EE392C Lecture 1: Introduction & Technology Background

Christos Kozyrakis
cristos@ee.stanford.edu

http://www.stanford.edu/class/ee392c

Today’s Menu

• Introduction to EE392C
  ▪ Course focus
  ▪ Basic information
  ▪ Assignments & projects

• Processor technology background
  ▪ General-purpose processors: not a solved problem
  ▪ The vision: a CMP processor with coarse-grain reconfiguration capabilities
  ▪ Contributing technologies

EE392C: Adv. Topics in Computer Architecture

• Focus: next-generation general-purpose processors
  ▪ Single-chip multiprocessors (CMPs) with coarse-grain reconfiguration capabilities (aka Polymorphic Processors)
  ▪ Topics: architecture, programming language, compilers, operating-systems, applications, fault-tolerance, …

• Goals: initiate research in this area
  ▪ Review previous and current work
  ▪ Identify open issues and key opportunities
  ▪ Propose initial approaches and demonstrate their potential
  ▪ If we work hard enough, publish some ideas and results

Basic Information

• Lectures: Tue-Thu, 1.15-2.30pm, Room 61A
  ▪ Bring your favorite coffee/beverage but not your lunch…

• Web page: http://www.stanford.edu/class/ee392c
  ▪ Latest schedule, papers, notes, info, etc
  ▪ Check regularly

• Communications channels
  ▪ Newsgroup (su.class.ee392c) for questions & discussion
  ▪ Mailing-list (ee392c-spr0203-all@lists.stanford.edu) for announcements only
  ▪ Email the instructor for more specific issues

• Read the information sheet for details!
The EE392C Instructors Team

- Christos Kozyrakis
  - Assistant professor of EE & CS
  - christos@ee.stanford.edu
- Teaching Assistant: Metha Jeeradit
  - Maintains web-page & (some of) the project tools
  - metha@stanford.edu
- Administrative support: Chris Lilly
  - clilly@cs.stanford.edu
- All of you...

Your Participation

- Every class meeting
  - Read papers before class meeting
  - Actively participate in the class discussion
- Lead one class meeting
  - 10 minute introduction to the topic
  - Guide the open discussion
- Keep notes of one discussion
- Mini-presentation on emerging applications
- Project (in groups of 3-4 students)
  - Original research on a open issue
  - Includes proposal, presentation, final paper
- Review one final paper from another group

Who Should Take EE392C

- Grad. students interested in systems research
  - Architecture, compilers, operating-systems, ...
  - Or students in application areas interested in system implications (networking, databases, graphics, ...)  
  - Diversity of interests and experiences is good
- Prerequisites: officially none
  - Unofficially: one of EE282, CS243, or CS240
  - Talk to instructor for specific questions
- Enrollment limited to 30
  - To allow for interesting round-table discussions

and now for something completely different!
Aren’t We Done with Processors Yet?

• Performance improving at 55% per year since 1982
  ▪ Similar improvements in cost
  ▪ Using the same sequential instruction set (x86)

• Contributing technologies
  ▪ Faster transistors: ~20% per year
    ► Moore’s Law
  ▪ Fewer gates per pipeline stage: ~12% per year
    ► Deeper pipelines, better circuits design
  ▪ More instructions per cycle (IPC): ~23% per year
    ► Wide instruction issue, multiple ALUs, caches, speculation

Historical Performance Trend

Trouble in Paradise

• Moore’s Law good for at least another decade
  ▪ But long wires will slow down compared to transistors

• Pipelining problems
  ▪ Very deep pipelining hurts IPC
  ▪ Already at 8-16 inverters per stage, difficult to go lower

• IPC problems
  ▪ Inherently limited instruction-level parallelism (ILP)?
  ▪ Large HW structures for ILP difficult to clock fast

• Other problems: power consumption, complexity
  ▪ Both translate to higher cost

Deep Pipelining Implications

• High clock frequency, but modest performance gains
  ▪ Due to memory latency and branch delays
• Power consumptions increases dangerously!
Current State of IPC

- IPC: 1.5x per generation up to Pentium 3
- Pentium 4 relies on clock frequency, not IPC
  - Still, a lot of hardware & power goes to IPC

Rely Only on IPC?

- Instruction-level parallelism is limited
  - Branches, date dependencies, memory latency, …
  - Even optimistic studies predict ILP < 10
- Instruction-level parallelism is expensive to exploit

A Look at the Future

- Superscalar processor scaling with technology
  - Goal: maximum clock frequency without hurting IPC
  - Option 1: keep ILP hardware same but use deeper pipelines
  - Option 2: use smaller ILP hardware to keep pipeline depth same

Power Consumption

- Despite reducing power supply and transistor size
- Power problems
  - Portable systems: battery life
  - All systems: cost (power distribution, packaging, cooling, …)
### Design Efficiency & Complexity

- 20% of area for registers & ALUs
  - What the ISA promises to software
- 80% of area ILP overhead
  - Caches, predictors, …
- Critical for performance but …
  - They take up area and burn power
  - Software has little control on them
  - Many time-critical, global wires
  - No design modularity
  - Large design effort
  - Even larger verification effort

### Who Cares about Performance?

- Embedded & portable systems
  - Need 10s of GOPs at a few mWatts and few $\$
  - Examples: speech & visual recognition, video processing, graphics, wireless communications, …
- Servers systems
  - Need 100s of GOPS at a few Watts and a few $\$
  - Examples: OLTP, data-mining, web servers, video servers, network routing, bio-computing, climate modeling, …
- Desktop systems are fine!
  - But watch out for Mr. Paperclip…

### It Has to Be a Chip Multiprocessor

- Design constraints set the architecture
  - A modular collection of processors
- Modular design
  - Limit design/verification time
  - Great scalability
  - Easier clocking
  - Redundancy
- No long wire problems
  - Wires scale with design
It Has to Have Lots of Local Memories

- **Access latency**
  - Reduce by exploiting locality
  - Temporal or spatial

- **Power density**
  - Memories cooler than processors

- **Still many questions**
  - Which processor core?
  - Which local memory?
  - How much local memory?
  - How many cores, memories?
  - What interconnect & IO?
  - Optimized for what kind of parallelism?

