Position Statement for Panel on Grand Challenges in Embedded Software

Joseph Sifakis
Verimag
2 Avenue de Vignate, centre Equation
GIERES, France
Joseph.Sifakis@imag.fr

Embedded software design is part of embedded system design, which by its very nature, requires a deep and coherent integration of competencies in software, hardware, and controller design. The scientific challenge is in setting up embedded systems as a new discipline, which systematically and even-handedly marries computation and physicality, performance and robustness. Our aim is not to discuss this grand challenge presented in detail in [1], but rather to identify missing pieces for applying the *formal methods* paradigm to embedded systems design. Formal methods, in particular *formal verification*, have been successfully applied to hardware design, and more recently, to software design. To what extent is it possible to adapt existing methods and tools to embedded systems?

Design is the process of deriving a system that meets given requirements. Correctness can be demonstrated using formal models meeting the requirements and representing a design at some level of abstraction. For some classes of systems, it is possible to derive a design from a model which *by-construction* meets the requirements (e.g. hardware synthesis). For others, a design is obtained as the result of a a creative process using existing algorithms and principles for organizing computation, pre-defined functions, data, components, etc. In this case, correctness may be established by *a posteriori* verification, to show that a model, which is an adequate abstraction of the design, meets the given requirements.

There are two non-trivial obstacles to transposing the formal methods paradigm to embedded systems.

Faithful modeling: Contrary to pure software or hardware, for a given embedded system, we do not know how to derive a model that faithfully represents its behavior at the proper level of abstraction. Embedded systems are *heterogeneous*. They are composed of a large variety of components, each having different characteristics, from a large variety of viewpoints, each highlighting different dimensions of a system.

We need models representing systems at varying degrees of detail and interrelated in an abstraction hierarchy. A key abstraction would relate application software to its implementation on a given platform. Another cause of heterogeneity in abstractions, is the use of different viewpoints e.g. combining computational and analytical models or different extra-functional dimensions such as timing and power consumption. At each level of abstraction, two main sources of heterogeneity may exist. a) Components may be fully synchronized, or asynchronous. Currently, we do not know how to consistently integrate synchronous and asynchronous models. b) There is a large variety of dispersed mechanisms used for coordinating interaction between components, including semaphores, monitors, message passing, remote call, protocols etc.

We need a *unified composition framework* for heterogeneous components. Such a framework should allow system designers to formulate their solutions in terms of tangible, well-founded and organized concepts. It should consider architectures as first-class entities, having their own properties, that can be studied independently of their components' behavior.

Achieving correctness: Current verification techniques focus mainly on invariance properties by analysis of abstractions. Embedded systems are concurrent, and their correctness is characterized by other classes of properties not preserved by abstractions. This is typically the case for progress properties or time-dependent properties. For instance, we do not have abstractions preserving even simple progress properties such as deadlock-freedom.

Furthermore, existing verification techniques work on flattened global models, whereas for embedded systems, extra-functional properties depend on architectural features. For all these reasons, we believe that *a posteriori* verification by itself is not sufficient for achieving correctness. As discussed in [1], emphasis should be put on results allowing *correctness-by-construction* in two complementary directions: a) Develop *reference architectures* guaranteeing generic properties *by-construction* such as security, robustness, diagnosability, adaptivity. b) Develop results allowing interference-free composition of different architectural solutions. Such results are essential for guaranteeing the stability of properties of integrated components, and are necessary for building reconfigurable systems.

1. REFERENCES

[1] T. A. Henzinger and J. Sifakis. The Embedded Systems Design Challenge. In *Formal Methods 06, LNCS 4085*, pages 1–15, 2006.