Advanced Computer Architecture

Instructor: Andreas Moshovos
moshovos@eecg.toronto.edu

Fall 2006

Some material is based on slides developed by profs. Mark Hill, David Woord, Guri Sohi and Jim Smith at the University of Wisconsin-Madison, and Dave Patterson at the University of California Berkeley.

Today's Lecture

- Course Content:
  - Building the **best** processor
- Who cares
- How to define "best"
- Needs/Metrics
- Forces that determine "needs"
  - Applications
  - Technology
- What is "Computer Architecture"
  - Implementation
- Role of the Architect
- **Overview of course policies**

Course Goal

- **Advanced uni-processor/single-chip architecture**
  - **Will use the term “processor”**

- Previous courses:
    - How to build a processor that works
    - Some optimization techniques

- This course:
  - What is the BEST processor?
  - Recent Research Developments

- Some overlap with the undergrad Comp. Arch.

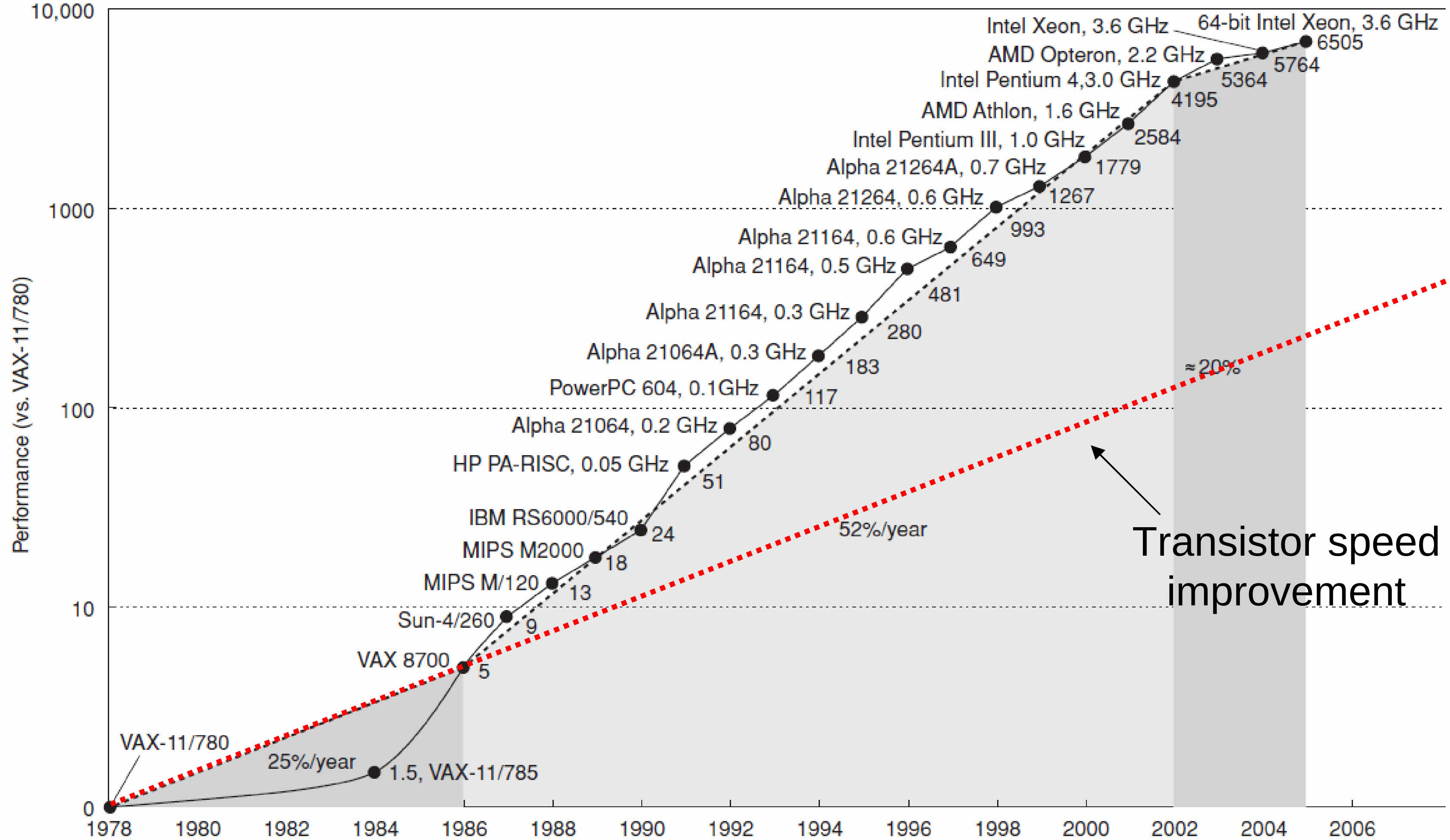
What is Computer Architecture

- Goal:
  - Build the **best** “processor”

