6.173
Fall 2010
Agarwal, Kaashoek, Morris, Terman
Lecture 1

Multicore Systems Laboratory

In This Course You Will

- Learn how to design and build multicore computers
- Learn how to write parallel programs
- Learn Verilog – a popular hardware design language
- Learn hardware design using FPGAs (field programmable gate arrays)

Adminstrivia

http://web.mit.edu/6.173

Lecturers: Anant Agarwal, Frans Kaashoek, Robert Morris, Chris Terman

TAs: Jonathan Eastep, Zviad Metreveli

Course admin: Cree Bruins

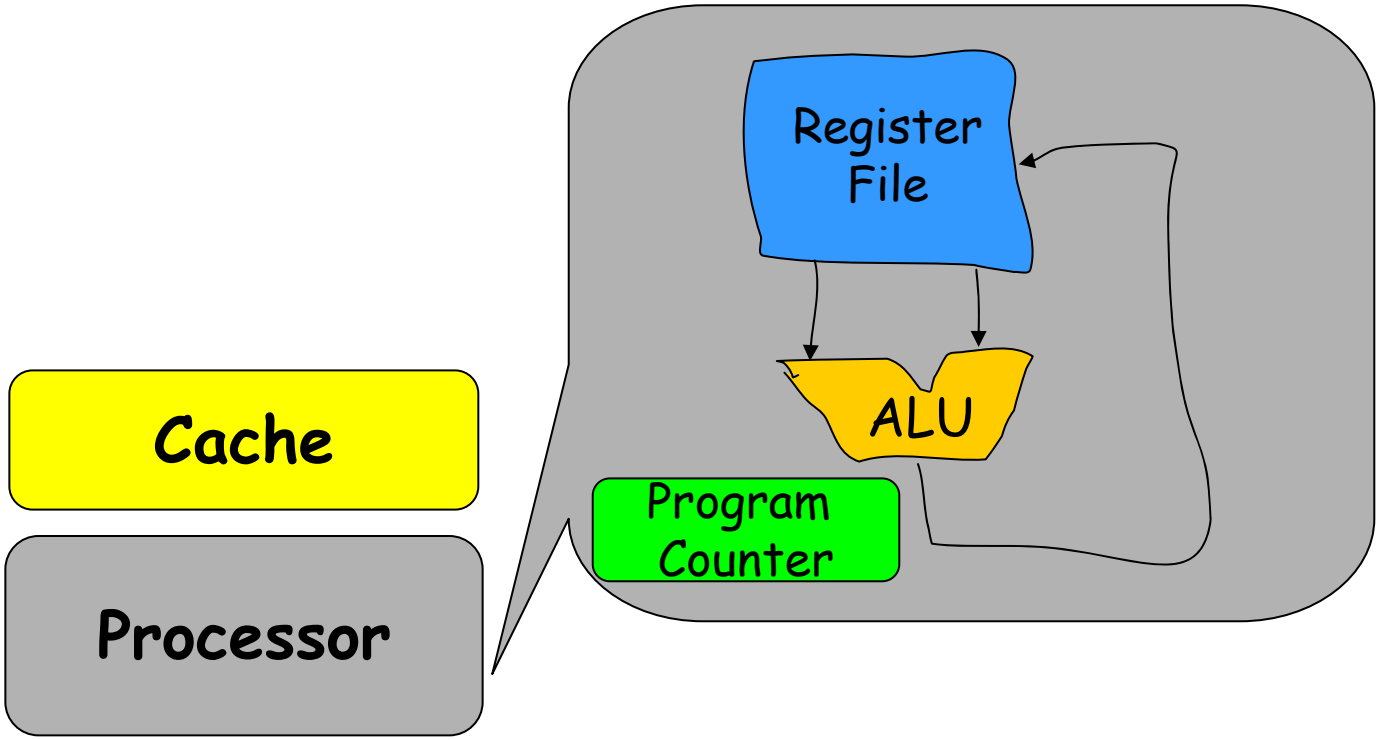
Grade based on:

final project	- 30%
6 labs	- 30%
2 quizzes	- 30%
class participation	- 10%

Our Approach in this Course

- Learn about general concepts in parallel computing – hardware, software, programming
- Apply concepts in hands-on hardware laboratory in which you will design and program a multicore computer by modifying an existing FPGA based design (Beehive)

What We Assume You Know



From 6.004

Outline of this Lecture

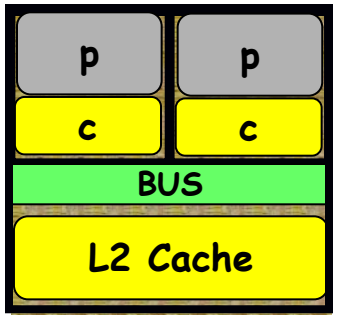
- What is multicore?
- What's the Problem with Single Core?
- Why multicore?
- Where is multicore going?
- Challenges of multicore

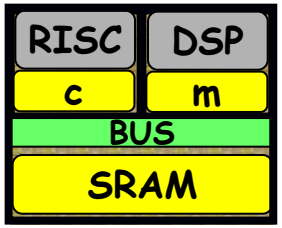
- Overview of the Lab and the Beehive infrastructure

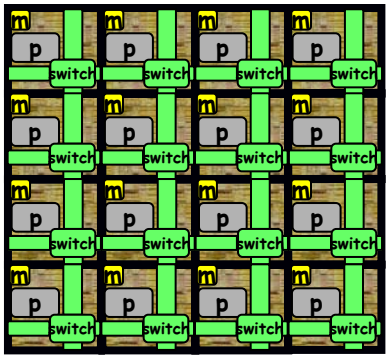
What is Multicore?

