Time-Predictable and Composable Architectures for Dependable Embedded Systems

Saddek Bensalem
Verimag,
University Joseph Fourier
saddek.bensalem@imag.fr

Kees Goossens
Eindhoven university of technology
k.g.w.goossens@tue.nl

Christoph M. Kirsch
University of Salzburg
ck@cs.uni-salzburg.at

Roman Obermaisser
University of Siegen
roman.obermaisser@uni-siegen.de

Edward A. Lee
University of California, Berkeley
eal@eecs.berkeley.edu

Joseph Sifakis
CNRS-Verimag
joseph.sifakis@imag.fr

ABSTRACT

Embedded systems must interact with their real-time environment in a timely and dependable fashion. Most embedded-systems architectures and design processes consider "non-functional" properties such as time, energy, and reliability as an afterthought, when functional correctness has (hopefully) been achieved. As a result, embedded systems are often fragile in their real-time behaviour, and take longer to design and test than planned. Several techniques have been proposed to make real-time embedded systems more robust, and to ease the process of designing embedded systems:

1. *Precision-timed and time-triggered architectures*, to make time a first-class citizen of system design.
2. *Deterministic* architectures for repeatable timing behaviour.
3. *Composability*, which guarantees that the (non)-functional behaviour of components is unchanged on integration in a larger system.

The tutorial *presents the state of the art and major approaches to time-predictability and composability*, such as BIP, TTA, PRET, PTIDES, Giotto, and CompSOC.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Reliability, availability, and serviceability, Performance Attributes, Modeling Techniques

General Terms

Design, Performance, Reliability, Theory, Verification

Keywords

System on chip, design, methodology, real-time, determinism, predictability, composability

EMSOFT'11, October 9–14, 2011, Taipei, Taiwan.
ACM 978-1-4503-0714-7/11/10.

1. Prof. Goossens: Composable Timing and Energy in CompSOC

This presentation will give an overview of the CompSOC platform that enables multiple applications to be implemented, verified, and executed independently. Any mix of best-effort, soft and firm real-time applications can be integrated, without any interference between them. Composability holds for functional behaviour, but also for timing, energy, and power [4]. Every application has its own time budget on processor tiles, network on chip [3], on-chip memories, and SDRAM off-chip memory [1], enabling end-to-end performance guarantees for real-time applications. Energy and power budgets per application on each processor tile allow each application to have its own (real-time) power manager, without causing interference.

2. Prof. Kirsch: The Logical Execution Time Paradigm

Since its introduction in 2000 in the time-triggered programming language Giotto, the Logical Execution Time (LET) paradigm has evolved from a highly controversial idea to a well-understood principle of real-time programming. This tutorial provides an easy-to-follow overview of LET programming languages and runtime systems as well as some LET-inspired models of computation. The presentation is intuitive, by example, citing the relevant literature including more formal treatment of the material for reference. This is joint work with Ana Sokolova.

3. Prof. Obermaisser: Time-Triggered Architecture – Concepts, Applications and Research Challenges

This presentation gives an overview of the concepts, applications and research challenges of the Time-Triggered Architecture (TTA). The TTA supports complexity management through the straightforward composition of systems out of components. As the foundation for robustness, the TTA supports fault isolation, the selective restart of components after transient faults, and active redundancy. Security is addressed at all levels of the architecture. Energy efficiency is enabled through integrated resource management that makes it possible to individually reduce the power-requirements of components. The TTA is a platform architecture that provides a minimal set of core services and

a plurality of optional services that are predominantly implemented as self-contained system components. Choosing a suitable set of these system components that implement optional services, augmented by application specific components, can generate domain-specific instantiations (e.g., automotive, avionics).

4. Prof. Lee: Repeatable Timing in Software and Networks

All widely used software abstractions lack temporal semantics. The notion of correct execution of a program written in every widely-used programming language today does not depend on the temporal behavior of the program. But temporal behavior matters in almost all systems. Even in systems with no particular real-time requirements, timing of programs is relevant to the value delivered by programs, and in the case of concurrent programs, also affects the functionality. In systems with real-time requirements, such as most cyber-physical systems, temporal behavior affects not just the value delivered by a system but also its correctness.

In this talk, we will argue that time can and must become part of the semantics of programs for a large class of applications. To illustrate that this is both practical and useful, we will describe two recent efforts at Berkeley in the design and implementation of timing-centric software systems. On the design side, we will describe PTIDES, a programming model for distributed real-time systems. PTIDES rests on a rigorous semantics of discrete-event systems and reflects the realities in distributed real-time, where measuring the passage of time is imperfect. PTIDES enables deterministic time-sensitive distributed actions. It relies on certain assumptions about networks that are not trivial (time synchronization with bounded error and bounded latency), but which have been shown in some contexts to be achievable and economical. PTIDES is also robust to subsystem failures, and, perhaps most interestingly, provides a semantic basis for detecting such failures at the earliest possible time. On the implementation side, we will describe PRET machines, which redefine the instruction-set architecture (ISA) of a microprocessor to include temporal semantics.

5. Prof. Sifakis: Rigorous Component-based System Design Using the BIP Framework

Rigorous system design requires the use of a single powerful component framework allowing the representation of the designed system at different levels of abstraction, from application software to its implementation. Such a framework allows to maintain overall coherency and correctness of the design flow by comparing different architectural solutions and their properties.

We present a rigorous design flow using the BIP (Behavior, Interaction, Priority) [2] component framework as a unifying semantic model to derive correct implementations from application software, a model of the target architecture and a mapping. Correctness of implementations is ensured by application of source-to-source transformations in BIP which preserve essential design properties. The design flow is fully automated and supported by a toolset including a compiler, the D-Finder verification tool and model transformers. It is illustrated through two non-trivial case studies.

1. REFERENCES

[1] B. Akesson and K. Goossens Architectures and modeling of predictable memory controllers for improved system integration. In *DATE*, 2011.

[2] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T. H. Nguyen, J .Sifakis. Rigorous Component-Based System Design Using the BIP Framework. In *IEEE Software* 28(3): 41-48 (2011).

[3] K. Goossens and A. Hansson. The Aethereal network on chip after ten years: Goals, evolution, lessons, and future. In *DAC*, 2010.

[4] A. Nelson, A. Molnos and K. Goossens. Composable power management with energy and power budgets per application. In *SAMOS*, 2011.