- Today this means:
  - Here’s a piece of silicon
  - Here are some of its properties
  - Tell me what to build
- Two challenges:
  1. **Understand your building blocks:**
     - today its semiconductors
  2. **Understand what best means**

- **Take into account design/production time**
  - Takes 4-5 years to design a new high-performance processor

Architecture and Technology



- Source: H&P, CA: A Quantitative Approach 4th Edition

Evolution of Microprocessors

	70's	80's	90's	2010?
xtor count	10k	100k-1M	1M-100M	1B
Freq.	0.2-2Mhz	2-20Mhz	20-1Ghz	10Ghz
IPC+	< 0.1	0.1-0.9	0.9-2.0	???
MIPS*	< 0.2	0.2-20	20-2k	100k?

(+) IPC = Instructions Per Cycle. How many instructions execute per machine cycle. This is <1 for the architectures you learned in undergrad courses. Can be >1 for others we will discuss later on.

(*) MIPS = Million Instructions per Second. Normalized. Later we will explain why this is a bad metric. Shown here to make a point and should be interpreted only as an indication.

Recent Designs

- **AMD Athlon 64 FX-62:**
  - 243M xtors, 90nm, 2.8Ghz, 220 mm^2, 2 cores
- **Intel Core Duo Extreme X6900**
  - 291M xtors, 65nm, 3.2Ghz, 143 mm^2, 2 cores
- **AMD Turion 64 ML-40**
  - 114M, 90nm, 2.2Ghz, 125mm^2, 1 core
- **SUN T1 “Niagara”**
  - 300M, 90nm, 1.2Ghz, 379 mm^2, 8 cores

Understanding the building Blocks

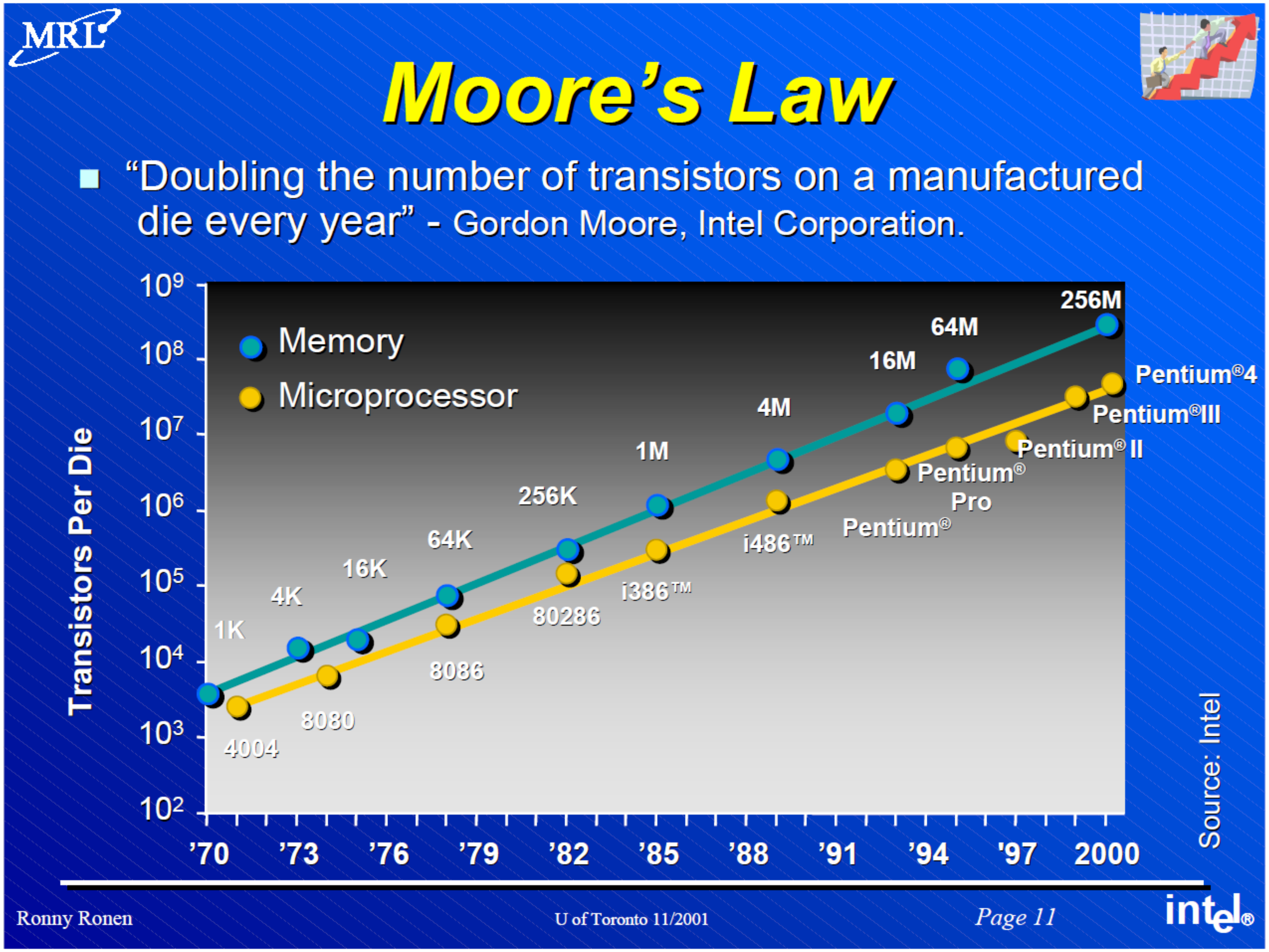
Moore’s “Law”

