Operating Systems Technology for Converged ECUs
Extended Abstract

André Hergenhan
OpenSynergy GmbH
Berlin
andre.hergenhan@opensynergy.com

Gernot Heiser
Open Kernel Labs and NICTA and UNSW
Sydney, Australia
gernot@ok-labs.com

Abstract

Introduction

The number of ECUs in modern cars is increasing, with top-of-the-line luxury cars featuring in excess of 100 processors. For example Volkswagen's Phaeton possesses about 70 to 100, depending on the version. The large number of ECUs is causing problems with cost, complexity and even space.

AUTOSAR [AUT] is the automotive industry's emerging standard software architecture, specifically designed to support merging separate control functions, formerly provided on individual ECUs, onto a single processor platform. It provides real-time operating-system (RTOS) functionality with hardware-enforced protection to isolate subsystems, and defines a component framework that is designed to integrate software components provided by different suppliers. These communicate via a *virtual functional bus* (VFB).

Infotainment

AUTOSAR is well suited for converging ECUs used for a vehicle's control and comfort functions. However, it fails to address the increasingly important infotainment functions. This not only limits the potential for ECU consolidation. It also gets into the way of the increasing need for interaction between infotainment and connectivity and comfort functions.

Imagine a driver assistance systems which exploits map materials provided by navigation systems in order to improve prediction of traffic situations. Or a park distance-control system may make use of a display controlled by the infotainment system. These examples show that even the classification of applications into specific car domains are becoming blurred.

In addition, the so-called head unit running the automotive infotainment system is normally the best-resourced ECU of the car. Hence, this ECU is ideally suited for computationally-intensive applications, as driver assistance systems are.

Thus, there are clear benefits from combining infotainment with driver systems and comfort functions on the same ECU.

The challenge in achieving this is a result of the dramatically-different nature of infotainment on one hand, and the functions served by AUTOSAR on the other. Specifically, control and comfort functions are characterised by strict requirements for real-time behaviour, and strong reliability, which imply strong temporal and spatial partitioning and an emphasis on worst-case performance.

Infotainment, on the other hand, is charactered by best-effort or soft real-time needs, high and varying resource requirements, and an emphasis on average-case performance. Furthermore, infotainment requires operating systems with rich functionality and high-level APIs, and consequently large and untrustworthy code bases.

Hence, there are conflicting requirements of strong separation yet close cooperation between infotainment and the AUTOSAR-based real-time world.

Virtualization

We argue that system virtual machines (see Figure 1) can be used to bridge this gap. A virtual machine can host a rich operating system, such as Linux, to support infotainment requirements. Other virtual machines can run an AUTOSAR environment. Such a setup, using the OKL4 micro-

kernel [OKL] to support concurrent rich-OS and RTOS environments, is successfully deployed in mobile phones and settop-boxes. While a strict virtualization approach is too inflexible to meet the requirements of embedded systems, such as cars, OKL4 is a flexible and general-purpose platform that provides the right mechanisms [Hei08].

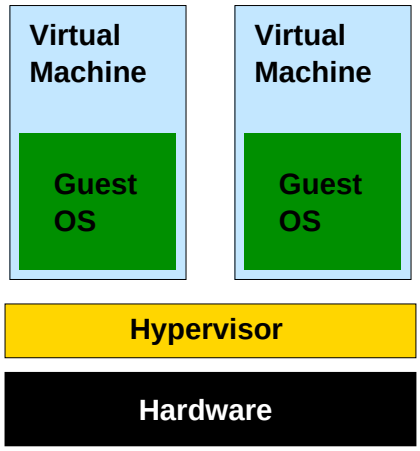

Figure 1: System virtualization creates a software environment where several "guest" operating systems, each running on virtual hardware, can share the same physical hardware.

COQOS

OpenSynergy is developing the COQOS software framework which uses virtualization to provide an environment where any combination of automotive computing functionality can be consolidated.

The essential elements of the COQOS architecture are shown in Figure 2. The framework mainly consists of three parts: the OKL4 microkernel and virtualization platform, any rich operating system that is able to serve for automotive infotainment (e.g., Linux) running in a virtual machine and a complete AUTOSAR environment.

