Exotic OpenMP Use Cases

Eduardo Quiñones, BSC
Heterogeneous and Parallel Computing

Massively parallel systems that operate **as fast as possible**

- Genomics
- Weather
- Big data

**Heterogeneous and Parallel computing becomes key to cope with performance requirements**

**HPC Domain (~300W)**
- NVIDIA A100 (GPU-based)
- AMD Instinct™ MI (GPU-based)
- Intel® Xeon® Series (40-core)
- AMD EPYC™ Series (up to 64-core)

**Embedded Domain (~10-20W)**
- NVIDIA Jetson Family (GPU-based)
- Kalray MPPA Coolidge (80-core fabric)
- Xilinx Versal (FPGA-based with DFX)

Network of HW/SW components that **must** operate **correctly** in response to its inputs from both **functional** and **non-functional** perspectives
Heterogeneous and Parallel Computing in Embedded Systems

- **Performance**: complex computations at high speed
- Real-time: end-to-end response time within budget
- Power/Thermal: energy/temperature within budget
- Safety: guarantee correctness and integrity of operation
- Security: prevent external elements from affecting correctness and integrity
Heterogeneous and Parallel Computing

Host-centric paradigm: The parallel computation is orchestrated by the general-purpose multi-core

- Accelerator for optimized graphical processing, linear algebra and deep learning
- General purpose multi-core for control-flow applications and parallel orchestration
- Optimised HW functions
- Network on Chip (NoC)

- GPU (Accelerator)
- FPGA (Accelerator)
- Multi-core (Host)
- Memory
- Interface with the (physical) world
- Addressable memory by the different computing elements
- Reconfigurable logic including dynamic partial reconfiguration
- Optimised HW functions
  - Accelerator for optimized graphical processing, linear algebra and deep learning
- General purpose multi-core for control-flow applications and parallel orchestration
- Optimised HW functions
- Network on Chip (NoC)

Peripherals
The SW Productivity Gap

1. **Efficiently exploit parallelism** and achieve the required performance
2. **Reason** about the functional and non-functional correctness

Source: ITRS & Hardware-dependent Software, Ecker et al., Springer
Addressing Complexity on System Development

Model Driven Engineering (MDE) in Embedded Systems

1. Construction of complex systems
2. **Formal verification** of functional and non-functional requirements (NFR) with **composability** features
   - Suitable Correct-by-construction paradigm by means of code generation
3. Only for single-core execution or with **very limited parallel support**

Gap between the MDE used for CPS and the PPM supported by parallel platforms

Parallel Programming Models (PPM) in HPC

1. Mandatory for **SW productivity** in terms of
   - Programmability: Parallel abstraction while hiding HW complexities
   - Portability: Compatibility multiple HW platforms
   - Performance: Exploiting parallel capabilities of underlying HW
2. **Efficient offloading** to HW acceleration devices for an energy-efficient parallel execution
AMPERE’s Vision

1. **Synthesis methods** for an efficient generation of parallel source code, while keeping NFR and composability guarantees

2. **Run-time parallel frameworks** that guarantee system correctness and exploit the performance capabilities of parallel architectures

3. **Integration** of parallel frameworks into MDE frameworks
AMPERE’s Main Contribution

A novel software architecture capable of

1. Capturing the component definition and NFR for the system model and transform it to parallel constructs
2. Fulfillment of NFR described in the CPS description
   - Real-time response, energy-efficiency, resiliency and safety and cyber-security
3. Efficient usage of advance parallel and heterogeneous embedded architectures

Productivity

- Programmability
- Portability/Scalability
- (Guaranteed) Performance
AMPERE’s Workflow Overview

1. System description (Domain Specific Modelling Language)
   - Meta Model Driven Abstraction
   - Model
   - Platform description

2. Synthesis Tool & Compiler
   - High-level Parallel Programming Model
   - Meta PPM Model
   - Multi-criterion Optimization
   - Parallel code (e.g., OpenMP)
   - Resource Allocation (i.e., mapping)
   - Meta-parallel programming model
   - Performance
   - Real-time
   - Energy
   - Resilience

3. Monitoring + dynamic resource allocation
   - Low-level Threading Library
   - Runtime + OS/Hypervisor
   - Monitoring

Domain-specific modeling languages: CAPELLA (AUTOSAR compatible), AMALTHEA

Synthesis tools/compilers: APP4MC, LLVM, NVCC, Vivado

Multi-criteria optimization

Run-time hypervisor: Linux, Erika, ROS2, MicroROS

Monitoring: PikeOS, ROS2 MicroROS, FRED

Analyzer, Optimizer, Profiler

Programming model:
- High-level parallel programming model
- Meta-parallel programming model

Threading library:
- CUDA
- KMP
- Xilinx Zynq Ultrascale+ ZCU102
- NVIDIA Jetson AGX Xavier

Operating systems:
- Linux
- Erika

Monitoring: PikeOS, ROS2 MicroROS, FRED

Analyzer, Optimizer, Profiler
AMPERE’s Workflow Overview

DSML addressing different phases of the V-Model

1. System description (Domain Specific Modelling Language)
   - Meta Model Driven Abstraction
   - Model
   - Platform description

2. Synthesis Tool & Compiler
   - High-level Parallel Programming Model
   - Meta PPM abstraction
   - Multi-criterion Optimization
   - Parallel code (e.g., OpenMP)
   - Resource Allocation (i.e., mapping)

