AXIOM

Agile, eXtensible, fast I/O Module for the cyber-physical era

The AXIOM Software Layers

DSD, Funchal, 26th August 2015

Carlos Alvarez, Eduard Ayguadé, Javier Bueno, Antonio Filgueras, Daniel Jimenez-Gonzalez, Xavier Martorell, Nacho Navarro
Barcelona Supercomputing Center, Spain

Carlos Segura, Carles Fernandez, David Oro, Javier Rodriguez Saeta
Herta Security, Barcelona, Spain

Dimitris Theodoropoulos, Dionisios N. Pnevmatikatos
FORTH-ICS

Davide Catani
SECO
Arezzo, Italy

Pierluigi Passera, Alberto Pomella
VIMAR SpA, Marostica, Italy

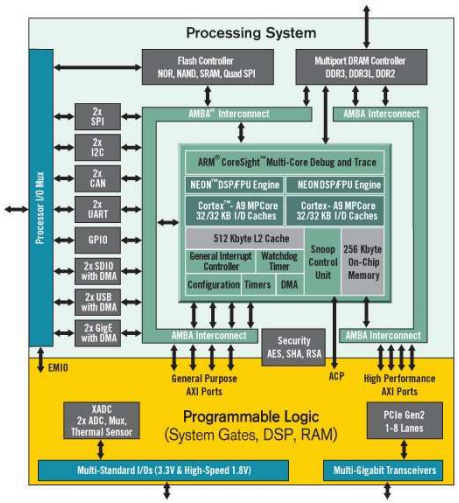
Claudio Scordino, Paolo Gai
Evidence Srl
Pisa, Italy

Antonio Rizzo
University of Siena
Siena, Italy

Coordinator:
Roberto Giorgi
University of Siena

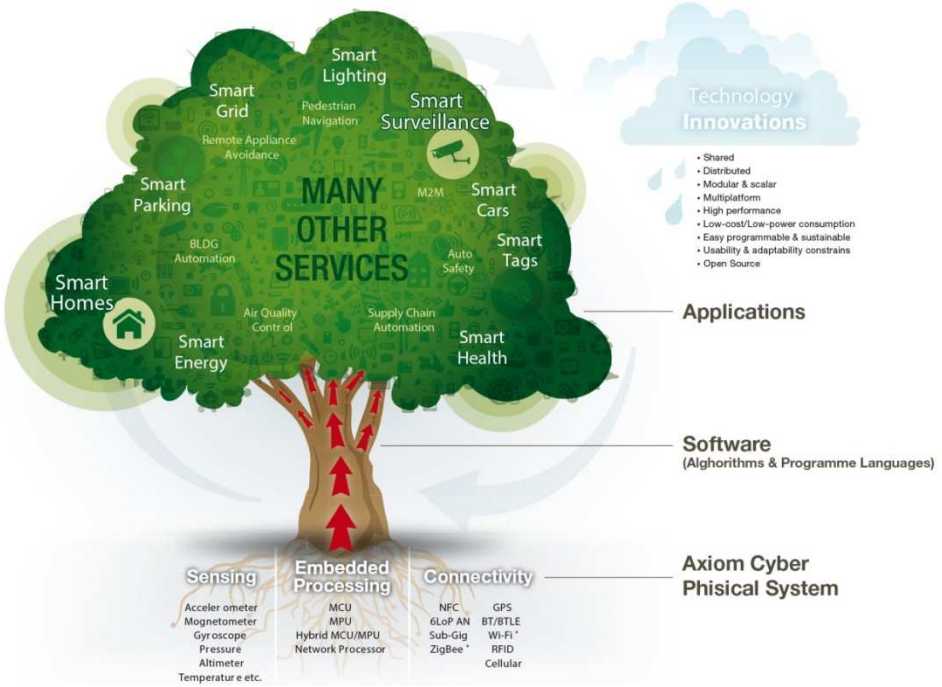
Scenario

- Heterogeneous many-core designs to fight the power wall
- E.g. CPU + GPU in a single chip
  - AMD Fusion APUs
  - < 20 watt, thousands of cores
  - GPU lacks flexibility
- CPU + Programmable logics
  - E.g., Xilinx Zynq
  - ARM Dual-core + Artix7 | Kintex7
  - Programmability opportunities



