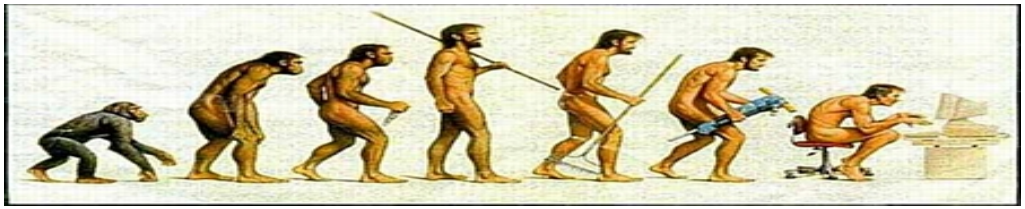
OmpSs@Zynq All-Programmable SoC Ecosystem

Antonio Filgueras, Eduard Gil, Daniel Jiménez-González*,
Carlos Álvarez, Xavier Martorell,
Barcelona Supercomputing Center-CNS – Universitat Politécnica de Catalunya
Jan Langer, Juanjo Noguera, Kees Vissers
Xilinx Research Labs
Speaker: Santhosh Kumar Rethinagiri
**Contact e-mail: djimenez at ac dot upc dot edu*
FPGA-2014
Feb 28th-2014

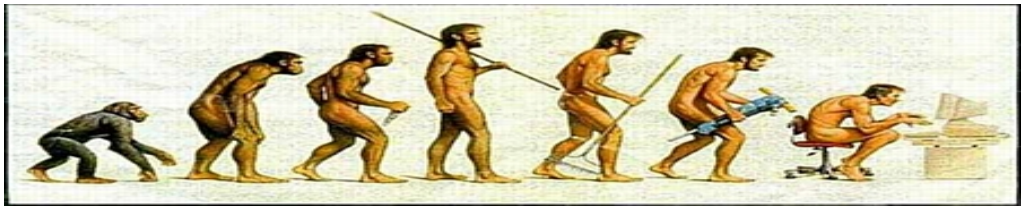
A programmer can be...



"Naive"
Programmer

My program is fine!!
The hardware has to accelerate my program!!

A programmer can be...

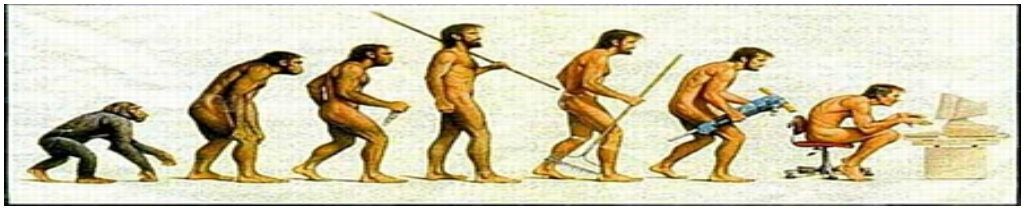


"Naive"
Programmer

Automatic
Acceleration

My program is fine!!
The hardware has to
accelerate my program!!

A programmer can be...



"Naive" Programmer

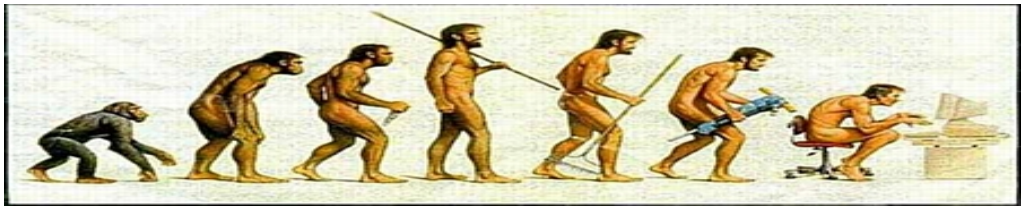
Parallel Programmer

Automatic Acceleration

My program is fine!!
The hardware has to accelerate my program!!

My program is fine?
My program should be **a parallel program!!**
Who is taking care of the parallel management?

A programmer can be...



