Workshops on Advancing Computer Architecture Research (ACAR)

Summary Slides

Organizers:

Josep Torrellas (University of Illinois)

Mark Oskin (University of Washington)

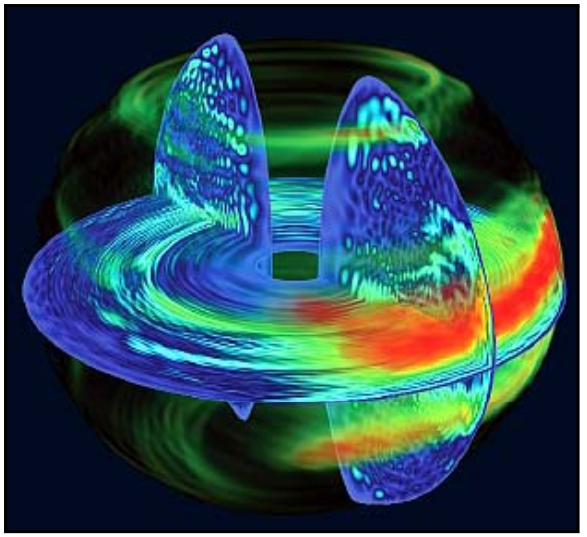
Contributors: Mark Oskin, Josep Torrellas, Sarita Adve, George Almasi, Luis Ceze, Almadena Chtchelkanova, Chita Das, John Davis, Sandhya Dwarkadas, Lieven Eeckhout, Bill Feiereisen, William Harrod, Daniel Jimenez, Mark Hill, Jon Hiller, Sampath Kannan, Krishna Kant, Martha Kim, Christos Kozyrakis, James Larus, Margaret Martonosi, Richard Murphy, Onur Mutlu, Satish Narayanasamy, Kunle Olukotun, Yale Patt, Andrew Putnam, Tim Sherwood, Anand Sivasubramaniam, Kevin Skadron, James Smith, Karin Strauss, Steven Swanson, Dean Tullsen, David Wood, Craig Zilles

Sponsored by Computing Community Consortium (CCC) Computing Research Association (CRA)
Held on February 22-23, 2010 in San Diego and on September 20-21, 2010 in Seattle

http://www.cra.org/ccc/acar.php

Extreme Scale Computing

- Energy and power consumption limitations require complete rethinking of computer architecture
  - Cloud computing:
    - Provide utility computing
    - Data centers will be used by billions of cost-conscious users and run applications written by many developers.
  - High-performance computing:
    - Revolutionize experimental computing techniques
- Goals:
  - Design machines that are 1000x more energy-efficient for the same performance and physical footprint
  - Consume only 20pJ/operation
  - Reduce the cost of data-center infrastructure to 1 Watt and $1 per month for the typical user
  - Enable individual programs to scale from a single-node system with tens of users to a full data-center deployment with millions of nodes and billions of users
- Wealth of research to do:
  - Chip and node architecture
  - Memory and storage
  - Runtime and OS
- ***Highly transformative:*** Extreme scale computing can provide cheap utility computing for billions of citizens



Energy and power consumption is the key limiter to progress

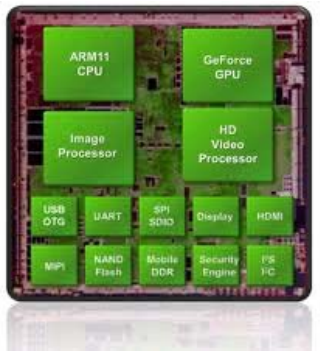
Architectures for Programmability

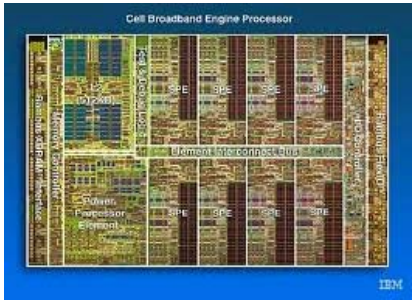
- The continued performance scaling of computer systems now requires extensive software and hardware changes to exploit parallelism
- Attaining a high-performance, correct 1000-core chip that is also highly programmable is a major challenge
- Goals:
  - Programming for parallel architectures should be as easy as it is now for sequential architectures
  - Maintain Moore's Law for performance (double the speed-up every 2 years)
  - Eliminate concurrency bugs
- Wealth of research to do:
  - Programming models and domain-specific languages
  - Correctness
  - Introspection
  - Scalable memory and communication fabric
  - Resource management
- ***Highly transformative***: Enabling programming for the masses of programmers

Historically, performance increases came without asking for significant software changes. However, performance scaling now requires extensive software and hardware changes to exploit parallelism

Specialized Architectures and Heterogeneity

- HW/SW specialization eliminates inefficiencies and overheads that come with the flexibility of general-purpose systems
- Want to develop the technologies necessary to deliver "turn-key" specialized computing systems quickly and economically
- Goals:
  - Design specialized architectures that deliver up to 10,000x speed-up for particular applications at an affordable cost
  - Manufacturing costs should be no greater than they are for conventional commodity computing systems.
  - Methodology should scale across form factors
  - Specialized system should integrate seamlessly into existing software systems
- Wealth of research to do:
  - Heterogeneous designs
  - Leverage existing reconfigurable designs with dynamic reconfigurability
  - Identification of the correct software abstractions
- ***Highly transformative:*** Obtain orders of magnitude improvements in performance, performance per Watt, and performance per dollar.