3. Monitoring + dynamic resource allocation
   - Low-level Threading Library
   - Runtime + OS/Hypervisor
   - Hardware Abstraction Layer

4. Monitoring
   - PikeOS
   - ROS2 MicroROS
   - FRED
   - Analyzer
   - Optimizer
AMPERE’s Workflow Overview

**Synthesis Tool & Compiler**

- **High-level Parallel Programming Model**
- **Meta PPM abstraction**
- **Parallel code (e.g., OpenMP)**
- **Resource Allocation (i.e., mapping)**
- **Multi-criterion Optimization**

**System description**
- (Domain Specific Modelling Language)

**Model**
- **Meta Model Driven Abstraction**

**Platform description**

**Domain-specific modeling languages**
- (AUTOSAR compatible) AMALTHEA
- CAPELLA

**Meta-model-driven abstraction**

**Multi-criteria optimization**
- Performance
- Real-time
- Energy
- Resilience

**Monitoring + dynamic resource allocation**

**Low-level Threading Library**

**Runtime + OS/Hypervisor**

**Hardware Abstraction Layer**

**Synthesis tools/compilers**
- APPAMC SLG
- LLVM
- NVCC
- Vivado

**Profiler**
- Analyzer
- Optimizer

**Operating system**
- Linux
- ERIKA
- ROS2
- MicroROS

**Hypervisor**
- NVIDIA Jetson AGX Xavier
- Xilinx Zynq Ultrascale+ ZCU102

**Parallel hardware**

**Monitoring**

**Time**

**Energy**

**Resilience**
AMPERE’s Workflow Overview

Runtime Monitoring and Implementation of
- Time
- Energy
- Resilience
- Safety and Security

System description (Domain Specific Modelling Language)

Model

Meta Model Driven Abstraction

Platform description

Synthesis Tool & Compiler

High-level Parallel Programming Model

Meta PPM abstraction

Multi-criterion Optimization

Parallel code (e.g., OpenMP)

Resource Allocation (i.e., mapping)

Meta-model-driven abstraction

Monitoring + dynamic resource allocation

Low-level Threading Library

Runtime + OS/Hypervisor

Hardware Abstraction Layer

Domain-specific modeling languages

(AUTOSAR compatible) AMALTHEA

CAPELLA

Synthesis tools/compilers

APPAMC SLG

LLVM

NVCC

Visado

Meta-parallel programming model

Profiler

Analyzer

Optimizer

Multi-criteria optimization

Performance
- Real-time
- Energy
- Resilience

High-level parallel programming model

Monitoring

Monitoring Profiling

PikeOS

ROS2 MicroROS

FRED

Analyzer

Optimizer

Operating system

CUDA

KMP

Linux

ERIKA

ROS2

MicroROS

Hypervisor

ROS2

PikeOS

Parallel hardware

NVIDIA Jetson AGX Xavier

Xilinx Zynq Ultrascale+ ZCU102

Monitoring + dynamic resource allocation

Low-level Threading Library

Runtime + OS/Hypervisor

Hardware Abstraction Layer

Monitoring

Monitoring Profiling

PikeOS

ROS2 MicroROS

FRED

Analyzer

Optimizer

Operating system

CUDA

KMP

Linux

ERIKA

ROS2

MicroROS

Hypervisor

ROS2

PikeOS

Parallel hardware

NVIDIA Jetson AGX Xavier

Xilinx Zynq Ultrascale+ ZCU102
AMPERE Use-cases

Obstacle Detection and Avoidance System (ODAS)
- ADAS functionalities based on data fusion coming from tram vehicle sensors

Predictive Cruise Control (PCC)
- Extends Adaptive Cruise Control (ACC) functionality by calculating the vehicle’s future velocity curve using the data from the *electronic horizon*
- Improve fuel efficiency (in cooperation with the powertrain control) by configuring the driving strategy based on data analytics and AI
The AMPERE Software Architecture: A Modular Design

- AMPERE aims to support different “instances” of the SW architecture
  - Increase exploitation opportunities

**SW Architecture applied to PCC**

- Domain-specific modeling languages: (AUTOSAR compatible) AMALTHEA, CAPELLA
- Synthesis tools/compilers: APP4MC SLG, LLVM, NVCC, Vivado
- Runtime: CUDA, Vivado
- Operating system: Linux
- Hypervisor: ROS2, MicroROS
- Parallel hardware: NVIDIA Jetson AGX Xavier, Xilinx Zynq Ultrascale+ ZCU102

**SW Architecture applied to ODAS**

- Domain-specific modeling languages: (AUTOSAR compatible) AMALTHEA, CAPELLA
- Synthesis tools/compilers: APP4MC SLG, LLVM, NVCC, Vivado
- Runtime: CUDA, Vivado
- Operating system: Linux
- Hypervisor: ROS2, MicroROS
- Parallel hardware: NVIDIA Jetson AGX Xavier, Xilinx Zynq Ultrascale+ ZCU102
Thank you

eduardo.quinones@bsc.es

www.ampere-euproject.eu

The AMPERE project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871669
OpenMP®

SC23 Booth Talk Series

openmp.org

OpenMP API specs, forum, reference guides,
and more

link.openmp.org/sc23

OpenMP SC23 booth talk videos
and presentations