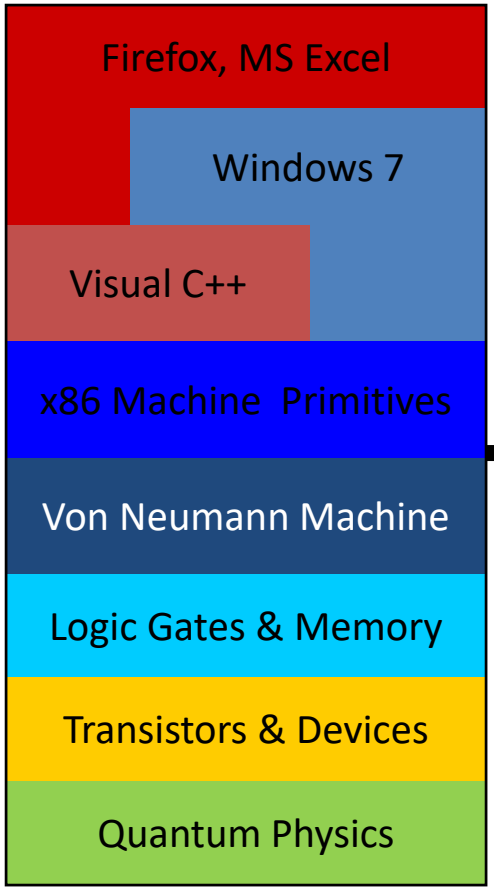
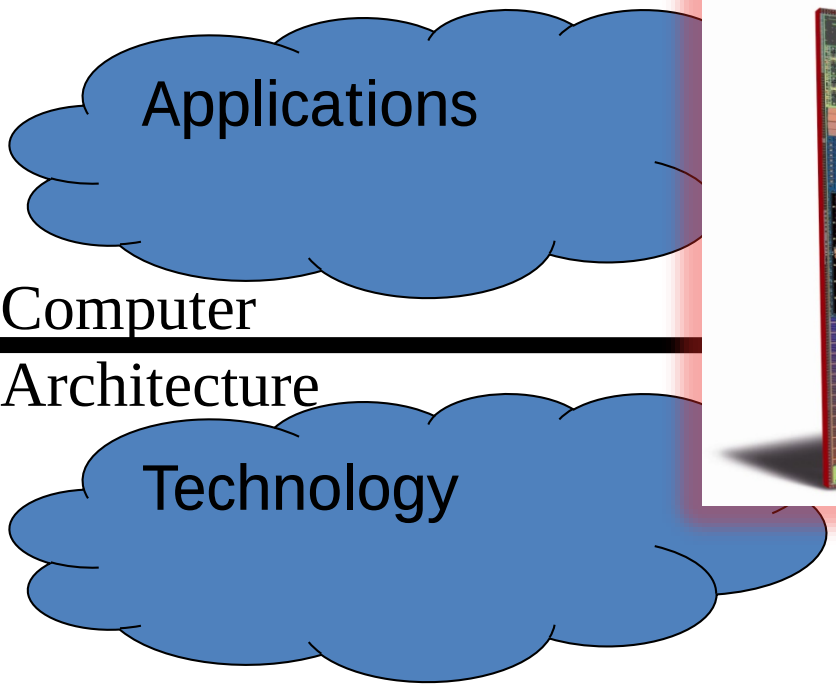
ECE/CS 752: Advanced Computer Architecture I

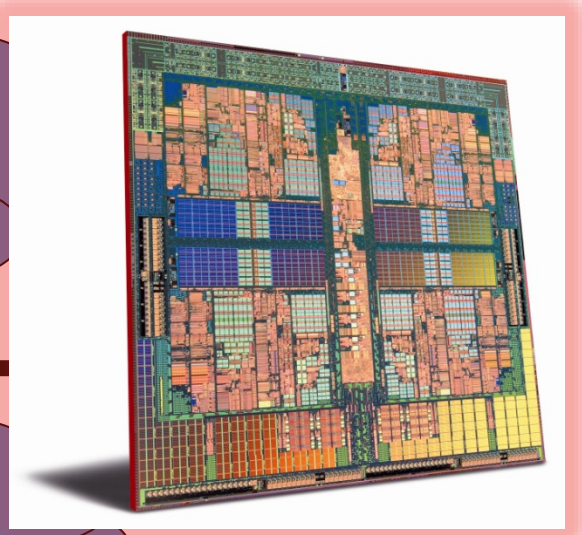
Fall 2017

Lecture notes based in part on slides created by John Shen, Ilhyun Kim, Mark Hill, David Wood, Guri Sohi, and Jim Smith, and others