"Naive" Programmer

Automatic Acceleration

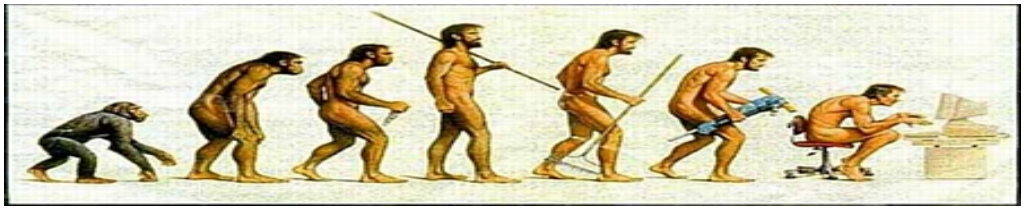
My program is fine!!
The hardware has to accelerate my program!!

Parallel Programmer

Parallel Programming Models

My program is fine?
My program should be **a parallel program!!**
Who is taking care of the parallel management?

A programmer can be...



"Naive" Programmer

Automatic Acceleration

My program is fine!!
The hardware has to accelerate my program!!

Non Expert FPGA and Parallel Programmer

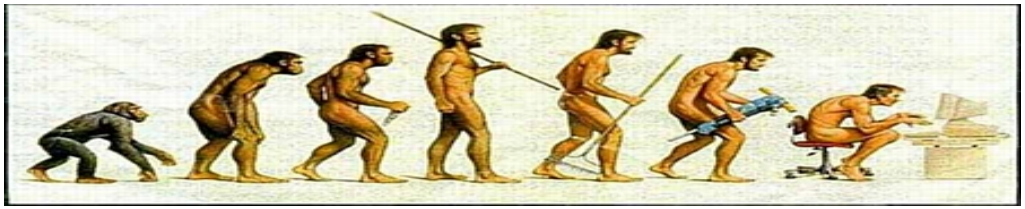
My **parallel** program is fine!!
I have **an FPGA**...
Who is programming the FPGA?
Heterogeneous Acceleration!!!

Parallel Programmer

Parallel Programming Models

My program is fine?
My program should be **a parallel program!!**
Who is taking care of the parallel management?

A programmer can be...



"Naive" Programmer

Automatic Acceleration

My program is fine!!
The hardware has to accelerate my program!!

Non Expert FPGA and Parallel Programmer

OmpSs@Zynq

My **parallel** program is fine!!
I have **an FPGA**...
Who is programming the FPGA?
Heterogeneous Acceleration!!!

Parallel Programmer

Parallel Programming Models

My program is fine?
My program should be **a parallel program!!**
Who is taking care of the parallel management?

Objectives: Easy Programming

- Easy Programming of Zynq platform
  - Using OmpSs parallel programming model
    - Influenced new OpenMP 4.0 w/
      - Fast and **AUTOMATIC** Task Dependency Management
      - **Heterogeneous PARALLEL** Execution (SMP + FPGA)
    - Using C with **ONLY two** simple OmpSs pragmas
      - #pragma omp task … and … target device
    - Avoiding programmability complexity and manual task dependency of other models like OpenCL
  - Automatic generation of host and FPGA binary
    - VIVADO HLS is transparently used

Objectives: Performance Analysis

- Execution Analysis
  - Execution traces that allow...
    - Analysis of the different levels of the runtime system and FPGA management libraries
    - Analysis of heterogeneous execution
      - Task level analysis
    - Performance analysis
      - Task performance analysis

Outline

- Objectives
- OmpSs Example
- OmpSs@Zynq Overview
- Results
- Conclusions

OmpSs@Zynq Example

Example: MxM original code

```
void matrix_multiply_hw(float a[32][32], float b[32][32], float c[32][32]){
    int const FACTOR = 32/2;
#pragma HLS inline
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
    // matrix multiplication of a A*B matrix
    for (int ia = 0; ia < 32; ++ia)
        for (int ib = 0; ib < 32; ++ib) {
#pragma HLS PIPELINE II=1
            float sum = 0;
            for (int id = 0; id < 32; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
}       }
```

