parMERASA

Multi-Core Execution of Parallelised Hard Real-time Applications Supporting Analysability

D6.5 – Report on Experiences with Tiny Automotive RTE for Multi-cores to the Standardisation Committees

Nature:	**R - Report**
Dissemination Level:	**PU - Public**
Due date of deliverable:	**30/09/2013**
Actual submission:	**30/09/2013**
Responsible beneficiary:	**UAU**
Responsible person:	**Theo Ungerer**

Grant Agreement number:	FP7-287519
Project acronym:	parMERASA
Project title:	Multi-Core Execution of Parallelised Hard Real-Time Applications Supporting Analysability
Project website address:	http://www.parmerasa.eu
Funding Scheme:	STREP SEVENTH FRAMEWORK PROGRAMME THEME ICT – 2011.3.4 Computing Systems
Date of latest version of Annex I against which the assessment will be made:	June 20, 2012
Project start:	October 1, 2011
Duration:	36 month

Project coordinator name, title and organisation: Prof. Dr. Theo Ungerer, University of Augsburg
Tel: + 49-821-598-2350 Fax: + 49-821-598-2359 Email: ungerer@informatik.uni-augsburg.de

Release Approval

name	role	date
Christian Bradatsch, Florian Kluge	author	2013-09-20
Sebastian Kehr, Bert Böddeker	contributor	2013-09-20
Ian Broster	WP6 leader	2013-09-20
Theo Ungerer	coordinator	2013-09-20

Deliverable Summary

Deliverable D6.5 “Report on Experiences with Tiny Automotive RTE for Multi-cores to the Standardisation Committees” concerns two recommendations to the standardization committees. The first recommendation “Tiny Automotive RTE Concept” is illustrated in more detail in deliverable D4.4. Both recommendations are implemented in the tiny automotive RTE prototype in deliverable D4.3.

Deliverable D6.5 summarises the work done in task T6.7 to fulfil milestone 16.

Task description of T6.7 (m24):
Report on experiences with Tiny Automotive RTE for multi-cores to the AUTOSAR Standardisation Committee and to ISO 26262 (see DoW, sect. 1.3.3, p. 59)

We plan to bring in the recommendations into the Software Architecture working group of AUTOSAR or a corresponding task force for new scheduling and communication approaches (concept 33) of DENSO AUTOMOTIVE GmbH Deutschland.

Conclusion of task T6.7:

All targets of task T6.7 and deliverable D6.5 have been reached.

Table of Contents

Experiences

One objective of the parMERASA project is to exploit fine grained parallelism in embedded industrial applications. In case of the automotive domain, these applications almost rely on the AUTOSAR software architecture. The smallest entity in the AUTOASR software architecture is a Runnable, which contains a sequence of instructions. Fine grained parallelized algorithms have a close interaction on data. For good execution performance, the access to the data has to be fast. Hence, the parallelized algorithms should be inserted in a very low layer of the software architecture to reduce overhead. In the AUTOSAR software architecture they should be encapsulated inside of Runnables.

Actually, a Runnable cannot span over more than one core. An obvious approach would be to distribute a parallelized algorithm over several Runnables. A Runnable Entity, which is executed in the runtime system, is at least assigned to one task. Again, a task is assigned to an OS-Application, which is located entirely on one core. So, a parallelized algorithm, which is distributed over multiple Runnable Entities, is also distributed over multiple OS-Applications. The communication between OS-Applications is done via the Inter OS-Application Communicator (IOC). The IOC is in charge of the communication crossing core or memory protection boundaries [1], which can imply additional overhead. On the other side, such a procedure bloats the application design.

In the next period we will try to figure out, how the AUTOSAR software architecture could be adapted to reduce the resulting overhead in the application design and compare the different approaches.

Our experiences while designing and implementing the tiny automotive RTE on a simulated predictable many-core with up to 64 cores lead to the following recommendations and requirements to enhance a future multi-/many-core AUTOSAR specification.

Recommendations

1 Tiny Automotive RTE Concept

1.1 Functional description

The tiny automotive RTE represents a subset of the AUTOSAR software architecture. The main purpose of the tiny automotive RTE is to provide an evaluation platform for execution of *parallelized* automotive applications and automotive applications executed *in parallel* on a multi-/many-core processor. One objective is to keep the tiny automotive RTE subset minimal concerning the basic functionalities required on each core and to its code basis. The other objective is to provide basic communication and synchronization features inside a multi-/many-core processor.

1.2 Requirements

1.2.1 Tiny automotive RTE shall only comprise a minimal set of BSW modules necessary on each core.

Description:	Tiny automotive RTE shall only comprise a minimal set of BSW modules necessary on each core.
Rationale:	To execute a (parallelized) application, a minimal OS and some additional BSW modules are needed on each core. To keep the code basis small, only necessary BSW modules are included in the tiny automotive RTE.
Use Case:	A parallelized application which is executed on a many-core processor and only requires basic functionalities.
Dependencies:	--

1.2.2 Each tiny automotive RTE BSW module shall provide the same API as its corresponding AUTOSAR BSW module to be compatible to other AUTOSAR BSW modules. If the API of the tiny automotive RTE BSW module is reduced, an additional module shall be provided, which contains the missing parts of the API to guarantee compatibility.

Description:	Each tiny automotive RTE BSW module shall provide the same API as its corresponding AUTOSAR BSW module to be compatible to other AUTOSAR BSW modules. If the API of the tiny automotive RTE BSW module is reduced, an additional module shall be provided, which contains the missing parts of the API to guarantee compatibility.
Rationale:	The tiny automotive RTE shall be extendible with any AUTOSAR BSW module. To do so, the BSW modules included in the tiny automotive RTE shall have the same API.
Use Case:	A parallelized application which is executed on a many-core processor and requires additional functionalities, e.g. a CAN communication stack.
Dependencies:	--

1.3 Comments

The functionalities that form the tiny automotive RTE, as seen by the parMERASA project, are defined in the public deliverable D4.4, see http://www.parmerasa.eu/index.php?menu=deliverables.

2 Time-predictable Spinlock Mechanism

2.1 Functional description

A spinlock as defined in [1] provides a mechanism for mutual exclusion of shared resources accessed by tasks from different cores. It lacks fairness and timing analysability. For static WCET analysis, the waiting time to access a shared resource must be bounded, which also implies that the access order

must be fair. For example, queuing spinlocks [2] and ticket locks [3] are mutual exclusion mechanisms which are fair and WCET timing analysable.

2.1.1 A mutual exclusion mechanism between cores shall be provided that allows determining bounded waiting times.

Description:	A mutual exclusion mechanism between cores shall be provided that allows determining bounded waiting times.
Rationale:	To synchronize access from different cores to shared resources, a mutual exclusion mechanism is needed. To assure that a task ever gets access to a shared resource, the mutual exclusion mechanism shall guarantee that tasks acquire a shared resource on a first-come first-served basis and hence an access from a task has a bounded waiting time.
Use Case:	Concurrent access to shared resources from different cores.
Dependencies:	--

References

[1] AUTOSAR, “Specification of Operating System, V5.1.0”.

[2] T. S. Craig, “Queuing Spin Lock Algorithms to Support Timing Predictability,” in *Proceedings of Real Time Systems Symposium*, Raleigh Durham, NC, 1993.

[3] M. Gerdes, Timing Predictable Execution of Parallel Hard Real-Time Programs, Augsburg: Universität Augsburg, 2013.