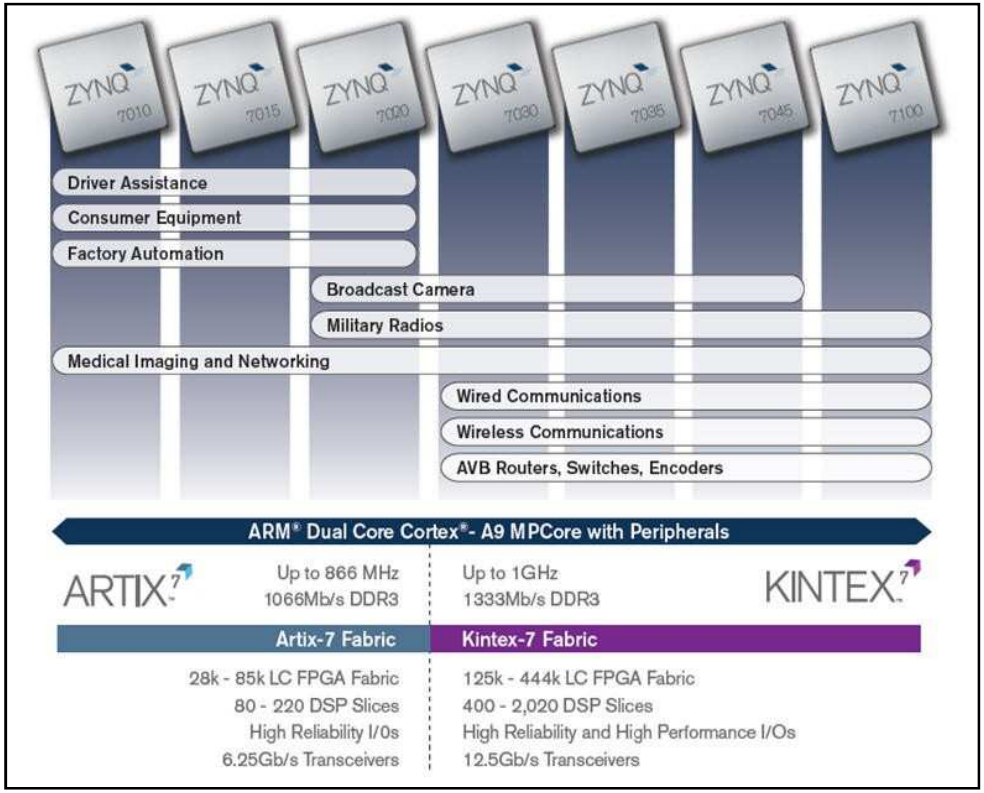
The AXIOM Project

- Realizing a small board that is flexible, energy efficient and modularly scalable
- Easy programmability of multi-core, multi-board, FPGA
- Leveraging Open-Source software to manage the board
- Easy Interfacing with the Cyber-Physical Worlds
- Enabling real-time thread scheduling
- Contribution to Standards



- BSC – Programming models
- EVI – Runtime, OS (Linux RT scheduler)
- FORTH – High-speed interconnects
- SECO – Hardware module realization
- UNISI – Simulation, evaluation, coordination
- VIMAR, HERTA – Applications

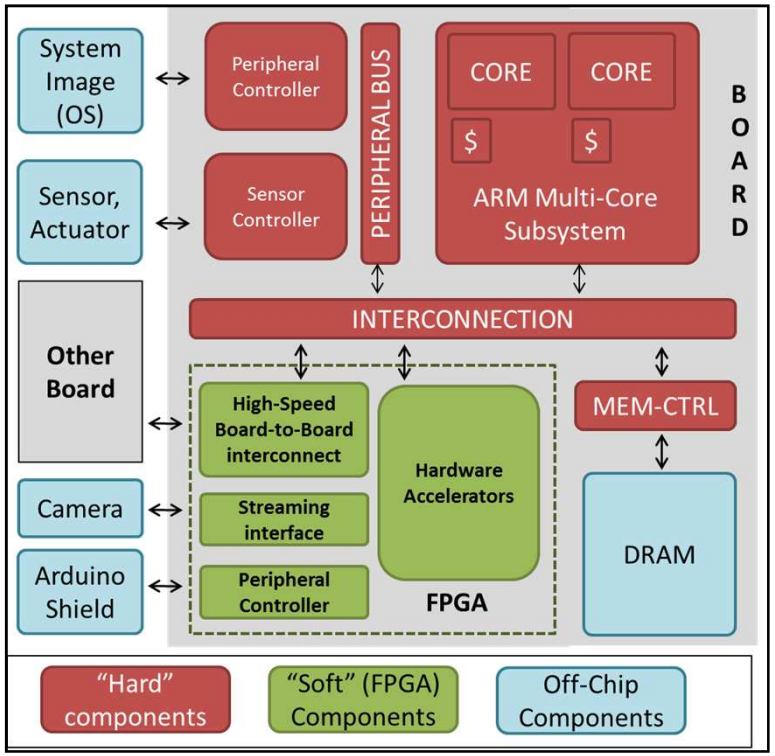
Xilinx Zynq



- Year 2011
  - 2013: Zynq-7100
- Broad range of high-end embedded applications
  - ADAS, medical, wireless, vision, ...
- ARM host to increase programmability
- Different FPGAs
  - Depending on requirements
  - Still, the Prog. Logic needs to be programmed!

The AXIOM board

- Integrate multiple cores + FPGA in the same chip
- Produce prototypes of single-board computers
  - connected with high-speed board-to-board comm medium
- Easily plug peripherals and components
  - Such as *Arduino Shields*



The AXIOM software stack

- Linux-based OS
- Highly-expressive programming model OmpSs
  - OpenMP-like
- Portions of applications on FPGA
  - Mapping of (HW) task to resources: OmpSs@Zynq

At cluster level:

- OmpSs@Cluster (+ OmpSs@Zynq)
- DSM (+ OmpSs@Zynq)

Current status: MxM example

```
void matrix_multiply(int BS, T a[BS][BS], T b[BS][BS],
                             T c[BS][BS]) {

    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
            T sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
        }

}
```