- “Cramming More Components onto Integrated Circuits”
  - G.E. Moore, Electronics, 1965
- Observation: (DRAM) transistor density doubles annually
  - Became known as “Moore’s Law”
  - Wrong, density doubles every 2 years
    - Had only four data points
- Corollaries
  - cost / transistor halves annually
  - power decreases with scaling
  - speed increases with scaling
  - reliability increases with scaling (??)
- Recent trends somewhat different
  - We will return to this throughout the lectures

The Other “Moore’s Law”

- **“Performance doubles every 18 months”**
  - common interpretation of Moore’s Law, not original intent
  - wrong, “performance” doubled every ~2 years
  - wrong, lately other parameters slowed down performance
- Self-fulfilling prophecy (Moore’s Curve)
  - doubling every 18 months = ~4% increase per month
  - **4% per month used to judge performance features,**
  - *if feature adds 2 months to schedule, it should add at least 8% to performance*

Intel Processor Family



Technology Scaling

- ICs characterized by **Feature Size**
  - *minimum xtor/wire size in x or y dimension*
  - *10 microns in 1971, 0.65 microns today, ~154x reduction*
- **Xtor density:**
  - quadratic w/ respect to feature size
- **Xtor performance:**
  - complex, but almost linear (lower Vdd required for correct operation)
- **Wire Delay:**
  - complex, distances shorter, but R and C higher/unit. Net effect, wires do not scale as well as xtors.
- **Power:**
  - dynamic and static. ~**C**x**F**x**V**^2 . Currently a big problem.
- **Die Size:**
  - Mostly unrelated

Technology Scaling

• **Feature Size**	30% every 2 to 3 years
• Transistor Density	~50% (0.7x0.7)
• Transistor speed	~50%
• Die size	10% - 20%
• Transistor Count	60%-80%