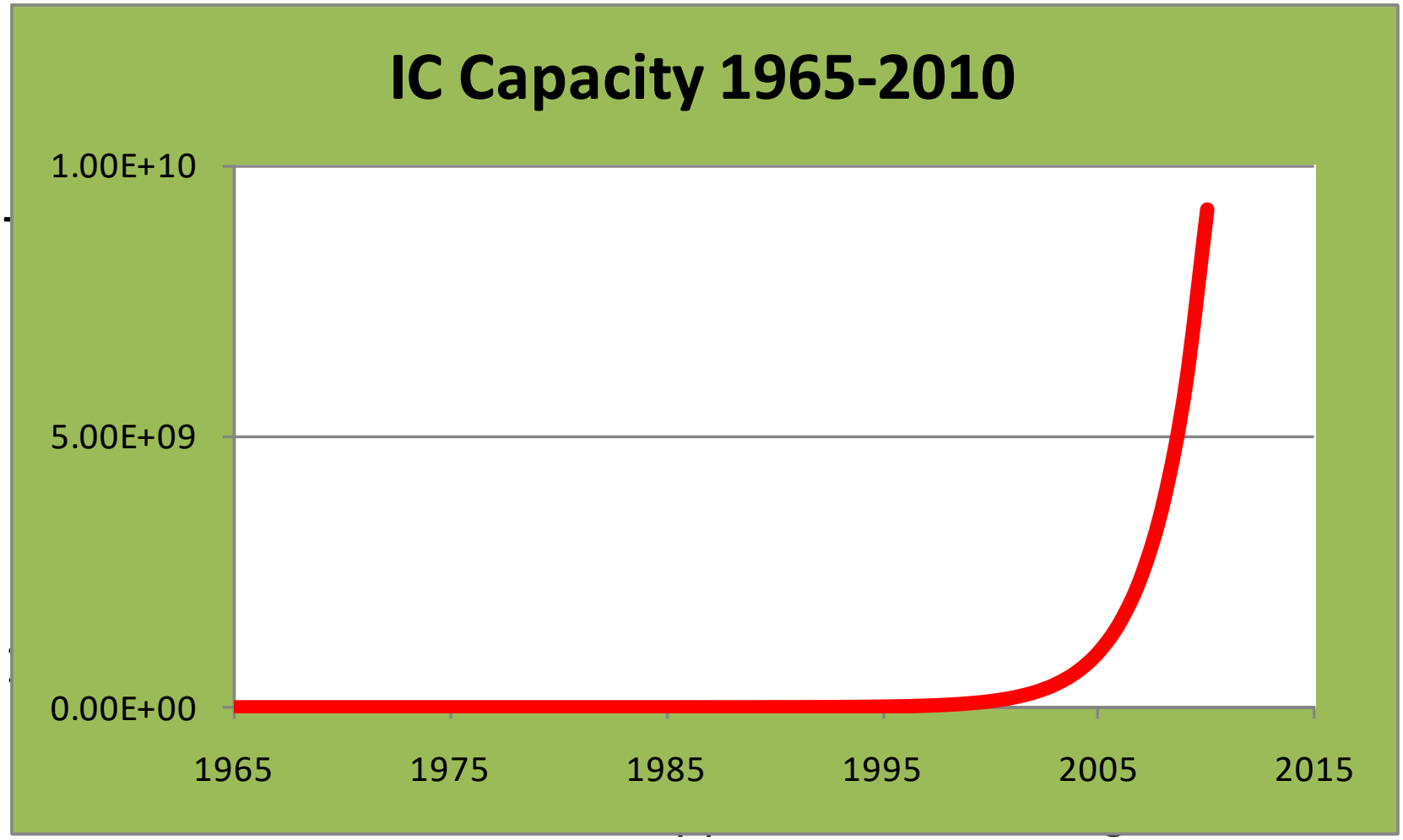
Computer Architecture







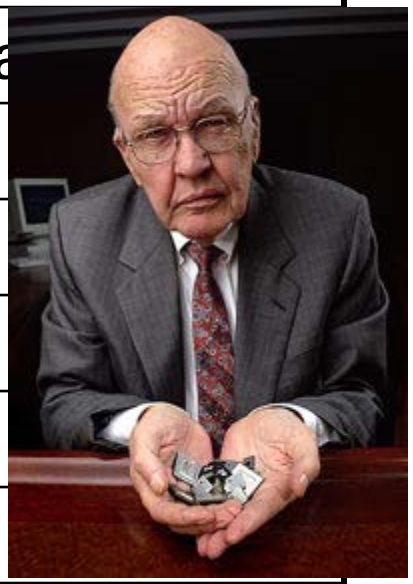
- Rely on *abstraction layers* to manage complexity
  - *Von Neumann Machine*



- Drives functionality, performance, cost
  - Exponential improvement for 50+ years