Multicore satisfies three properties

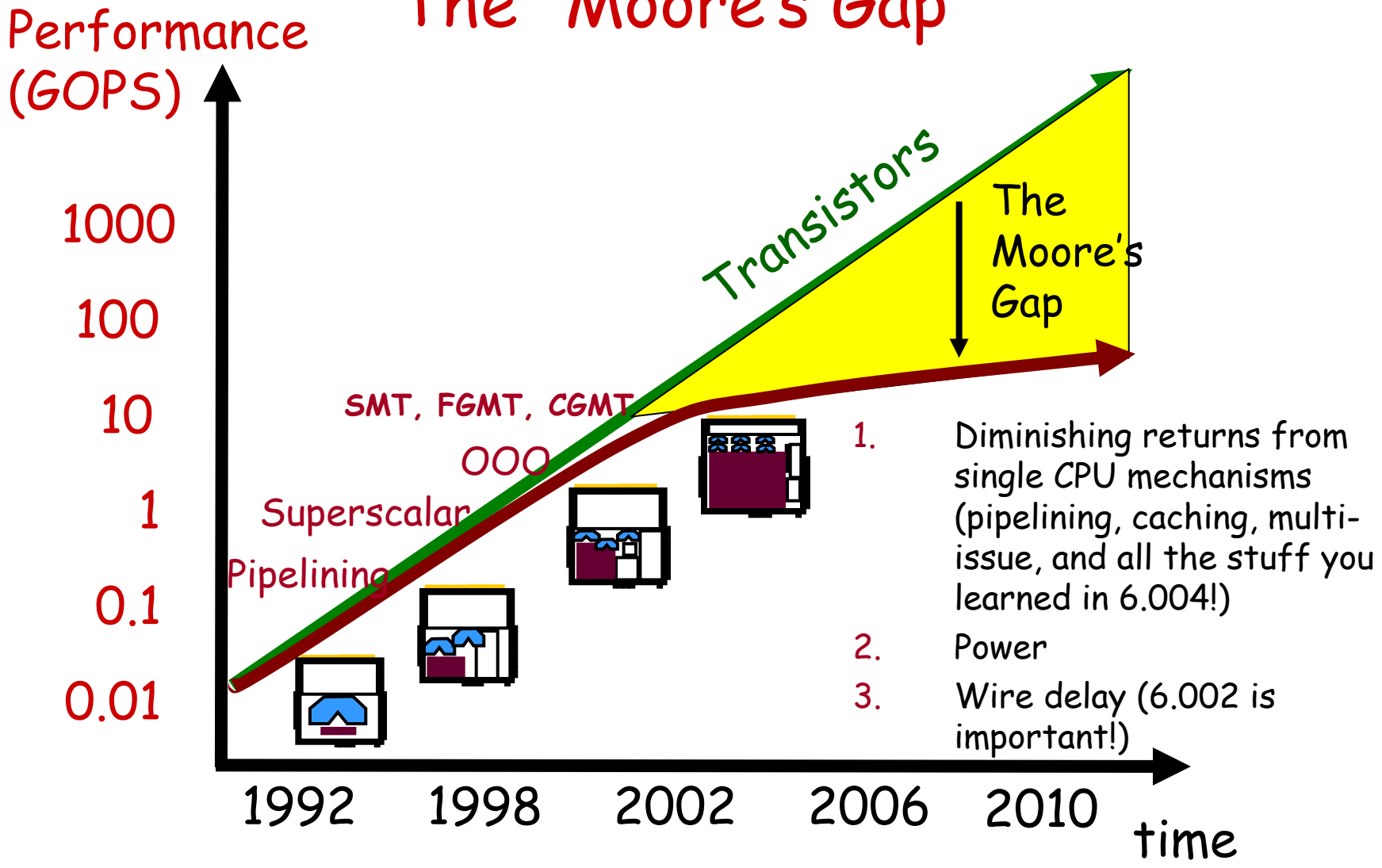
- Single chip
- Multiple distinct processing engines
- Multiple, independent threads of control (or program counters)





