Session 3:
Compiler and Library Technologies

Joe Germann / Sky Computers
Compiler and Library Technology Themes

- Compilers and their associated Libraries have made Significant Performance Enabling Advances in addressing the Complexities of Algorithm Mapping to Processor Architectures

- Historically the Solutions were addressed with:
  - 1980’s - The Brute Force Times: Fortran and Assembly Code, Macro Preprocessors, Simple Code Translators, and FPS Libraries; many of which required Assembly Code interventions to get Performance
    - Most Application were on uniprocessors and used Fortran and Assembly Code
  
  - 1990’s – Smart Tools Emerge onto the HPEC scene: Uniprocessor Vectorizers, Small Cluster Parallelizers, Memory Hierarchy Management, Performance Profiling, and Optimization Guiding Tools such as Event Analyzers
    - Most Applications migrated to Multiprocessors and used Fortran or C and Assembly Code

  - 2000’s - Opaque Software and Highly Specialized Hardware: Object Oriented Languages, Opaque Vector Libraries, Generalized Communication API’s, Class Templates, SIMD Processor Architectures, and Application Specific ASICS and FPGA’s.
    - Most Applications will be on Multiprocessors and will use C, C++, and possibly some JAVA
 Compiler and Library Technology Challenges

- The Efficient Algorithm Mapping and Performance Optimizations to Differing Solution Architectures (Microprocessors, FPGA’s, Interconnect Fabrics, High Availability Hardware)

- The Special Role and Relationship between Algorithm Processing and Interprocessor Communications in Large-Scale HPEC Applications

- The Ability for Reusable Software (i.e., C, C++, VSIPL, MPI) to be Optimized for Maximum Efficiency with Minimal Effort; preferably Completely Transparent to the Developer
  - All the While Exploiting the Maximum Potential of:
    - The Processor’s Architecture and it’s Utilization of Memory and I/O
    - The Interconnect Fabrics Performance and Reliability Characteristics
  - While insuring Application Performance, Reliability, and Survivability
Short Vector SIMD Code Generation for DSP Algorithms
- Markus Pueschel / Carnegie Melon University
- Franz Franchetti / Technical University of Vienna
- Jose Moura / Carnegie Melon University
- Christopher Ueberhuber / Technical University of Vienna

The advent of multiple SIMD instruction sets (Intel’s SSE/SSE2, AMD’s 3DNow, and Motorola’s Altivec) necessitate close cooperation between Algorithm Knowledge and SIMD Instruction choices.

The application of a SPIRAL algorithm definition front end has shown good performance gains when implemented using Advanced Compiler Techniques.
SC2 C to FPGA Compiler

- Maya Gokhale / Los Alamos National Laboratory
- Jan Stone / Stone Ergonaut
- Jan Frigo / Los Alamos National Laboratory
- Christine Ahrens / Los Alamos National Laboratory

This paper looks at the Streams-C Language and Compiler optimizations as targeted for use with the Xilinx/Virtex technology.

The work presented focuses on stream-oriented computing to include high-data-rate flow and compute intensive operations usually at low precision fixed-point arithmetic operations.

The efficiency of implementation between Streams-C and other approaches (manual hardware and software) will be presented.
Monolithic Compiler Experiments using C++ Expression Templates
- Lenore Mullin / MIT Lincoln Laboratory
- Edward Rutledge / MIT Lincoln Laboratory
- Robert Bond / MIT Lincoln Laboratory

This paper focuses on compiler optimizations for indexing operations that are implemented utilizing C++ Expression Templates.

Comparisons are made between Hand Coded Optimizations and Optimizations utilizing an Enhanced Portable Expression Template Engine.
Streaming and Dynamic Compilers for High Performance Embedded Computing

- Peter Mattson / Reservoir Labs. Inc.
- Jonathan Springer / Reservoir Labs. Inc.
- Charles Garrett / Reservoir Labs. Inc.
- Richard Lethin / Reservoir Labs. Inc.

Two Trends are Driving Next-Generation HPEC:
- 1) Multiple Processing Units on a Single Chip, and
- 2) Explicit Data Transfers between Processing Elements and/or Memory

This paper addressing the role of Stream Data Processing utilizing Streaming Languages, Streaming Compilers, and Dynamic Compilers