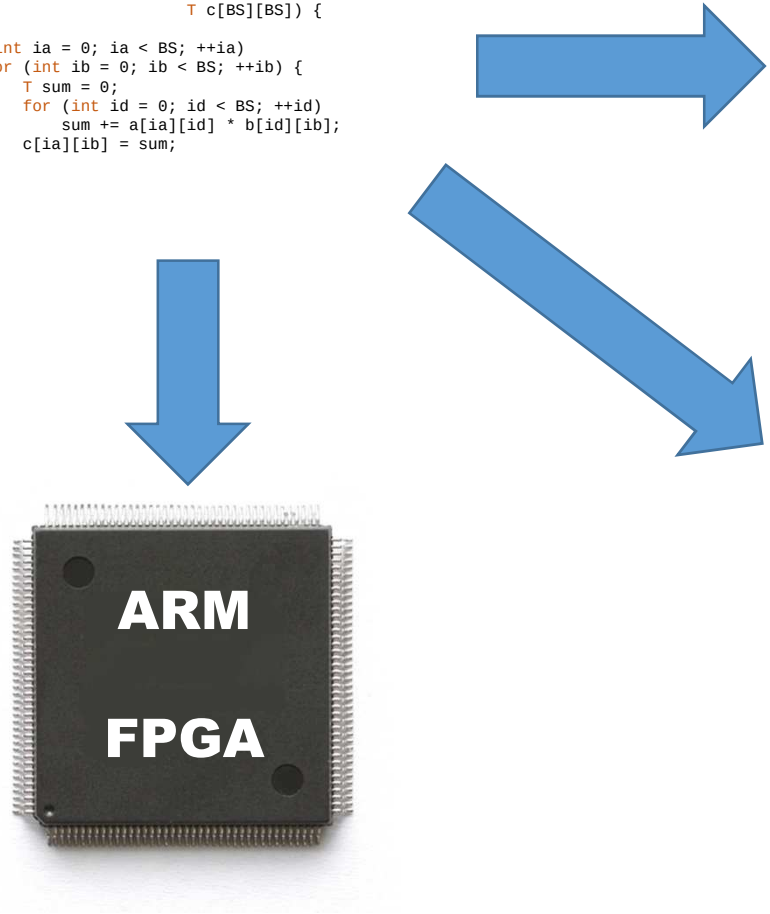
OmpSs Pragmas: Programmability

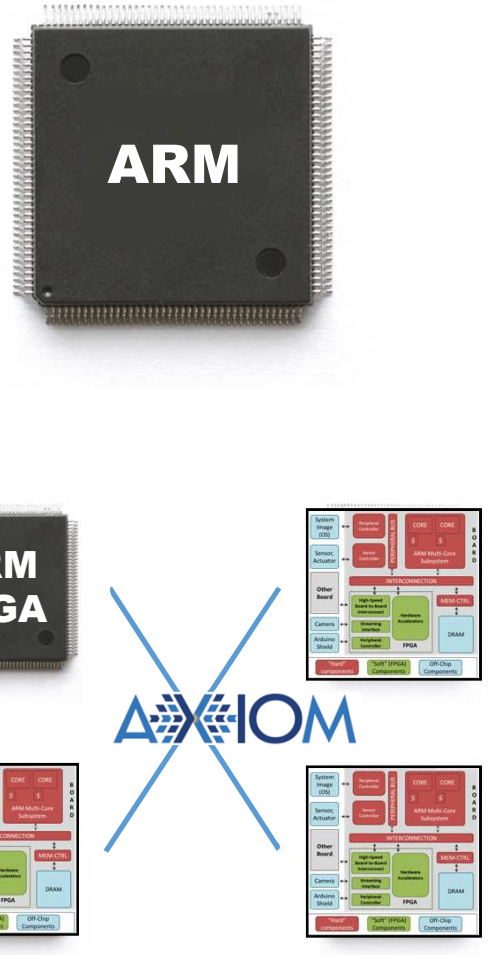
```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in(a[0:BS*BS-1], b[0:BS*BS-1]) \
                 inout(c[0:BS*BS-1])
void matrix_multiply(int BS, T a[BS][BS], T b[BS][BS],
                             T c[BS][BS]) {

    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
            T sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
        }

}
```

We can do it with only two OmpSs pragma directives!

How we want to execute

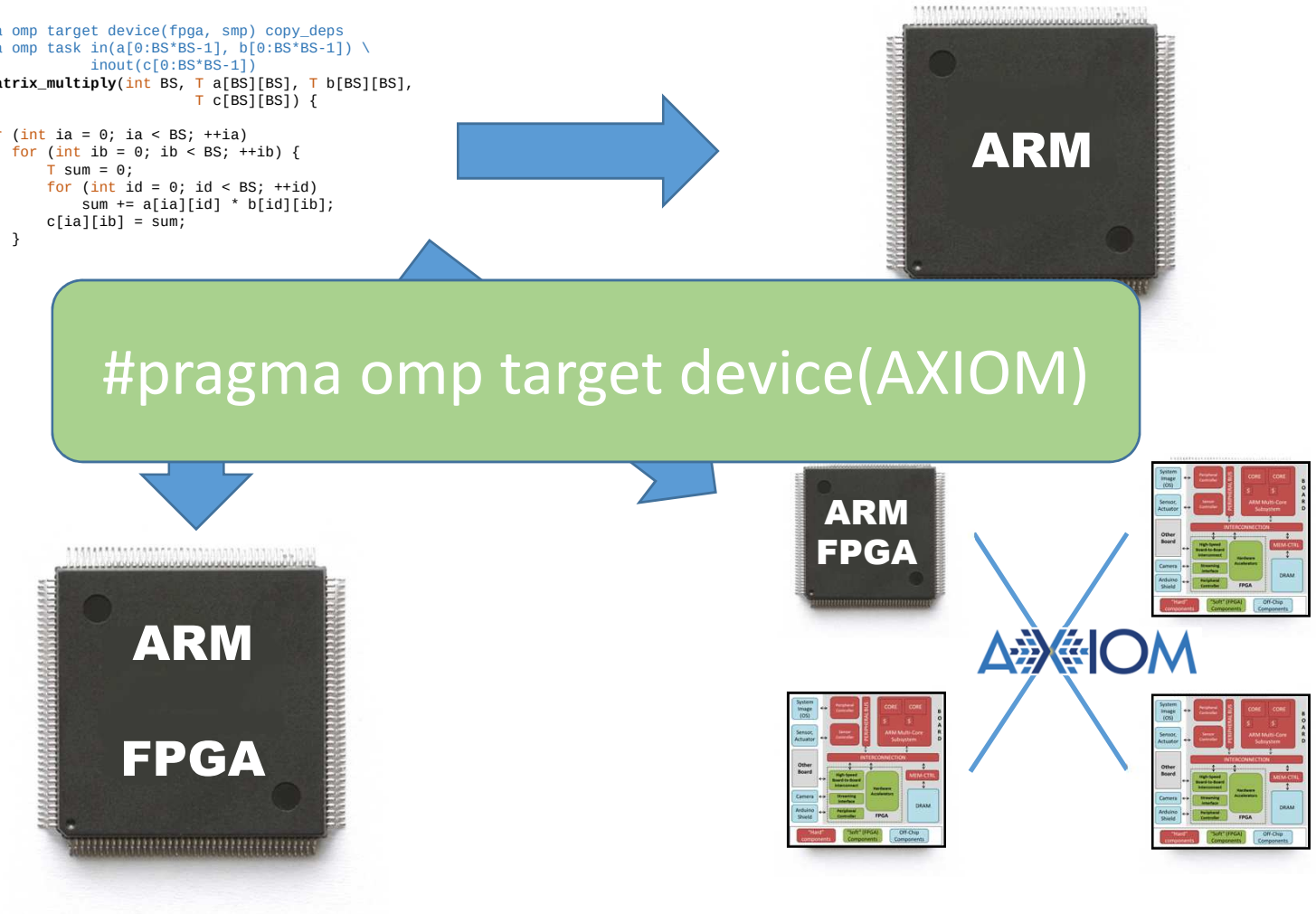
```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in(a[0:BS*BS-1], b[0:BS*BS-1]) \
                 inout(c[0:BS*BS-1])
void matrix_multiply(int BS, T a[BS][BS], T b[BS][BS],
                             T c[BS][BS]) {

    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
            T sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
        }

}
```





