
Advanced Computer Architecture

Instructor: Andreas Moshovos
moshovos@eecg.toronto.edu

Fall 2005

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

All other material (c) A. Moshovos.

Today's Lecture #1

- 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**

What this course is about

Advanced Uniprocessor Architecture
Build the Best Processor

What is this course about

- **Advanced Computer Architecture**
- Really is on: **Advanced uni-processor architecture**
 - Previous courses:
 - How to build a processor that works?
 - This course: What is the BEST processor I can build?
 - This is not as easy as it sounds.
- To explain why let's consider an example:
 - What is the best means of transportation?
 1. Porche? / 2. SUV? / 3. Truck? / 4. Train? / 5. Plane? / 6. Bike?

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

It's the interface

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, μ Arch 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
- **μ March (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 for computers 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 foresee how technology/application trends may create problems in the future
 - Delay slots

Architecture vs. μ March 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

- μ March: How it does it:

 - Ripple carry

 - Carry lookahead

 - Carry prediction

- Implementation

 - static, dynamic, CMOS, Synthesized, Custom, etc...

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

Role of the Computer (μ)Architect

- **Architect:** Define hardware/software interface
- **μ Architect:** 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 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 2005-2010?
- 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**

Why Study Computer Architecture #2

- Implementation technologies change rapidly
- Technology Annual improvement
- Transistor density 50%
- Die size 10% - 20%
- Transistor Count 60%-80%
- Transistor speed 20-25%
- 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)

- They also change relative to one another

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)**

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.

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.

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 18 months (had only 4 data points)
- corollaries
 - cost / transistor halves annually
 - power decreases with scaling
 - speed increases with scaling
 - reliability increases with scaling (??)
- Recent trends different: technology or marketing?

The Other “Moore’s Law”

- “performance doubles every 18 months”
- • common interpretation of Moore’s Law, not original intent
- • wrong, “performance” doubles every ~2 years
- • 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*

Technology Scaling

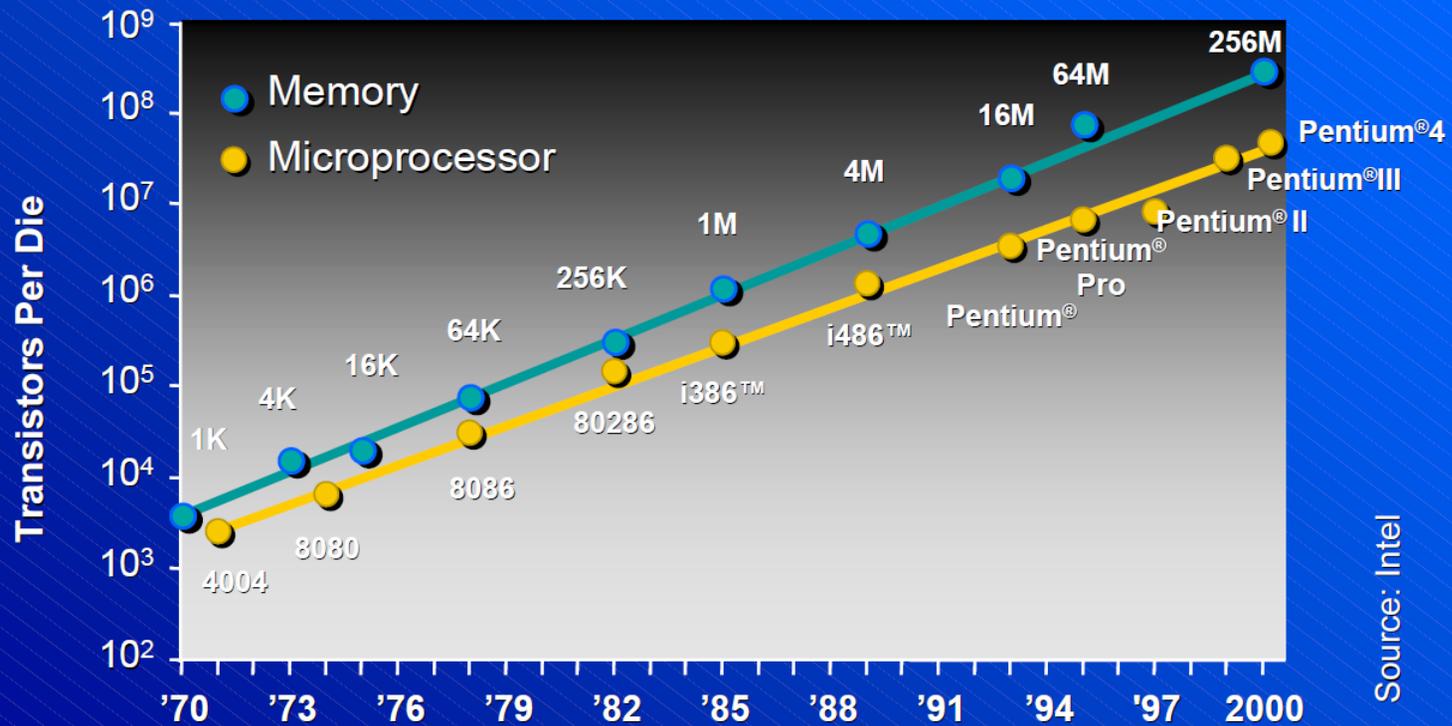
- ICs characterized by *Feature Size*
- *minimum xtor/wire size in x or y dimension*
- *10 microns in 1971, 0.13 microns today, ~76x reduction*
- **Xtor density:** quadratic w/ respect to feature size
- **Xtor performance:** complex, but almost linear (lower V_{dd} required for correct operation)
- **Wire Delay:** complex, distances shorter, but R and C higher/unit. Net effect, wires do not scale as well as xtors.
- **Power:** dynamic and static. $\sim CxFxV^2$. Currently a big problem.