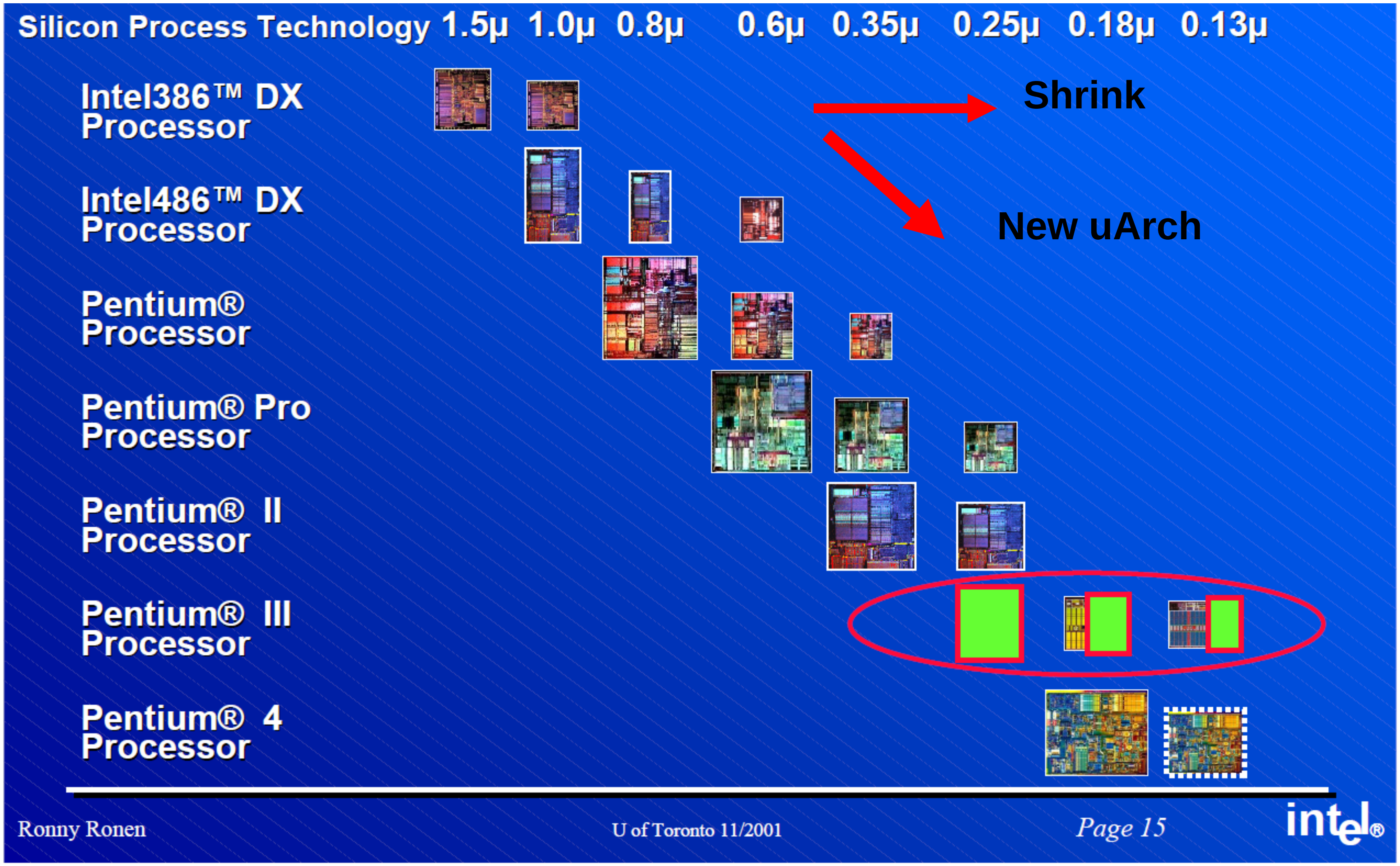
- **IDEAL Shrink:**
  - 1x xtors
  - 0.5x area
  - 1.5x frequency
  - 1x IPC
  - 1.5x performance
  - 0.5x power

- **IDEAL New Design:**
  - 2x xtors
  - 1x area
  - 1.5x frequency
  - 2x IPC
  - 3x performance
  - 1x power