Example: MxM OmpSs code

```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in([32*32]a, [32*32]b) out([32*32]c)
void matrix_multiply_hw(float a[32][32], float b[32][32], float c[32][32]){
   int const FACTOR = 32/2;
#pragma HLS inline
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
   // matrix multiplication of a A*B matrix
   for (int ia = 0; ia < 32; ++ia)
      for (int ib = 0; ib < 32; ++ib) {
#pragma HLS PIPELINE II=1
         float sum = 0;
         for (int id = 0; id < 32; ++id)
            sum += a[ia][id] * b[id][ib];
         c[ia][ib] = sum;
}     }
```

```
…
matrix_multiply_hw(A,B,OUT);
…
matrix_multiply_hw(B,OUT,C);
….
```

Example: MxM OmpSs code

```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in([32*32]a, [32*32]b) out([32*32]c)
void matrix_multiply_hw(float a[32][32], float b[32][32], float c[32][32]){
   int const FACTOR = 32/2;
#pragma HLS interface
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
   // matrix multiplication of a*b, results in c
   for (int ia = 0; ia < 32; ++ia)
      for (int ib = 0; ib < 32; ++ib) {
#pragma HLS PIPELINE II=1
         float sum = 0;
         for (int id = 0; id < 32; ++id)
            sum += a[ia][id] * b[id][ib];
         c[ia][ib] = sum;
}     }
```

- **Now the kernel can be transparently executed in one SMP...**
- **...or in two SMPs**
- **...or in one FPGA accelerator**
- **...or in two FPGA accelerators**

```
...
matrix_multiply_hw(A,B,OUT);
...
matrix_multiply_hw(B,OUT,C);
....
```

Example: MxM OmpSs code

```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in([32*32]a, [32*32]b) out([32*32]c)
void matrix_multiply_hw(float a[32][32], float b[32][32], float c[32][32]){
  int const FACTOR = 32/2;
#pragma HLS interface
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
  // matrix multiplication of a*b, results in c
  for (int ia = 0; ia < 32; ++ia)
    for (int ib = 0; ib < 32; ++ib) {
#pragma HLS PIPELINE II=1
      float sum = 0;
      for (int id = 0; id < 32; ++id)
        sum += a[ia][id] * b[id][ib];
      c[ia][ib] = sum;
}     }
```

- **Now the kernel can be transparently executed in one SMP...**
- **...or in two SMPs**
- **...or in one FPGA accelerator**
- **...or in two FPGA accelerators**

