High-level Synthesis of Memory Bound and Irregular Parallel Applications with Bambu

High Performance Reconfigurable Computing

- **Application Domain**
  - Several emerging application are irregular
  - Process large pointer-based data structures (graphs, trees, grids)
  - Generate unpredictable memory accesses
  - Memory bandwidth bound, high synchronization intensity
  - Feature high degrees of (task-level) parallelism
  - Examples: bioinformatics, knowledge discovery, data analytics

- **Hybrid Architectures**
  - CPUs + FPGA co-processors (e.g., Convey HC-x systems)
  - Idea: execute performance critical kernel on dedicated hardware components
  - Promising for memory bound and irregular applications

High Level Synthesis (HLS)

- Allows the automatic implementation of a digital circuit, starting from its behavioral description (C/C++ code)
- May fill the gap between development costs and benefits of hardware acceleration (performance/power)

Main Limitations of Typical HLS Frameworks

- Coarse-grained parallelism exploitation
  - Most HLS approaches focus on Instruction-level Parallelism
  - Limitations to
    - Input language semantics
    - Architectural model (e.g., Finite State Machine with Datapath, FSM)
- Both inherently serial

Support for complex memory subsystems

- Large amounts of data to process
- Memory operations may represent a bottleneck for performance
- Fundamental to support shared/distributed memories
- Higher memory availability/bandwidth

Proposed Solutions

- Novel architectural model for HLS
  - Exploit a Parallel Controller for managing concurrent execution flows
- Hardware modules for interfacing shared parallel memories
  - Concurrency and synchronization management
  - Modules automatically integrated in the final design

Parallel Controller Architecture

- Set of communicating control elements (Execution Managers, EM)
- Each EM establishes when operations/tasks can start at runtime
- Dynamic execution paradigm
- Dedicated hardware for checking
- Satisfaction of dependency constraints
- Resource availability – interaction with arbiters (Resource Managers, RM)
- Linear complexity with respect to the number of operations/tasks
- Regardless of their latency

Memory Interface Controllers

- Map unpredictable memory accesses to several memory ports
- Dynamic resolution of memory addresses – support for pointer arithmetics
- Manage concurrency
  - Lightweight arbiters associated with no delay penalties
- Manage synchronization
  - Hardware implementation of atomic memory operations

Proposed High-level Synthesis Flow

- Evaluation on a set of OpenMP benchmarks
  - Up to six concurrent kernels
- Comparison (Performance/Area) against Bambu 0.9 and LegUP 3.0 (accelerators only)
- Target device: Altera Cyclone II
- Target frequency: 100 MHz
- Target memory architectures: single and 4-banked shared memories

Experimental Evaluation

Extends the Bambu framework
- Input: C-code specification
- Optionally annotated
- Output: Verilog implementation
- Hierarchical/modular synthesis process
- Follows the call structure of the input
- Enhances modules reusability
- Reduces final design complexity
- Exploits both Parallel and FSM controllers
- FSM is preferred for serial kernels
- Exploits both local and external memories benchmarks
- Local BRAMs on FPGAs
- Shared memories
- Distributed/multi-ported memories

Bambu is freely available at http://panda.dei.polimi.it/