ECE 2162
Intro & Trends

Jun Yang – Fall 2014
Prerequisites

• CoE/ECE 0142: Computer Organization; or CoE/CS 1541: Introduction to Computer Architecture
• I will assume you have detailed knowledge of
  – Pipelining
    • Classic 5-stage pipeline, pipeline hazards, stalls, etc.
  – Caches
    • Tag/Index/Offset, hit/miss, set-associativity, replacement policies, write-through/write-back, etc.
  – Assembler and MIPS ISAs
    • RISC, load/store, instruction encoding, caller-saved/ callee-saved registers, stack pointer, frame pointer, function call/return code, etc.

• If you don’t remember a few of these
  – Review the 142 or 1541 textbook: Computer Organization and Design: The hardware/software interface
  – Read Appendices A, B, and C

• If you don’t know what some of these are
  – Take the prerequisites before you take this class
Logistics

• Internet
  – Course Webpage: http://www.pitt.edu/~juy9/2162/2162.html
  – Email: juy9@pitt.edu
  – Phone: 412-624-9088, x49088
  – 4175 PSVR Hall

• Textbook
  – Computer Architecture: AQA, 5th Edition by Hennessy and Patterson

• Meeting time
  – Tuesdays 5:20 — 7:50
Logistics

• Midterm (30%) – 10/21 (tentative)
• Final (30%) – 12/2
• Project (40%) – 12/8 demo
What is Architecture?

• Original sense:
  – Taking a range of building materials, putting together in desirable ways to achieve a building suited to its purpose

Adapted from David Clark’s, What is “Architecture”?
In Computer Science:

• Similar: how parts are put together to achieve some overall goal
• Examples: the architecture of a chip, of the Internet, of an enterprise database system, an email system, a cable TV distribution system
Why Computer Architecture?

• Exploit advances in technology
  – Make things Faster, Smaller, Cheaper, ...

• Which enables new applications
  – 3D movies 20 years ago?

• Make new things possible
  – Accurate one-month weather forecasts? Cure for cancer? Life-like virtual reality?

• The advancement of computer architecture is vital for the advancement of all other areas of computing!
Constantly Changing Definition of CA

• 50s to 60s: Computer Architecture ~ Computer Arithmetic

• 70s to mid 80s: Instruction Set Design, especially ISA appropriate for compilers

• 90s: Speculation: Predict this, predict that; memory system; I/O system; Multiprocessors; Networks

• 2000s: Power efficiency, Communication, On-die Interconnection Network, Multi-this, Multi-that. (We are here)

Job Description of a Computer Architect

• Used to be “Performance, performance, performance”
• Make trade-off of performance, complexity effectiveness, power, technology, cost, etc.

New Fads

• Availability
  – Where you store your photos, emails and shared docs today?
  – Cloud computing
• Reliability
  – Toyota blamed soft errors for the sudden acceleration problem
• Security
  – Intel acquired McAfee
• Power management
  – It is about money!
Stack of A Computing Problem

Problems
- Algorithms
- Programming Languages
- Compilers
- ISA

System Architecture
- Implementation
- MicroArchitecture

Logic and Circuits

Transistors

Manufacturing

Apps Trend
Architects’ Territory
Technology Trend
Today:

- Trends in Computer Industry
  - Technology
  - Performance
  - Power
  - Cost
Technology Trend

• Transistor density $\uparrow$ by 35% per year; Die area $\uparrow$ by 10-20% per year
  – More functionality
  – Transistor speed $\uparrow$ linearly with $\downarrow$ size $\rightarrow$ clock speed $\uparrow$. 
Moore’s Law (a.k.a. Intel’s Roadmap)

Exponential growth

Source: Intel Corp.
Moore’s Law (1965)

- “Cramming More Components onto Integrated Circuits”
  - Gordon Moore, Electronics, 1965
- Related trends
  - Processor performance
  - Memory capacity
    Twice as much in <2 years
Moore’s Not-Exactly-Law

• Not a law of nature
  – But fairly accurate over 42 years

• More about Moore’s Law at

• Transistor doubling every 18-24 months
  – But does not get you 2x performance

• Wire delays do not scale as well as logic devices
  – Global wire delay scales UPWARD w.r.t. gate delays.
How to use billions of transistors?

Computer architecture! How you organize resources to get more work done.

1. Instruction set architecture (ISA)
   - interface between HW and SW
   - different ISAs may be more/less effective for different target application areas

2. Microarchitecture
   - Techniques below the ISA level (transparent)
   - ex. pipelining, caching, branch prediction, superscalar, dynamic scheduling, clock-gating, ...

