ALMA Project Overview

ALMA Consortium
Outline

- ALMA EU Project
  - Motivation
  - Project Overview
  - Target Architectures
  - ALMA Toolchain & Development Flow
  - Application Test Cases
  - Current Status
  - Summary
ALMA Project ID Card

- **Three year project:** 01/09/2011 – 30/08/2014
- **Funded by FP7:** 3.2 Million Euros
- **Official web site:** [http://www.alma-project.eu/](http://www.alma-project.eu/)
- **Coordinator:** Juergen Becker (KIT)
- **Technical Coordinator:** Nikos Voros (TMES)

![Logos of participating institutions]
Why do we need multi-core processors?

- Until ~2005 processor performance increase driven by
  - Clock speed
  - Execution optimization
  - Cache
- Power wall
- ILP wall

→ Led to multicore processors
→ Parallelism must be exposed by the programmer

(source: http://www.gotw.ca/publications/concurrency-ddj.htm)
Processor Breakthroughs ....

A major architecture disruption: multiprocessing and specialization will have a strong impact on software.

Technology limitations: perf. by parallelism no more by frequency => disruption in programming model, long term research challenges.

Domain oriented architectures: eg with predictable performance to control the timeliness in RT critical applis, dynamic reconfiguration for adaptive, distributed critical architectures (multilevel RT composability).

Source: G. Edelin (Thales), 2009
## Motivation

<table>
<thead>
<tr>
<th>End user perspective</th>
<th>Target architecture perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Explore/Develop algorithms</td>
<td>• Multi-Processor System-on-Chip</td>
</tr>
<tr>
<td>• Use a simple, comfortable language</td>
<td>• Parallel processor cores</td>
</tr>
<tr>
<td>• E.g. Matlab, Scilab, ...</td>
<td>• Parallel programming model</td>
</tr>
<tr>
<td>• Don’t want to care about</td>
<td>• E.g. pthreads, MPI, OpenMP</td>
</tr>
<tr>
<td>• data types</td>
<td>• Parallelism with the processor cores</td>
</tr>
<tr>
<td>• parallelism</td>
<td>• Single Instruction Multiple Data</td>
</tr>
<tr>
<td>• End result</td>
<td>• Very Long Instruction Word</td>
</tr>
<tr>
<td>• Performance</td>
<td>• Native data types</td>
</tr>
<tr>
<td>• Energy efficient</td>
<td>• E.g. 32-bit integer</td>
</tr>
<tr>
<td>• Cost efficient</td>
<td>• Other data types perform inefficient</td>
</tr>
<tr>
<td>• Fast development time</td>
<td></td>
</tr>
</tbody>
</table>
ALMA in a Nutshell

- **Hide the complexity** of the underlying hardware to the end user
- ALMA will develop an approach for compiling annotated Scilab code to **MPSoc architectures**
- **Algorithms and tools** for
  - **High-level**, platform-independent application code performance estimation and optimization
  - Identification of possible **partitions** and their placement on different resources of the underlying architectures
  - **Data type binding** and data **parallelization** to exploit data-level parallelism
- Develop an unified **SystemC simulation framework** to provide an environment for simulating MPSoCs
- Two **state-of-the-art architectures** provided by RECORE and KIT
- **Net result**: smaller application **development time/effort** and faster **time-to-market**
Objectives

- Extend Scilab for optimization on high-level system models
- Develop a parallelization and optimization environment
- Employ and extend two different architectures
- Parallel code generation
- Parallel code simulation
Challenges for Compiling Scilab to MPSoCs

- Scilab (Matlab-like) programming language
  - Dynamic typing (scalars, vectors, matrices)
  - Pointer-free, i.e. no memory aliasing problems
  - End users typically use floating-point data types
  - Natural parallelism within vector operations

- MPSoC target architectures
  - Exploit coarse-grain parallelism (task-level)
    - Distributed memory
  - Exploit fine-grain parallelism (instruction-level)
ALMA Architectures (1/2): Recore X2014

- **Scalability by virtue of**
  - Packet-switched Network-on-Chip
  - Distributed memories & I/O
  - Distributed control
  - Distributed processing cores

- **Xentium® processing tile**
  - Fixed-point DSP processing
  - 10-issue VLIW processor
  - SIMD capability
  - Streaming communication services

- **Reconfigurability**
  - Smart memory tile (RAM/FIFO)
  - Separate applications from each others
  - Guarantee QoS
  - Fault tolerant application mapping
ALMA Architectures (2/2): KIT Kahrisma

