Chapter 28
Affordable Safe and Secure Mobility Evolution

Alexander Viehl, Udo Gleich, and Hendrik Post

28.1 Mobility Systems Evolution

Many of today’s societal challenges such as global warming, tightening energy supplies, aging society, or security have an impact on transportation, be it automotive, rail, or aviation. ERTRAC, the European Road Transport Research Advisory Council, aims at a 50% more efficient transport system by 2030 [1]. In terms of fatalities and severe injuries, the automotive industry targets a 60% reduction until 2030.

In the premium market’s segments, innovative functions are the most important factor to influence buying decisions. Future mobility solutions will increasingly rely on smart components that continuously monitor the environment and assume more and more responsibility for a convenient, safe, and reliable operation in parallel to an additional facilitation of energy consumption optimization and emission reduction. A major step for the evolution of autonomous systems is the transition
from assistance systems to automated hands-off systems. Public perception moves
toward higher expectations with regard to the safety of highly autonomous systems.
With hands-off systems, a failure rate clearly below the one of a human actor is
expected. For automotive, the self-driving car is the next big revolution, and it is
still unclear how functional and nonfunctional guarantees can be given probably
zero defect for those new class complex-automated functions. Consequently, novel
design and verification methods for such highly automated systems are needed to
satisfy future safety-relevant systems requirements.

During the design process of a complex distributed system, system level require-
ments are broken down into more fine-grained technical hardware and software
requirements and mapped to subsystem parts. During this requirement break down,
various models are created to represent distinct aspects of the developed system.
Hence, traceability between different abstraction levels of requirements and system
parts must be established. The designs models and hence design decisions have to
be verified with respect to their associated functional and safety requirements, and
it has to be ensured that the implementation does not violate requirements.

One major limitation today is the unavailability of synthesis and verification tools
for these specific models. While research prototypes show the general feasibility
of formally proving the correctness of models or code with respect to given
requirements, these tools have not been adopted by the industry widely.

Tools working on source code or implementation level that check for specific
errors or design flaws are available and well applied today. However, higher-level
requirements and the correctness of design decisions cannot be checked effortlessly
and completely with high confidence.

In addition to that, further challenges arising with the shift to more auton-
omy are the increasing complexity and performance requirements of autonomous
systems. On the way to realize this vision, the need for computing power will
drastically increase far beyond what can be provided by conventional sequential
single-core hardware. While the required efficiency and scalability compel future
embedded microcontrollers (μC) to rely on multi- and many-core architectures, the
change in hardware architecture also necessitates fundamental advances to software
development methodology. Replacing today’s essentially sequential technology by
interconnected cores and omnipresent communication poses the colossal challenge
in software development to identify and exploit means for concurrency still
guaranteeing reliable and predictable behavior. One problem here is that analysis
techniques and flows have to be extended to support parallelism and system
complexity on several design and implementation levels.

Current analysis techniques are severely limited by the size and complexity of
the embedded systems to be analyzed. Tools using abstract interpretation to prove
the absence of runtime defects usually become difficult to use for code sizes above
200,000–300,000 lines of code depending on the used programming features and
the code complexity. Model checking techniques are currently limited to much
smaller sizes of the program state space since they enumerate overall possible
program states without abstraction. Moreover, tools for the analysis of concurrent
and multi-core software currently present a large number of false positives to
the user. Consequently, an efficient assessment of analysis results for concurrent systems in an industrial environment is currently not feasible.

Combinations of different analysis methods and tools for concurrent systems are mostly in a premature state of research and not viable for industrial application yet. One reason for this situation is the absence of standardized interfaces between verification and modeling tools that support verification tasks. Formal verification tools typically only apply one technology and support one implementation language. A close collaboration of these tools is needed, allowing the exchange of analysis findings and given assumptions across modeling languages and tooling borders.

28.2 Objectives

The main goal of the ASSUME project is the affordable, standard-compliant development and verification of highly automated, safety-relevant, and performance-critical mobility systems. A strong focus is on development methods for concurrent systems and static verification techniques. The ASSUME algorithm portfolio will be the key technology to bring innovative solutions from sandboxes into consumers’ daily lives. ASSUME provides a seamless engineering methodology to overcome this roadblock. The problem is addressed on the constructive and on the analytic side. For efficient construction and synthesis of embedded systems, the project provides new tools, standards, and methodologies to cover most of the challenges by design. In addition, ASSUME provides a well-integrated sound static analysis solution that allows proving the absence of problems even in a multi-core environment. New algorithms will be integrated in exploitable tools. New interoperability standards and requirement formalization standards will facilitate cooperation between different market players.

The major field of innovation for ASSUME’s industrial partners (end users) resides in the model-based parallel software engineering for multi- and many-core processors. The project enables the effective use of formal verification and synthesis technology along the design flow. Methods and tools are developed that support the safe migration of sequential programs into parallel paradigms. Safety in parallel execution environments will further be ensured by the extension of formal methods to support concurrency for multi- and many-core CPUs.