The ultimate goal is a fully automated generation of application-specific hardware for each program

The End of the Road for Conventional Instruction Sets

- The foundation of computing needs revisiting
  - The biggest problems in computing (security, scalability, programmability, reliability) are hamstrung by architectures built for completely different problems
- Current hardware abstractions developed 30-50 years ago
  - Technology constraints *very* different today: abundance vs. scarcity; battery power-constrained vs. build-your-own power-plant;
  - Instruction granularity forces processors to view applications through a narrow window
    - Obfuscates concurrency, intent, and knowledge the compiler/runtime/application writer knew
- Revisit fundamental assumptions of computer architecture:
  - Instruction-by-instruction Execution
  - Memory being typeless and monolithic
  - Virtual memory as only means of security and isolation
- Industry will not invest in such fundamental research.
- ***Highly transformative***: rethinking the fundamental assumptions about ISA can unlock the solutions of many important challenges in modern systems, among them: security, reliability, manageability, energy consumption, and possibly even performance.

Modern systems are skyscapers built on the ISA foundation of a bungalow

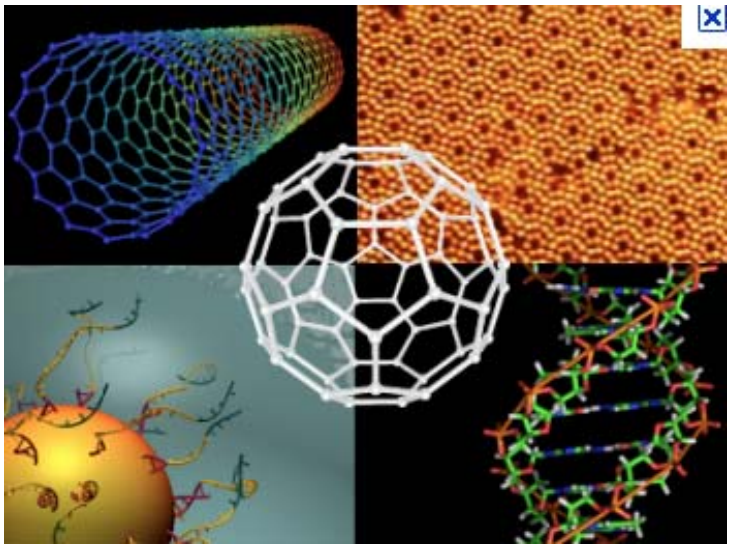
Secure, Reliable, and Predictable from the Hardware Up

- Current architectures are fragile
  - do not fail gracefully, poor isolation mechanisms
  - unsecured by default; impossible to truly keep a secret (information leaks everywhere)
  - unpredictable and unspecified behavior (non-deterministic actions lead to bugs and frustration)
  - Goal: machines worthy of the trust we place in them
  - Reliability: verifiably correct in the face of faults and errors and software bugs
  - Security and privacy: new hardware foundations to support information containment, privacy, cryptography, and self-monitoring
  - Programmable & predictable: provide users and developers the hardware/software tools they need to understand a program's behavior and performance
- Wealth of research to do:
  - Strong containment: provable assurance about information leakage
  - Mutually distrustful parties: user need not trust the operating environment; environment need not trust the users
- Trustworthy systems today cost $10,000 / line of code because they must be built on a shaky foundation. Typical systems cost $10/loc. A 2 orders of magnitude reduction in cost will **transform** the set of things we can trust machines to do. However, it will require *fundamentally new architectures* and ISAs to close gap.

The foundation of computing is breaking apart, and malicious parties exploit this fact for their gain. Without a serious commitment to science and engineering on this front, the losses to our economy and safety will only continue to grow

Exploiting Emerging Technologies

- Emerging technologies offer orders of magnitude improvement for challenging architectural design aspects; e.g. on-die light channels.
- Researching the architectural use-cases and implementation details early, engineering research brings science to market quicker and helps steer scientific inquiry.
- *Goal*: play a useful role in shaping the directions and providing solutions for problems in emerging technologies, including: quantum, synthetic biology, PCM, 3D stacking, photonics, nanotubes, etc.
- **Transformative and high-risk:** not all new technologies will be successful. Architecture research increases the likelihood of success, however, and some technologies can provide substantive leaps in performance and efficiency over CMOS.



Architecture research enables new technologies to enter the market quicker by identifying use cases and implementation challenges early.