Vision

Effective Many-Tile System-Level Programming Environment

- Parallelism: instruction-level, task-level, application-level
- Model of Computation (MoCs): sequential, parallel, dynamic, concurren, scalable
- Many-tile execution with high performance and efficiency

Fault-Tolerance at System-Level

- Conservative Analysis: providing reliability guarantees on mappings at design time (e.g., maximum temperature)
- Over-Provisioning: for non-critical applications
  - Empty (disengaged) tiles and clusters
  - In case of a fault: restart application on an empty tile or cluster

Platform Specification

- Scalable many-tile platform
- Hierarchical communication infrastructure:
  - First level (cortical columns): instruction-level parallelism, intra-process parallelism
  - Second level (cortical areas): process-network parallelism
  - Third level (neo-cortex): concurrent applications

Task-Duplication

- For safety-critical applications
- Distinction between pure computation processes and I/O processes
- Duplication of computation processes

Application Specification

- Hierarchical model of computation
- FSM controls multiple concurrent Kahn process networks
- Events causing scenario transition

Design Space Exploration

- Each “application scenario” is linked to one set of mappings
- Mapping optimization using PISA and EXPO
- MOEA (multi-objective evolutionary algorithm) module to compute the Pareto front of optimal mappings
- Performance analysis using MPA (modular performance analysis) framework to provide real-time behavior and temperature guarantees while optimizing average power consumption, data throughput, and latency

Run-Time Environment

- Hierarchically centralized system of controllers:
  - Many-core architecture divided into several clusters
  - Each cluster has a single (local) controller cluster
  - Cluster controllers under the control of a main controller
  - Cluster controller: receives events from running applications
    - Produces commands to the distributed system that lead to pausing/starting of tasks and queues

Hardware-dependent Software (HdS)

- Applications and run-time environment independent of the target platform
- HdS: software stack to abstract the hardware by the application code:
  - Operating system (OS)
  - Communication primitives (middleware)
  - Hardware abstraction layer (HAL)
- Multiprocessor targets:
  - Intel x86-based HPC platform
  - RISC (IRISC-based) embedded platform

Virtual EURETILE Platform (VEP)

- Virtual platform scalable to many simulated tiles/cores
- Non-intrusive controllability and visibility to foster debug and programming better than in HW
- Fault injection and many-tile concurrency debug frameworks
- Two simulation modes: fast host-compiled and accurate ISS-based

References


Contact

- Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
  - Email: firstname.lastname@inf.ethz.ch
- System-Level Synthesis Group, Laboratory TIMA, France
  - Email: firstname.lastname@imag.fr
- INFN-Laboratory, Italy, email: firstname.lastname@roma1.infn.it
- INFN-Laboratory, Germany, email: firstname.lastname@inf.ethz.ch

EURETILE: http://euretile.roma1.infn.it

EURETILE HPC platform

- Based on QUONiG (Quantum chromodynamics ON GPU)
- PC mesh based on Intel multi-core CPUs accelerated with high-end GPU and interconnected via 3-d torus network
- Communicating with custom interconnect (APEnet+); DPNs on FPGA-based PCI Express card
- Software-programmable accelerators in the form of ASICs

System Structure

- DNA-OS: component-based operating system
- Provides high-level mechanisms such as:
  - Threads (based on POSIX-I:2001 API standard)
  - Semaphores
  - Dynamic memory (malloc and free)
  - Inputs/outputs management
  - Low memory footprint
  - Minimal impact on the overall performances

EURETILE!