Multi-Core Execution of *Parallelised* Hard Real-Time Applications Supporting Analysability parMERASA

Project Overview

Theo Ungerer (UAU)
Overview

- parMERASA vision and consortium
- parMERASA targets and overview of achievements
- parMERASA system architecture
- Conclusions
- **Increasing demand** for functionality in current and future (hard) real-time embedded systems
- Today obviously demand for **mixed criticality application** execution
### Hard Real-time Systems

- **Hard real-time:**
  - a deadline should never be missed
  - if missed it may cause harm to humans or equipment

- **Mixed criticality in multi-cores:**
  - combining functionalities with different levels of criticality within multi-core systems
  - e.g. sub-systems to be combined that have different automotive safety integrity levels (ASIL)
parMERASA concerned with time predictability of mixed criticality applications: run hard real-time and non real-time tasks in parallel without interference.

parMERASA goes one step beyond mixed criticality demands:

We target future complex control algorithms by parallelising hard real-time programs to run on predictable multi-/many-core processors.

Currently, timing behaviour of parallel applications is not analysable with current programming paradigms and timing analysis techniques.
parMERASA research is based on multi-core systems:

- parallelisation of industrial applications,
- in concert with WCET technology, verification tools, parallelisation support, and system architecture,
- predictable embedded multi-core design.

MERASA project (2007-2010) was hardware-driven and targeted the now state-of-the-art multi-cores with 4-8 core.

parMERASA project (2011-2014) is application-driven targeting parallelisation of hard real-time applications for future many-cores with 16-64 core.
Project Partners

- **University of Augsburg** (Project Coordinator) Germany
- **Barcelona Supercomputing Center** Spain
- **Université Paul Sabatier** Toulouse, France
- **Technical University of Dortmund** Germany
- **Rapita Systems Ltd.** York, UK
- **Honeywell International s.r.o.** Brno, Czech Republic
- **BAUER Maschinen GmbH** Schrobenhausen, Germany
- **DENSO AUTOMOTIVE Deutschland GmbH** Eching, Germany
Industrial Advisory Board

- **Airbus**, Toulouse, France
- **Infineon Technologies UK Ltd**, Bristol, UK
- **Infineon Technologies AG**, Dept. Industrial & Automotive, Munich, Germany
- **BMW Group**, Munich, Germany
- **Quamcom Research & Technology AB**, Sweden
- **Elektrobit Automotive GmbH**, Munich, Germany
- **Daimler AG**, Stuttgart, Germany
Overview

- parMERASA vision and consortium
- **parMERASA targets and overview of achievements**
- parMERASA system architecture
- Conclusions
Select and parallelise industrial hard real-time applications.

Find ways to efficiently parallelise industrial applications for embedded real-time systems.

Provide hard real-time support in system software, WCET analysis and verification tools for multi-cores.

Develop techniques for time predictable multi-cores with 16 to 64 cores which are commercially feasible.

Contribute to Standards and Open Source Software.
Three Project Phases

  Select applications, define requirements for system SW, tools, and multi-core design

- **Phase 2 (July 2012 – September 2013): Full Specification and Implementation**
  Parallelise for maximum parallelism

- **Phase 3 (October 2013 – September 2014): Optimisation and Refinement**
  Agglomerate for optimal parallelism
Parallelisation of Hard Real-time Applications

- Use cases for parallelisation selected:
  - **Avionics (Honeywell Int.)**
    - 3D Path Planning for airborne collision avoidance
    - Stereo Navigation for aircraft localization when in loss of GNSS
    - Global Navigation Satellite System (GNSS)
  - **Automotive (DENSO Automotive)**
    - Engine control for diesel fuel injection
  - **Construction Machinery (BAUER Maschinen)**
    - Control algorithm for dynamic compaction machine

- Requirements for multi-core architecture, system software, WCET and supporting tools defined
- Parallelisation approach and parallel pattern catalogue developed by University of Augsburg
- Parallelisation started towards suitable parallel design patterns in all application domains

**Talk at ARPA:** parMERASA - Parallelized Avionics and Automotive Software, João Fernandes (Honeywell) and Sebastian Kehr (DENSO Automotive)
Verification and Profiling Tools