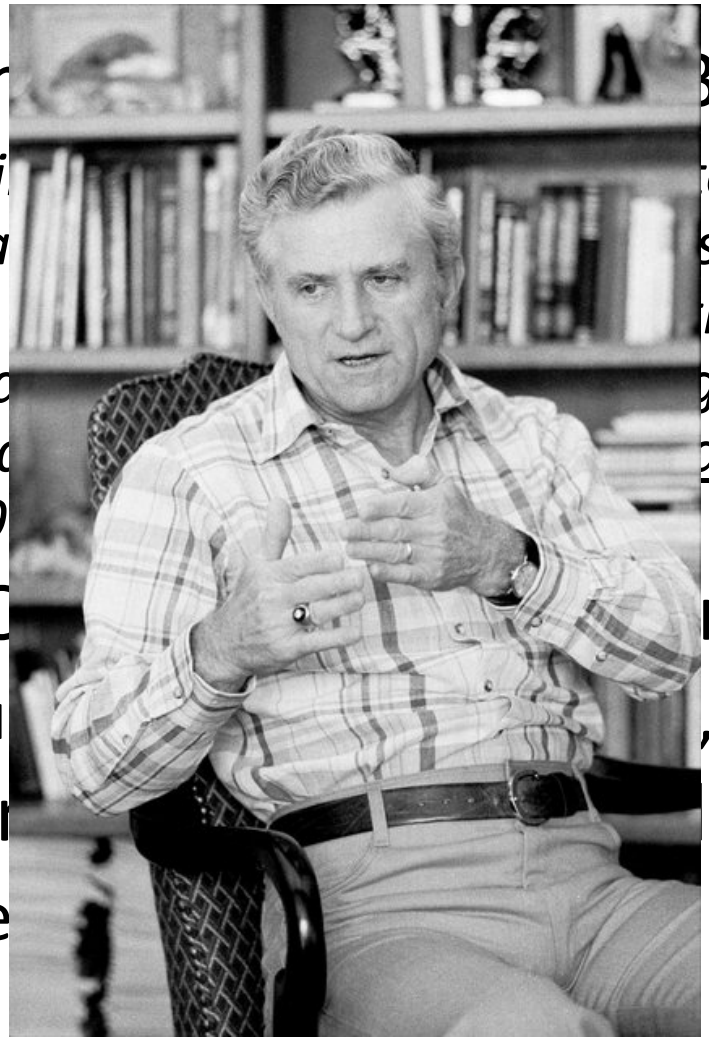
Semiconductor History

Date	Event	Comments
1947	1st transistor	Bell Labs
1958	1st IC 	Jack Kilby (MSEE '50) @TI Winner of 2000 Nobel prize
1971	1st microprocessor	Intel (calculator ma
1974	Intel 4004	2300 transistors
1978	Intel 8086	29K transistors
1989	Intel 80486	1M transistors
1995	Intel Pentium Pro	5.5M transistors
2006	Intel Montecito	1.7B transistors
2015	Oracle SPARC M7	10B+ transistors



Computer Architecture

- Instructio BM 360)
  - *… the attri tem as seen by the progra structure and functional he organization of the data gic design, and the physica ahl, Blaaw, & Brooks, 19*
- Machine C rchitecture)
  - ALUS, Bu , etc.
- Machine I ization)
  - Gates, ce



752 In Context

- Prior courses
  - 352 – gates up to multiplexors and adders
  - 354 – high-level language down to machine language interface or instruction set architecture (ISA)
  - 552 – implement logic that provides ISA interface
  - CS 537 – provides OS background (co-req. OK)
- This course – 752 – covers advanced techniques
  - Modern processors that exploit ILP
  - Modern memory systems that exploit MLP
- Additional courses
  - ECE 757 covers parallel and multiprocessing
  - ECE 755 covers VLSI design

Why Take 752?

- To become a computer designer
  - Alumni of this class helped design your computer
- To learn what is *under the hood* of a computer
  - Innate curiosity
  - To better understand when things break
  - To write better code/applications
  - To write better system software (O/S, compiler, etc.)
- Because it is intellectually fascinating!
  - What is the most complex man-made single device?

Computer Architecture

- Exercise in engineering tradeoff analysis
  - Find the fastest/cheapest/power-efficient/etc. solution
  - Optimization problem with 100s of variables
- All the variables are changing
  - At non-uniform rates
  - With inflection points
  - Only one guarantee: Today's right answer will be wrong tomorrow
- Two high-level effects:
  - Technology push
  - Application Pull

Technology Push

- What do these two intervals have in common?
  - 1776-1999 (224 years)
  - 2000-2001 (2 years)
- Answer: Equal progress in processor speed!
- The power of exponential growth!
- Driven by Moore's Law
  - Devices per chip doubles every 18-24 months
- Computer architects turn additional resources into
  - Speed
  - Power savings
  - Functionality

Performance Growth

Unmatched by any other industry !
[John Crawford, Intel]

- Doubling every 18 months (1982-1996): 800x
  - Cars travel at 44,000 mph and get 16,000 mpg
  - Air travel: LA to NY in 22 seconds (MACH 800)
  - Wheat yield: 80,000 bushels per acre

- Doubling every 24 months (1971-1996): 9,000x
  - Cars travel at 600,000 mph, get 150,000 mpg
  - Air travel: LA to NY in 2 seconds (MACH 9,000)
  - Wheat yield: 900,000 bushels per acre