Not what is possible most of the time

R. Ronnen et. al. IEEE Proceedings 2001

FYI: Actual Scaling

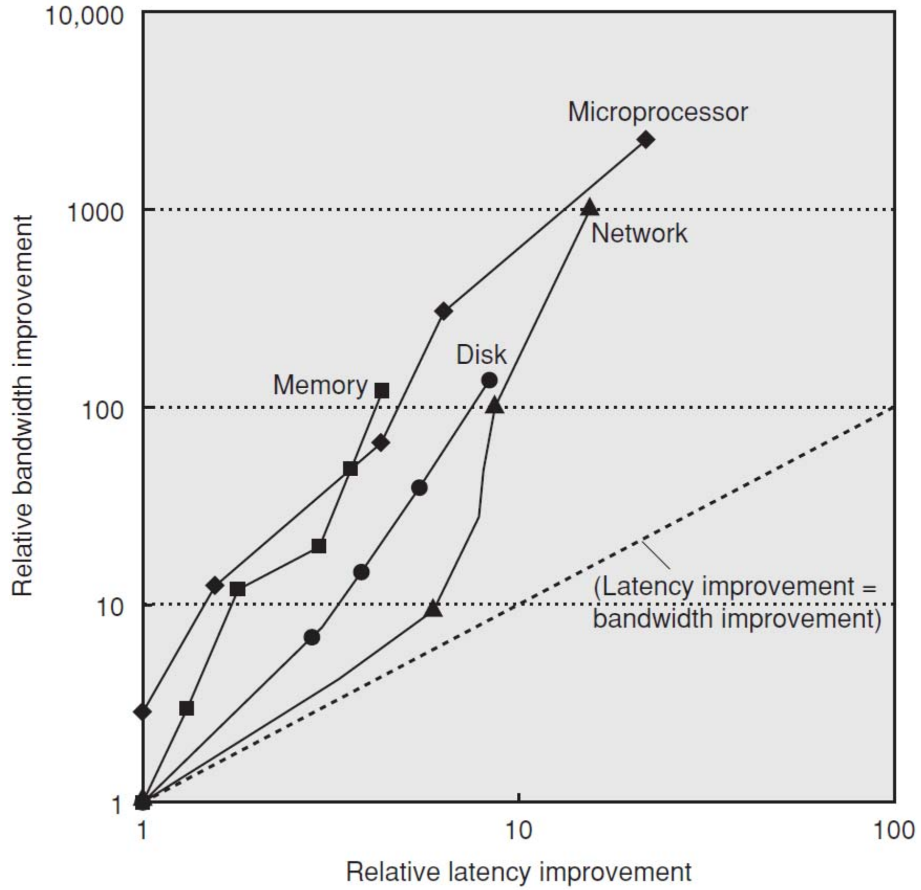


Many factors determine what the new arch should be.

Technology Scaling Contd.

• DRAM density	40% - 60% (4x in 3 years)
• DRAM speed	4% (1/3 in 10 years)
• Disk density	100% (4x in 2 years)
• Disk speed	4% (1/3 in 10 years)

Technology Scaling: Latency vs. Bandwidth



- **Not all technologies scale similarly**
- Source: H&P, CA: A Quantitative Approach 4th Edition

FYI: DRAM/Disk Technology Evolution

Memory module	DRAM	Page mode DRAM	Fast page mode DRAM	Fast page mode DRAM	Synchronous DRAM	Double data rate SDRAM
Module width (bits)	16	16	32	64	64	64
Year	1980	1983	1986	1993	1997	2000
Mbits/DRAM chip	0.06	0.25	1	16	64	256
Die size (mm^2)	35	45	70	130	170	204
Pins/DRAM chip	16	16	18	20	54	66
Bandwidth (MBit/sec)	13	40	160	267	640	1600
Latency (ns)	225	170	125	75	62	52

Hard disk	3600 RPM	5400 RPM	7200 RPM	10,000 RPM	15,000 RPM
Product	CDC WrenI 94145-36	Seagate ST41600	Seagate ST15150	Seagate ST39102	Seagate ST373453
Year	1983	1990	1994	1998	2003
Capacity (GB)	0.03	1.4	4.3	9.1	73.4
Disk form factor	5.25 inch	5.25 inch	3.5 inch	3.5 inch	3.5 inch
Media diameter	5.25 inch	5.25 inch	3.5 inch	3.0 inch	2.5 inch
Interface	ST-412	SCSI	SCSI	SCSI	SCSI
Bandwidth (MBit/sec)	0.6	4	9	24	86
Latency (ms)	48.3	17.1	12.7	8.8	5.7

Local area network	Ethernet	Fast Ethernet	Gigabit Ethernet	10 Gigabit Ethernet
IEEE standard	802.3	803.3u	802.3ab	802.3ac
Year	1978	1995	1999	2003
Bandwidth (MBit/sec)	10	100	1000	10000
Latency (μsec)	3000	500	340	190

•Source: H&P, CA: A Quantitative Approach 4th Edition

Putting things into Perspective

	Cray® X-MP	Intel Pentium® 4
Year:	1982	2000
Freq.	105MHz	2000MHz
Perp.	**400M FLOPS**	**2G FLOPS**
Power	? KW	100W
Size	2m	2 cm^2
Weight	>100kg	<1g
Cost	**$15M**	**<500$**
Cooling	Freon	Air

← Relative sizes →

* The Cray® X-MP system sits in UPC, Barcelona

Classes of Computers

Feature	Desktop	Mobile	Server	Embedded
Price	\$400-\$10k	\$900-\$7k	\$10k-\$10M	\$10-\$100K
CPU	\$70-\$1k	\$400-\$2k	\$200-\$5k /cpu	\$0.20-\$200 /cpu
Volume	150M	?	4M	300M
Critical attributes	price/perf. graphics perf.	power price/perf. graphics. perf.	through-put availability scalability	price power appl-spe-cific perf.

