CS184: Computer Architecture (Structure and Organization)

Day 1: January 3, 2005
Introduction and Overview

Today

- Matter Computes
- Architecture Matters
- This Course (short)
- Unique Nature of This Course
- Relation to other courses
- More on this course

Review: Two Universality Facts

- Turing Machine is Universal
  - We can implement any *computable* function with a TM
  - We can build a single TM which can be programmed to implement any computable function
- NAND gate Universality
  - We can implement any computation by interconnecting a sufficiently large network of NAND gates

Review: Matter Computes

- We can build NAND gates out of:
  - transistors (semicondutor devices)
    - physical laws of electron conduction
  - mechanical switches
    - basic physical mechanics
  - protein binding / promotion / inhibition
    - Basic biochemical reactions
  - …many other things

Starting Point

- Given sufficient raw materials:
  - can implement any computable function
- Our goal in computer architecture
  - is **not** to figure out how to compute new things
  - rather, it is an ***engineering*** problem

Engineering Problem

- Implement a computation:
  - with least resources (in fixed resources)
    - with least cost
  - in least time (in fixed time)
  - with least energy
- Optimization problem
  - how do we do it best?

Quote

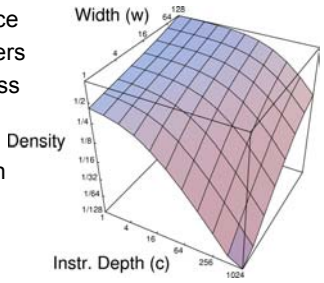
- "An Engineer can do for a dime what everyone else can do for a dollar."

Architecture Matters?

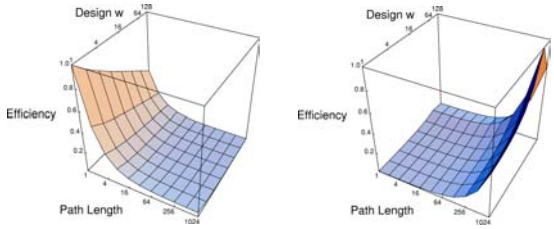
- How much difference is there between architectures?
- How badly can I be wrong in implementing/picking the wrong architecture?
- How efficient is the IA-32, IA-64?
  - Is there much room to do better?
- Is architecture done?
  - A solved problem?

Peak Computational Densities from Model

- Small slice of space
  - only 2 parameters
- 100× density across
- Large difference in peak densities
  - large design space!



Yielded Efficiency



FPGA (c=w=1) "Processor" (c=1024, w=64)

- Large variation in **yielded** density
  - large design space!

Architecture **Not** Done

- Many ways, not fully understood
  - design space
  - requirements of computation
  - limits on requirements, density...
- …and the *costs* are changing
  - optimal solutions change
  - creating new challenges and opportunities

Personal Goal?

- Develop systematic design
- Parameterize design space
  - adapt to costs
- Understand/capture req. of computing
- Efficiency metrics
  - (similar to information theory?)
- …we'll see a start at these this quarter

Architecture **Not** Done

- Not here to just teach you the forms which are already understood
  - (though, will do that and give you a strong understanding of their strengths and weaknesses)
- **Goal**: enable you to design and synthesize new and better architectures

This Course (short)

- How to organize computations
- Requirements
- Design space
- Characteristics of computations
- Building blocks
  - compute, interconnect, retiming, instructions, control
- Comparisons, limits, tradeoffs

This Course

- Sort out:
  - Custom, RISC, SIMD, Vector, VLIW, Multithreaded, Superscalar, EPIC, MIMD, FPGA
- Basis for design and analysis
- Techniques
- [more detail at end]

Graduate Class

- Assume you are here to learn
  - Motivated
  - Mature
  - Not just doing minimal to get by and get a grade
- Problems
  - May not be fully, tightly specified

Uniqueness of Class

Not a Traditional Arch. Class

- Traditional class
  - focus RISC Processor
  - history
  - undergraduate class on uP internals
  - then graduate class on details
- This class
  - much broader in scope
  - develop design space
  - see RISC processors in context of alternatives

Authority/History

- ``Science is the belief in the ignorance of experts." -- **Richard Feynman**
- Traditional Architecture has been too much about history and authority
- Should be more about engineering evaluation
  - physical world is “final authority”
- **Goal**: Teach you to think critically and independently about computer design.

