Revolution on Demand: Push-button Specialized Supercomputers

Chita Das, Yale Patt, Kevin Skadron, Karin Strauss, and Steven Swanson

Man on the Moon Goals

- Push-button, drop-in co-processor for every application
  - 10,000X speedup over general-purpose for $10K
- Push-button, application specific reconfiguration
  - 1000X speedup in a commodity node
- Stay within today's power envelope, chip area, system form factor
- Scalable
  - All form factors -- handheld to datacenter
  - Scale out
  - Scale forward
- Man on Neptune goal
  - Custom data center for every application
  - Reconfigure entire system

Scenario

- Go to website, give algorithm, data size, target perf, physical constraints (form factor, power, cooling), push button
- FedEx delivers the next day
- Or, for cloud service, you just get an IP address
- (Mostly) transparent software layer

Problem Statement

- Many applications still require orders of magnitude improvement in performance, perf/W, or perf/$ to enable new capabilities
  - Mobile: Speech recognition, language translation, data analysis, diagnostics (think Star Trek tricorder), situational awareness
  - Desktop/Notebook: Still the sweet spot for programmers and users: Video processing, rich UIs/VR, interactive problem solving/data analysis (eg MATLAB)
  - Data center: Large-scale problems
    - Computational fluid dynamics
    - Drug discovery
    - Simulation and modeling (weather, geology, tsunami prediction, multi-agent modeling, etc)
    - Mining and machine learning (graph analytics, EXAMPLES...)

Problem statement, cont.

- Performance within a processing node matters
  - Sweet spots: mobile, single computer, single rack, small data center
  - Infrastructure, utility costs
  - Compute vs communication balance
- Specialization provides 10X-10,000X improvements in performance as well as perf/W, perf/$
  - Specialized computational units, memory hierarchies, interconnects, etc.
  - Examples, SIMD/MIMD/task pipelines; custom operations (FFT, IDCT, transcendentals, etc.); support for fine-grained synch...

Problem statement cont.

- Specialization will also be necessary as general-purpose scaling stops
  - Power delivery, cooling, pin-B/W all hitting walls due to fundamental physical limits
  - Beyond Moore's Law

Some Key Requirements

- Programs should be portable across diverse hardware
  - Same code should be portable across platforms and generations
    - Separate correctness from performance
  - System software (compiler/runtime/OS) must automatically map to specific resources
- Programmers should be able to drill down to optimize for specific hardware

Research Questions

- How do we pick which heterogeneous resources to use?
  - How to identify application-specific needs
  - How to generate appropriate specialized hardware units
- How do we connect them?
- How do we abstract it to the software?
  - Interface design/abstractions?
  - Programming models?
- How do we build it in a cost-effective manner?
  - Huge design automation and VLSI challenges
- How do we automate all this?

Architecting Components of Heterogeneous Systems

- Components must have
  - Clean interfaces
  - Scalability
  - Reusable
  - Composable
- A menu of components
  - Form factors
  - Memory interfaces
- Select from a menu of components and press go
  - Humans are involved only at the highest level of design

Opportunities for specialization

- Computing resources
  - ISA specialization
  - ASIC/ASIP cores
  - Reconfigurable logic blocks
  - Specialized serial/CPU/control cores
  - Non-silicon computation
- Communication resources
  - Specialized NoC, NIC
- Storage resources
  - Specialized/programmable memory interfaces
  - Access pattern-specific memory organization
    - Streaming, scatter/gather, etc.
- System topology
- Software!Software!Software!
  - Programming models and system software

Abstracting Heterog to Software

- Abstractions are key to integrating heterogeneity into the larger system
  - Language?
  - Hardware capability descriptions?
- Well-defined, boundaries
  - Minimize changes needed in upper layers to leverage heterogeneity

Driver applications: There's an App (and chip) for that!

- Large scale
  - Climate modeling
  - Multiscale modeling of the human body
    - From proteins to gross mechanics (muscles, bones)
  - Genomics and drug discovery
  - Graph analytics
  - Video analytics
  - Multi-agent simulations
- Portable
  - A supercomputing laptop for signal intellegence
- Embedded
  - "Tricorder"
- ...

Approach

- Select some specific, strategic application drivers
- Develop specialized accelerators for those applications
  - Emphasize design reuse
  - System-level building blocks
  - Distill HW/SW co-design principles
- Develop new SW abstractions, maybe DSLs
- Build and deploy in a small scale "real" system
  - Custom supercomputers-in-a-rack
  - Maybe Bluegene-style cards in backplanes
- Iterate!

Must Work with Real Applications (and Developers)

- Must work with real applications
  - Continuous feedback loop between our research and outcomes for strategic applications
  - Gain experience in exploiting hetero throughout the system
  - Enable specialization in a wider range of applications
    - Reduce DA cycle for specialized hardware
  - Understand the cost of abstraction (improve efficiency)
- Must work with real developers
- Translational research is how we build credibility
- We want the hardware to be transparent, not the benefits!

Other Challenges

- Design Automation
  - Automatically synthesizing specialized units
  - There's a wealth of technology to leverage
- Manufacturing
  - NREs are large
  - They need to become almost zero.

Timeline

- 5 Years
  - Implementation of *N* prototype systems
  - 1000x performance improvement
- 10 years
  - First fully-automated design completed
- 15 years
  - Customized computing systems become defacto line item for startups and research grants in computation-intensive fields.