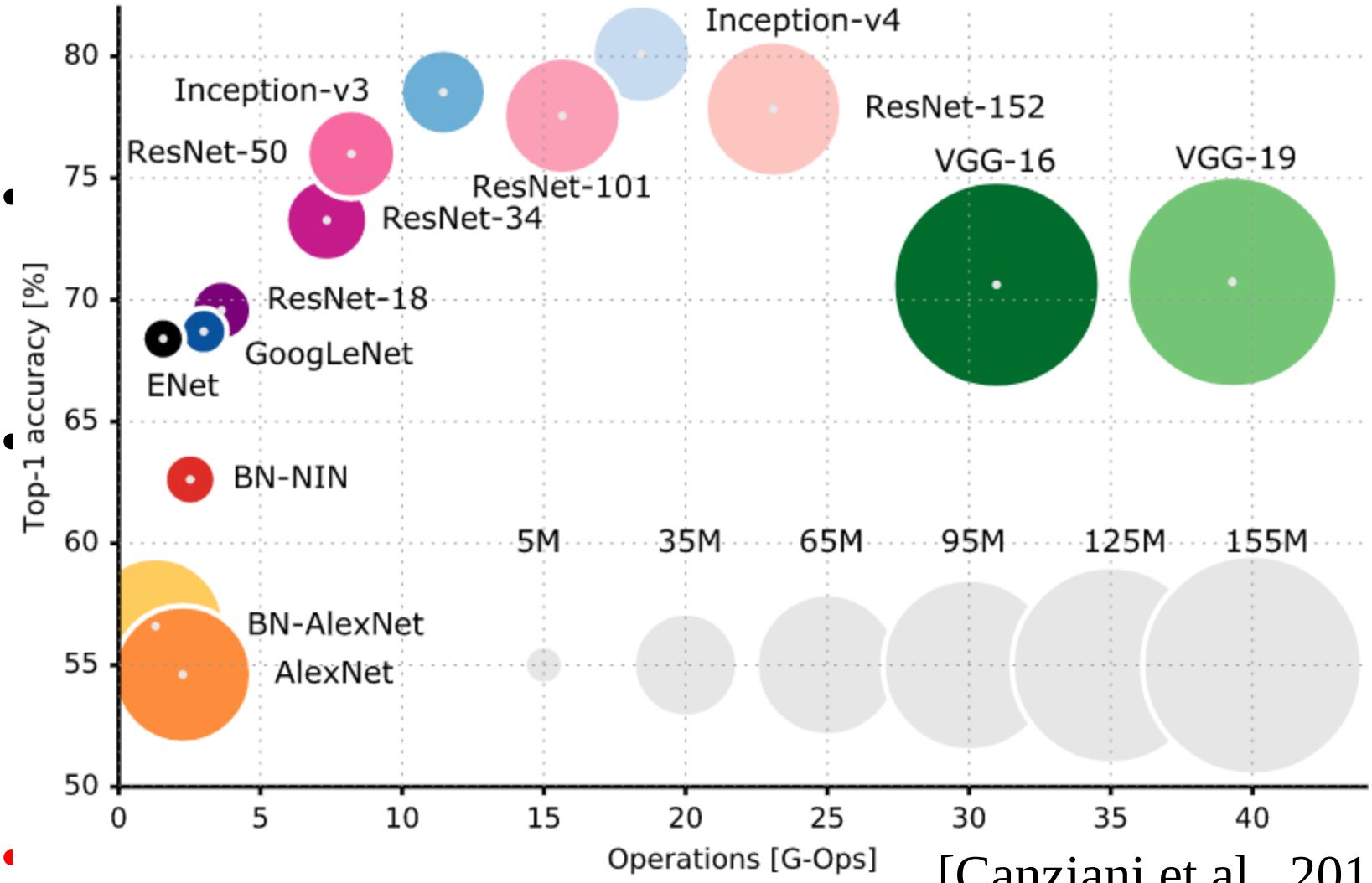
Technology Push

- Technology advances at varying rates
  - E.g. DRAM capacity increases at 60%/year
  - But DRAM speed only improves 10%/year
  - Creates gap with processor frequency!
- Inflection points
  - Crossover causes rapid change
  - E.g. enough devices for multicore processor (2001)
- Current issues causing an “inflection point”
  - Power consumption
  - Reliability, variability
  - Packaging innovations