- Dynamic reconfigurable MPSoC
- Modules can be reconfigured to processors or DSPs
- Dynamic clustered VLIW processor instances
- Local scratchpad memory
- Non-coherent access to main memory
- Communication between cores through a network
Outline

- Agenda
- ALMA EU Project
  - Motivation
  - Project Overview
  - Target Architectures
  - ALMA Toolchain & Development Flow
  - Application Test Cases
  - Current Status
  - Summary
ALMA Development Flow (overview)

Embedded application design

Translation to Scilab & annotations

ALMA algorithm parallelization tools

PARAMETERS FOR ALGORITHM OPTIMIZATION

C-based code with parallel descriptions

Recore C-compiler

KIT C-compiler

Executable binary (for simulator and HW)

Multi-core simulator

Feedback for optimization

Optimized application code on multi-core platform

Multi-core hardware design

Abstract hardware description (ADL)

Structural hardware description
Input for the ALMA tools

ALMA dialect of the Scilab language

- Subset of Scilab language

- Extended by a preprocessing language
  - Variables declaration
  - Static types specification
  - Maximum size of vector and matrix data types definition

- Extended by an annotation language for supporting parallelism extraction
**ALMA Front-end tools**

- **Scilab Front-End (SAFE)**
  - Parses Scilab source code and produces high level intermediate representation (HLIR) expressed in C

- **ALMA profiler (aprof)**
  - Early *performance estimation* at the HLIR level

- **High-Level Optimizer (HLO)**
  - Applies platform *independent optimizations* to the HLIR
Parallelization Tools (Fine grain extraction)

- **Floating point to fixed point**
  - No hardware support for FP in embedded multi-core systems
  - Provide a automated floating to fixed point conversion tool.

- **SIMD/SWP parallelization**
  - Loop parallelization and layout optimization for SIMD ISA.
  - Explore perf./accuracy trade-off in fixed point encodings
Parallelization Tools (Coarse-grain extraction)

Coarse-grain parallelism extraction and optimization

- Responsible for the **global** optimization
- Transformation of ALMA IR CFDG to Hierarchical Task Graph (HTG)
- Resource availability from ADL
- **HTG partitioning** to cores
- **Optimal mapping** and **scheduling** of tasks to architecture resources
- Exploits profiling information from the simulator for better resource usage estimation
Parallel platform code generation

- Generates target-specific C code
  - Maps Scilab variables to memory locations
  - Expresses communication
  - Expresses SIMD instruction as intrinsics
- Uses Recore/Kahrisma C compiler
  - Exploits ILP by VLIW compilation
  - Generates executable for the hardware and simulator
Multicore architecture simulation

- Simulation of **ALMA target architectures**
- Retargetable
  - Structure defined by ADL
  - Implementation by library of SystemC modules
- Mixed-accuracy simulation
  - Behavioural or cycle-accurate
  - For individual modules (**processor core, memory subsystem, network**)
- Collect **profiling and tracing information**
ALMA Architecture Description Language (ADL)

ALMA ADL

- **Architecture Description Language (ADL)**
- **Tailored to** the requirements of **ALMA**
  1. Enables **target independence** of the compilation toolchain
  2. Used as architecture description for the **simulator**
  3. Enables design-space exploration
- **Compact specification** of regular MPSoC structures by for and if constructs
- **Structural specification** annotated with **behavioural information**
Outline

- Agenda
- ALMA EU Project
  - Motivation
  - Project Overview
  - Target Architectures
  - ALMA Toolchain & Development Flow
  - Application Test Cases
  - Current Status
  - Summary
IEEE 802.16e PHY Layer in $N_T \times N_R$ MIMO Configuration

- Typical example of a state-of-the-art wireless communication system
  - Application requirements impose hard, real-time constraints.
  - Design time must follow shrinking time-to-market.
Test Case (2/2): Image Processing

- Feature based algorithm for object recognition and **multi-object tracking**
- Use of Scale Invariant Feature Transform (SIFT)
- The final goal is to run such applications in smart cameras
Summary

- **ALMA Goal:** Hide the complexity of the underlying hardware to the end user
- ALMA will develop an approach for compiling annotated Scilab code to **MPSoCs**
- **ALMA toolchain components**
  - Front-end tools (Scilab parser, high-level optimizer, profiler)
  - Fine-grain parallelism extraction
  - Coarse-grain parallelism extraction
  - SystemC multi-core simulator
- **ALMA toolchain** is kept platform independent by a novel **Architecture Description Language**
- Two **state-of-the-art architectures** provided by RECORE and KIT
- Evaluated by two test cases from **Telecommunications** and **Image Processing** domain
Thank you!