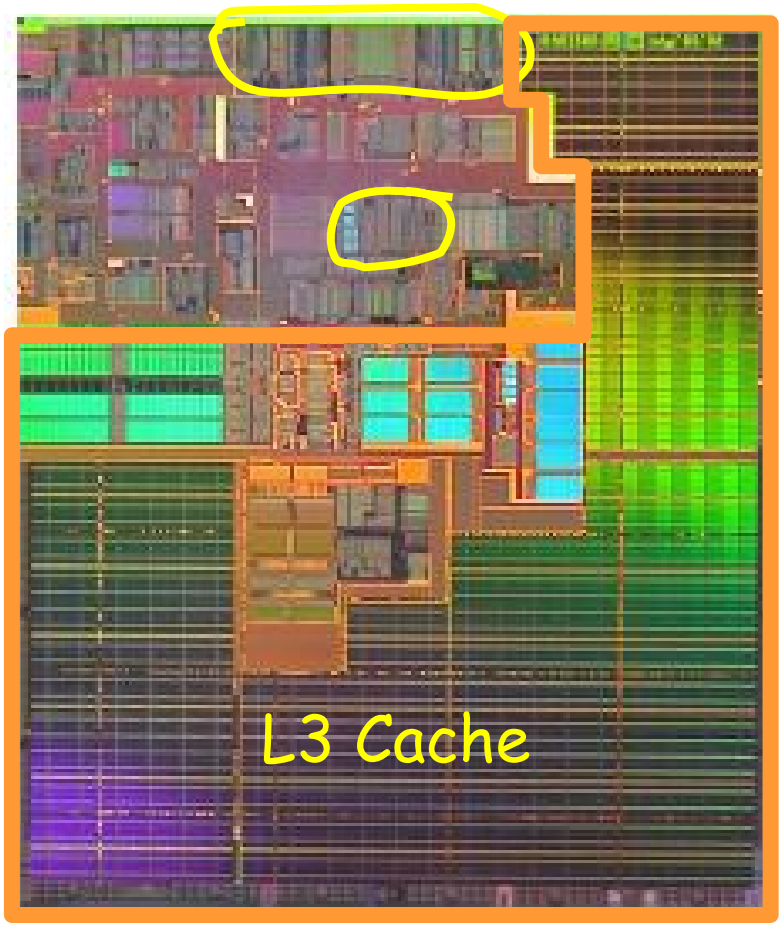


The Problem with Single Core
The "Moore's Gap"



1. Diminishing returns from single CPU mechanisms (pipelining, caching, multi-issue, and all the stuff you learned in 6.004!)
2. Power
3. Wire delay (6.002 is important!)

1. Diminishing Returns from Caches



Madison Itanium2
0.13μm

Less than 4% to
ALUs and FPUs

Majority of die area
devoted to cache

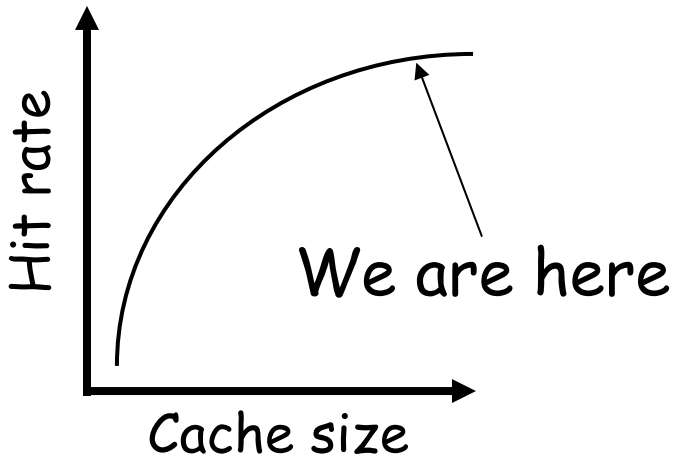
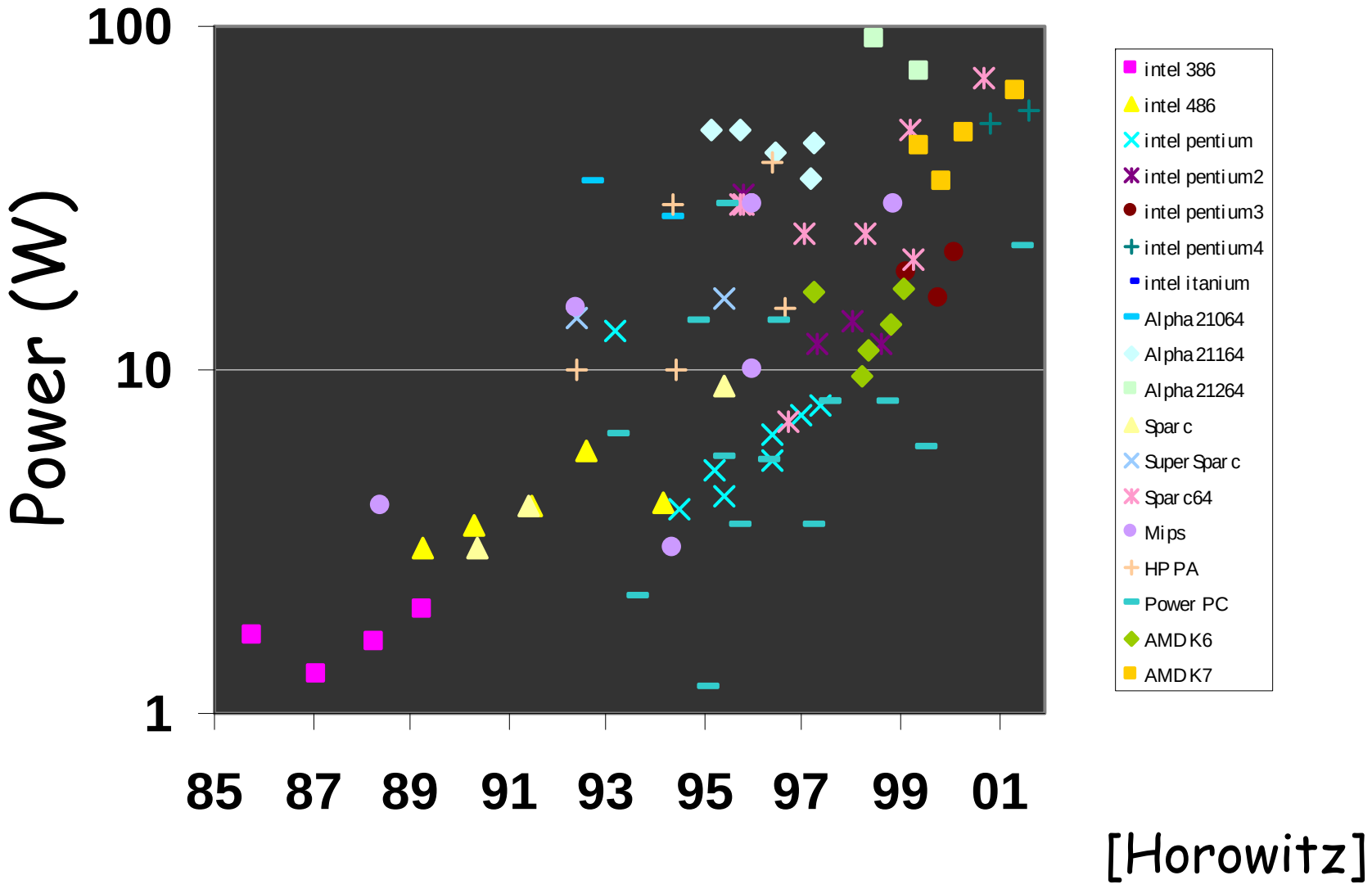