Application Pull

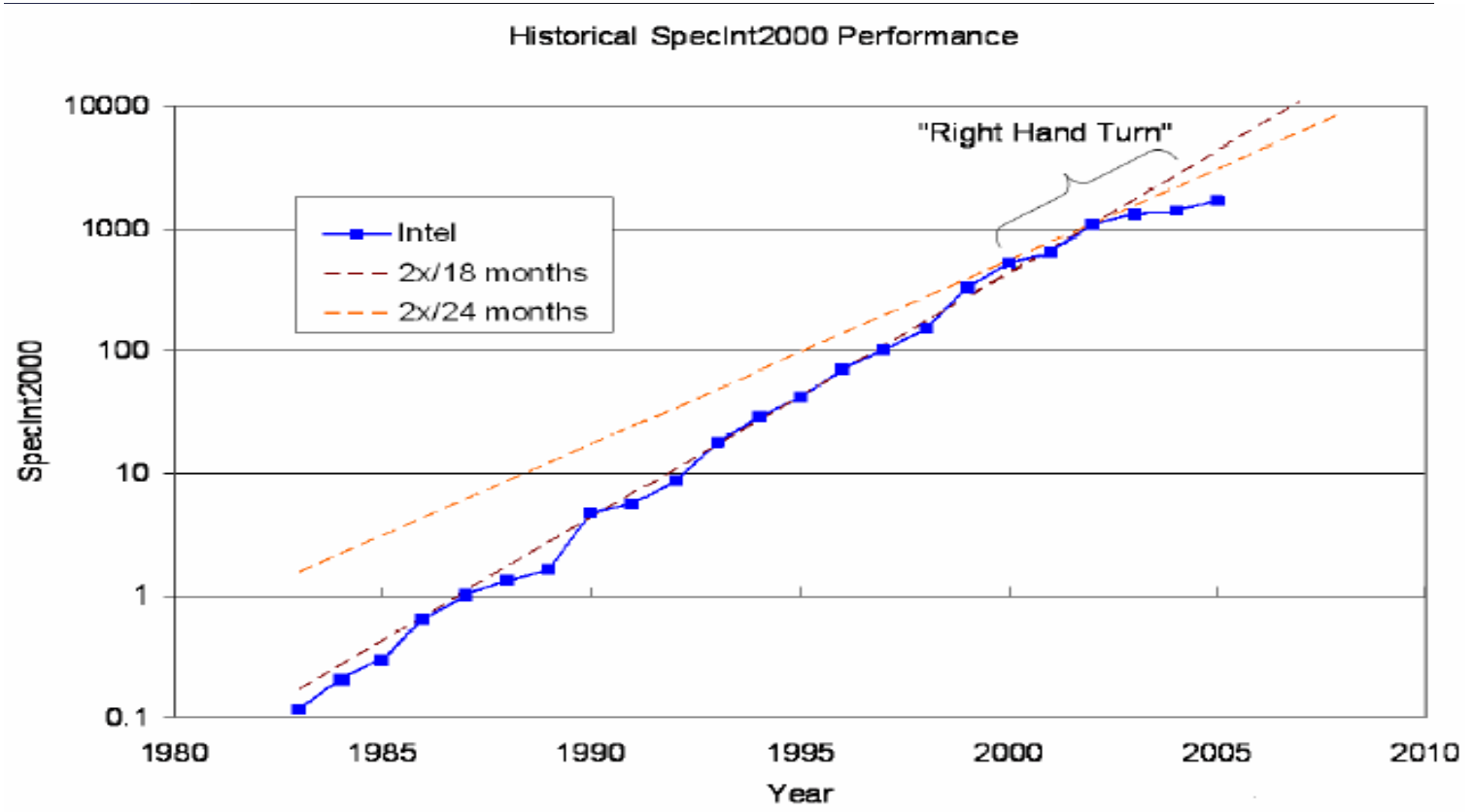
- Corollary to Moore's Law:
  Cost halves every two years

  In a decade you can buy a computer for less than its sales tax today. –Jim Gray
- Computers cost-effective for
  - National security – weapons design
  - Enterprise computing – banking
  - Departmental computing – computer-aided design
  - Personal computer – spreadsheets, email, web
  - Mobile computing – GPS, location-aware, ubiquitous
  - Wearable computing – activity/health monitoring, etc.
  - Voice web search



[Canziani et al., 2016]

Trends



- Moore's Law for device integration [source: Intel]
- Chip power consumption
- Single-thread performance trend

