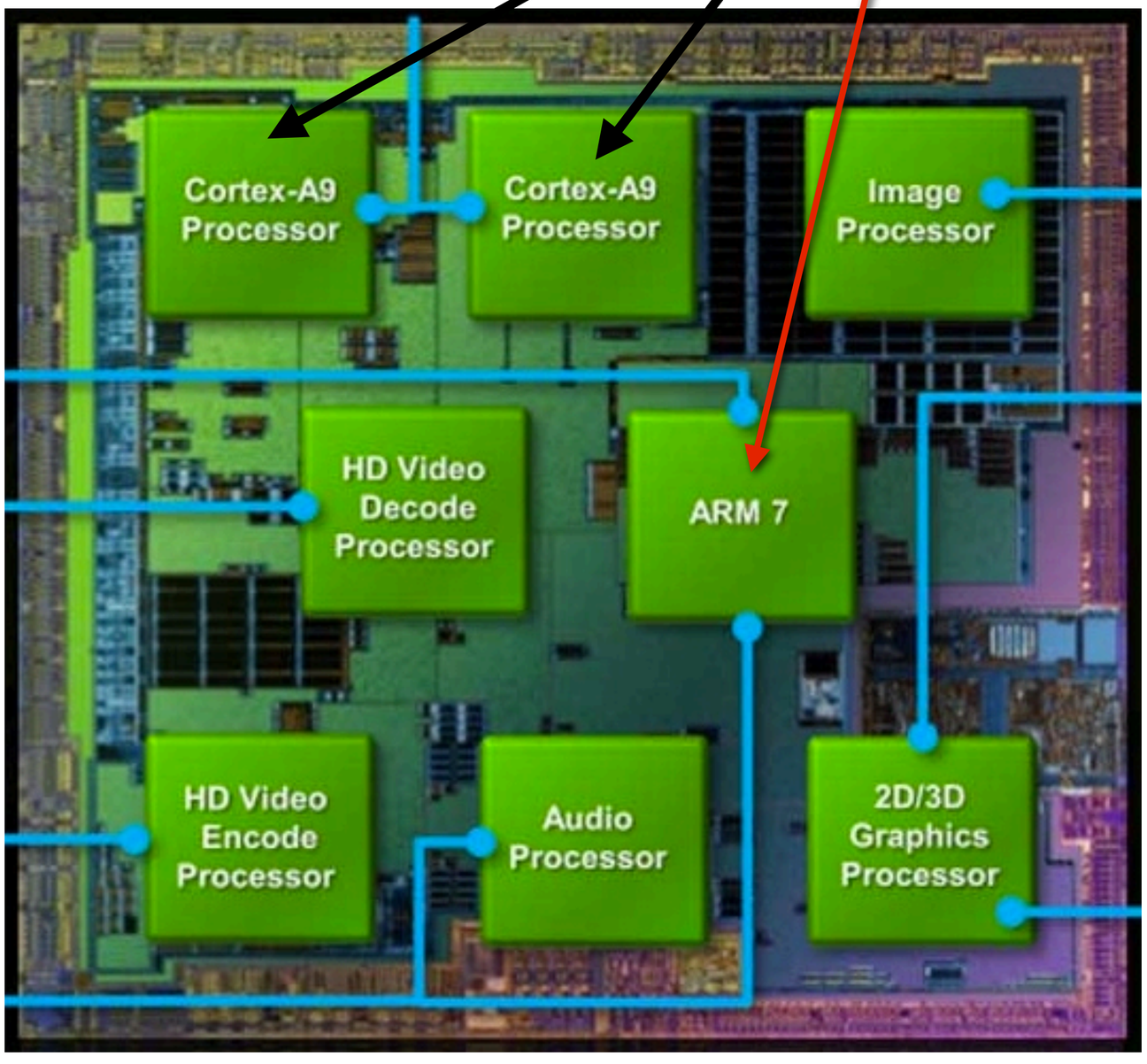


Image credit: NVIDA

Texas Instruments OMAP 5 (2012)