Photo courtesy Intel Corp.

2. Hitting Power Limits

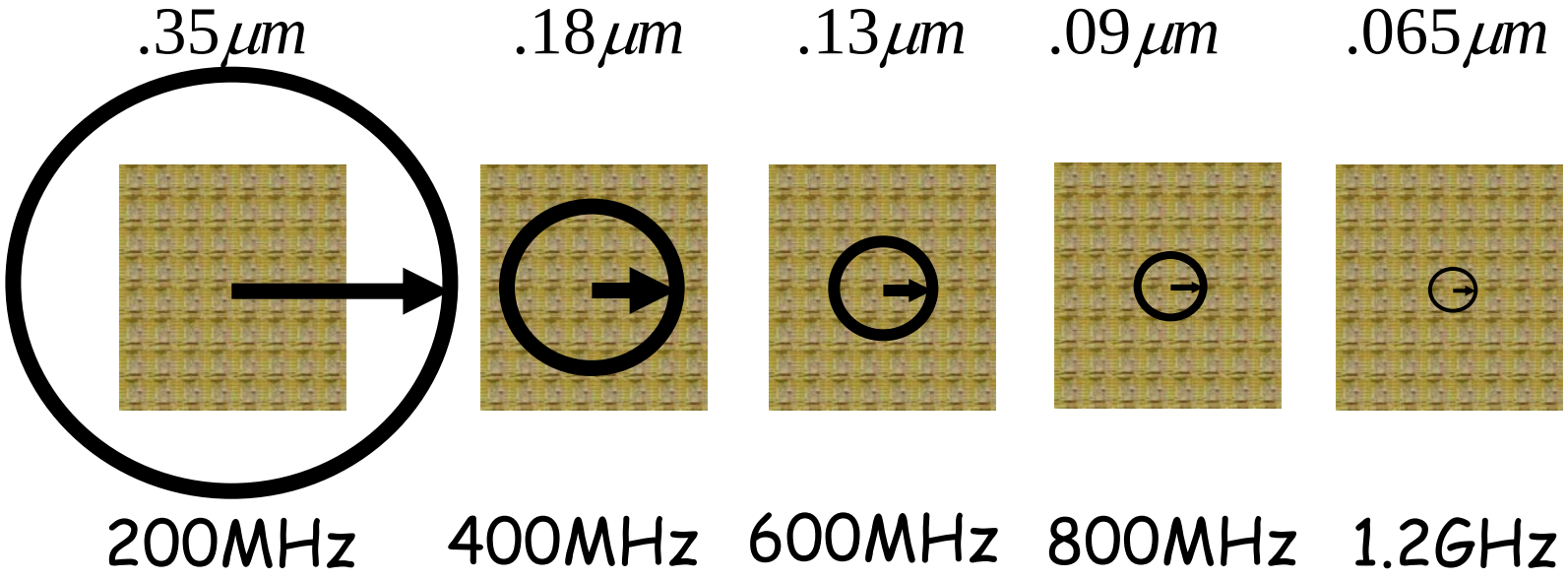


[Horowitz]

Hitting the Power Wall

- $P \propto CV^2f$ (from 6.002)
- Already at 100W chip; cannot go much higher
- Reduce C? If chip area is more or less constant, then C is more or less constant
- Reduce f? Will reduce power, but performance goes down! So, cannot reduce f significantly without losing performance
- Reduce V? Will reduce power, but will reduce f; So, cannot reduce V
- Stuck

