Computing with Billion Transistor Chips

Guri Sohi

Computer Sciences Department
University of Wisconsin — Madison
URL: http://www.cs.wisc.edu/~sohi

The Computing Problem

- Aspect 1: How to create a static representation for the desired computation?
  - HLL, compilers, instruction set, etc.
- Aspect 2: How is the dynamic computation recreated and performed?
- Aspect 3: How are “program external” interactions performed?

A billion transistors on a chip can (fundamentally?) change Aspect 2. How?

How might change in Aspect 2 affect Aspects 1 and 3?

Program "Execution" (Aspect 2)

- How to create the dynamic sequence of operations from the static representation? (Program Sequencing)
- How to perform the effects of the operation? (Operation Execution)

Higher performance means speeding up the above

How might a billion transistors affect this?

Digression: Limited Hardware

- Have limited hardware resources. How does that impact computing?
  - Direct impact on Aspect 2: simplify operation execution and simplify program sequencing => RISC instructions.
  - Consequent impact on Aspects 1 and 3.

Program Sequencing

- Sequence through static representation
  - currently single sequencer with control speculation
- Speedup options
  - Multiple sequencers
  - Speculation (don’t wait until decision known)
  - Don’t redo work (learn as you go along)

Operation Execution

- How to execute operations efficiently?
  - dependences determine order of execution
  - operation latencies influence execution time
- Speedup options
  - Reduce operation latencies
  - Facilitate multiple operations per cycle (alleviate dependences)
  - Don’t re-execute operations that don’t produce new values

Reducing Operation Latencies

- Learn from the past and "speculate"
  - benefits depend upon costs of mis-speculation
  - Example 1: Memory location speculation. Predict storage element where a particular memory location resides (a.k.a caches)
  - Example 2: Operand value speculation. Guess the value generated by an operation. (load address speculation, load value speculation)
- Don't perform operation if result can be obtained in some other manner (e.g., is available from a previous execution)

Facilitating Multiple Ops per Cycle

- Alleviate dependences through data dependence speculation

Digression: Evolution of Speculation

- Storage location speculation: guess where a particular memory location resides (caches)
  - common since 1968
- Control speculation: guess value of Boolean variable
  - common since about 1990
- Speculate on values of non-Boolean variables
  - will be common when?
- Speculate on relationships between values
  - will be common when?
- Other forms of speculation?

Plentiful Hardware Scenario

- Individual instruction/operation de-emphasized
- Emphasis on efficient program sequencing
- Emphasis on (dependence) relationships between operations

Plentiful Hardware Microarchitectures

- Multiple sequencers
- Extensive support for a variety of forms of speculation
- Support for reusing computations

Current Paradigms

- Superscalar
  - single sequencer, with speculation (control then data)
- Multiprocessor
  - multiple (explicit) sequencers, no speculation
- Multithreaded
  - multiple (explicit) sequencers, no speculation
- Multiscalar
  - multiple (implicit) sequencers, aggressive speculation
- Known paradigms will need to evolve

Impact on Aspect 1

- What does the static program representation look like?
- How does static representation impact software used to create it (e.g., compilers)?

Static Program Representation

- Currently
  - Control Flow Graph, with single flow of control
  - Emphasis on representation of computation in node of CFG
- Future Needs
  - Multiple flows of control
  - Emphasis on node to node sequencing
  - Support for speculation
- Future Representations
  - CFG with annotations (e.g., Tera)
  - Hierarchical Task Graphs (HTGs)
  - Program Dependence Graphs (PDGs)
  - Others?

Creating Static Representations

- New static representations/hardware support will change compiler mindset
  - from "can do this in parallel" to "try and do this in parallel"
  - from "i know this is true" to "i think this is true"
- New mindset will demand newer (perhaps simpler?) compiler methods

Impact on Aspect 3

- Operating systems allow a program to interact with external entities
- Operating system mindset is sequential execution, i.e., single sequencer, with little or no speculation
- Operating system mindset will have to change to adapt to the multiple sequencer, heavily speculative hardware
  - what does it mean to handle multiple exceptions simultaneously?
  - what does it mean to handle exceptions speculatively?

It's the Memory System, Stupid

- What is memory?
  - a way of naming storage used to pass values from producers to consumers
- What are registers?
  - a way of naming storage used to pass values from producers to consumers
- Notion of registers has changed
  - were a way of naming a storage location
  - now a way of naming a value
- Notion of memory also needs to change from a location-oriented view to a value-oriented view

Future Memory Hierarchies

- Many on-chips resources devoted to a pool of storage used to create and maintain a value-oriented view of storage
  - storage pool will be used for much of inter-operation communication performed through “memory”
- Mappings will be used to go back and forth between location-oriented and value-oriented views

Summary

- Billion transistor chips will dramatically alter the hardware and microarchitecture structures used to compute
  - multiple sequencers
  - aggressive use of speculation
  - value-oriented notion of inter-operation communication
- New hardware structures will impact compilers and operating systems in ways that are not being thought about today
- Hardware evolution will happen
- Time to start thinking about impact of new hardware structures on compilers and operating systems.