The Transformation Hierarchy in the era of Multi-core

Yale Patt The University of Texas at Austin

HiPC 2007 Goa December 21, 2007 Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

Why Multi-core?

- Today:
 - One billion transistors, 4 GHz
- Tomorrow:
 - 10 to 100 billion transistors, over 10 GHz
- How can we harness all those transistors
- Manufacturers are building multi-core
- BUT we are not seeing very much benefit

Why so little benefit?

In my humble opinion:

- The transformation hierarchy
- Parallel programming

Until Recently (Phase I)

- Maintain the artificial walls between the layers
- Keeps the abstraction layers secure
 - Makes for a better comfort zone
- (Mostly) Improves the Microarchitecture
 - Pipelining, Caches
 - Branch Prediction, Speculative Execution
 - Out-of-order Execution, Trace Cache
- Today, we have too many transistors

BUT, do we use the transistors wisely?

- The Past: Commercial chips of the last few years

 poorly utilized area (i.e., large L2 Cache)
 Unwarranted accusation: Diminishing returns
- Today: Recent flurry of CMPs (deja vu cf.1982)
- Tomorrow: Terascale integration: Even worse!
 - Bandwidth: Too many cores
 - Power, energy: Too many transistors
 - What can we do?

Hint: Massive integration does NOT imply massive replication

The Answer: Break the Layers

- (We already have in limited cases)
- Pragmas in the Language
- The Refrigerator
- X + Superscalar
- The algorithm, the compiler, & the microarchitecture – The Alpha 21164 experiment

IF we break the layers:

- Compiler, Microarchitecture
 - Multiple levels of cache
 - Block-structured ISA
 - Part by compiler, part by uarch
 - Fast track, slow track
- Algorithm, Compiler, Microarchitecture
 - X + superscalar the Refrigerator
 - Niagara X / Pentium Y
- Microarchitecture, Circuits
 - Verification Hooks
 - Internal fault tolerance

Unfortunately:

- Computer People work within their layer
- Too few understand outside their layer

and, as to multiple cores:

• People think sequential

At least two problems

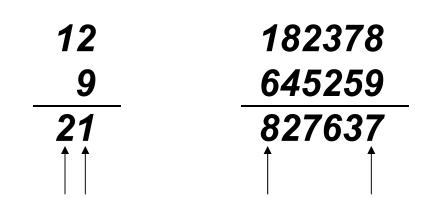
Conventional Wisdom Problem 1: "Abstraction" is Misunderstood

- Taxi to the airport
- The Scheme Chip (Deeper understanding)
- Sorting (choices)
- *Microsoft developers (Deeper understanding)*
- Wireless networks (Layers revisited)

Conventional Wisdom Problem 2: Thinking in Parallel is Hard

- Perhaps: Thinking is Hard
- What if the Programmer understood shared memory, and Synchronizing Primitives
 - Would it matter?
- Some simple programs for freshmen
 - Pipelining (aka Streaming)
 - Factorial
 - Parallel Search

Addition



On Education

- Object-oriented FIRST does not work
 - Students do not get it
 - *Memorizing isn't Learning (or, Understanding)*
- Motivated bottom up
 - Students build on what they already know
 - Continually raise the level of abstraction
- Don't be afraid to work the student hard
 - Students can digest serious meat
 - Students won't complain if they are learning
- No substitute for: Design it wrong,

Debug it yourself, Fix it, and see the working result.

We have an Education Problem We have an Opportunity

- Too many computer professionals don't get it.
- Applications can drive Microarchitecture
 IF we can speak the same language
- Thousands of cores, Special function units

 Ability to power on/off under program control
- Algorithms, Compiler, Microarch, Circuits all talking to each other ...
- Harnessing Terascale integration
 - Not Necessarily massive replication

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

Micro-40 Panel: What after von Neumann?

- Other panelists proposed
 - Quantum Computing
 - Biological Computing
 - Mimic the Human Brain
- I proposed
 - von Neumann

My conjecture:

In this era of multi-core and beyond, we have a better shot at making a difference by breaking the transformation hierarchy and teaching people to think (and therefore program) in parallel than by figuring out how the logic circuits work that make up the human brain.

Questions & Responses

- Does software or hardware drive development?
 I did not understand the question
- Entirely new arch or iterate current architecture?
 What is "entirely new"?
- Apps different at "massive integration scale"?
 FIRST: "massive integration scale" is NOT "massively replicated cores"
 - SECOND: Some old apps, some from dreamers

Questions/Responses (continued)

- Energy, Power?
 - Very important issue. Can we turn it off?
- Automatic tools transform code to parallel?
 Important, but do not hold your breath
- How does hardware drive architecture?
 Everything should drive everything else
- Optical interconnect
 - On-chip: It would be nice
 - Off-chip: it would be nice

Questions/Responses (continued)

• Three-dimensional?

- cube root (x) < square root (x)</pre>

- Reconfigurability: (Today's over-hype Number 1)
 Be careful (delay, energy, bw)
- Asynchronous: (Today's over-hype Number 2)

 Asynch structures will take multiple cycles
 AND will be synchronously controlled
- What new things can we do with a terachip?
 - Let the dreamers tell us

More Questions/Responses

- Metrics?
 - As always:
 Speed
 Cost
 Energy
 Reliability
 Availability

- BUT NOT: Utilization