3. Wire Delays Make Bigger Processors Hard to Scale



Signal reach in a single clock cycle

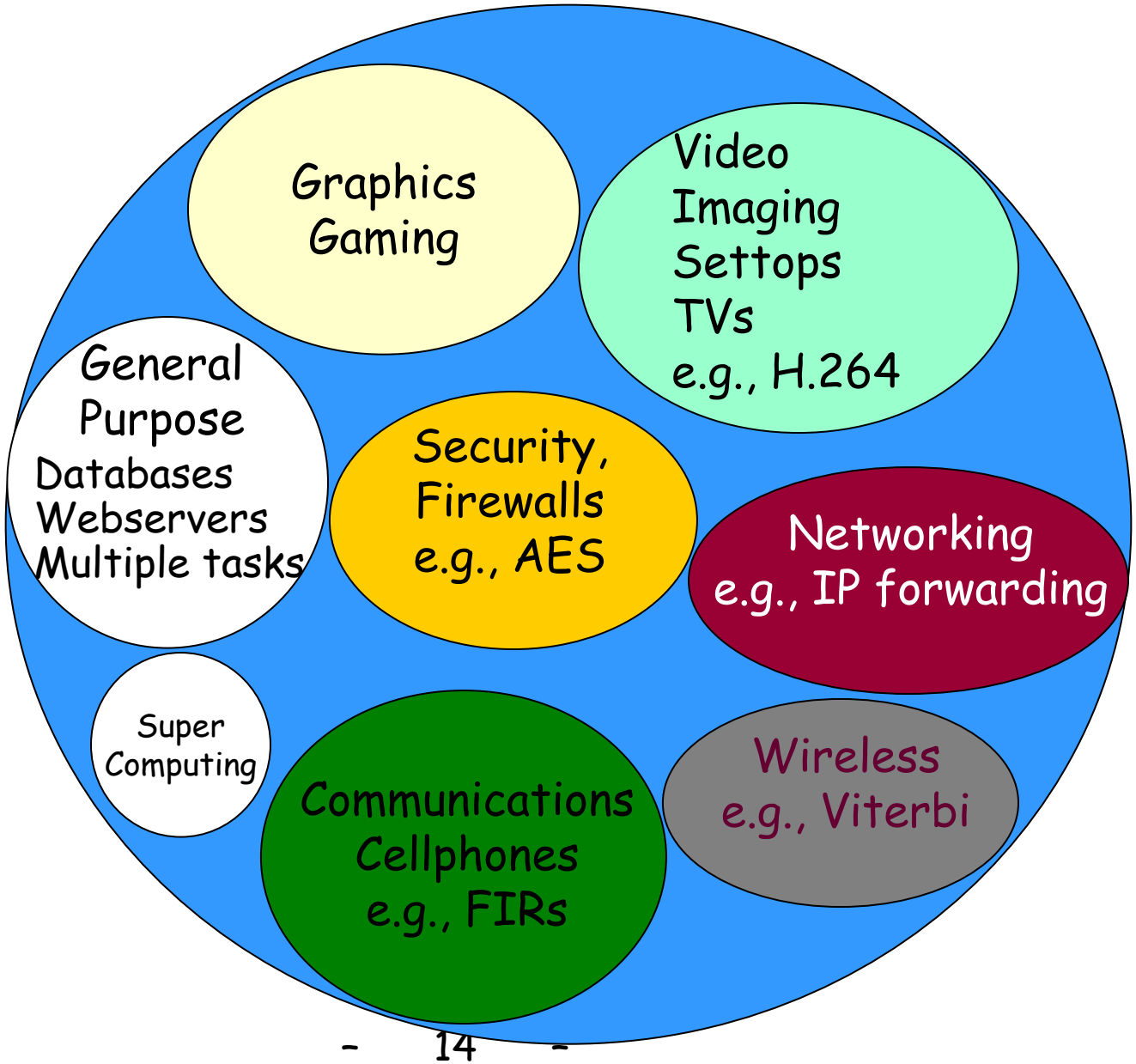
Large structures common to sequential processors do not scale

Why Multicore?

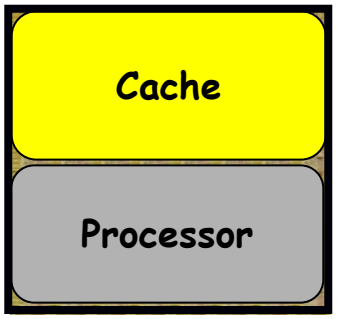
Let's look at the opportunity from two viewpoints