or in ANY # and type of resources

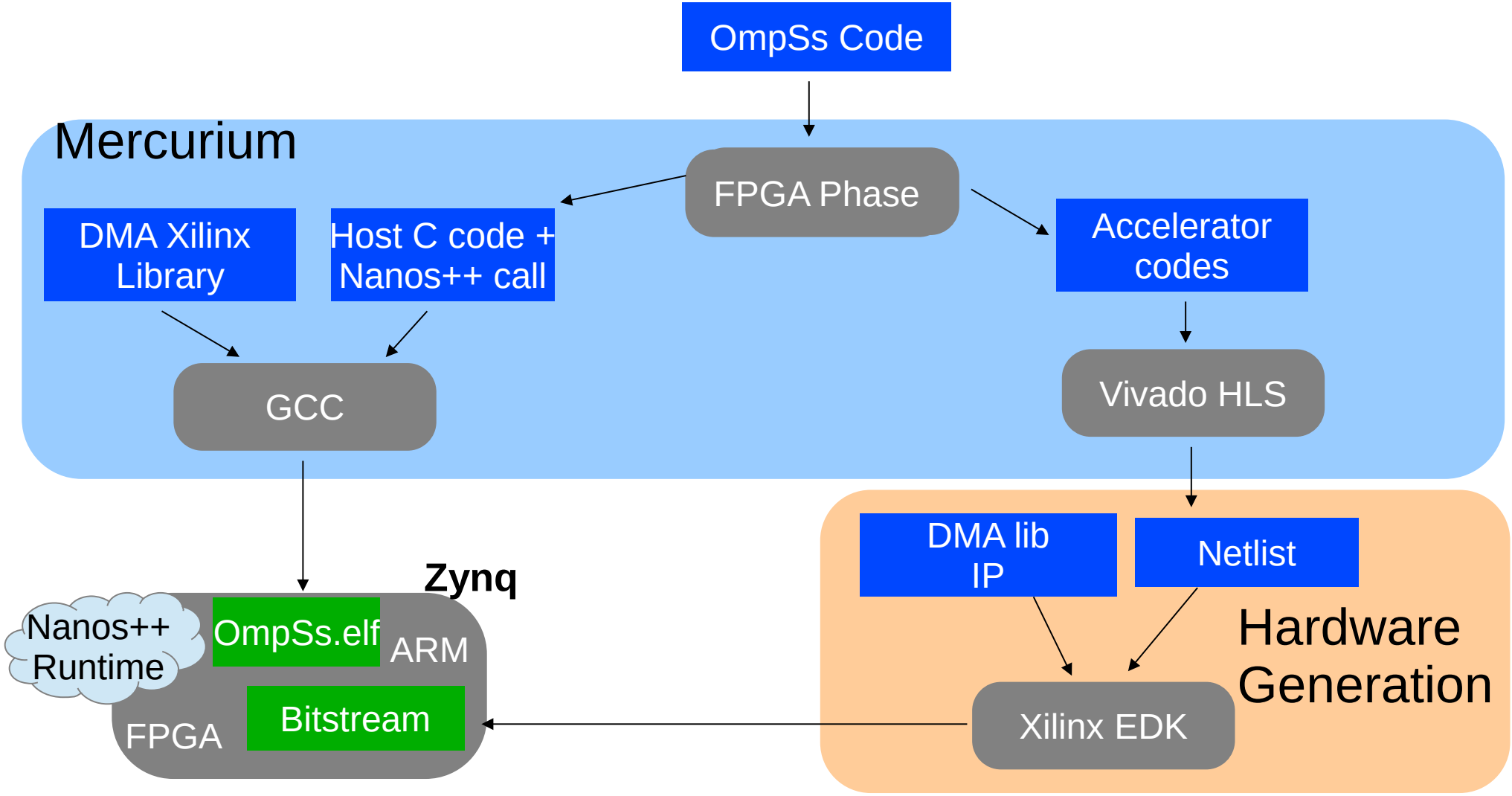
```
...
matrix_multiply_hw(A,B,OUT);
...
matrix_multiply_hw(B,OUT,C);
....
```