Not to be taken literally.

Recall

- Goal:
  – Build the **best** “processor”

- Two challenges:

1. **Understand your building blocks:**
   - today its semiconductors
2. **Understand what best means**

What BEST means?

- Really depends on what your goal is:
  - Moving: Best take truck - unless you have nothing...
  - SUV? I don't know, you tell me
  - Porche? Have money to burn - cruising
- **Observation #1:**
  - Before we can decide what is best we need to know the **Needs** are.
- Moving vs. cruising
- **Observation #2:**
  - Then we need to be able to judge how well each option serves these needs. **Metrics**
- Truck vs. Porche
- What if you had to build the best car for a given purpose?

What BEST processor means?

- Needs:
  - **Performance:** word processing vs. weather simulation
  - **Cost:** would you pay 5x $ for 2x performance?
  - **Complexity:** Design/validation time -> cost and perf.
  - **Power:** PDA, laptop, server
  - **Reliability:** Must work correctly
- There are a number of forces at work:
  - 1. What does the user needs: **applications**
  - 2. What does technology offers: **semiconductors**
- Why this is challenging:
  - Many applications, some yet to be developed
  - Technology changes

What is Computer Architecture?

- **Architecture:** How are things organized and what you can do with them (functionality)

- Many different “Architectures” exist in a system
  - Application/System architecture
    - Structure of the application itself
  - Interface to outside world (API, libraries, GUIs, etc.)
  - Operating system calls
  - Often appear as layers

- **For our purposes Computer architecture is** the Interface between hardware and software

What is Computer Architecture?

System attributes as seen by the programmer

The term architecture is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation.

Gene Amdahl, IBM Journal of R&D, April 1964

- What you the user needs to know to reason about how the machine behaves

- **A contract between users and the designer/architect**
  - Architect: I guarantee these features, anything else can change across different designs
  - User can develop applications and reason about what they will do having a guarantee that they will work across different designs

Architecture, µArch and Implementation

- **Computer "Architecture": HW/SW interface**
  - instruction set
  - memory management and protection
  - interrupts and traps
  - floating-point standard (IEEE)
  - Could include others: designer beware

- **µMarch (micro-Arch)**: also called organization
  - number/location of functional units
  - pipeline/cache configuration
  - programmer transparent techniques: prefetching

- **Implementation (Hardware)**: low-level circuits

Architecture vs. Implementation

- AND Gate:
  - Architecture is the interface:
    - 2 inputs - 1 output and function
    - Truth table defines behavior
  - Implementation?
    - Transistor based (How many can you think?)
      - static, dynamic? CMOS, NMOS?
    - Moshovos™ implementation
    - others?

Computer Architecture

- **The big question is what goes into Architecture**
- **Too much:**
  - Too restrictive
    - Additions take 1 cycle to complete
- **Too little:**
  - Lost opportunity
  - Substandard performance
    - Subtract and branch if negative is good enough
    - Multimedia instruction set extensions

- Challenge is to forsee how technology/application trends may create problems in the future
  - Delay slots

Architecture vs. μMarch vs. Impl.

The boundaries are a bit blurred, still

64-bit Adder:

- — Arch: What it does
  - — take two 64-bit numbers produce 64-bit sum
- — μMarch: How it does it:
  - — Ripple carry
  - — Carry lookahead
  - — Carry prediction
- — Implementation
  - — static, dynamic, CMOS, Synthesized, Custom, etc...

- **This course: Architecture, μMarch and its interactions/ implications to software and implementation**

Role of the Computer (μ)Architect

- **Architect:** Define hardware/software interface
- **μArchitect:** Define the hardware organization, usually same person as above
- **Goal:**
  - 1. Determine important attributes (e.g., performance)
  - 2. Design machine to maximize those attributes under constraints (e.g., cost, complexity, power).
- **How :** Study applications
  Consider underlying technology
  Cost
  Performance
  Complexity
  Power
  Reliability

Two Aspects of CA

- **Techniques:**
  - This is the accumulated experience
  - Typically, there is no formal way of developing these (innovation)
  - Know how to evaluate
- **When to use them?**