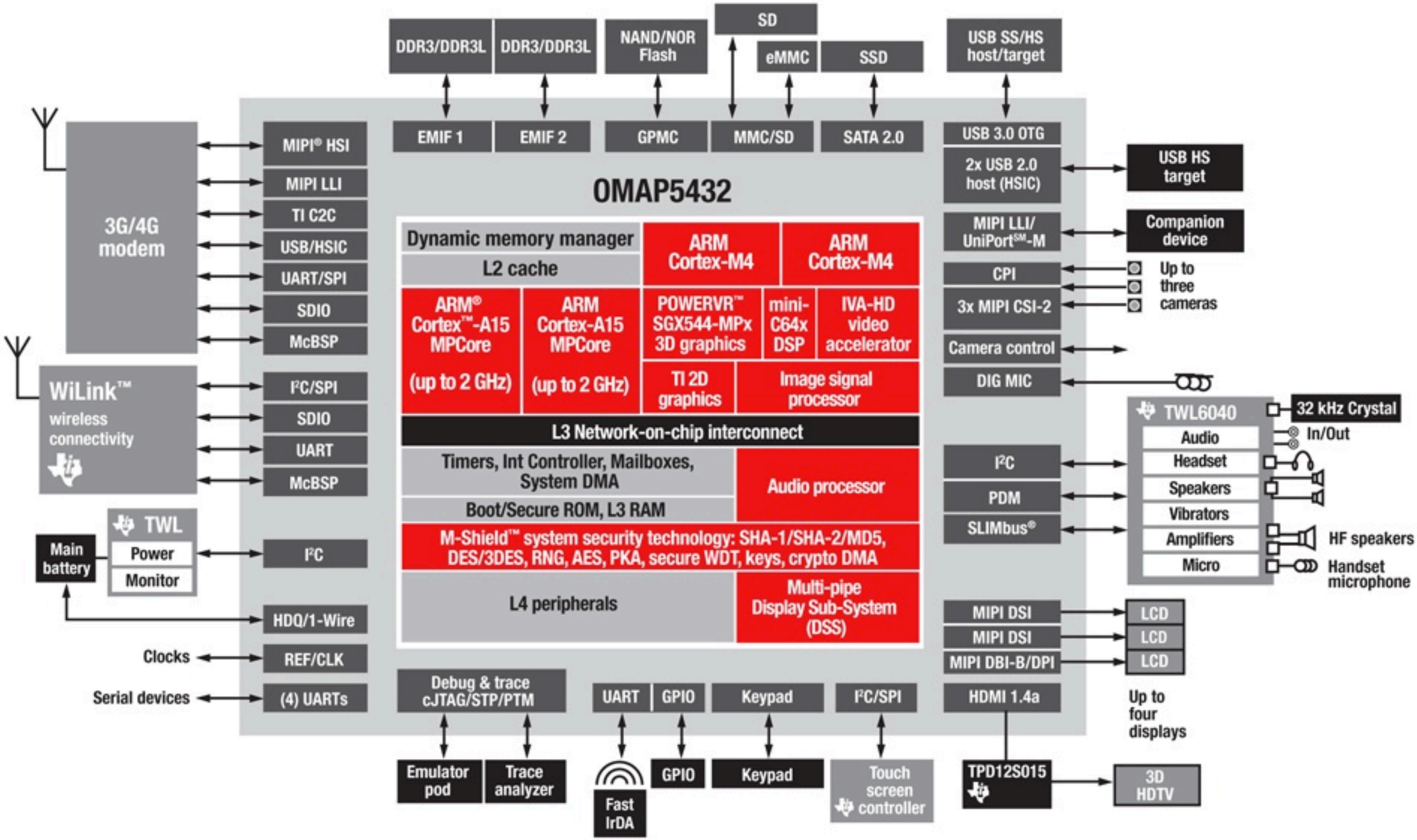


Image credit: TI

Performance 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.