Outline

- Objectives
- OmpSs Example
- OmpSs@Zynq Overview
- Results
- Conclusions

OmpSs@Zynq Overview

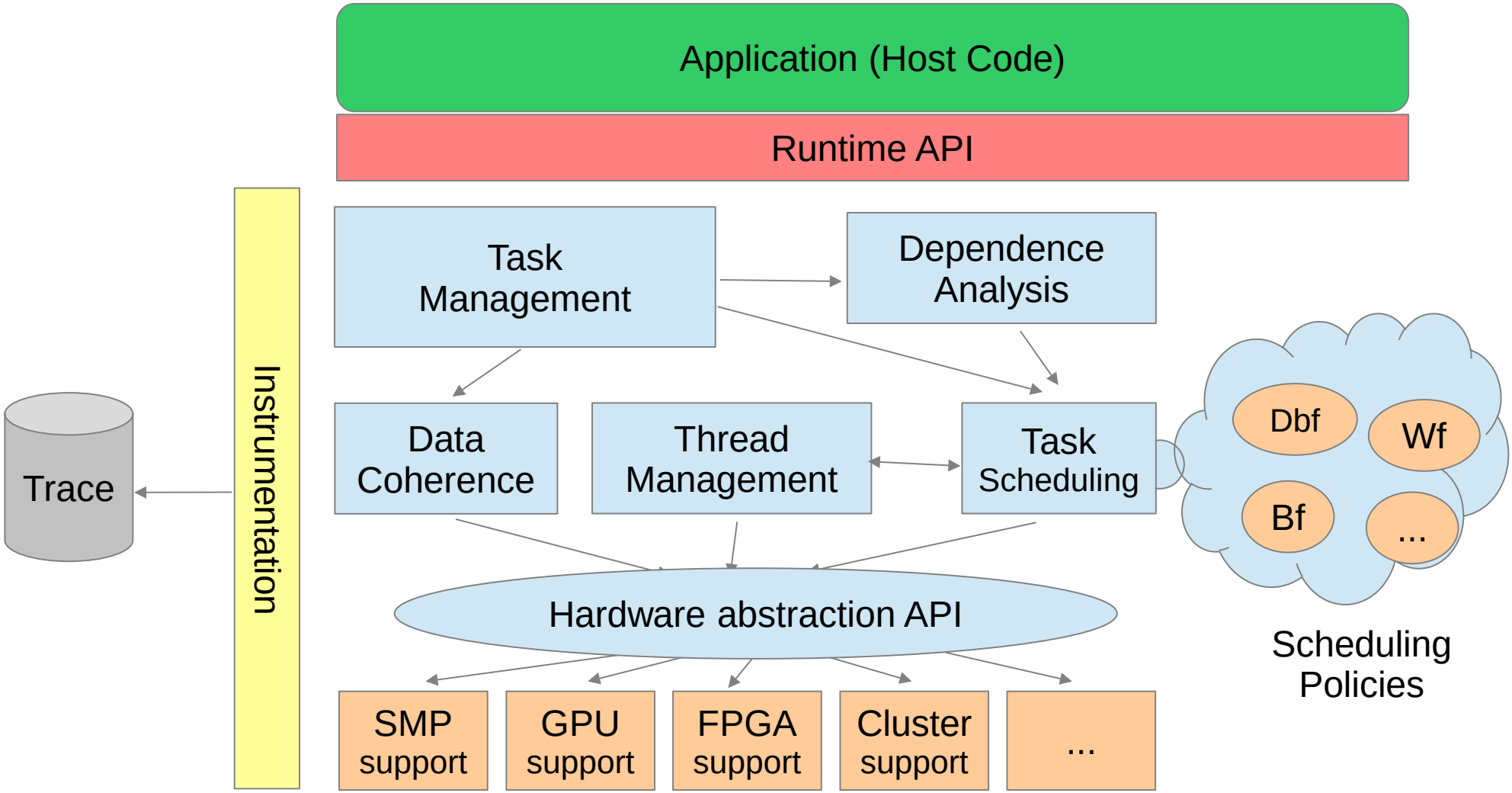
OmpSs@Zynq Ecosystem



OmpSs@Zynq Features Overview

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

Nanos++ Runtime Overview



Compilation Output

- Host binary (ARM binary)
  - Setup (system configuration)
  - Communication A9's and Programmable Logic
- FPGA bitstream implementing the task
  - Accelerators created using High Level Synthesis from C/C++ code
  - EDK design generated from system templates

How to use this tool?

- Non instrumentation
  - Compilation: *$ fpgacc --ompss -o main.elf main.c*
  - Execution: *$ ./main.elf*

- Instrumentation
  - Compilation: *$ fpgacc --ompss --instrument -o main.elf main.c*
  - Execution: *$ NX_ARGS="--instrumentation=extrae" ./main.elf*

