An innovative compilation tool-chain for embedded multi-core architectures

M. Torquati, Computer Science Department, Univ. Of Pisa Italy
29/02/2012, Nuremberg, Germany
This is a joint work with...

- M. Amini, S. Guelton, R. Keryell, V. Lanore, F.-X. Pasquier from **HPC Project**
- M. Barreteau, R. Barrere, T. Petrisor, E. Lenormand from **Thales Research & Technology** Paris Area, France
- C. Cantini, F. De Stefani from **SELEX Sistemi Integrati**, Italy
- M. Torquati, M. Vanneschi from **University of Pisa**, Italy

... within the **SMECY project** (www.smecy.eu)
Talk outline

- Introduction
- The proposed compilation Tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results
- Conclusion and Future Works
Introduction

The proposed compilation Tool-chain

Spear-DE and Par4All

High-level Intermediate Representation

Application scenarios

First results

Conclusion and Future Works
Scenario: HW

- **Multi-cores**
  - Increasing number of cores per socket
  - Decreasing single core complexity
  - Increasing complexity of memory hierarchy

- **GP-GPUs & Accelerators**
  - Better control management and increasing n. of cores on GP-GPUs
  - Increasing n. of GP accelerators (MIC, Tilera, P2012...)
  - Many FPGA-based platforms
Scenario: SW

- **Multi-cores**
  - `#pragma`-based approaches (OpenMP)
  - Threads + parallel pattern/containers (Cilk, TBB, FastFlow, CnC, ...)

- **GPUs and Accelerators**
  - From CUDA to OpenCL
  - From vendor specific code acceleration to OpenACC

Developing efficient parallel code on many-core platforms is still very difficult and time consuming
The SMECY project

- European initiative started Feb, 2010 (3-yr project)
- Mix of research, industrial and SMEs (29 partners)
- 3 application domains and 2 reference platforms:
  - EdkDSP academic (UTIA)
  - P2012 industrial (ST)
The SMECY project

▶ Vision:
  ▶ No single universal compilation tool-chain exists
  ▶ Once both application domain and target platform are fixed, we can have an efficient tool-chain

▶ Objective:
  ▶ Providing compilation tool-chains adapted both to application domain and to targeted platform.
  ▶ www.smecy.eu
Progress ...

- Introduction
- The proposed compilation tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results
- Conclusion and Future Works
Tool-chain targets and motivations

- Radar signal processing application domain
- x86 multi-core + GPUs platform
  - shared-memory platform model
  - P2012 multi-cluster platform coming soon

The tool-chain leverages on a high-level Intermediate Representation (IR) that is able to:
  - clearly decouple the functional view from the low-level platform details
  - allow efficient code generation and debugging
Tool-chain principles

- 3 tired tool-chain: front-end, IR, back-end
- Explicitly parallel approach offered to the application programmer:
  - separation of concerns between application & system programmer
- IR devised within the SMECY project
- Efficient back-end tool that is able to produce efficient code for multiple target platforms using all the information provided by the IR
The Tool-chain picture

SpearDE

Modelling

Application Architecture

Parallelisation

Task parallelism

Data parallelism

Space / Time optimisations

Code generation

Intermediate Representation (IR)

Preprocessing

IR static analysis (loop fusion, loop parallelisation, ...)

Code generation

OpenCL optimised code

Target architectures

Front-end

Back-end

#pragma omp parallel section // OpenMP-based pragmas (high level IR)
{
    #pragma omp section
    {
        #pragma smecy map(G4,0) arg(3, [4][32][196], in) arg(5, [4][32][193], out)
        //call to convol_CF
        fusion_ApplicationModel_F_Comp_imp(64,64 (Cplfloat(*))[32][256]&UG.SegB. ....

Parallelised application

Functional application

GPU

Multi-core chip

P2012

ARTEMIS Joint Undertaking
Progress ...

- Introduction
- The proposed compilation Tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results
- Conclusion and Future Works
Tool chain front-end: Spear-DE

- A graphical tool to implement high performance data-streaming applications onto embedded parallel machines
  - High level application and architecture models
    - Only kernel codes are provided by the user (e.g. libraries)
  - Expert-driven approach regarding parallelisation
  - Automatic coherency checks (e.g. data availability, memory occupancy)
  - Performance simulation (using SystemC TLM)
    - Design Space Exploration
  - Automatic code generation on multiple targets (e.g. Multi-cores DSP, FPGA, GPU) using various languages (e.g. C, MPI, OpenCL)
    - No need to maintain the glue code in comparison with legacy code
Spear-DE design flow