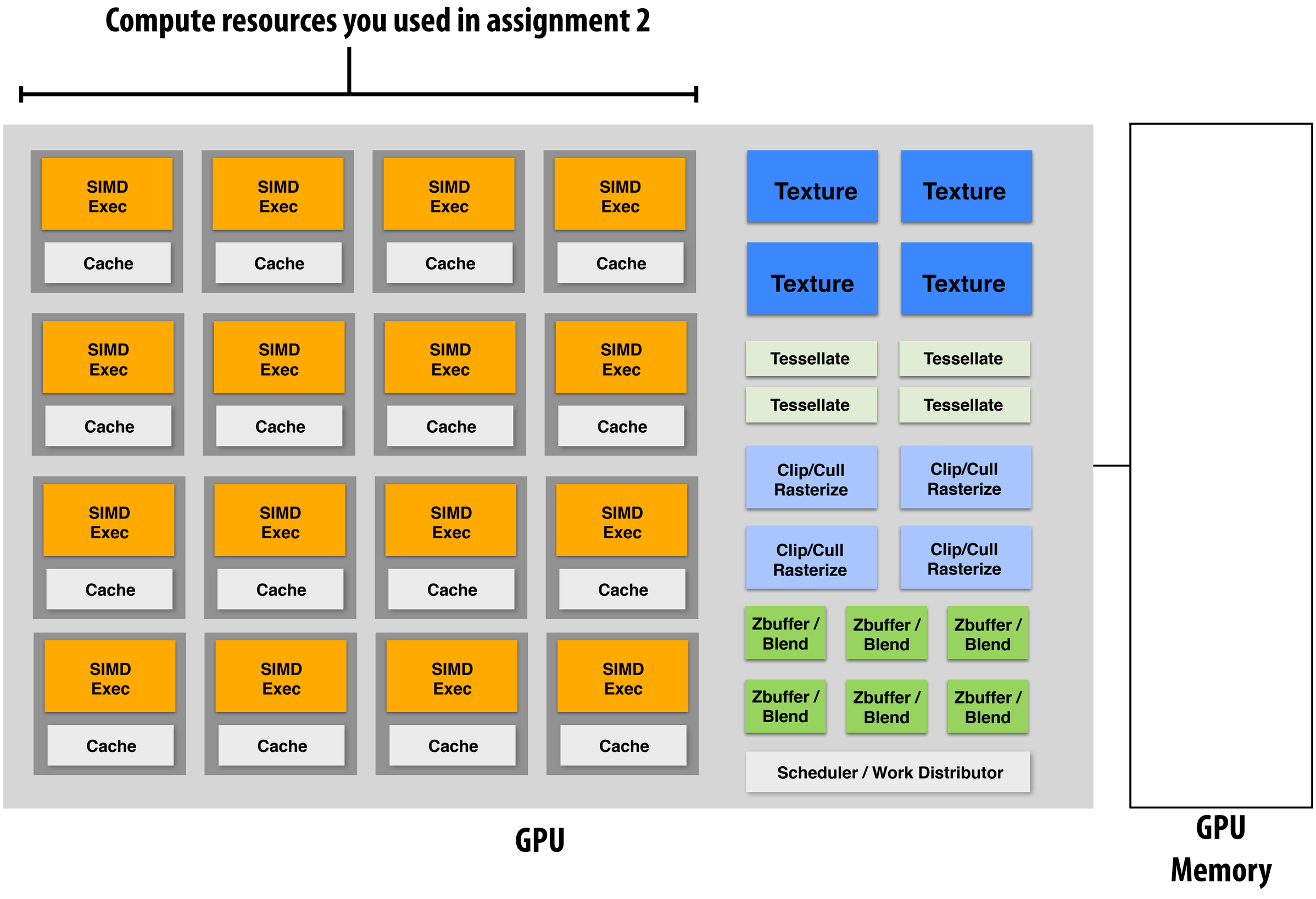
When websites re-encode their videos using H.264, they can offer them without using Flash at all. They play perfectly in browsers like Apple's Safari and Google's Chrome without any plugins whatsoever, and look great on iPhones, iPods and iPads.

Steve Jobs' "Thoughts on Flash", 2010
http://www.apple.com/hotnews/thoughts-on-flash/