Outline

- Objectives
- OmpSs Example
- OmpSs@Zynq Overview
- Results
- Conclusions

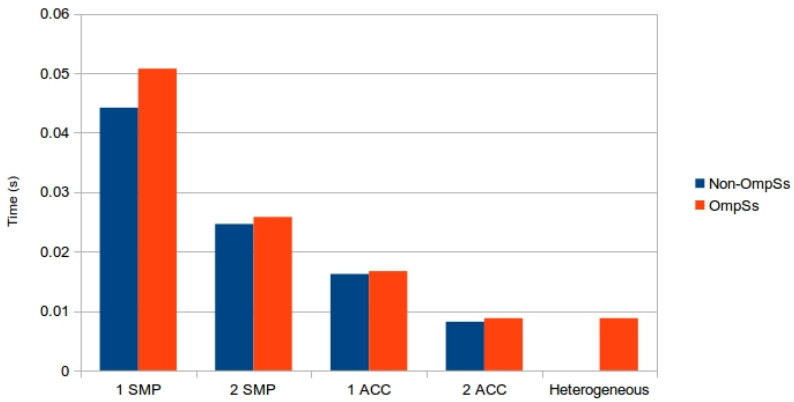
OmpSs@Zynq Results

Experimental Setup

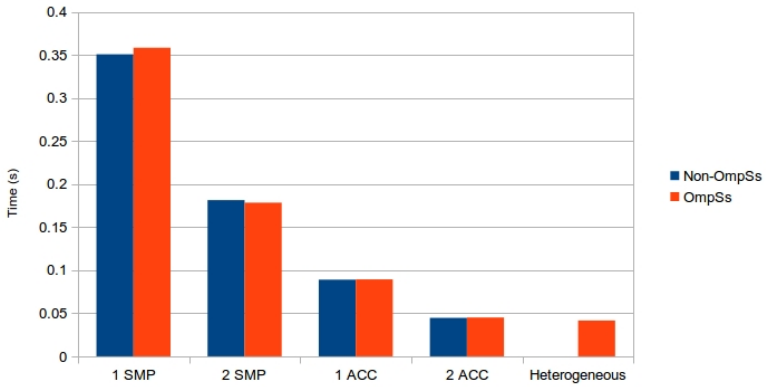
- Zynq All-Programmable SoC 702 Board
- Mercurium 1.99.0
- Nanos nanox 0.7a
- Extrae Instrumentation Library 2.3.4
- Paraver Tool
- Proof of concept-Applications:
  - 32x32 MxM, 64x64 MxM, 64x64 Cholesky, Covariance