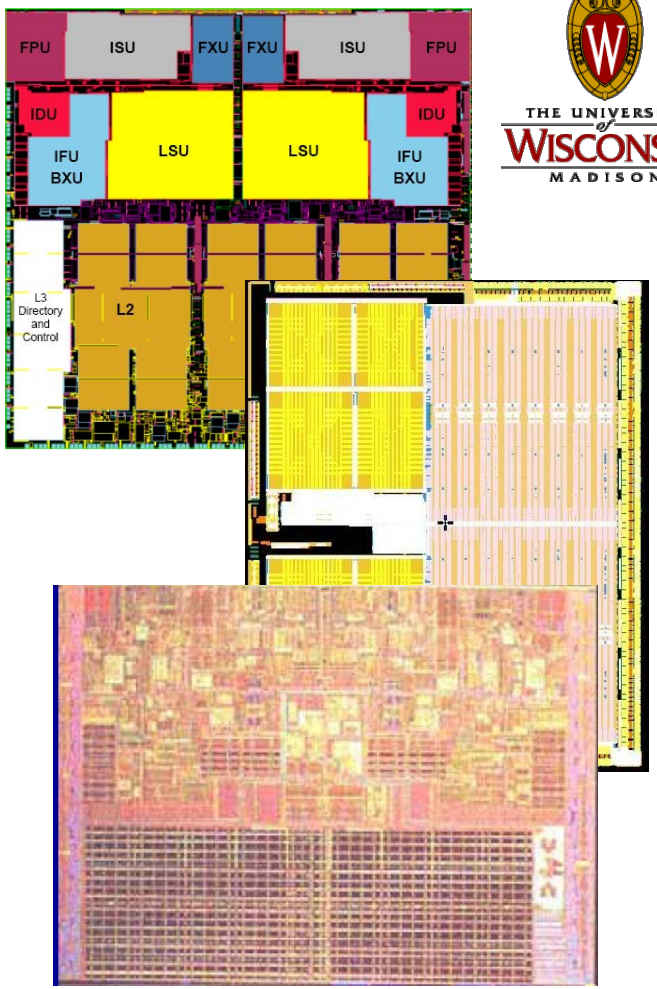
Dynamic Power

$$P_{dyn} \approx \sum_{i \in units} k_i C_i V^2 A_i f$$

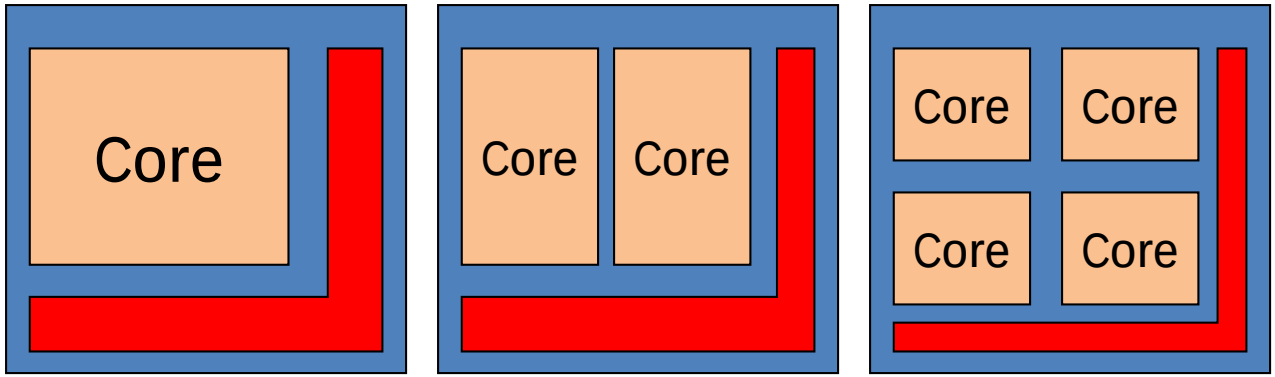
- Static CMOS: current flows when active
  - Combinational logic evaluates new inputs
  - Flip-flop, latch captures new value (clock edge)
- Terms
  - C: capacitance of circuit
    - wire length, number and size of transistors
  - V: supply voltage
  - A: activity factor
  - f: frequency
- Future: Fundamentally power-constrained

Multicore Mania

- First, servers
  - IBM Power4, 2001
- Then desktops
  - AMD Athlon X2, 2005
- Then laptops
  - Intel Core Duo, 2006
- Cellphones
  - Dual/quad/octo, big.LITTLE

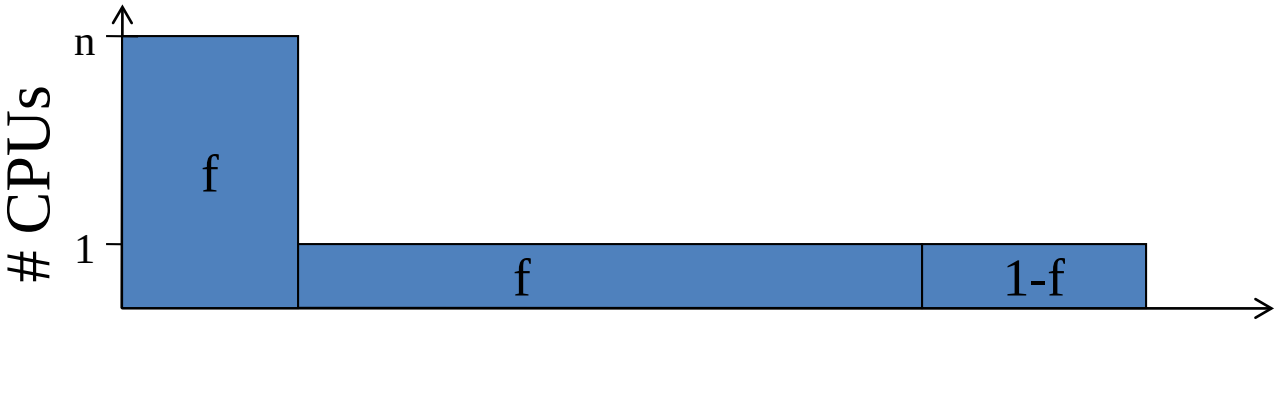


Why Multicore



	Single Core	Dual Core	Quad Core
Core area	A	~A/2	~A/4
Core power	W	~W/2	~W/4
Chip power	W + O	W + O’	W + O’’
Core performance	P	0.9P	0.8P
Chip performance	P	1.8P	3.2P

