Microprocessor Architecture: Looking Forward

Bill Joy
Sun Microsystems, Inc

I've Worked on Big Chips and Software Designs

- UNIX
- SPARC-I
- UltraSPARC
- Java (language and libraries)

Very Different Experience

- Hardware
  - » Simulate
  - » Physical Laws
  - » Multiple Levels
- Software
  - » Undefined Semantics
  - » Violates Causality
  - » Complex: Program and OS

Foundations for Reliable Software

- Well-defined Semantics
- Causality
  - » Bounds Checking
  - » Pointer Checking
  - » Abstractions for locality
- Describe as Programs in Language

Why GC? (Auto. Storage Mgmt.)

- Otherwise no semantics
  - » if **free** why still in use
  - » and reallocate with **malloc**
  - » then effect not explainable by the language semantics
- Otherwise unreliable software
- Like doing chips without CAD

For Modern, Quality Software

- Need Safe Testable Language
- Static and Dynamic Checking
- Concurrent (applications are)
- Aps built from components
  - » Implies Late Optimization / Codegen

Java Addresses this Problem

- Carefully Completely Specified
- Necessary Properties
  - » Concurrent
  - » Safe
  - » Strongly Typed
  - » Dynamically Linked

Some Difficult Parts of Java Specification

- Order of Evaulation and Exceptions
- Semantics of Loading, Linking, Initialization
- Parallelism and Memory Model
- Binary Compatibility
  - » in presence of strong typing

Next Generation Applications

- Machine/System Independent
- Testable
- Concurrent
- Media-Oriented
- Dynamically Linked
  - » Not by SQL or Scripting...

Implications for μproc. Architecture

- Plenty
- We've been studying / overoptimizing old APs with SPEC
- Things that will change
  - » mixes
  - » address traces
  - » branch distributions
  - » ...

Forces Toward Runtime Optimization

- Machine Independence
- Extension and Dynamic Assembly
- Better Performance
- Can't Optimize till RT if Program isn't assembled until RT

Forces Toward Parallel Apps

- Supported In Language
- Limits to Sequential Performance
- Cost of Scalarity
  - » N^2 area complexity
  - » Diminshing Returns (Amdahl)
- Pressure of Memory Hierarchy
  - » Double cache only 33% less misses
  - » Mostly idle transistors
  - » Increasing miss costs (in cycles)

Forces Toward New Kinds of Apps

- Media
  - » the last big thing
- The Internet
  - » the current big thing
- Embedded
  - » the future big thing

10 Ideas for Future μprocessors

- Ideas that are "with the trend"
  - » language
  - » technology
  - » applications
- Appropriate for New Designs

1. Don't Need Single Instruction Set

- use a Virtual Machine ala Java VM
  applications are
  - » instruction set independent
  - » pipeline independent (performance)
  - » operating system independent

  (not just applets)

2. Assume Dynamic Optimization

- necessary for dynamically assembled programs
- will affect statistics
- gets rid of lots of calls
- do easy branches in software
- use to move complexity to software

3. Support Concurrency Better

- low latency synchronization
  - » esp. with MP's (real or virtual) on one chip
- e.g. Java monitor support in hardware
- use to take pressure off memory (μthreading)

4. Microthread

- mitigate memory stalls
- better with simpler cpus (less state)
- seems necessary given memory hierarchy trends
- very relevant to database like applications

5. Consider Lots of Registers

- compilers can manage
- needed for media aps
- simpler than renaming

6. Let Software Handle Interlocks

- compiler models pipe anyways
- i.e. M.I.P.S.
  - » didn't work before because of fragile binaries
  - » with Virtual Machine, time has come

7. Consider If You Need an MMU

- separate concepts of mapping, protection
- with HLL (e.g. Java) need some (little) protection
- for many embedded aps don't need mapping
  - » realtime prevents use of "paging"
  - » object paging can be done in software...

8. Support Media Datatypes

- e.g. media (SIMD) ala VIS, MMX
- need more for 3D
- help dealing with compressed formats

9. Support Compression and Decompression

- between processor and memory
- to save memory
- to save bandwidth

10. Optimize Memory Hierarchy

- Hardware
  - » Study Object-Oriented Traces
  - » Study New-Media Traces
- Software
  - » New Algorithms with less misses
  - » e.g. trade computes for loads

Conclusion: Software Changes Coming

- Software Design Revolution Coming
- Safety and GC will be Ubiquitous
- Dynamic Assembly of Component Software
- At Low End: Direct Execution Simple, Efficient
- At High End: Dynamic Optimization

Conclusion: Hardware Opportunities

- Many Instruction Sets
- Cheap Concurrency, μthreading
- Lots of Usable Registers
- Move Interlocks to Software
- Eliminate Much of MMU

Conclusion: Work to Be Done

- Build Dynamic Optimizers
- Study Options for Microthreading
- Structures for Media Support
- Study Traces of New Applications
  - » After Dynamic Optimization!