Demo: image processing on Nikon D7000

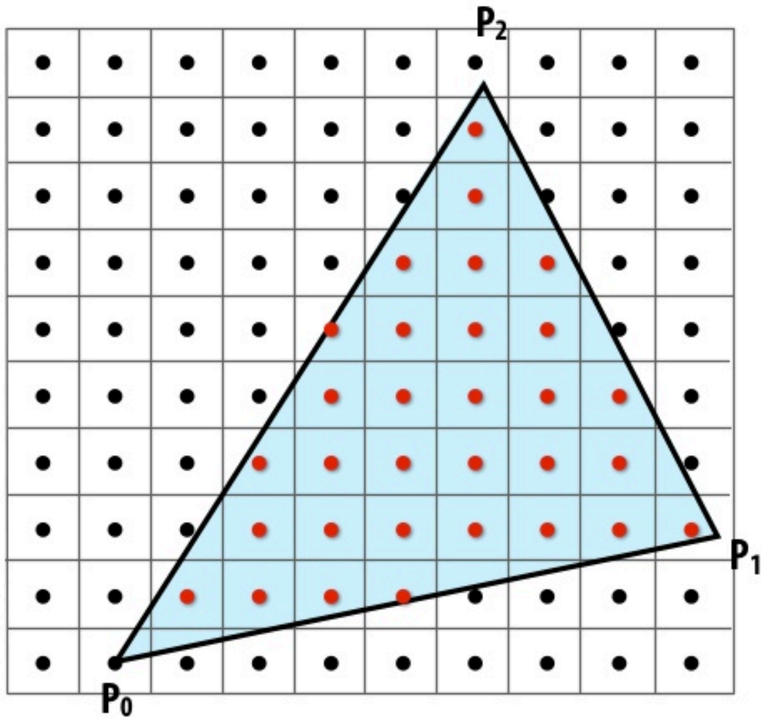
16 MPixel RAW image to JPG image conversion:
Quad-core Macbook Pro laptop: 1-2 sec
Camera: ~ 1/6 sec

GPU is itself a heterogeneous multi-core processor

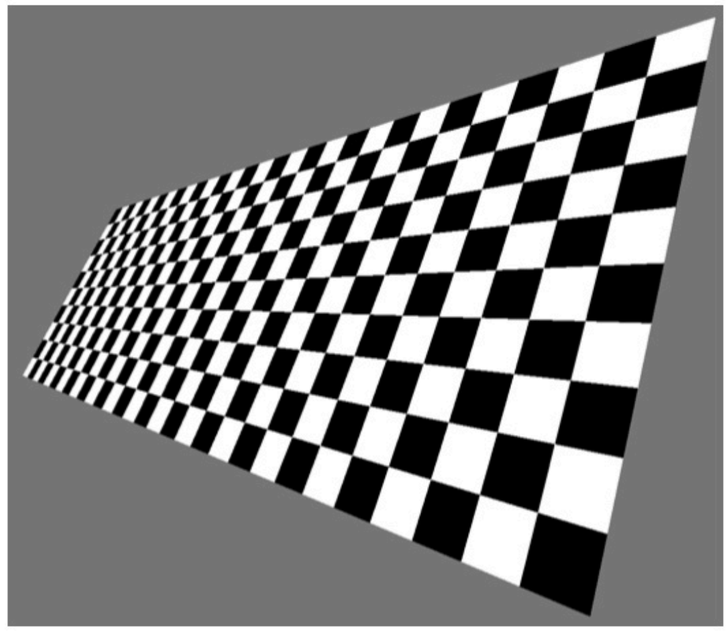


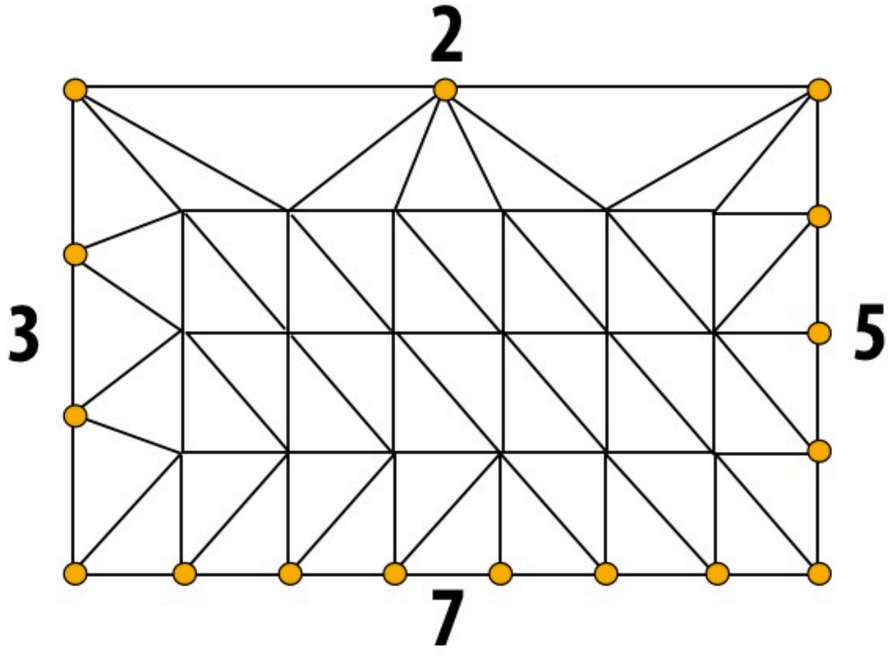
Example graphics tasks performed in fixed-function HW