Amdahl's Law



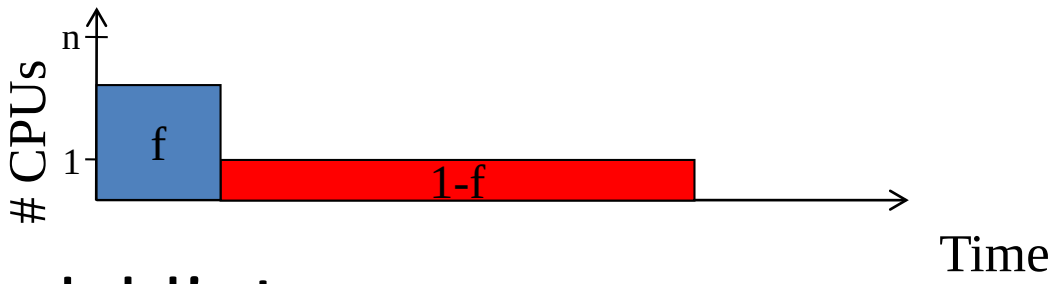
f – fraction that can run in parallel

1-f – fraction that must run serially

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

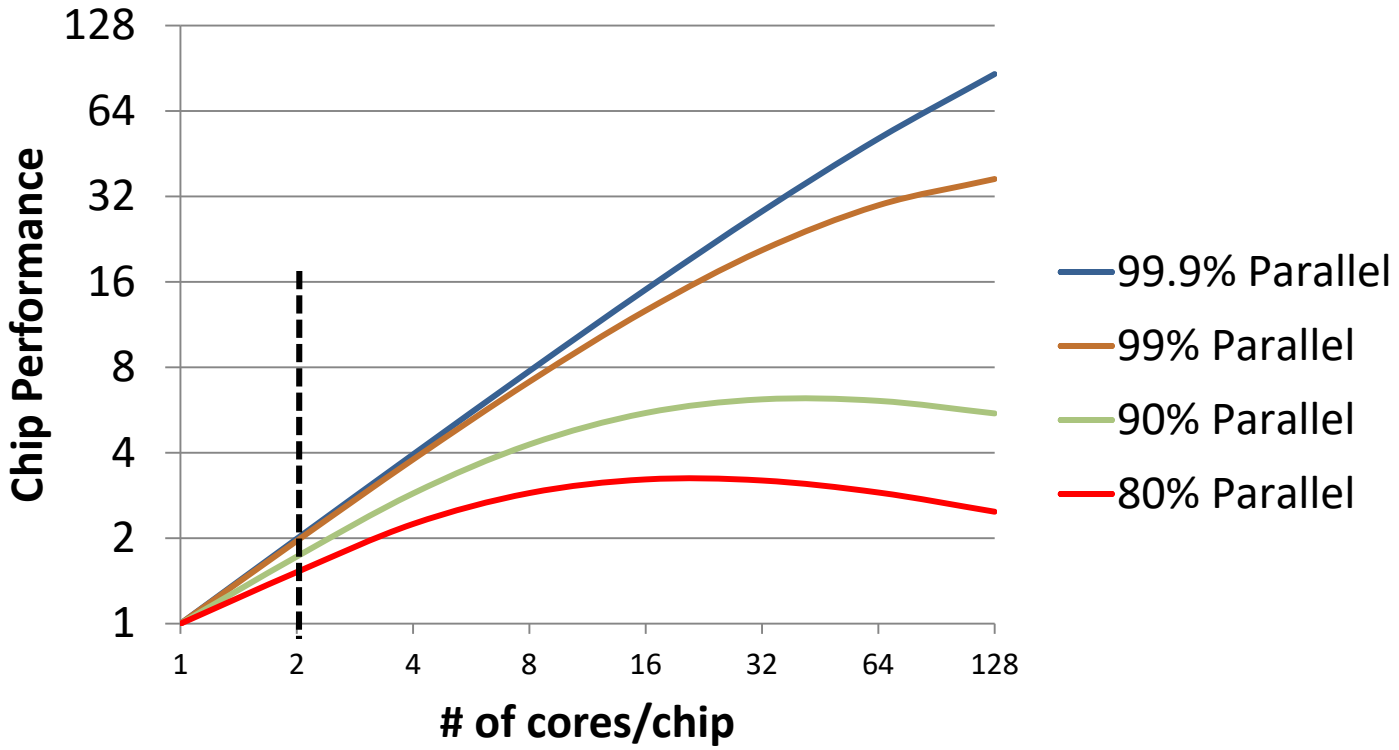
$$\lim_{n \to \infty} \frac{1}{1-f+\frac{f}{n}} = \frac{1}{1-f}$$

Fixed Chip Power Budget



- Amdahl's Law
  - Ignores (power) cost of n cores
- Revised Amdahl's Law
  - More cores → each core is slower
  - Parallel speedup < n
  - Serial portion (1-f) takes longer
  - Also, interconnect and scaling overhead

Fixed Power Scaling



- Fixed power budget forces slow cores
- Serial code quickly dominates

Focus of this Course

- How to make serial portion fast
  - Fast serial portion also helps parallel portion!
- State-of-the-art processor design
  - Pipelining review (online lectures)
  - Superscalar, out-of-order processors
  - Branch prediction
- Advanced memory systems
  - Cache review (online lecture)
- Multicore and multithreaded processors

Instruction Set Processing

The ART and Science of Instruction-Set Processor Design

[Gerrit Blaauw & Fred Brooks, 1981]

ARCHITECTURE (ISA) programmer/compiler view

- Functional appearance to user/system programmer
- Opcodes, addressing modes, architected registers, IEEE floating point

IMPLEMENTATION (µarchitecture) processor designer view

- Logical structure or organization that performs the architecture
- Pipelining, functional units, caches, physical registers