How we want to execute

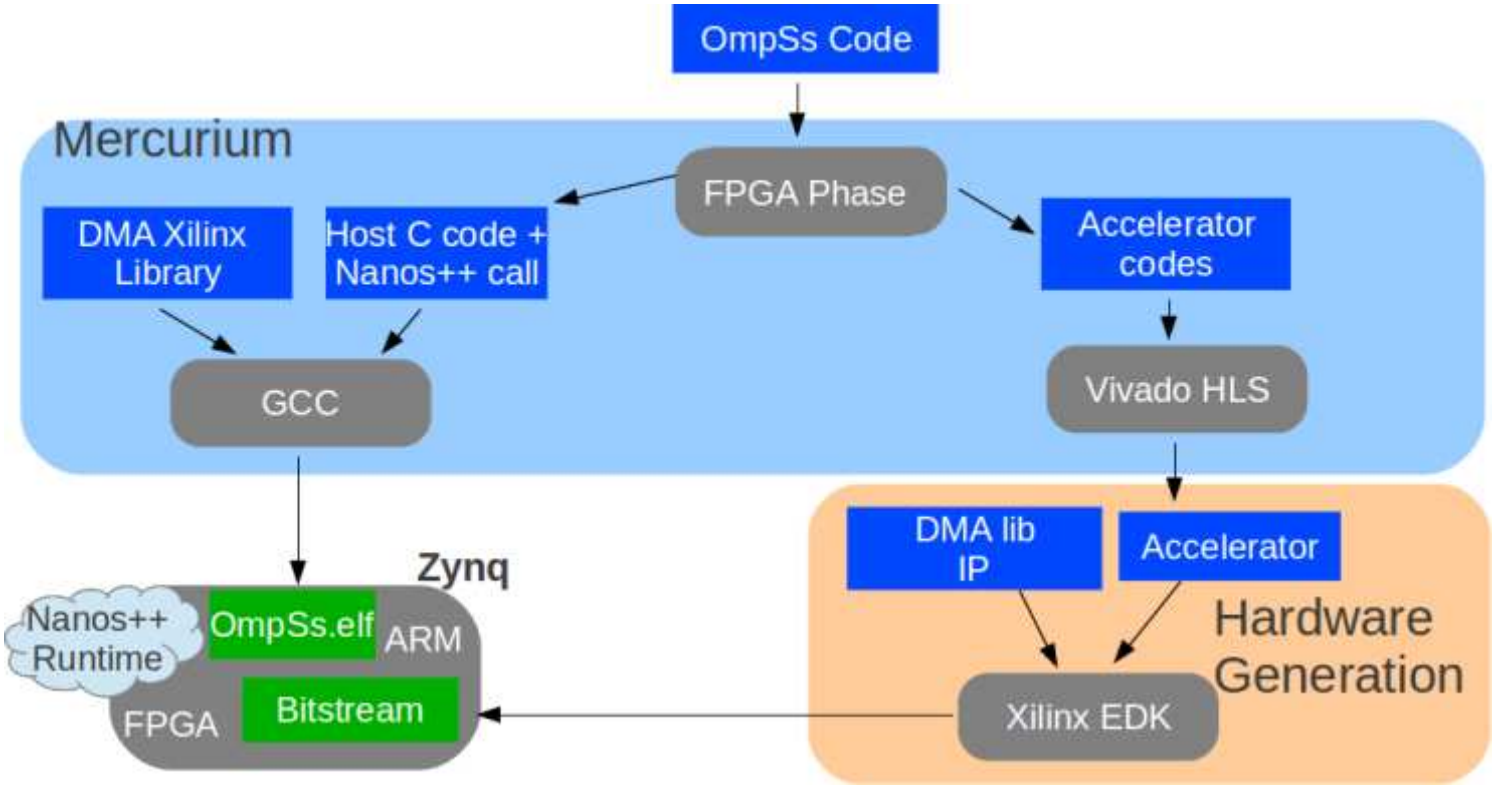
```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in(a[0:BS*BS-1], b[0:BS*BS-1]) \
                 inout(c[0:BS*BS-1])
void matrix_multiply(int BS, T a[BS][BS], T b[BS][BS],
                             T c[BS][BS]) {

    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
            T sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
        }

}
```