Execution Time

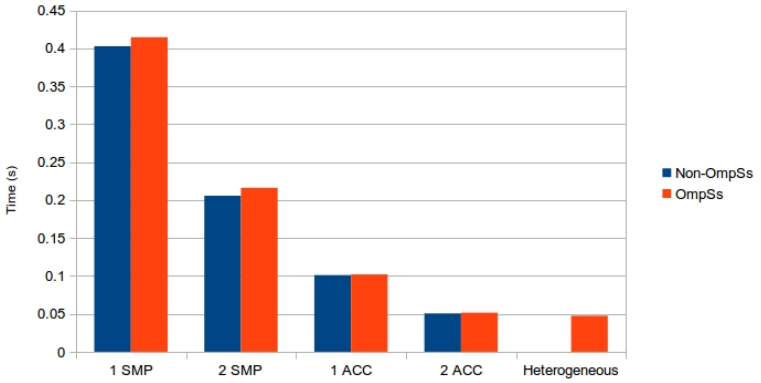
32x32 MxM



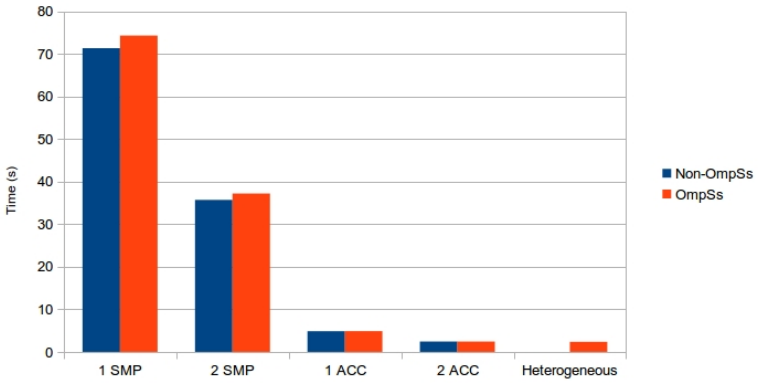
64x64 MxM



64x64 Cholesky



Covariance



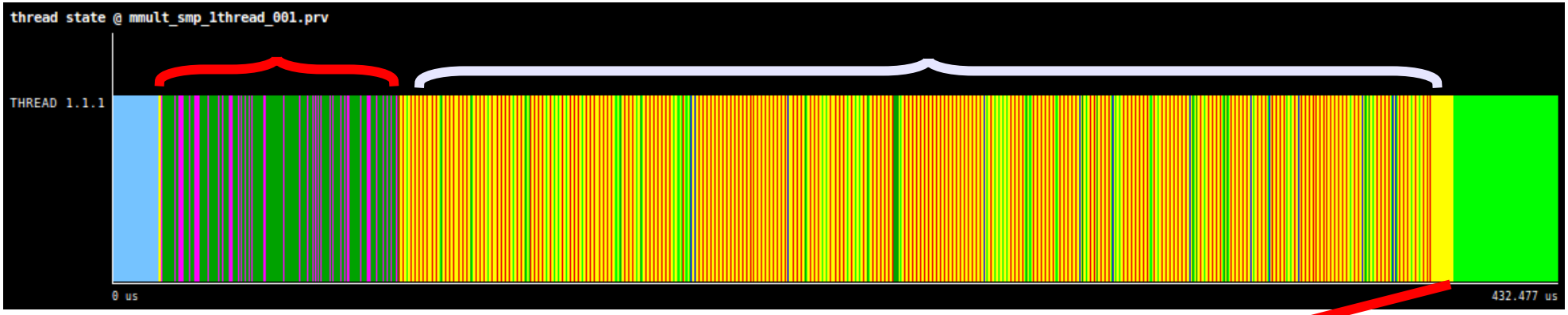
- Non-OmpSs (Hand Coded) versions: SMP: pthreads, ACC: sequential + DMA management
- OmpSs version: annotated sequential version

Objectives: Performance Analysis

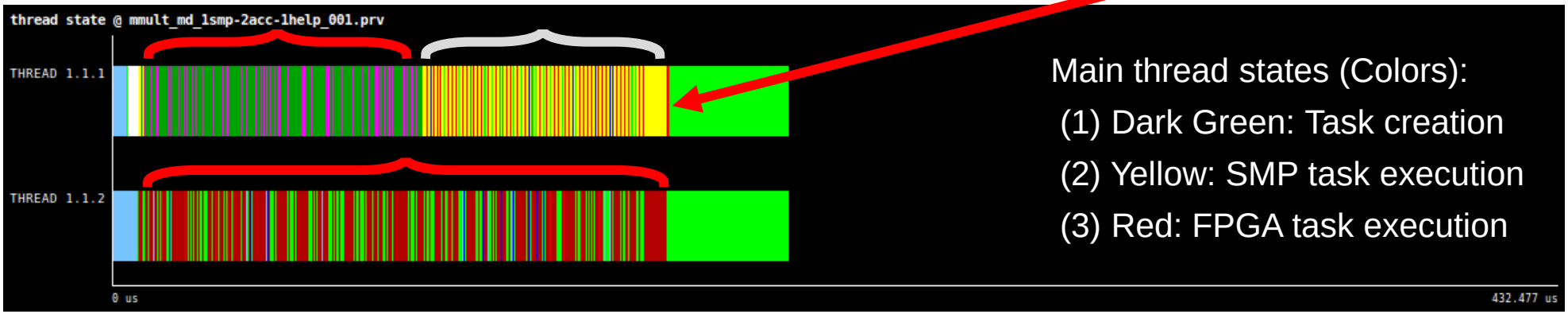
- Execution Traces Analysis that allows:
  - Analysis of the different levels of the runtime system and FPGA management libraries
  - Analysis of heterogeneous execution
    - Task level analysis
  - Performance analysis
    - Task performance analysis
- Paraver: Execution Trace visualization tool
  - Software tool for multi-core performance analysis

Paraver Trace on Zynq
MxM: 1 SMP vs 1 SMP + 2 acc

- Quick Phase Identification and Performance Comparison



Execution Trace: 1 SMP (A9)



Main thread states (Colors):