- **OTAWA tool** of Univ. of Toulouse
  - adapted to PowerPC ISA
  - specification of annotation format for source code annotations,
  - analysis of synchronisation primitives in progress

- **Five verification and parallelisation support tools selected** by RapitaSystems Ltd., and two are in the state of implementation
  - On-target timing and WCET analysis tool RapiTime enhanced for parallel programs;
  - On-target code coverage tool for parallel programs;
  - Memory, cache and stack analysis tool for parallel programs;
  - Tool to assist with the parallelisation of existing sequential software;
  - Visualisation and profiling tool for parallel programs.

**Talk at ARPA:** parMERASA - WCET Analysis Tools, Haluk Ozaktas, (University Paul Sabatier Toulouse) and Ian Broster (Rapita Systems)
Predictable Multi-core Processor

- **parMERASA Generic Multi-core Architecture** defined by Barcelona Supercomputing Center in collaboration with Technical University of Dortmund and University of Augsburg
  - Clustered multi-core architecture based on simple cores and predictable interconnect
  - Hard real-time support by full isolation of tasks or bounding interferences
- Common parMERASA Architecture Simulator
  - Refined SoCLib version installed at most partner sites

**Talk at ARPA:** parMERASA - Hardware Architecture, Eduardo Quiñones (Barcelona Supercomputing Center)

System Architecture and System-level Software

- System architecture defined and kernel library implementation done by University of Augsburg
- TinyAUTOSAR OSEK support for DENSO application in progress
Overview

- parMERASA vision and consortium
- parMERASA targets and overview of achievements
- **parMERASA system architecture**
- Conclusions
## RTE Requirements

<table>
<thead>
<tr>
<th></th>
<th>Automotive</th>
<th>Avionic</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling</td>
<td>FPP, (Earliest Deadline First)</td>
<td>Fixed cyclic &amp; Pre-emptive techniques</td>
<td>Round-robin</td>
</tr>
<tr>
<td>strategies</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Resources, Events, buffered/unbuffered messages</td>
<td>Messages; Events, Buffers, Blackboards, Semaphores</td>
<td>Events, Semaphores, Spin-Locks</td>
</tr>
<tr>
<td>and Synchronisation techniques</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O requirements</td>
<td>Low latency</td>
<td>Predictability</td>
<td>Low latency</td>
</tr>
<tr>
<td>Protection</td>
<td>OS-Application, (Task)</td>
<td>Partition</td>
<td>Task</td>
</tr>
<tr>
<td>domains</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Blocking sync./com. techniques need interaction with system scheduler ↔ different scheduling algorithms
System Architecture

Application

Domain Specific Interface

Domain Specific RTE (Non-Critical Services)

Protection Boundary

Domain Specific RTE (Critical Services)

Kernel Library

Simulated Hardware
Kernel Library

- Abstraction of underlying hardware
- Provides RTE-independent services
  - WCET-analysable primitives
  - Synchronisation functions for parallel applications
  - Common base for RTE implementations

- OS Kernel consists of
  - Kernel Library
  - RTE-specific isolation-critical services
  - Executed on every core

- Non-critical RTE services: no influence on other partitions
System Architecture

User Mode

Application Layer

Domain-specific Interface

Non-critical RTE Services

Critical RTE Services

Scheduling

Protection

Communication & Synchronisation

I/O

Context Management

Memory Management

Synchronisation Mechanisms

Interrupt Handling

Kernel Library Services

Simulated Hardware
Implementation of AUTOSAR stack
- Multi-core OS functionality
  - Task management
  - Interrupt handling
  - Resource management
  - Event control
  - OS execution control

- I/O functionality: virtual CAN

- Multi-core communication
  - Sender-receiver
  - Client-server
  - Inter OS-Application Communicator
parMERASA targets **parallelising hard real-time programs** to run on predictable multi-/many-core processors.

- **1st project phase** (Oct. 2011-June 2012) “Requirement Specification and Concept” **completed**
- **2nd project phase** (July 2012-September 2013) “Parallelisation and Implementation” currently in **month 7 of 15**