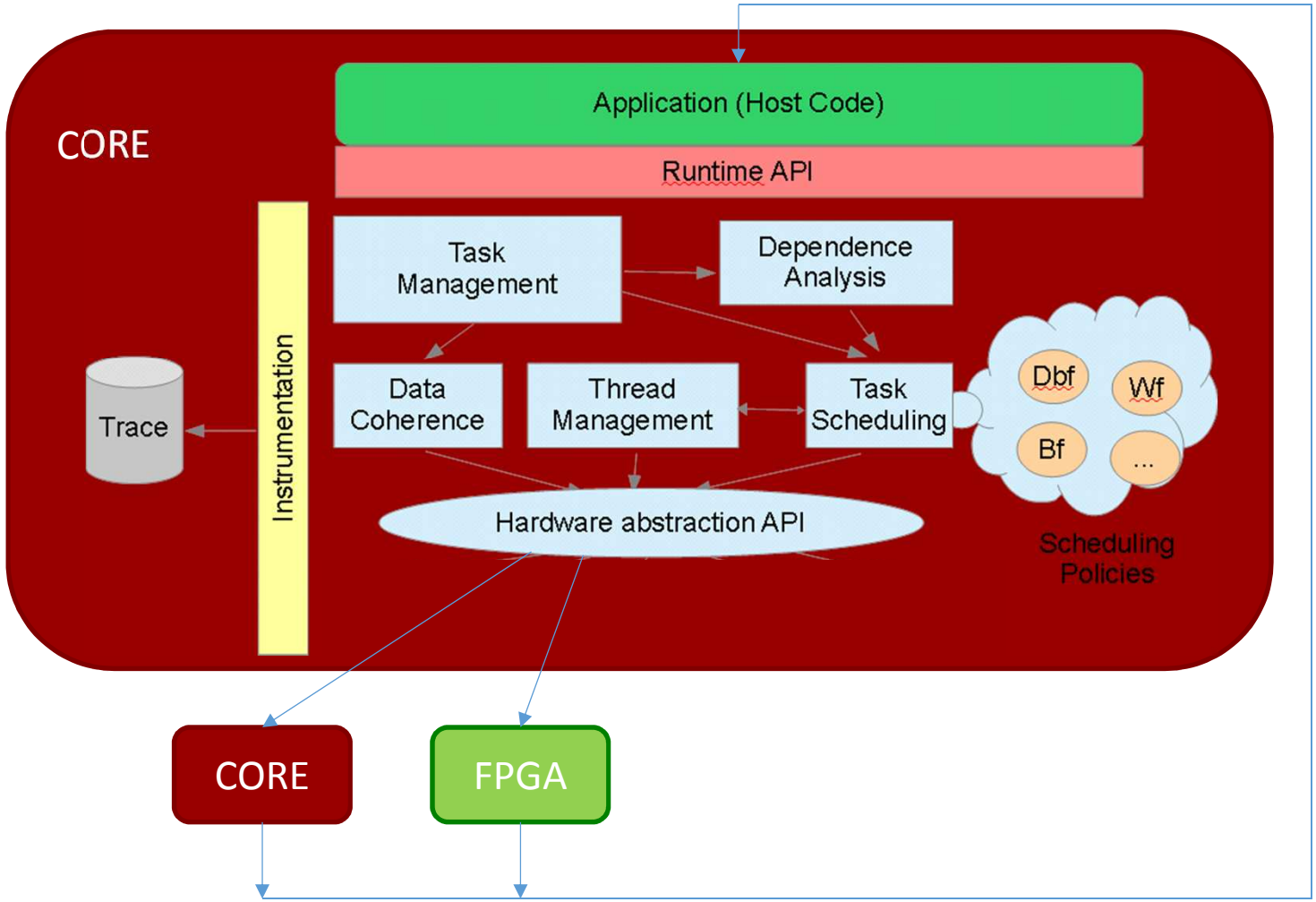
OmpSs@Zynq

- Mercurium source to source compiler
  - C code transformation, Nanos++ call insertions and bitstream generation automation
- Nanos++ runtime
  - Task dependency management and data movement (DMA lib management)
- Extrae library
  - Task execution instrumentation

OmpSs@Zynq infrastructure



OmpSs@Zynq runtime (Nanos++)