(1) Dark Green: Task creation

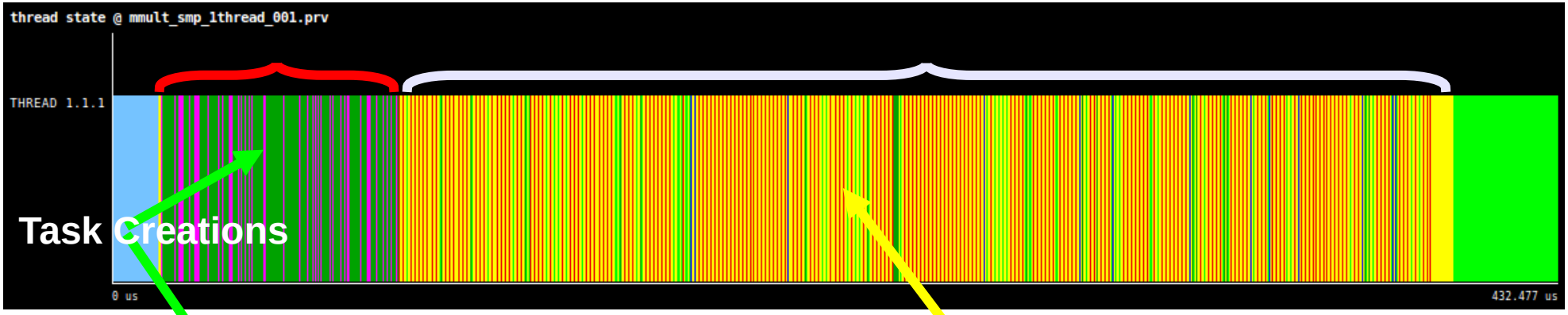
(2) Yellow: SMP task execution

(3) Red: FPGA task execution

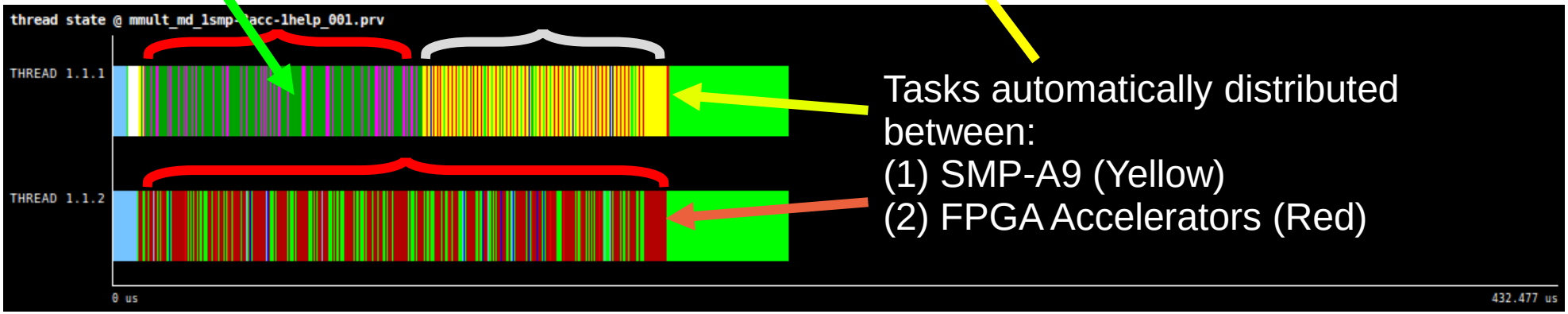
Execution Trace: 1 SMP (A9) + 2 Accelerators

Paraver Trace on Zynq
MxM: 1 SMP vs 1 SMP + 2 acc

- Heterogeneous Execution performance analysis



Execution Trace: 1 SMP (A9)



Tasks automatically distributed between:
(1) SMP-A9 (Yellow)
(2) FPGA Accelerators (Red)

Execution Trace: 1 SMP (A9) + 2 Accelerators

Outline

- Objectives
- OmpSs Example
- OmpSs@Zynq Overview
- Results
- Conclusions

Conclusions

- With a ***simple*** programming model...
- From the source to the executable in just ***"one" pragma***...
- We use ***all the resources*** of the Zynq automatically and transparently to the programmer...

  … achieving very good speedups
- We are as ***fast*** as the non-OmpSs specific code

OmpSs@Zynq All-Programmable SoC Ecosystem

Antonio Filgueras, Eduard Gil, Daniel Jiménez-González*,
Carlos Álvarez, Xavier Martorell,
Barcelona Supercomputing Center-CNS – Universitat Politécnica de Catalunya
Jan Langer, Juanjo Noguera, Kees Vissers
Xilinx Research Labs
Speaker: Santhosh Kumar Rethinagiri
**Contact e-mail: djimenez at ac dot upc dot edu*
FPGA-2014
Feb 28th-2014