Model-based engineering of high-performance embedded applications on heterogeneous hardware with real-time constraints and energy efficiency

Tommaso Cucinotta – Scuola Superiore Sant’Anna, Pisa (Italy)
Introduction & Motivations

- CPSs have higher and higher computational performance & reliability requirements

- Use of increasingly heterogeneous & interconnected, battery-operated platforms
  - non-SMP multi-core
  - GP-GPU/TPU acceleration
  - FPGA

- Heterogeneous platforms needed in soft and hard real-time use-cases
  - automotive, railways, aerospace, robotics, gaming, multimedia, ...
Problems & Challenges

- Development of software for CPSs is cumbersome!
  - Optimum usage of underlying hardware parallelism & acceleration
  - Performance vs energy consumption trade-offs
  - Real-time constraints
  - Safety & certification
Model-Driven Engineering (MDE)
- Fill the gap between high-level specifications and actual system behavior

MDE embraces
- Formal specification language(s)
- Model transformation engine(s)
- Model refinements & composability
- Automatic code generator(s)
- Model verifiability

=> Correctness-by-construction
MDE & Formalisms in Embedded System Design

- Model-Driven Engineering (MDE)
  - Fill the gap between high-level specifications and actual system behavior
- MDE embraces
  - Formal specification language(s)
  - Model transformation engine(s)
  - Model refinements & composability
  - Automatic code generator(s)
  - Model verifiability

Traditional MDE limitations
- Single-processor systems or very limited support for multi-core systems
- Struggles at coping with nowadays complex heterogeneous embedded boards

=> Correctness-by-construction
AMPERE Project Goal

- Fill the gap between
  - MDE techniques with no/limited parallelism support
  - Parallel-programming models with efficient HW offloading (OpenMP, CUDA, ...)
  - Heterogeneity in hardware

- In presence of non-functional requirements
  - High-Performance
  - Real-Time Constraints
  - Energy Efficiency
  - Fault Tolerance
1. **Synthesis methods** for an efficient generation of parallel source code, while keeping non-functional 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 Vision

Bridge the gap

Parallel Execution Model

Parallel Programming Models
(e.g. OpenMP, OpenCL, CUDA, COMPSs)

Run-time parallel frameworks

MDE
(e.g. CAPELLA, AMALTHEA, AUTOSAR)

Sensors
Logic Controller
Actuators

AUTOSAR
- SW-C
- Runnables
- Client-server
- ASIL

AMALTHEA
- Performance
- Tasks
- Scheduling
- Platform

CAPELLA
- Functional components
- Allocation of resources
- Data models
- View points
- validation rules

Meta-model Driven Abstractions
Components, Communications, Timing Characteristics, Integrity Assurance, ...

Model Transformation Engine

Meta-parallel Programming Abstraction
Parallelism, Synchronization, Data Dependencies, Data Attributes, ...

OpenMP
- Task construct
- Dependencies
- Parallel construct

OpenCL
- clGetDeviceId
- clCreateBuffer
- __kernel_exec

COMPSs
- Compute resource
- Data movements
- Task annotations

Parallel Run-Time Frameworks

AMPERE MDE Framework

Tommaso Cucinotta – Real-Time Systems Laboratory - Scuola Superiore Sant’Anna - VHPC 2020
AMPERE Software Architecture

**AMPERE use-cases**
- DSMLs
- Code synthesis tool
- Parallel programming models
- Compilers
- Analysis tools for multi-criteria optimization transformations

**AMPERE Ecosystem**
- System Design
- Computing Software
- Run-time libraries for efficient (and guaranteed) parallel heterogeneous execution
- Operating Systems
- Hypervisor

**AMPERE Software Architecture**

<table>
<thead>
<tr>
<th>Software Layer</th>
<th>Tool</th>
<th>Owner (License)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DSMLs</strong></td>
<td>AUTOSAR</td>
<td>AUTOSAR (Proprietary)</td>
</tr>
<tr>
<td></td>
<td>AMALTHEA</td>
<td>BOSCH (Open-source)</td>
</tr>
<tr>
<td></td>
<td>CAPELLA</td>
<td>TRT (Open-source)</td>
</tr>
<tr>
<td><strong>Parallel programming models</strong></td>
<td>OpenMP</td>
<td>OpenMP ARB (Proprietary)</td>
</tr>
<tr>
<td></td>
<td>CUDA</td>
<td>NVIDIA (Proprietary)</td>
</tr>
<tr>
<td></td>
<td>OpenCL</td>
<td>Khronos (Proprietary)</td>
</tr>
<tr>
<td></td>
<td>COMPSs</td>
<td>BSC (Open-source)</td>
</tr>
<tr>
<td><strong>Artificial Intelligence</strong></td>
<td>TensorFlow</td>
<td>Google (Open-source)</td>
</tr>
<tr>
<td><strong>Code synthesis tools</strong></td>
<td>Synthesis tools</td>
<td>AMPERE (Open-source)</td>
</tr>
<tr>
<td><strong>Analysis and testing tools</strong></td>
<td>NFP analysis</td>
<td>AMPERE (Open-source)</td>
</tr>
<tr>
<td><strong>Compilers and hardware synthesis tools</strong></td>
<td>Mercurium</td>
<td>BSC (Open-source)</td>
</tr>
<tr>
<td></td>
<td>GCC/LLVM</td>
<td>GNU/LLVM (Open-source)</td>
</tr>
<tr>
<td></td>
<td>Vivado</td>
<td>Xilinx (Proprietary)</td>
</tr>
<tr>
<td><strong>Run-time libraries</strong></td>
<td>GOMP</td>
<td>GNU-GCC (Open-source)</td>
</tr>
<tr>
<td></td>
<td>KMP</td>
<td>LLVM (Open-source)</td>
</tr>
<tr>
<td></td>
<td>Vivado</td>
<td>Xilinx (Proprietary)</td>
</tr>
<tr>
<td><strong>Operating systems</strong></td>
<td>Linux</td>
<td>Linux-Foundation (Open-source)</td>
</tr>
<tr>
<td></td>
<td>ERIKA Enterp.</td>
<td>EVI (Open-source/commercial)</td>
</tr>
<tr>
<td><strong>Hypervisors</strong></td>
<td>PikeOS</td>
<td>SYSGO (Proprietary)</td>
</tr>
</tbody>
</table>
AMPERE Software Development Workflow Overview