Types of Parallelism

- **ILP**
  - 1-10x?

- **DLP**
  - 10s-100s?

- **TLP**
  - 10s-100s?

- My App

Focus on DLP & TLP

- **DLP & TLP advantages**
  - They scale with our applications
  - They can be explicit

- **How to balance DLP/TLP resources: look at custom processors!**
  - DSPs, ASICs, ASIPs, graphics chips
  - 10x to 1,000x better in performance/power than general-purpose
  - Efficiency by matching resources to application characteristics
    - In computation, memory, and communication structures

- **Customization comes at a price**
  - Minimum flexibility (programmability)
  - High development cost

The Challenge for Future CMPs

- **Can we get close to the efficiency of ASIC**
  - E.g. within 10x across all interesting applications

- **with the programmability**
  - Good for all “important” applications
  - Write code in HLL and compile

- **and the cost of a general-purpose processor?**
  - “Single” architecture but potentially many implementations

- **In other words:**
  - Can we design a CMP that adapts to the application needs?
  - A CMP that is “polymorphic”?
Examples of Coarse-grain Reconfiguration

• Processor side
  - Use a few cores as a vector engine for multimedia (DLP)
  - Use a few cores as a multithreading engine for OLTP (TLP)
  - Use a few cores as a VLIW engine for control (ILP)

• Memory side
  - Configure parts the local memory as cache, scratchpad, or hardware FIFO, etc
  - Configure a set of local memories as L2 cache, frame buffer, stream buffer, packet buffer, etc

Application Example: Set-top Box

To Make this Work We Need

• A configurable CMP architecture
  - That supports DLP, TLP, and some ILP

• A programming model for parallel applications
  - That is easy to use and compile

• A parallelizing compiler
  - That can figure out the types of parallelism in applications

• An operating system
  - That can handle the configurable nature of the system

• and, of course, applications
  - That work great in this environment

contributing technologies to investigate
(aka the ee392 outline)
Architecture

- Data-parallel architectures (Lec. 2)
  - Hardware for DLP (characteristics, benefits, issues)
  - Which features do we want to keep?

- Thread-parallel Architectures (Lec. 3)
  - Hardware for TLP (characteristics, benefits, issues)
  - Which features do we want to keep?

- Speculative Multithreading (Lec. 4)
  - How about difficult to analyze applications?
  - Parallelize optimistically, rollback if dependencies exist
  - How much hardware support is necessary?

Emerging Applications (Lec. 5)

- Which applications will dominate processor cycles?
  - E-commerce, web, video-on-demand, OLTP, data-mining…
  - Network routing, cryptography & security, wireless, …
  - Video processing, 3D graphics, voice and image recognition, probabilistic signal processing, …
  - Bio-computing, protein folding, DNA sequencing, …

- What are their characteristics?
  - Types of parallelism, working set, memory access pattern, scaling trends, bottlenecks, …

- Will explore with homework & mini-presentations

Polymorphic Architectures (Lec. 7-8)

- Review some recent proposals
  - What are their strengths and weaknesses?
  - Hardware and software trade-offs
  - Open issues?

- Project brainstorming in Lecture 6
- Project proposal due in Lecture 7

Software, Software, Software

- Programming models (Lec. 9)
  - Can we express parallelism in a way that is easy for the programmer to write and easy for the compiler to handle?

- Virtual machines (Lec. 10)
  - How do we virtualize polymorphic hardware?
  - What is the minimal OS support needed?

- Project reviews in Lecture 11

- On-line profiling (Lec. 12)
  - How do we measure & summarize execution behavior?

- Dynamic compilation (Lec. 13-14)
  - Can DCs help with parallelizing difficult apps?
  - Can DCs help with selecting optimal configuration?
Reliable & Adaptive Systems

- Fault-tolerance & reliability (Lec. 15)
  - HW & SW support for reliable system operation
  - Can we handle both transient and permanent faults?
- Machine-learning in system design (Lec. 16)
  - Can we use ML/genetic techniques to build better systems?
  - How do we write & run ML code efficiently?
- “Adding value to future processors” (Lec. 17, TBD)
  - Power management, debugging, …

- Project deadlines
  - Presentations in Lectures 18 & 19
  - Papers due in Lecture 19
  - Paper reviews due on 6/6

---

Project

- Topic
  - Architecture or software topic related to class focus
  - See list of proposed topics on class web-page
  - Can also propose your own topic
    - Design-only topics are discouraged
- Groups: 3-4 students
  - You are all expected to do a fair share of implementing, writing, and presenting
- Tools
  - Tensilica tools available for class projects
  - Can also use any public tool if right for your project
    - But check with instructor first

---

Final Notes

- Before you leave
  - Fill in class sign-up sheet
  - Sign up for discussion leading and note scribing
  - Pick up 2 papers for next lecture

- Next Lecture
  - Data-parallel architectures
    - Read the required papers & come prepared!
  - Class photos
    - Help your instructor learn your name...

- Start thinking about your project topic!!