On Prerequisites

- Suggested:
  - CS21 (compute models, universality)
  - EE4 (boolean logic, basic logic circuits)

Next Few Lectures

- Quick run through logic/arithmetic basics
  - make sure everyone remembers
  - (some see for first time?)
  - get us ready to start with observations about the key components of computing devices
- Trivial/old hat for many
  - But will be some observations couldn’t make in EE4
- May be fast if seeing for first time
- Background quiz intended to help me tune

Relation to Other Courses

- CS181 (VLSI)
- EE4 (Fundamentals of Digital Systems)
- CS184 (Architecture)
- CS137 (Electronic Design Automation)
- CS24 (Introduction to Computing Systems)
- CS134 (Compilers and Systems)
- CS21 (Computational Theory)

Content Overview

- This quarter:
  - building blocks and organization
  - raw components and their consequences
- Next quarter:
  - abstractions, models, techniques, systems
  - will touch on conventional, single-threaded architecture (ISA Processor)
  - Emphasis likely to be on parallel architectures

Themes (this quarter)

- Design Space
- Parameterization
- Costs
- Change
- Structure in Computations

This Quarter

- Focus on raw computing organization
- **Not** worry about
  - nice abstractions, models
- Will come back to those next quarter

Change

- A key feature of the computer industry has been rapid and continual change.
- We must be prepared to adapt.
- For our substrate:
  - capacity (orders of magnitude more)
    - what can put on die, parallelism, need for interconnect and virtualization, homogeneity
  - speed
  - relative delay of interconnect and gates

What has changed?

- [Discuss]
- Capacity
  - Total
  - Per die
- Size
- Applications
  - Number
  - Size/complexity of each
  - Types/variety
- Speed
  - Ratio of fast memory to dense memory
  - Wire delay vs. Gate delay
  - Onchip vs. inter-chip
- Joules/op
- Mfg cost
  - Per transistor
  - Per wafer

1983 (early VLSI)

- Early RISC processors
  - RISC-II, 15Mλ^2, 40K transistors
  - MIPS, 20Mλ^2, 24K transistors
  - ~10MHz clock cycle
- Xilinx XC2064
  - 64 4-LUTs

Today

- CPUs
  - Multi-issue, 64b processors
  - GHz clock cycles
  - MByte caches
- FPGAs
  - >100,000 bit processing elements
  - Mbits of on-chip RAM

More chip capacity?

- Should a 2005 single-chip multiprocessor look like a 1983 multiprocessor systems?
  - Processor→processor latency?
  - Inter-processor bandwidth costs?
  - Cost of customization?

Memory Levels

- Why do we have 5+ levels of memory today?
  - Apple II, IBM PC had 2
  - MIPS-X had 3

Class Components

Class Components

- Lecture
- Reading [1 required paper/lecture]
  - No text
- Weekly assignments
- Final design/analysis exercise
  - (2 weeks)

Lecture Schedule

- Scheduled MWF 1.5 hrs
- To allow for lost days
  - Holidays
  - Conferences
- Target use 22 of ideally 30 lectures
- (standard MW would ideally have 20)

Feedback

- Will have anonymous feedback sheets for each lecture
  - Clarity?
  - Speed?
  - Vocabulary?
  - General comments

Fountainhead Quote

Howard Roark's Critique of the Parthenon

-- Ayn Rand

Fountainhead Parthenon Quote

"Look," said Roark. "The famous flutings on the famous columns---what are they there for? To hide the joints in wood---when columns were made of wood, only these aren't, they're marble. The triglyphs, what are they? Wood. Wooden beams, the way they had to be laid when people began to build wooden shacks. Your Greeks took marble and they made copies of their wooden structures out of it, because others had done it that way. Then your masters of the Renaissance came along and made copies in plaster of copies in marble of copies in wood. Now here we are making copies in steel and concrete of copies in plaster of copies in marble of copies in wood. Why?"

Computer Architecture Parallel

- Are we making:
  - copies in submicron CMOS
  - of copies in early NMOS
  - of copies in discrete TTL
  - of vacuum tube computers?

Big Ideas

- Matter Computes
- Efficiency of architectures varies widely
- Computation design is an engineering discipline
- Costs change $\Rightarrow$ Best solutions (architectures) change
- Learn to cut through hype
  - analyze, think, critique, synthesize