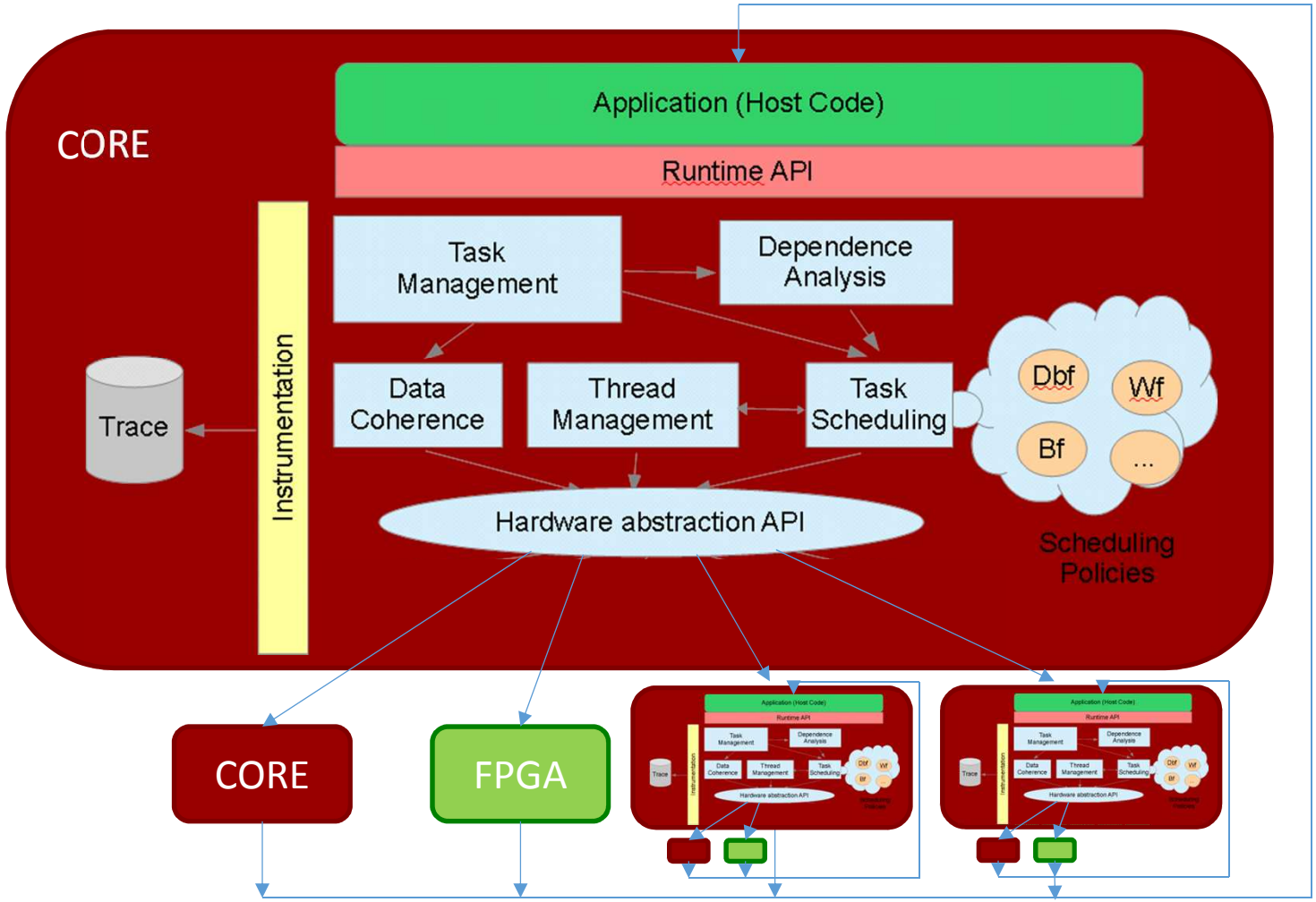


Going cluster

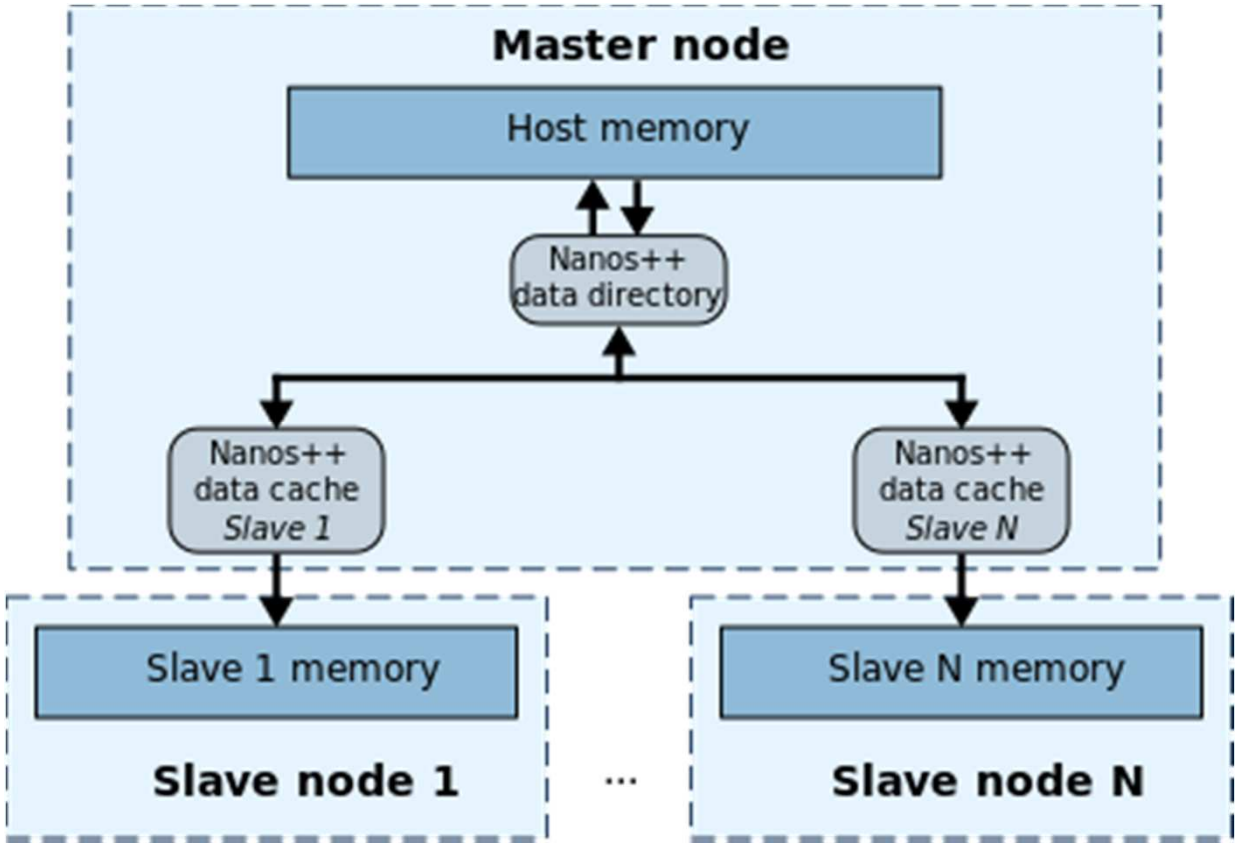
Two alternatives are being evaluated

- OmpSs@Cluster uses additional modules as heterogeneous accelerators
  - Nanos takes care of the data movements between modules
  - No global variables shared between the main module and the accelerator modules
- A DSM approach where the OS takes care of the data movement and behaves like a large multicore