**System description**
- Components/communication
- Functional/NFP
- Etc.

**Platform description**
- Accel. devices
- Cores/clusters
- Memory model
- Etc.

**Model**
- Meta MDE abstraction
- Meta PPM abstraction

**Code Synthesis + Multi-criteria Optimization**
- Performance
- Time-predictability
- Energy-efficiency
- Resiliency

**Compiler**
- Correctness + Refined Parallel Structure

**Parallel code (e.g. OpenMP, CUDA graphs)**

**Resource Allocation**
(i.e., mapping/scheduling)
- Monitoring
- Dynamic resource allocation

**Run-time framework + OS+Hypervisor**

**Execution Profile**

Tommaso Cucinotta – Real-Time Systems Laboratory - Scuola Superiore - VHPC 2020
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
FPGA System-on-Chip (SoC)

FPGA-based system-on-chips are a very promising solution to enable **predictable** HW acceleration of complex computing workloads:

- Multiprocessors can host multi-OS software systems
- FPGA fabric can be used to deploy HW accelerators
HW Accelerators on FPGAs

- Programmable logic exhibits very regular, clock-level behavior (differently from other HW accelerators, e.g., GP-GPUs)
- Internal control logic of several HW accelerators is typically based on state machines

Possibility to deploy custom **bus logic**

- Bus/memory contention can be made **predictable**

We can monitor & supervise bus transactions to shield the systems from misbehaviors

We can realize custom bus arbitration policies that help meet timing constraints

FIR and Sobel filters from Xilinx IP library (screenshot from Vivado 2017.4)
Dynamic Partial Reconfiguration

- Modern FPGAs offer dynamic partial reconfiguration (DPR) capabilities
- DPR allows reconfiguring a portion of the FPGA at runtime, while the rest of the device continues to operate
- In essence, reconfiguration requires programming a memory
  - Simplifying, an image of the FPGA configuration (bitstream) is copied from one memory to another
Enable **predictable** HW acceleration on FPGA system-on-chips

Collection of technologies developed at the ReTiS Lab

http://fred.santannapisa.it/

**Supported platforms**
- Zynq Ultrascale+
- Zynq-7000 series
FRED Programming Model

periodic/sporadic real-time tasks

Fixed-priority scheduling

work on shared-memory buffers

SW-Task

CPU

System-on-Chip

FPGA Fabric

HW accelerators implemented as programmable logic

non-preemptive execution

HW-Task

SW-Task

TASK(myTask) {
   <prepare input data>
   EXECUTE_HW_TASK(myHWtask);
   <retrieve output data>
}

Suspend the execution until the completion of the HW-task
Time-predictable DNN Inference

- **CHaiDNN**: HLS based DNN Accelerator Library for Xilinx Ultrascale+
  - Designed for maximum compute efficiency at 6-bit integer data types (it also supports 8-bit integer data types)
  - The inference time in isolation exhibits **very little fluctuations**
  - The real issue for time predictability is **bus/memory contention**

**Setup**: Xilinx ZCU102 (Ultrascale+), Vivado2018.2, GoogleNet, DMA from Xilinx IP lib
The FRED framework is a combination of several technologies:

- Run-time FPGA manager & scheduler for Linux (both C and Python API)
- Bus monitors and budget enforcers
- Automated FPGA floor-planning
- Automatic synthesis of bus interconnections
Conclusions

- AMPERE aims to bridge the gap between MDE and PPM on HHW by
  1. Providing a **development framework** for CPS targeting parallel heterogeneous architectures for an increased productivity compliant with current MDE practices
  2. Providing an **execution framework** for an efficient exploitation of parallel and heterogeneous architectures, fulfilling functional and non-functional constraints
  3. **Integrating** AMPERE software solutions into relevant industrial use-cases (automotive and railway) with HPC and real-time requirements
Thanks for Listening
Any Questions?

https://www.linkedin.com/company/ampere-project

https://twitter.com/ampereproject

www.ampere-euproject.eu

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