Rasterization:
Determining what pixels a triangle overlaps



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

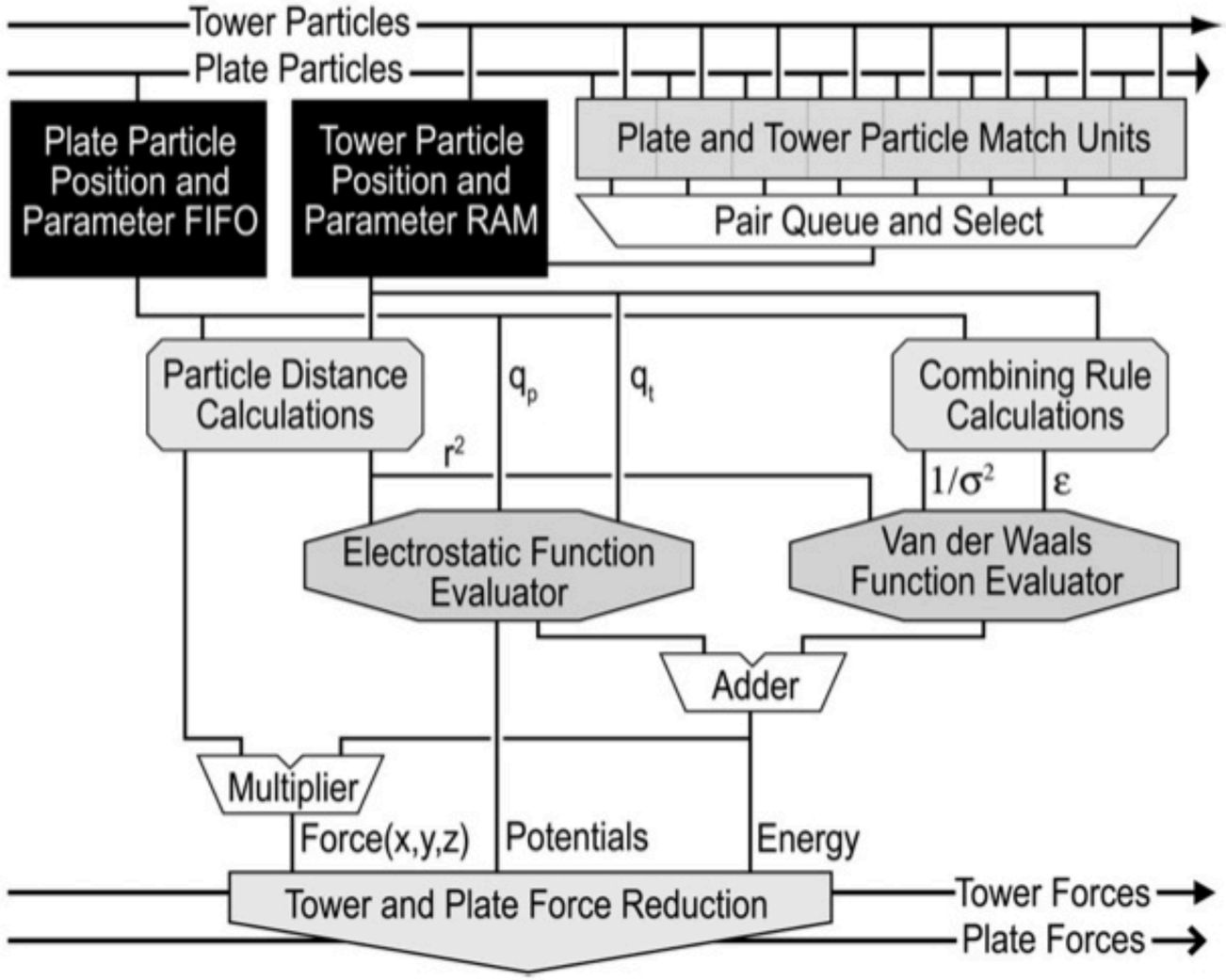




Geometric tessellation:
computing fine-scale geometry from coarse geometry

DESRES Anton supercomputer

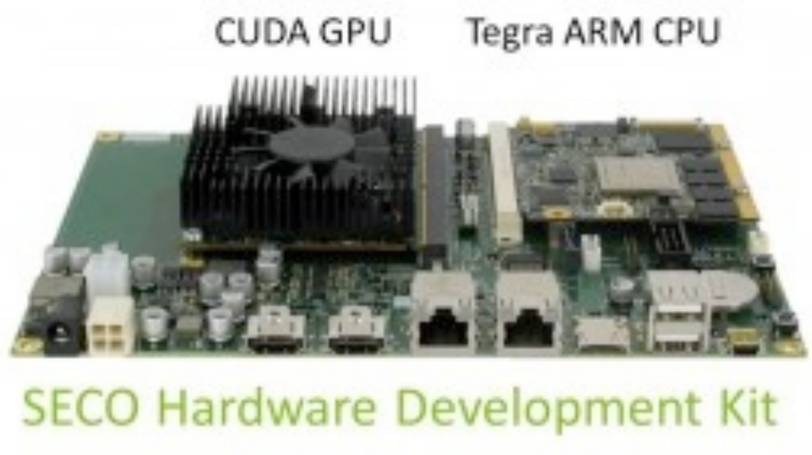
- Supercomputer highly specialized for molecular dynamics
  - Simulate proteins
- ASIC for computing particle-particle interactions (512 of them)
- Throughput-oriented subsystem for efficient fast-fourier transforms
- Custom, low-latency communication network



ARM + GPU Supercomputer

- Observation: 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 + GPU cores
- Goal: 7 GFLOPS/Watt efficiency
- Project underway at Barcelona Supercomputing Center