Intel Processor Family



Moore's Law

- “Doubling the number of transistors on a manufactured die every year” - Gordon Moore, Intel Corporation.



Ideal Technology Scaling Scenario

■ Ideal “Shrink”

- Same μ arch
- 1X #Xistors
- 0.5X size
- 1.5X frequency
- 0.5X power
- 1X IPC (instr./cycle)
- 1.5X performance
- 1X power density

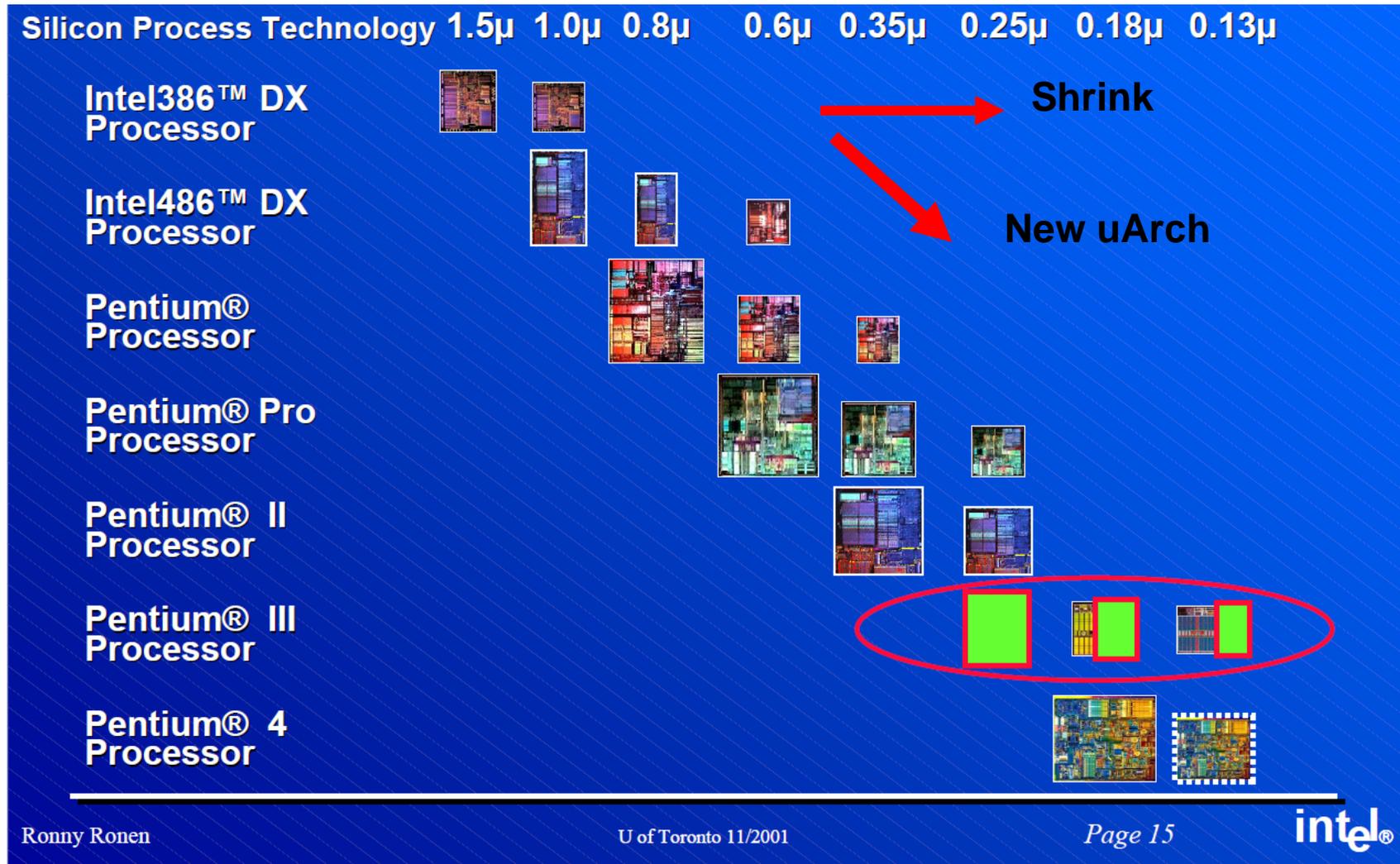
■ Ideal New μ arch

- Same die size
- 2X #Xistors
- 1X size
- 1.5X frequency
- 1X power
- 2X IPC
- 3X performance
- 1X power density

Looks good. Isn't it?

We will see why this is not the case most of the time.

Actual Scaling



Many factors determine what the new arch should be.

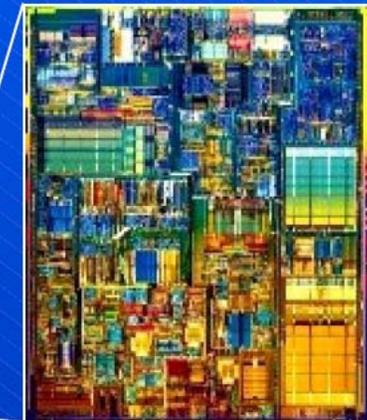
Putting things into Perspective



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

	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 →



Perpetually Open Problems in CA

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

About the Course

- Instructors: Andreas Moshovos
- Office hours: via appointment only
- 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/ACA05
- nothing there yet
- There is no TA
- Meet twice a week: Tue. 2 hour lecture, Thursday 3 hours.
 - Probably not throughout the semester
- **You are responsible for all material discussed in class**

Texts

- *These slides*
- • ***Computer Architecture: A Quantitative Approach, Hennessy and Patterson, Third Edition, Morgan Kaufmann, 1-55860-596-7 (cloth), 1-55860-724-2 (paper)***
- • ***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

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 X1%
- Homeworks X2%
- Presentations X3%
- If needed (Intention is NOT to have one):
 - Take Home Exam X4%
- 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 5-6 assignments
- They will include assignments from the H&P book
- 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/WinXP or access to linux cluster

Policies

- **No late work will be accepted**
 - You will be given ample time to complete all coursework
 - There is no TA, dealing with late work is a logistical nightmare
- **All work must be your own unless otherwise specified**
 - — Please take this seriously
 - — Make sure to reference any external sources