Review appendix C
Review appendix B
Review 1.3 and appendix A
Today:

• Trends in Computer Industry
  – Technology
  – Performance
  – Power
  – Cost
Performance Trend

- Fast clock speed and architecture innovation
- Multiple cores
Clock Speed Increases

- **Digital VAX-11/780**: 5 MHz in 1978
  - **15% per year**

- **MIPS M2000**: 25 MHz in 1989
  - **40% per year**

- **Digital Alpha 21064**: 150 MHz in 1992

- **Digital Alpha 21164A**: 500 MHz in 1996

- **Intel Pentium III**: 1000 MHz in 2000

- **Intel Pentium 4 Xeon**: 3200 MHz in 2003

- **Intel Nehalem Xeon**: 3330 MHz in 2010

1% per year
Today:

• Trends in Computer Industry
  – Technology
  – Performance
  – Power
  – Cost
Hitting a Power Wall

- Power/thermal limit has been reached with ↑ in frequency

Source: Intel Corp.
Power/Thermal Limit

- Intel 80386 consumed ~ 2 W
- 3.3 GHz Intel Core i7 consumes 130 W
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air
Power Consumption Trends

- Dynamic power: first term

- Capacitance per transistor and $V_{dd}$ are decreasing, but number of transistors is increasing at a faster rate;

- Static power: second term; is a function of transistor count, $I_{st}$, and $V_{dd}$

- Power consumption is already between 100-150W in high-performance processors today

- Energy = $P \times$ time

\[ P \propto \alpha CV_{dd}^2 f + V_{dd} I_{st} \]
Power vs. Energy

• Energy is the ultimate metric: it tells us the true “cost” of performing a fixed task

• Power (energy/time) poses constraints; can only work fast enough to max out the power delivery or cooling solution

• If processor A consumes 1.2x the power of processor B, but finishes the task in 30% less time, its relative energy is 1.2 X 0.7 = 0.84; Proc-A is better, assuming that 1.2x power can be supported by the system
Reducing Power and Energy

- Can gate off transistors that are inactive (reduces leakage)

- Design for typical case and throttle down when activity exceeds a threshold

- DFS: Dynamic frequency scaling -- only reduces frequency and dynamic power, but hurts energy

- DVFS: Dynamic voltage and frequency scaling – can reduce voltage and frequency by (say) 10%; can slow a program by (say) 8%, but reduce dynamic power by 27%, reduce total power by (say) 23%, reduce total energy by 17% (Note: voltage drop → slow transistor → freq drop)
Problem 1

• For a processor running at 100% utilization at 100 W, 20% of the power is attributed to leakage. What is the total power dissipation when the processor is running at 50% utilization?
Problem 1

- For a processor running at 100% utilization at 100 W, 20% of the power is attributed to leakage. What is the total power dissipation when the processor is running at 50% utilization?

Total power = dynamic power + leakage power
= 80W x 50% + 20W
= 60W
Problem 2

• If processor A consumes 1.4x the power of processor B, but finishes the task in 20% less time, which processor would you pick:
  (a) if you were constrained by power delivery constraints?
  (b) if you were trying to minimize energy per operation?
  (c) if you were trying to minimize response times?
Problem 2

• If processor A consumes 1.4x the power of processor B, but finishes the task in 20% less time, which processor would you pick:
  
  (a) if you were constrained by power delivery constraints? Proc-B
  
  (b) if you were trying to minimize energy per operation? Proc-A is 1.4x0.8 = 1.12 times the energy of Proc-B
  
  (c) if you were trying to minimize response times? Proc-A is faster
Problem 3

- Processor-A at 3 GHz consumes 80 W of dynamic power and 20 W of static power. It completes a program in 20 seconds.
What is the energy consumption if I scale frequency down by 20% (assume CPU bound)?

What is the energy consumption if I scale frequency and voltage down by 20%?
Problem 3

- Processor-A at 3 GHz consumes 80 W of dynamic power and 20 W of static power. It completes a program in 20 seconds.

What is the energy consumption if I scale frequency down by 20%?

New dynamic power = 64W; New static power = 20W
New execution time = 20/0.8 = 25 secs (assuming CPU-bound)
Energy = 84 W x 25 secs = 2100 Joules

What is the energy consumption if I scale frequency and voltage down by 20%?