This engineering methodology, which will be supported by code generators and static analyzers, enables system/software developers to move from the current engineering for sequential functions to a concurrent realization, in order to benefit from the increased performance and hardware integration that result from the use of modern parallel processors.

Sound static analysis holds the promise to perform fully automated and exhaustive detection of important classes of errors (such as runtime errors) with full control and data coverage. Despite important past progress, related tools still suffer in many contexts from limitations in precision and in domain of application. ASSUME delivers analysis methods and tools with improved precision and efficiency. The
currently available technologies for the verification of nonfunctional properties (absence of runtime errors) are extended to larger and more complex systems. Moreover, through improvement and combination of analysis techniques, the verification of functional properties (such as requirements) through static analysis will be implemented. This is possible through a complete traceability of formalized requirements to their implementation. It is made possible to find example scenarios that might violate a requirement. These examples are very valuable information to identify the cause of defects and errors in the models and implementation. Static analysis techniques are extended and improved for the industrial use of concurrent software on multi- and many-core platforms. The ASSUME project proposes to extend existing static analyzer tools and prototypes for concurrent software along several lines:

- The sound support for true concurrency offered by scheduling on multi-core hardware
- The precise handling of the associated models of memory consistency
- The efficient migration of sequential software to parallel applications
- The support for modern embedded operating system

Technological innovations are driven by the industrial use cases provided by industrial partners to achieve high precision through specialization.

### 28.3 Expected Outcomes

The expected outcome of ASSUME is the design of a static analysis platform (SAP), the first of its kind, able to check for both classic and concurrency-related runtime errors on large embedded industrial multi-core software, with a high efficiency and low rate of false alarms.

ASSUME advocates the idea that static analysis of software can be extended to provide affordable means to prove the absence of defects. The newly developed static analysis platform (SAP) will extend the state of the art in the following aspects:

- SAP scalability is improved to obtain a high enough precision-runtime tradeoff for future software complexity.
- SAP supports fully parallel software running on multi- and many-core μ.C.
- SAP integrates a new framework for the formalization of safety and security requirements.
- SAP allows for seamless traceability and impact analysis of functional and extra-functional properties in model-driven and conventional development.

The key advancement of the ASSUME's analysis platform is the improved integration of verification tools across language and tool boundaries. The ASSUME platform will facilitate various available models along the embedded development chain to improve traceability and associate requirements at different levels,
solutions, and validation and verification activities. The main benefits of this integration are higher analysis precision and the ability to verify larger and concurrent systems, to check functional properties, and at the same time decrease the verification effort. The versatility of the platform will be achieved by the provisions of exchange formats but is not limited to this. The SAP will allow the implementation of meta-algorithms for an intelligent (re-) combination of algorithms to get overall improved analysis results. Another valuable feature is the improved exploitation of available hardware resources by the development of methods and tools that enable efficient parallelization due to the more precise estimation of the worst-case execution time (WCET) and other properties in parallel systems.

Last but not least, the widespread use of multiprocessor systems-on-chip architectures (including multi-/many-cores) imposes significant changes to the models and methods used to perform scheduling and code generation for embedded platforms. The adoption of multi-/many-core architectures is driven by scalable performance arguments (concerning speed, power, etc.), but this scalability comes at the price of increased complexity of both the software and the software mapping process (including scheduling and code generation). Part of this complexity can be attributed to the steady increase in the size of software that is run by a single system. But there are also significant qualitative changes concerning both the software and the hardware. In software, more and more applications include parallel versions of signal processing, simulation, and control algorithms, which are best modeled using dataflow models (as opposed to independent tasks). Providing functional and real-time correctness guarantees for parallel code requires an accurate control of the interferences due to concurrent use of communication resources. Depending on the hardware and software architecture, this can be very difficult. There are two main reasons to this. The first one concerns communications: as the tasks are more tightly coupled and the number of resources in the system increases, the on-chip networks and shared memory banks become critical resources, which need to be explicitly considered and managed during scheduling. The second reason concerns automation: the complexity of large many-cores and of the (parallel) applications mapped on them is such that code generation, allocation, and scheduling must be largely automated. Formal verification techniques for these automated tools are in great need.

ASSUME proposes a set of methods and tools that enable the efficient use of synthesis techniques in the design flow of parallel software. The common denominator of these techniques is formalization and full automation. The technical focus is placed on formal compiler verification and on correct real-time implementation for parallel applications. In both cases, hardware modeling is recognized as a major issue and dedicated specific attention.

Figure 28.1 shows the proposed fields for technology innovations of the ASSUME project along the development chain of embedded systems.

The circular arrow represents the system development or more precisely the design flow. The proposed technological contributions of ASSUME are arranged around and in the center of this circle. The development chain differs from classical
software development in terms of distribution of development roles as contained in the market value chain but also technically due to the close integration into physical processes. Because of the severe consequences of failures in these systems, development processes are strictly regulated in the specific standards for the several mobility domains.

Acknowledgments  This work was partially supported by BMBF under grant number 01IS15031.

Reference

1. ERTRAC Strategic Research Agenda 2010, Towards a 50% More Efficient Road Transport System by 2030, May 2010, www.ertrac.org