- A. Performance
- B. Power efficiency

Parallelism is Everywhere



A. The Performance Opportunity



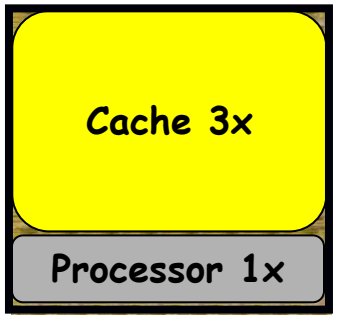
90nm

CPI: 1 + 0.01 x 100 = 2

1 + m x T

Smaller CPI (cycles per instruction) is better

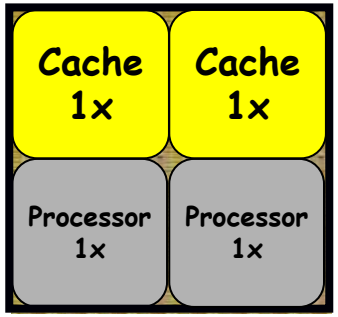
Push single core



65nm

CPI: 1 + 0.006 x 100 = 1.6

Go multicore



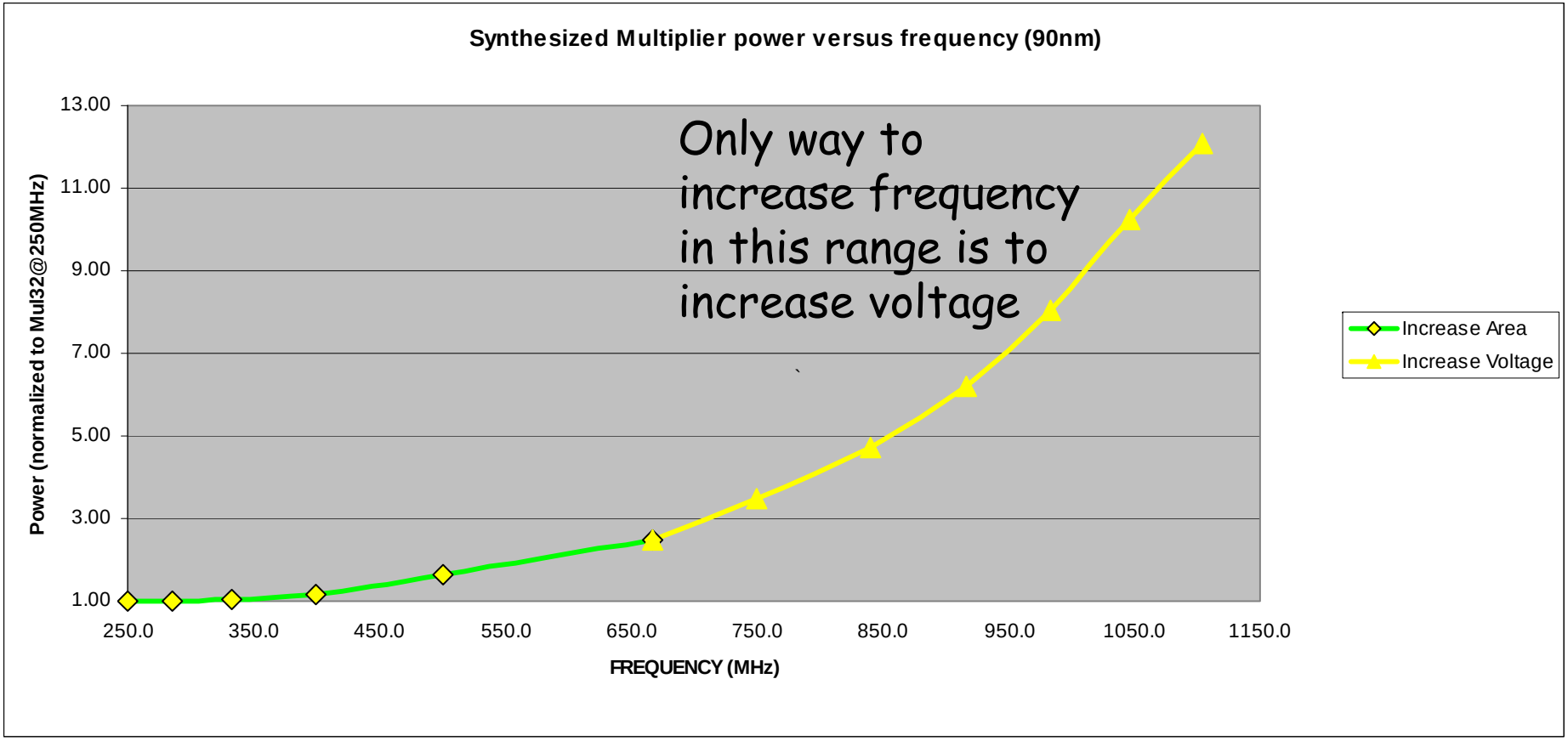
65nm

CPI: (1 + 0.01 x 100)/2 = 1

Single processor mechanisms yielding diminishing returns

Prefer to build two smaller structures than one very big one

B. The Power Cost of Frequency



Frequency $\propto V$

Power $\propto V^3$ (V^2F)