  http://www.montblanc-project.eu

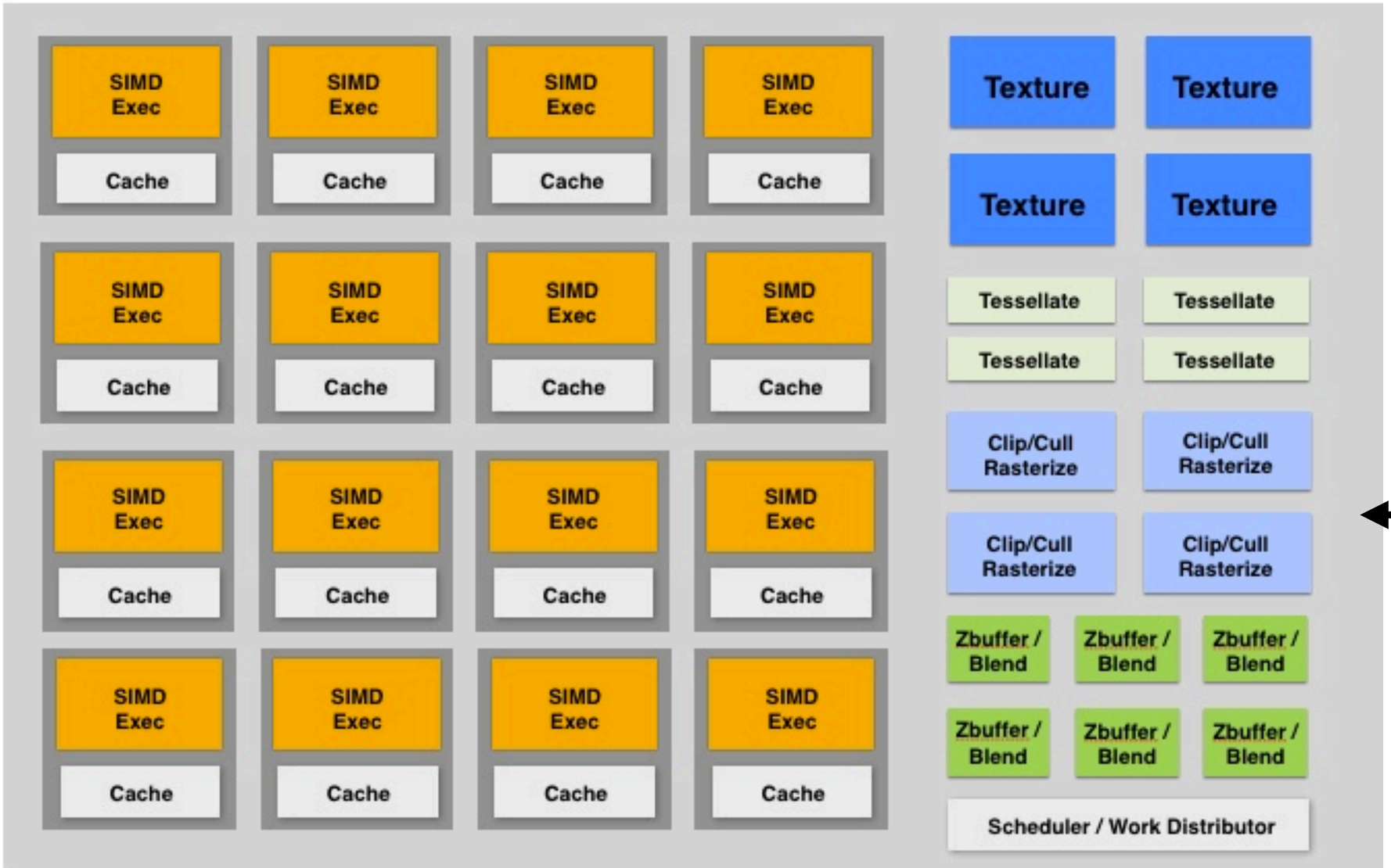


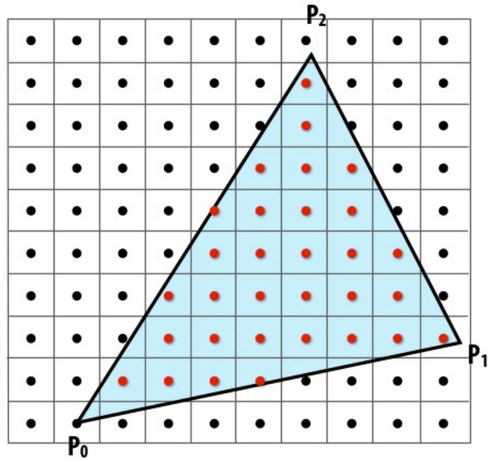
Challenges of heterogeneity

- To date in course:
  - Goal: to get best speedup, keep all processors busy
  - Homogeneous system: every processor can be used for every task
- Heterogeneous system: preferred processor for each task
  - Challenge for system designer: what is the right mixture of resources?
    - Too few throughput-oriented resources (fast sequential processor is underutilized--- should have used resources for more throughput cores)
    - Too little sequential processing resources (bit by Amdahl's Law)
    - How much chip area should be dedicated to a specific function, like video? (these are resources taken away from general-purpose processing)
- Work balance must be anticipated at chip design time

GPU heterogeneity design challenge

[Molnar 2010]





Say 10% of the computation is rasterization. (most of graphics workload is computing color of pixels)
Consider the error of under-provisioning fixed-function component for rasterization.
(1% of chip used for rasterizer, really needed 1.2%)

Problem is that if rasterization is bottleneck, the expensive programmable processors 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 (fast sequential processor is underutilized)**
    - **Too little sequential processing resources (bit by Amdahl's Law)**
    - **How much chip area should be dedicated to a specific function, like video? (these are resources taken away from general-purpose processing)**
    - **Work balance must be anticipated at chip design time**
      - **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?**
    - **Makes scheduling decisions complex**
    - **Mixture of resources can dictate choice of algorithm**
    - **Software portability nightmare**

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, desktop processors, supercomputers

- Traditional rule of thumb in system design is to design simple, general-purpose components. This is not the case with emerging processing systems (perf/watt)

- 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?