Goals

- Across a multitude of reconfigurable (RC), multicore, and manycore architectures (Examples: ElementCXI ECA-64 – 64 physical ‘elements’ / 512 virtual, IBM Cell BE – 9 cores, nVidia Tesla GPU – 128 cores, Tilera – 64 cores)
- Develop a set of productivity-enhancing development tools and environments for the architectures
- Construct a suite of applications so that performance can be compared across architectures
- Characterize the computational capabilities of the architectures using the newly developed applications
- Research and develop reconfigurable (RC) application-to-hardware mappings (e.g., process-to-core), leveraging the above development tools and environments

Motivation

- How to map application processes to processor cores?
- Not all cores are created equal with respect to effective capability.

Results

- Hybrid Multicore: Smith-Waterman on Cell B.E.
  - 82x to 118x speedup
- Manycore: Smith-Waterman on GPGPU
  - 5.5x speedup

Approaches

- Analyze and build upon existing tools and environments for the targeted architectures, e.g., Element CXI
- Alchemy SDK
- Study process-to-core mapping (or more generally, application-to-hardware mapping) in RC, multicore, & manycore environments
- Characterize application performance across the targeted architectures to understand which application-to-hardware mappings perform the best … and then automate the process in the longer term

Future Work

- Continue to implement development tools by leveraging common abstraction patterns across architectures for better productivity and performance
- Reconfigurable: ElementCXI
  - Target autonomous computing
- Hybrid Multicore: Cell
  - Study autonomic RC scheduling of tasks for performance
- Manycore: AMD/ATI Radeon 3970 and Tilera TILE64
  - Implement data-parallel applications like atmospheric modeling
  - Transition from nVidia to AMD/ATI Radeon 3970 and Tilera TILE64

Conclusion

- Mapping reconfigurable (RC) software on advanced architectures is a difficult problem
- Intelligent RC process-to-core mapping on traditional multicore  100% improvement in application performance
- Efficient application-to-hardware mapping on hybrid multicore, manycore and RC platforms 116x speedup in application performance
- Hybrid multicore and manycore systems are amenable to SIMD-like processing
- Manycore GPGPU architecture is more appropriate for data-parallel applications, i.e., limited dependency between data elements and no branching between application threads
- Examples of data parallel applications: atmospheric modeling or high-speed network intrusion detection

V2: Application-to-Hardware Mapping for Advanced Architectures

Faculty: Dr. Peter Athanas (athanas@vt.edu) and Dr. Wu-chun Feng (feng@cs.vt.edu)

Students: Ashwin M. Aji (aaji@cs.vt.edu) and Saurabh Kulkarni (saurabh6@vt.edu)