For a 1% increase in freq, we suffer a 3% increase in power

Multicore's Opportunity for Power Efficiency

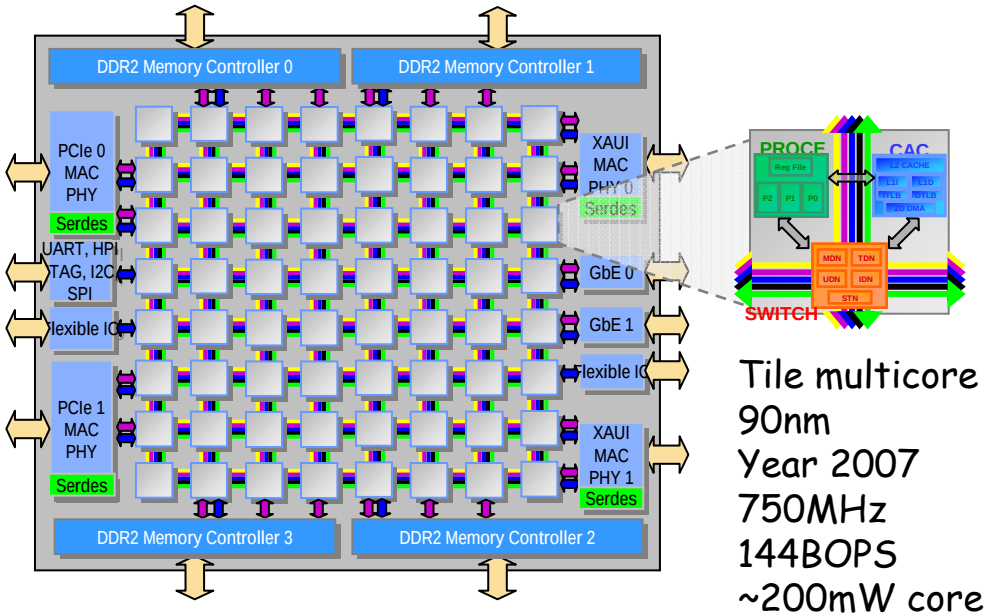
	Cores	Freq	V	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X
Multicore	2X	0.75X	0.75X	1.5X	0.8X	1.88X

(Bigger # is better)

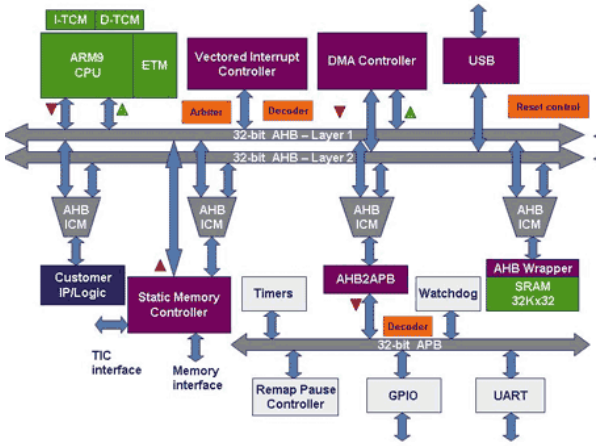
50% more performance with 20% less power