Modelling
Application
Architecture

Parallelisation
Task parallelism
Data parallelism
Space & time optimisations

Code generation
Performance Simulation

Functional application
Parallelised application
Architecture model
Gantt chart
Tool-chain back-end: Par4All

- Par4All is an automatic parallelizing compiler
- Starting from well-written sequential code, Par4All can generate parallel code (OpenMP, CUDA, OpenCL...) for several parallel architectures.
  - The IR follows specific coding rules that Par4All is able to “understand”
- Based on PIPS parallelizer engine
- Open-source project (www.par4all.org)
Par4All design flow

- Par4All is a Source-to-Source compiler
  - Final code generation leverages on highly optimized vendor compiler
  - Able to capture or generate architectural details when needed (SIMD or intrinsic instructions)
  - Performs polyhedral model-based compilation via the PIPS framework
Progress ...

- Introduction
- The proposed compilation Tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results
- Conclusion and Future Works
Intermediate Representation

Requirements (from the SMECY project):

- Extensible
- Able to cope with big number of tools
- Close to standards

Objectives:

- Easy to manipulate (for low level optimizations and debugging)
- Providing a good balance between performance and portability
SME-C Intermediate Representation

- SME-C follows a `#pragma`matic approach:
  - Embedded C standard
  - OpenMP 3.0 `#pragma(s)` (task extensions)
  - Added some `#pragma(s)` to manage **mapping** (i.e. where a function will be executed)
  - Added **memory usage** annotations
  - **Communication** `#pragma(s)` to detail source and sink memories
SME-C parallelization schemas

- **OpenMP directives:**
  - `#pragma omp parallel for private (....)`
  - `#pragma omp parallel section`

- **Mapping directives:**
  - `#pragma smecy map(GPU,0) ....`

- **Memory access hints:**
  - `.... arg(2, [32][256],in) arg(3, [16][512],out)`

- **Communication directives:**
  - `#pragma smecy communication src(x86,0,F) dst(GPU,0,F)`
Progress ...

- Introduction
- The proposed compilation Tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results
- Conclusion and Future Works
Test cases

► Radar Signal Processing application domain
  ▶ data-streaming
  ▶ strict latency and bandwidth requirements

► We tested 2 applications:
  ▶ The Cholesky decomposition (used for the RT-STAP radar algorithm)
  ▶ The CHIRP filter (a simplified radar signal processing chain)
Parallel schemas

- Block-Cholesky decomposition steps

- CHIRP
Progress ...

- Introduction
- The proposed compilation Tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results
- Conclusion and Future Works
Results obtained on CPU

**Cholesky speedup**

- **1 Intel i7 980X @3.33 GHz**
- **16x16 block size**
- **~4X speedup**

**CHIRP speedup**

- **2 Intel X5670 @2.93GHz**
- **256x32x8 elements**
- **~7X speedup**
Results obtained on GPU

- On nVidia GTX 480, generating OpenCL 1.1

<table>
<thead>
<tr>
<th></th>
<th>Seq. Time (ms)</th>
<th>OpenCL Time (ms)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky (2x2 block size)</td>
<td>8440</td>
<td>560</td>
<td>15X</td>
</tr>
<tr>
<td>CHIRP</td>
<td>7</td>
<td>0.3</td>
<td>23X</td>
</tr>
</tbody>
</table>

By hand-tuning the OpenCL code generated automatically, we obtain a **35x speedup** for the Cholesky application (239 ms).

- We have to optimize code generation
Progress ...

- Introduction
- The proposed compilation Tool-chain
- Spear-DE and Par4All
- High-level Intermediate Representation
- Application scenarios
- First results

Conclusion and Future Works
Conclusion and Future Works

▶ A new tool-chain leveraging on the SME-C IR has been proposed and tested
   ▶ SME-C definition is still an ongoing work within the SMECY project

▶ The results obtained are encouraging
   ▶ but more work has to be done to optimize code generation

▶ Code generation for the P2012 multi-cluster platform has to be addressed.
Thank you for your attention!

Massimo Torquati  <torquati@di.unipi.it>
Spear-DE

► Semantics
  ▶ Model of Computation: Array-OL formalism (close to Multi-Dimensional SDF)
  ▶ Model of Execution: Multi-SPMD

► Usage
  ▶ Choosing the right architecture for a given application domain (rapid prototyping)
    ▶ The same high level modelling enables to quickly address several targets
  ▶ Sizing future architectures (through performance simulation)
    ▶ e.g. increase memory sizes, add computing resources, ...
  ▶ Generating code for real time execution or simulation acceleration
Tool-chain actors

- **Spear-DE** from Thales France as front-end tool
- **SME-C** Intermediate Representation devised within the SMECY project
  - ...with the contribution of all SMECY partners
- **Par4All** from HPC project as back-end tool