  RISC architectures: Could fit a CPU within a single chip in the early 80's

Architecture is a "science" of tradeoffs

No underlying one-truth - we build our own world and mess

Too many options -> too many different ways of being wrong

Why Study Computer Architecture

- Build faster/better processors
  - Why? my MS-Word, Latex runs quite fast on my Pentium 166 MMX thank you very much
  - How about weather simulation? Speech recognition? MRI Processing? MPEG-4 (7?), Your Killer-App circa 2010?
- Bottom line:
  - Historically, faster processors facilitated new applications
  - Similarly, novel applications created a need for faster machines
  - Cost is factor
  - Facilitate further scientific development
  - Any reason why this will change?
- Also performance not the only requirement
  - **#1: User requirements are constantly changing**

Implications of Implementation Technology

- **Caches (“bad” for IBM-XT, “a must” for Pentium 4):**
- 70’s: thousands of xtors, DRAM faster than 8088 microprocessor
- nice way of slowing down your program
- 80’s: depends on machine
- 90’s: millions of xtors, what to do with them, DRAM much slower than processor
- a must, otherwise your ~3Ghz processor spends most of its time waiting for memory

- **#2: Technology changes rapidly making past choices often obsolete**

- **#3: Also opens up new opportunities (e.g., out-of-order)**

Perpetually Open Problems in CA

- Performance
- Cost
- Complexity
- Power
- Reliability
- Architectural Support for…

Texts

- *These slides*
- ***• Computer Architecture: A Quantitative Approach, Hennessy and Patterson, 4th Edition, Morgan Kaufmann***
- ***Readings in Computer Architecture, Hill, Jouppi and Sohi.***
- Related conference papers - both classic and cutting-edge
- Conferences:
- • ISCA (international symposium on CA)
- • ASPLOS (arch. support for progr. languages & OSes)
- • MICRO (microarchitecture)
- • HPCA (all encompassing?)
- • Others: PACT, ICS...
- GENERAL INFO: www.cs.wisc.edu/~arch/www
- Online papers: **www.computer.org, citeseer.nj.nec.com**

About the Course

- Instructors: Andreas Moshovos
- Office hours: via appointment only, but feel free to stop by, EA311
- best way to communicate with me: e-mail
  - Persist if I don’t respond the “first” time
  - moshovos@eecg.toronto.edu

- Please use **“ACA: Your header here”** for all your e-mails
- Course web site: www.eecg.toronto.edu/~moshovos/ACA06
- nothing there yet
- There is no TA

- **You are responsible for all material discussed in class**
- **Notes will not be provided for all discussions**

Schedule

- **First half (you attend lectures):**
- Lectures on advanced architecture topics
- Some assignments

- • **Second half (you give lectures and discuss):**
- In groups you select among a set of research papers
- You give a presentation
- We discuss them in class
- You work on a project
- (you define or pick from a set of suggestions)

Expected Background

- Organization and Comp. Arch. (some overlap)
- Design simple uniprocessor
- Instruction set concepts: registers, instructions, etc.
- Organization
- Datapath design
- Hardwired/microprogrammed control
- Simple pipelining
- Basic caches, main memory
- High-level programming experience (C is a must)
- Compilers (back-end) and VLSI highly desired
- You are expected to read on your own and fill-in any gaps

Topics

1. Technology Trends / Performance Metrics / Methodology
2. Pipelinining
3. Advanced Instruction Level Parallel Processing
4. Control Flow Prediction
5. Memory System
6. Instruction Set Principles
7. New Challenges: Power/Reliability
8. State-of-the-Art Research Papers and Classics

1 through 7 is my responsibility

8: I provide pointers, you make the presentation, we discuss the papers in class

Marking

- This is a grad course: *You are expected to be able to seek information beyond what is discussed in class.*
- Project 1/3
- Homeworks 1/3
- Presentations 1/3

  – If needed (Intention is NOT to have one):
    - Take Home Exam ½ (and everything else x ½)

- You must score at least 5/10 in all of the above separately to pass

Project

- This is probably the most important part of the course
- You will be required to propose and conduct “research” in computer architecture
- — I will provide some suggestions
- — You are strongly encouraged to suggest your own:
- **Validate data in some paper**
- Evaluate extension to existing work
- Propose something completely new (difficult)
- Since this is a class project negative results are OK
- — In general it is hard to publish negative results
- You will probably have to use the simplescalar simulator
- Requires strong programming skills in C
- You must be familiar with UNIX or learn your way through it
- Groups of 2 or 3 if necessary (depends on class size too)
- More details coming “soon”