Preferable to use multiple slower devices, than one superfast device

Cores can be Power Efficient



Tile multicore
90nm
Year 2007
750MHz
144BOPS
~200mW core

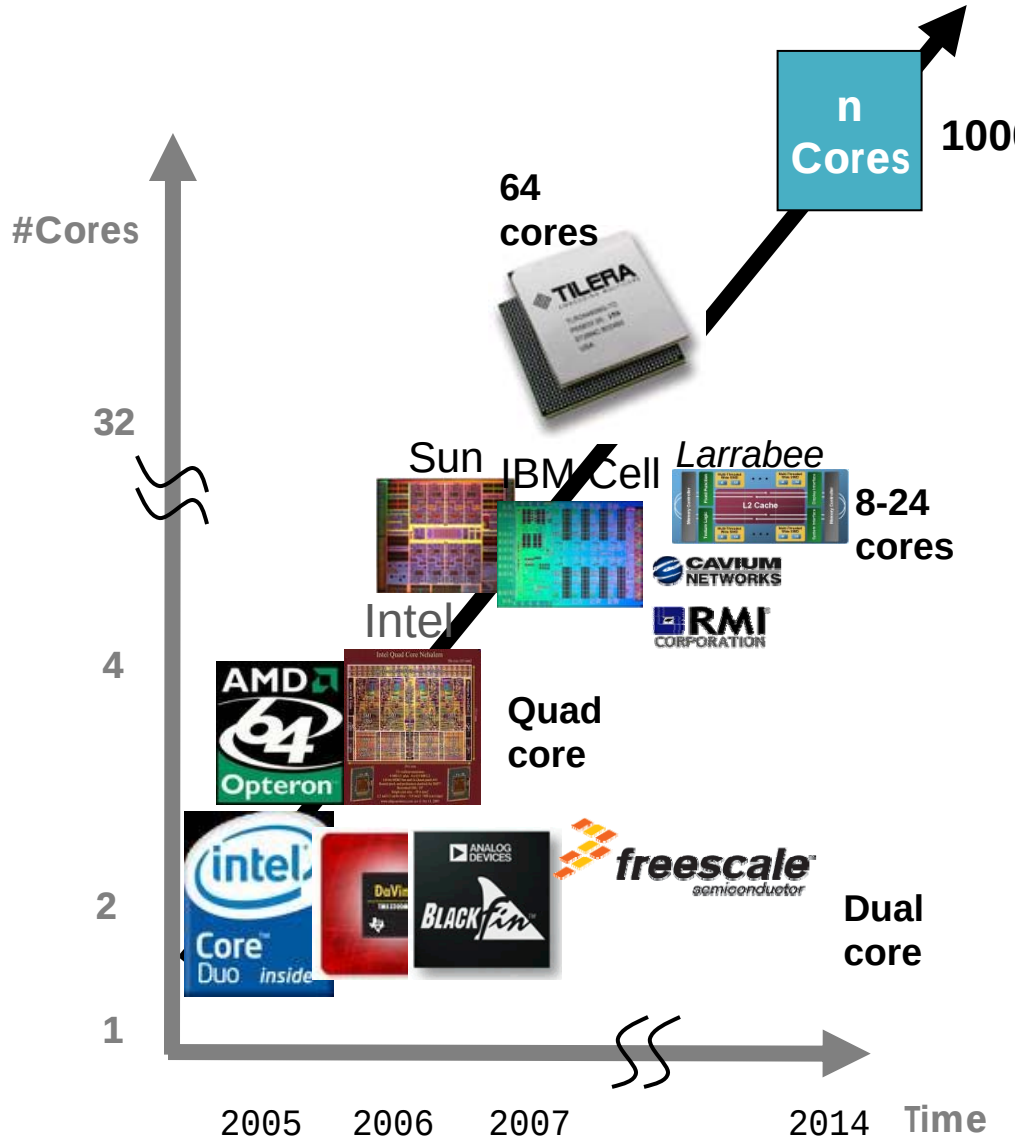


ARM core

These cores consume a few 100 milliwatts and clock at speeds in the vicinity of 1GHz

Compare to desktop cores which consume 10W or more per core and clock at speeds around 3GHz

Where is Multicore Going?



- Parallel computing no longer the province of a few rocket scientists – it is now mainstream
- The computing world is ready for radical change

The Future of Multicore

A corollary of Moore's law:
Number of cores will double every 18 months

Year	'02	'05	'08	'11	'14
Cores	**4**	**16**	**64**	**256**	**1024**

Vision for the Future

- The 'core' is the logic gate of the 21st century

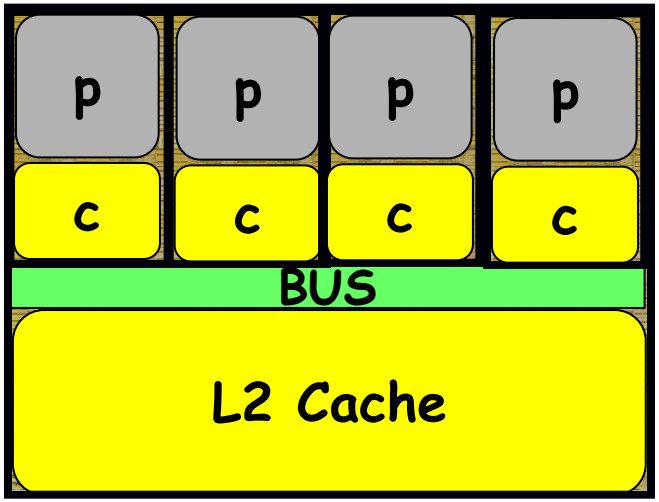
Multicore Challenges
The 3 P's

- P1: Performance challenge
- P2: Power efficiency challenge
- P3: Programming challenge

Rich area for academic or industry research and development
We will address each of these in this course

P1: Performance Challenge

- Interconnect is a new mechanism in multicore
- Cache coherence is another new mechanism
- Big performance impact



- We will cover interconnection networks and cache coherence in this course

P2: Power Efficiency Challenge

- Existing CPU cores at 0.2W to 10W
- 1000 CPU cores at 200W to 10 KWatts!
- Does not count interconnect and I/O power!

We will touch on power efficiency in this course
More about power in 6.846

P3: Multicore Programming Challenge

- "Multicore programming is hard". Why?
  - New
  - Misunderstood- some sequential programs are harder
  - Current tools are in the dark ages – Debugging 1000 cores with 1000 gdb windows is not an option!

Existing Programming Approaches – Reasonable, but Hopefully Better Ones in the Future

- **Cache-coherent shared memory (e.g., pthreads, TBB, Cilk, Wool)**
  - Easy to get started, convenient
  - Intel webinar likens pthreads to the assembly of parallel programming
  - Data races are hard to analyze, and no encapsulation or modularity
  - But evolutionary, and OK in the interim
- **Software shared memory (e.g., DMA with external shared memory)**
  - Hard to program, but DSP programmers favor this model
  - Explicit copying from global shared memory to local store, and explicit invalidation of items from local store
  - But, evolutionary, simple, modular and small core memory footprint
- **Message passing (e.g., MPI)**
  - Province of HPC users
  - Based on sending explicit messages between private memories
  - Lacks the convenience of shared memory
  - Modular, but has high overheads
- **Streams**
  - Great for producer-consumer or client-server models
  - Modular
  - Lacks the convenience of shared memory

You will see variants of the first three in this course

Summary

- Sequential processors have run out of steam
- Multicore can close the "Moore's Gap"
- Industry is aggressively pursuing multicore
- Most developed code will be parallel in the future
- We will see 1000 core manycores in a few years
- But there are many fundamental challenges
- We have our work cut out for us

Outline of this Course

- Overview of parallel architectures
- Message passing architectures and programming
- Application parallelization approaches and issues
  - Partitioning, placement, load balancing, locality
- Shared memory architectures and programming
  - Intro to software shared memory
  - Intro to cache coherent shared memory
- Overview of synchronization
- Latency tolerance
  - Multithreading, prefetching
- Caches and coherence
- Synchronization
- Interconnection networks

You will apply these concepts in 6 labs and a final project using Beehive
Next, intro to Beehive – our lab infrastructure