New DP = 80*0.8^3 = 41W; New static power = 20*0.8^2 = 13W;
New exec time = 25 secs; Energy = 1350 Joules
Google Server Farms (Oregon)
Heat Dissipation

Pure copper

Cooler jet

3D Cooler Pro

Cooligy's µch

Cooking-Aware (or Colwell’s Charcoal-aware) Co

PS3 Grill
Today:

• Trends in Computer Industry
  – Technology
  – Performance
  – Power
  – Cost
# Depends on the Class of Processor

<table>
<thead>
<tr>
<th>Feature</th>
<th>Desktop</th>
<th>Server</th>
<th>Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price of system (USD)</td>
<td>$500-$5K</td>
<td>$5K - $5M</td>
<td>$10 - $100K (ex. high-end network routers)</td>
</tr>
<tr>
<td>Price of CPU (per processor)</td>
<td>$50 - $500</td>
<td>$200 - $10K</td>
<td>$0.01 - $100</td>
</tr>
<tr>
<td>Critical design issues</td>
<td>Price-performance, graphics performance</td>
<td>Throughput, availability, scalability</td>
<td>Price, power, application-specific performance</td>
</tr>
</tbody>
</table>
Desktop Systems

• Examples
  – Intel Core i7 (Quad-core)
  – AMD FX (Eight-core)

• Applications: everything (general purpose)
  – Office, Internet, Multi-media, Video Games...

• Goals
  – performance, performance/price
  – power \Rightarrow affects cost, noise, size
Servers

• Examples
  – IBM Power (up to 8 cores)
  – Sun Niagara (T1, T2, 8 cores)
  – Intel Xeon (up to 8 cores)

• Applications
  – infrastructure: file server, email server, ...
  – business: web, e-commerce, databases

• Goals
  – Throughput (transactions/second)
  – Availability (reliability, dependability, fault tolerance ...)
  – Cost not a major issue
Embedded

• Examples
  – Xscale, ARM, MIPS, x86, ... (many varieties)

• Applications
  – cell phones, mp3 players, game consoles, consumer electronics (refrigerator, microwave), automobiles, ... (many varieties)

• Goals
  – Cost, Power
  – Sufficient performance, real-time performance
  – Size (CPU size, memory footprint, chip count...)
Fabrication Costs

• CPU (die) size greatly affects cost of all systems (desktop/server/embedded)
  – Current CPUs 1-2 cm²
  – Embedded much smaller

  • cost and footprint really matters in cell phone or iPad

http://news.bbc.co.uk/olmedia/1140000/images/_1144917_dom_joly150pa.jpg
Yield

The % of manufactured devices that survives the testing procedure

13/16 working chips
81.25% yield

1/4 working chips
25.0% yield
Yield (2)

52 die, 81.25% yield \(\rightarrow\) 42.25 working parts / wafer

17 die, 25.0% yield \(\rightarrow\) 4.25 working parts / wafer

Assuming $250 per wafer:
- $5.92 per die
- $58.82 per die
Cost/Yield Equations (approximations)

Cost of Die = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}

Dies per wafer = \frac{\pi \times (\text{Wafer diameter} / 2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}

Die yield = \text{Wafer yield} \times \left( 1 + \frac{\text{Defects per unit area} \times \text{Die area}}{\alpha} \right)^{-\alpha}

Accounts for number of completely bad wafers

Typical: 0.4 defects per cm² in 90nm, but improves with time

Parameter related to complexity of manufacturing, typical $\alpha = 0.4$
Interaction of Price and Performance

• Add a new architectural feature to chip
  – for more performance, for less power, etc.

• Chip die size increases
  – Fewer dies per wafer
  – More defective dies

• Die testing more expensive
  – Must test whether feature works

• Die package more expensive
  – Larger package, maybe more pins
  – If feature needs more power, may need better heat sink
Goal of Processor Design

• Maximize performance
• Within the constraints of
  – Peak power, average power, thermals, reliability, manufacturing costs, implementation complexity, verification complexity, time-to-market, cost to manufacturer (Intel), cost to OEM (Dell), cost to end-customer (you)
  – Which really just says:
    • Maximize performance per $$$ (performance/price)

• Huge, multi-variable optimization problem!
  – Not all variables are equal and independent
The Rest of this Course

• Not as much *explicit* focus on price, although it will be kept in mind

• How do you organize these millions/billions of transistors to implement the ISA
  – data-processing (workers)
  – control-logic (managers)
  – memory (warehouse)
  – parallel systems (multiple worksites)