The OKL4 microkernel ensures strict partitioning of the hardware resources between the two sides, while providing low-latency, high-bandwidth communication via its native high-performance message-passing mechanism and the ability to set up shared memory buffers.

The right side of the COQOS architecture depicts two layers of the AUTOSAR architecture for ECUs. The AUTOSAR Basic Software layer provides hardware and microcontroller abstraction

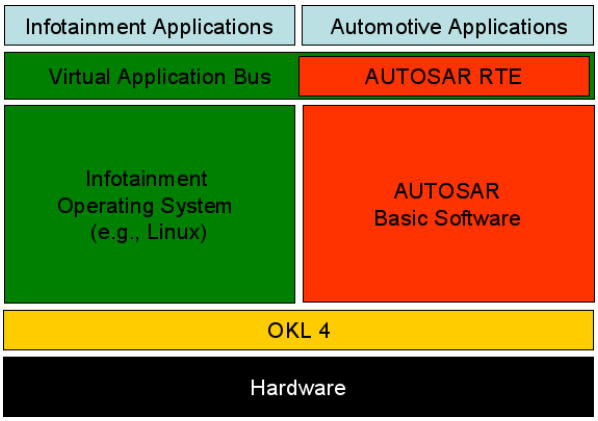

Figure 2: The software framework COQOS enables the integration of automotive multimedia/infotainment as well as AUTOSAR systems onto a single ECU.

as well as higher-order services. The so-called AUTOSAR RTE (run-time environment), however, denotes a middleware layer that provides a component framework that supports the integration of software components. All in all, AUTOSAR provides an environment within COQOS for automotive open- and closed-loop applications with hard real-time requirements.

Due to this design, the AUTOSAR layers are tightly integrated with OKL4. This requires that OKL4 can satisfy the requirements of automotive software paradigms.

One interesting issue here is that OKL4 traditionally uses priority-based scheduling. This naturally supports event-triggered architectures, but does not map easily onto the time-triggered approach favoured by the automotive industry. RBED [BBLB03], a generalisation of EDF scheduling, has recently been implemented in OKL4 [LSP08]. It naturally supports OSEK-style scheduling [OSE] and can be extended to fully support time-triggered architectures.

The OKL4-based COQOS architecture specifically addresses the needs of automotive applications to communicate across domain boundaries (Linux/AUTOSAR), and that a specific application may be integrated on either side. For this purpose, COQOS extends the AUTOSAR virtual functional bus (or RTE on an particular ECU) into the Linux side.

This concept of a unified *virtual application bus* will simplify the integration of infotainment applications in a similar manner as the VFB/RTE does

for the AUTOSAR side. In addition, the virtual application bus will enable transparent communication between any applications irrespective of their location (on top of COQOS) or classification.

Summary

We made a case for a convergence of infotainment with control/convenience functionality in automobiles. We presented the COQOS software framework which supports this convergence, with the help of virtualization and microkernel technology. A COQOS prototype is to be demonstrated at this year's IAA and a first release will be launched by the end of the year.

References

[AUT] AUTOSAR. http://www.autosar.org.

[BBLB03] Scott A. Brandt, Scott Banachowski, Caixue Lin, and Timothy Bisson. Dynamic integrated scheduling of hard real-time, soft real-time and non-real-time processes. In *Proceedings of the 24th IEEE Real-Time Systems Symposium*, Cancun, Mexico, December 2003.

[Hei08] Gernot Heiser. The role of virtualization in embedded systems. In *1st Workshop on Isolation and Integration in Embedded Systems (IIES)*, pages 11–16, Glasgow, UK, April 2008. ACM SIGOPS.

[LSP08] Martin P. Lawitzky, David C. Snowdon, and Stefan M. Petters. Integrating real time and power management in a real system. In *Proceedings of the 4th Workshop on Operating System Platforms for Embedded Real-Time Applications*, Prague, Czech Republic, July 2008.

[OKL] OKL4 community site. http://okl4.org.

[OSE] OSEK VDX portal. http://www.osek-vdx.org/.