REALIZATION (Chip) chip/system designer view

- Physical structure that embodies the implementation
- Gates, cells, transistors, wires

Iron Law

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) (CPI) (cycle time)

Architecture --> Implementation --> Realization

Compiler Designer **Processor Designer** **Chip Designer**

Iron Law

- Instructions/Program
  - Instructions executed, not static code size
  - Determined by algorithm, compiler, ISA
- Cycles/Instruction
  - Determined by ISA and CPU organization
  - Overlap among instructions reduces this term
  - Constrained by energy per instruction (EPI)
- Time/cycle
  - Determined by technology, organization, clever circuit design
  - Constrained by power limitations

Our Goal

- Minimize time, which is the product, NOT isolated terms
- Common error to miss terms while devising optimizations
  - E.g. ISA change to decrease instruction count
  - BUT leads to CPU organization which makes clock slower
  - Reduced CPI causes large increase in EPI
- Bottom line: terms are inter-related

Textbooks

- **Recommended course textbook:**
  - John Paul Shen and Mikko H. Lipasti, *Modern Processor Design: Fundamentals of Superscalar Processors*, First edition, McGraw-Hill.
- **Recommended textbook:**
  - Mark Hill, Norm Jouppi, and Guri Sohi. *Readings in Computer Architecture.* Morgan Kauffman, 1999

Expected Background

- ECE/CS 552 or equivalent
  - Design simple uniprocessor
  - Simple instruction sets
  - Organization
  - Datapath design
  - Hardwired/microprogrammed control
  - Simple pipelining
  - Basic caches
- High-level programming experience
  - C/UNIX skills – modify simulators

Course Context

- Assume canonical RISC ISA
  - Register-register ALU ops
  - Load from memory (cache)
  - Store to memory
  - Branches, jumps, calls, returns
- Modern CISC (x86) processors
  - Translate to equivalent primitives
    - Later: how the translation is done

About This Course

- Readings and Paper Reviews
  - Will be posted on website (one list for each midterm)
  - Make sure you keep up with these! Not necessarily discussed in lecture.
- Lecture
  - Attendance required
  - Some lectures will be delivered on line
  - Overscheduled in first half; will cancel many lectures in 2nd half
- Homework
  - Homework assigned but not graded
  - Learning tool to help prepare for midterm

About This Course

- Pop Quizzes
  - Not announced ahead of time
  - Will drop one for final grade to accommodate occasional absence
  - Make sure you are ahead on readings!
- Exams
  - Midterm 1: Wed 10/25 in class
  - Midterm 2: Wed 12/20 10:05am-12:05pm (final exam time slot)
  - Keep up with reading list!

About This Course

- Course Project
  - Research project
    - Replicate results from a paper
    - Or attempt something novel
- Final project includes a written report and an oral presentation
  - Proposal due 10/30
  - Progress report due 11/22
  - Presentations during class time 12/11, 12/13
  - Final reports due 12/13

About This Course

- Grading
  - Quizzes & paper reviews 20%
  - Midterm 1 25%
  - Midterm 2 25%
  - Project 30%
- Web Page (check regularly)
  - http://ece752.ece.wisc.edu

About This Course

- Office Hours
  - Prof. Lipasti: EH 3621, TBD
  - Or, catch me after class
- Communication channels
  - E-mail to instructor, class e-mail list
    - compsci752-1-f17@lists.wisc.edu
  - Web page
  - Office hours

About This Course

- Other Resources
  - Computer Architecture Colloquium – Tuesday 4-5PM, 1221 CSS
  - Computer Engineering Seminar – Friday 12-1PM, EH4610
  - Architecture mailing list:
    http://lists.cs.wisc.edu/mailman/listinfo/architecture
  - WWW Computer Architecture Page
    http://pages.cs.wisc.edu/~arch/www/

About This Course

- Lecture schedule:
  - MWF 11:00-12:15
  - Cancel approx. 1 of 3 lectures, mostly in second half of semester
  - Allows us to get ahead on topics to enable broader range for project work

Tentative Schedule

Week 0	Intoduction, Technology challenges
Week 1	Superscalar Organization
Week 2	Instruction Flow
Week 3	Register Data Flow
Week 4	Memory Data Flow
Week 5	Advanced Register Data Flow
Week 6	Case Studies
Week 7	Midterm 1 in-class on 10/25, Case Studies
Week 8	Advanced Memory Hierarchy
Week 9	Multiple threads, Case studies
Week 10	Advanced topics
Week 11	Lecture canceled, project work
Week 12	Lecture canceled, project work
Week 13	Lecture canceled, project work
Week 14	Project talks, Course Evaluation, Final reports
Finals Week	Midterm 2 Wednesday 12/20 10:05pa

Wrapping Up

- Next lecture on technology challenges
  - Sets the stage for the whole course
- View review lecture online
  - Pipelining Review, 2 lectures with audio narration
  - http://ece752.ece.wisc.edu
- Reading list and review schedule on web page
- Be prepared for discussion/pop quiz

Final thought:

Talking about music is like dancing about architecture.
(Thelonius Monk)