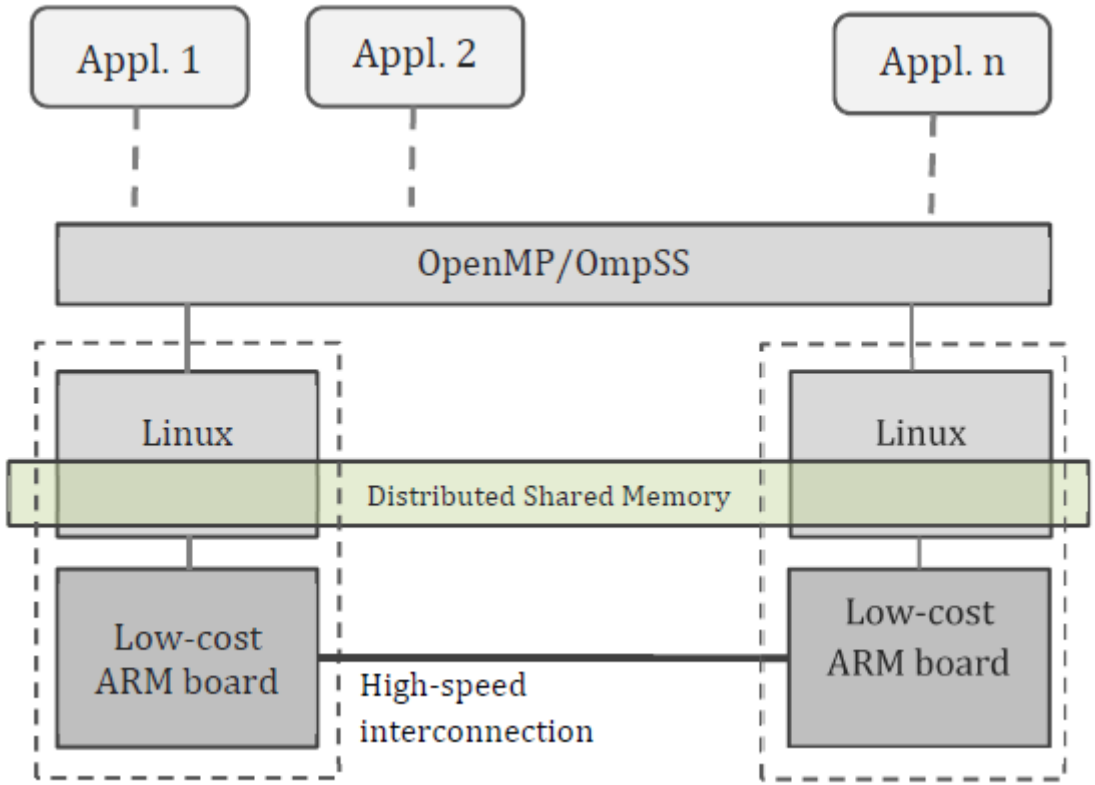
OmpSs@Cluster + OmpSs@Zynq



OmpSs@Cluster memory management



DSM + OmpSs@Zynq

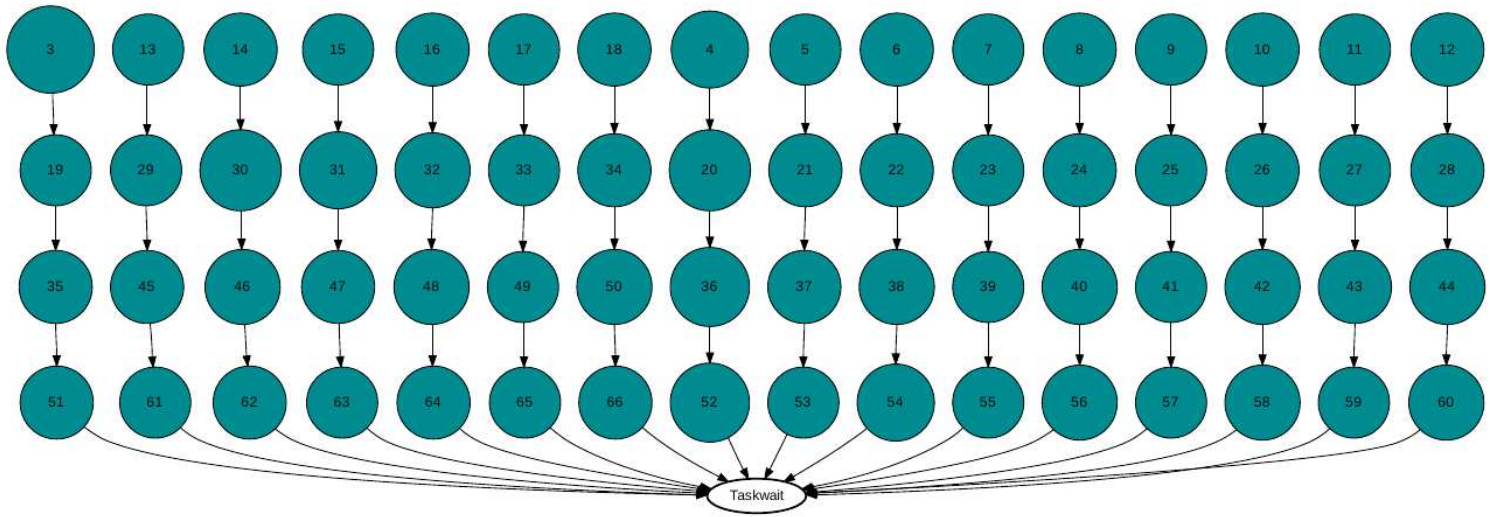


Compilation & Execution

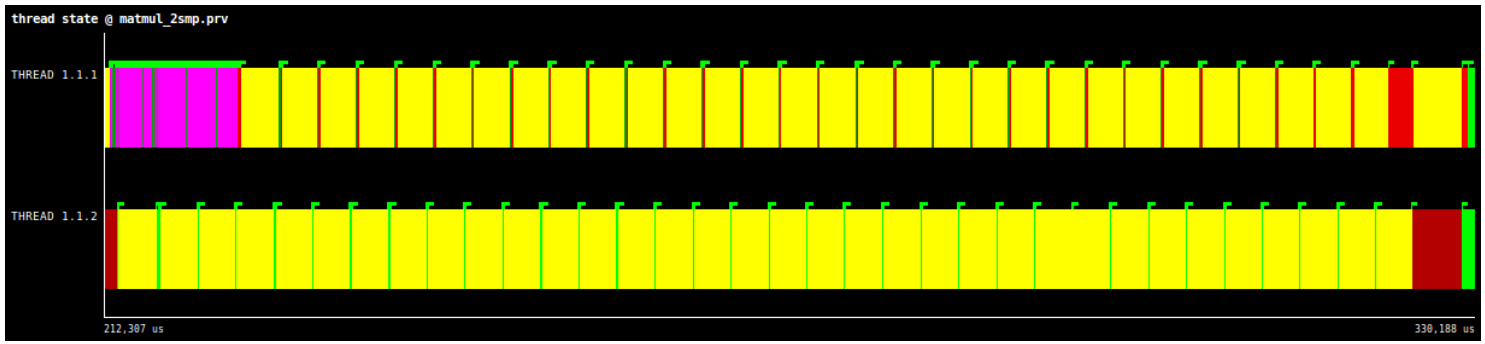
- Non instrumentation compilation
  - `$ fpgacc --ompss -o main.elf main.c`
- Instrumented compilation
  - `$ fpgacc --ompss --instrument -o main.elf main.c`
- Non instrumented execution
  - `$ ./main.elf`
- Instrumented execution
  - `$ NX_ARGS="--instrumentation=extrae" ./main.elf`