Homeworks

- There will be 3-4 assignments
- May require material that we do not cover in depth in class
- There will be series of programming assignments that are designed to help you learn the simulation infrastructure that is commonly used in our research community: www.simplescalar.com
- Assignments require strong programming skills primary in C
- Also require that you are familiar with UNIX systems
- Environment to be determined within two weeks:
  - Either cygwin/windows or linux

Policies

- **No late work will be accepted**
  — You will be given able time to complete all coursework

- **All work must be your own unless otherwise specified**
  – Please take this seriously
  – Make sure to reference any external sources
  – I will not go looking for plagiarism, but often it’s obvious and CAN’T BE IGNORED

Integrated Circuit Costs

$$\text{cost (IC)} = \frac{\cos t(die) + \cos t(testing) + \cos t(packaging)}{FinalTestYield}$$

$$\text{cost (die)} = \frac{\cos t(wafer)}{(die/\ \ wafer) \times yield(die)}$$

$$\text{yield (die)} = yield(Wafer) \times \left\{1 + \frac{defects/\ \ cm^2 \times area}{\alpha}\right\}^{-\alpha}$$

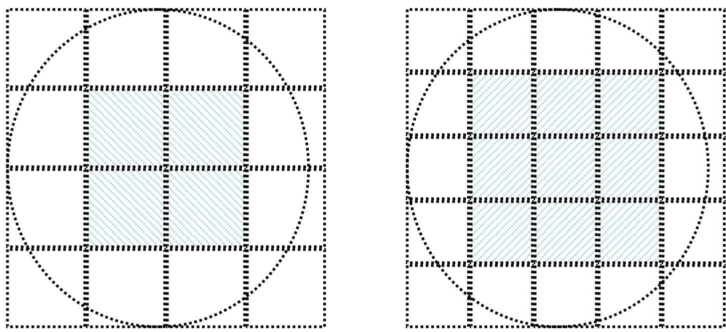
often α is 0.40

cost (die) = f (die area^4)

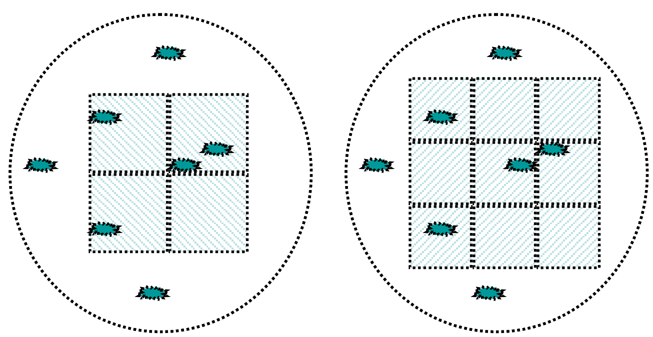
$$Dies/Wafer = \frac{\pi \times \left(\frac{wafer\_diameter}{2}\right)^2}{Die\_area} - \frac{\pi \times wafer\_diameter}{\sqrt{2 \times die\_area}} - test\_dies$$

Die Size, Wafer and Yield

- Bigger die → less dies per wafer



- Bigger die → defects much more probable



IC Cost Examples

Chip	Metal layers	Line width	Wafer cost	Defect /cm²	Area mm²	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

- From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15
- New products end up being much more expensive to manufacture

Early Steps: Reading 1

- Arthur W. Burks, Herman H. Goldstine, and John von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument", 42pp, Inst. for Advanced Study, Princeton, N. J., June 28, 1946
- Reprinted in: "Computer Structures: Readings and Examples", (1971 edition) by C. Gordon Bell & Allen Newell

- **Interesting Discussions:**
  - Selection of word length and number base.
  - Discussion of the instructions needed.
  - Concern for the input/output structure and the idea of displays
  - Rationale for not including floating-point arithmetic (caution about the technology).
  - The lack of necessity for the rather trivial binary-decimal conversion hardware and the idea of cost effectiveness.
  - Analysis of the addition, multiplication, and division hardware implementation. (This description includes a nice, one-page discussion of the average carry length for addition.)

The Task of the Referee: Reading #2

- Evaluating research/engineering work in computer architecture

Strong Inference: Reading #3

- “Strong Inference”, John R. Platt, Science, Vol. 146, No. 3642, Science
- Possible alternative explanations of an observation.
- How to discriminate between alternative explanations.

- 1. Devise alternative hypotheses.
- 2. Devise crucial experiment.
- 3. Carry out the experiment so as to get a clean result.
- Go back to 1 as necessary