Performance Analysis Tools

Dynamic task graph



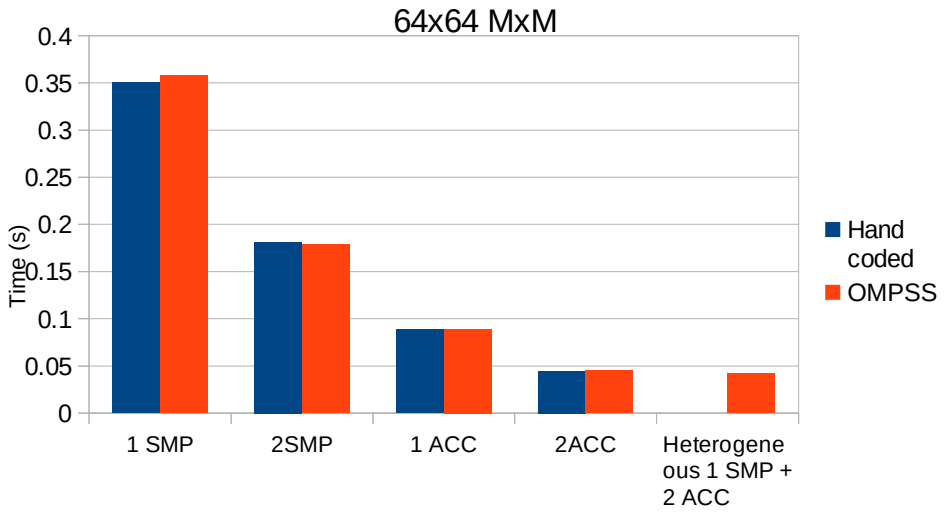
Analysis with Paraver

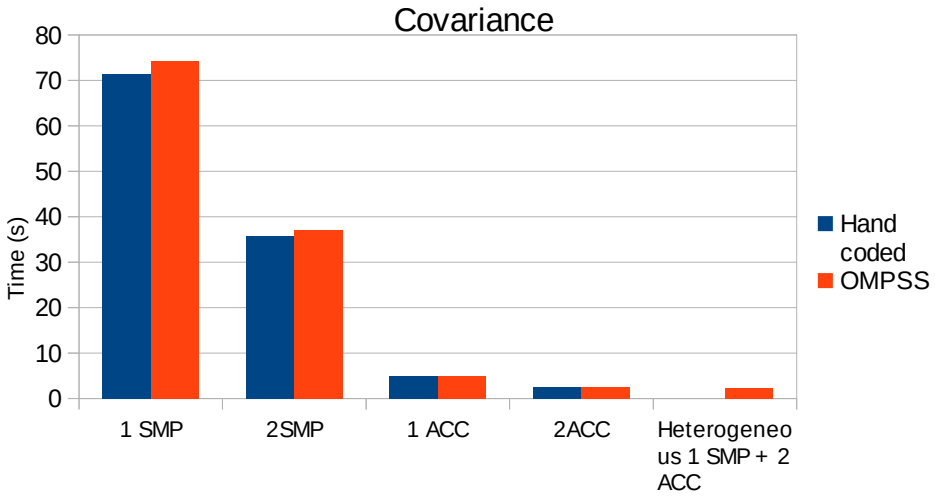


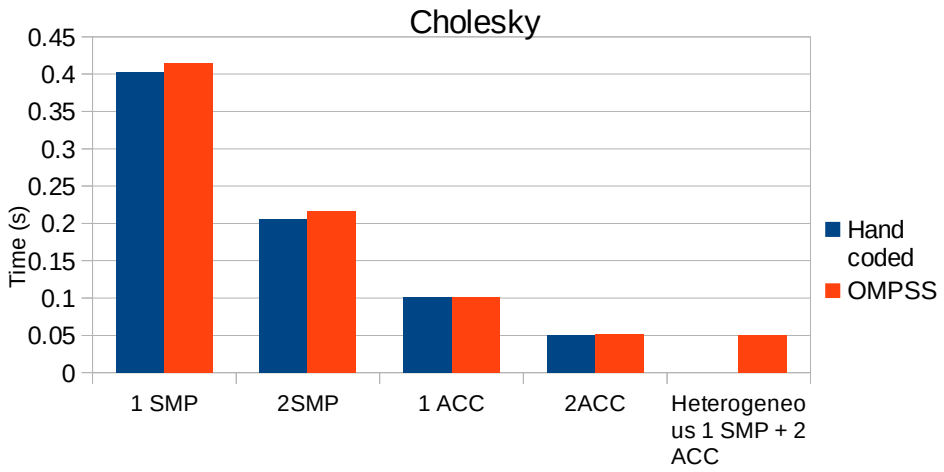
Results

Programming effort

Application	Seq w/ DMA	Pthread	OmpSs
Cholesky	71	26	3
Covariance	94	29	3
Matrix Multiply	95	39	3







Conclusions

- H2020 AXIOM to explore heterogeneous architectures for next-generation Cyber-Physical Systems
- Need for a programming model that relieves the programmer from the burden of dealing with micro-managing hardware details: OmpSs
- Two possible alternatives of managing inter-module task scheduling
- Our preliminary system to build a software framework for the AXIOM project shows promising programmability and performance results

AXIOM

Agile, eXtensible, fast I/O Module for the cyber-physical era

PROJECT ID: 645496

University of Siena
(Coordinator Partner)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EVIDENCE
EMBEDDING TECHNOLOGY

herta security

FORTH
Institute of Computer Science

SECO

VIMAR
energia positiva