Microelectronics Journal 34 (2003) 989–1000

Microelectronics Journal

www.elsevier.com/locate/mejo

A general time model for the specification and design of embedded real-time systems[☆]

Ralf Münzenberger[a,1], Matthias Dörfel[a,1], Richard Hofmann[a,1], Frank Slomka[b,*]

[a]*Department of Computer Science 7, University of Erlangen—Nuremberg, Martensstr. 3, 91058 Erlangen, Germany*
[b]*Department of Computer Science, University of Oldenburg, Ammerländer Heerstraße 116-118, 26111 Oldenburg, Germany*

Accepted 3 April 2003

Abstract

Design of complex embedded systems feasible with current and upcoming semiconductor technologies necessitates consideration of real-time from the beginning. However, the commonly used specification techniques do not consider temporal aspects in general like fulfillment of high level timing requirements or dynamic reactions on timing violations. In this paper, we discuss the restrictions of current specification techniques for embedded real-time systems and present a general time model that solves this issue. The time model contains the progress of time, the measurement of time and the specification of timing requirements based on event traces. In contrast to earlier techniques, preconditions determine the actual relevance of a specific timing bound. Exemplified for SDL, a solution for the specification of temporal aspects is shown. The advantages of this solution are discussed in a hardware/software co-design case study from the mobile communication area.

Keywords: Real-time; Embedded systems; Hardware/software co-design; Specification of clocks

1. Introduction

The high transistor count of modern VLSI chips makes it feasible to increase the functionality of embedded systems. This enables the integration of complete systems like cellular phones or mechatronic systems on a single chip. The design process of such real-time systems has to cover many different aspects: VLSI design of the complete system architecture, embedded real-time software design, simulation and verification of the system's hardware and software, rapid-prototyping of the whole system, and a validation in the real environment. Finally, the system has to be partitioned into hardware and software in a way that minimizes costs while fulfilling all requirements.

Such complex tasks require a specification language that allows for automated implementation and verification of *functional and temporal behaviour* of the real-time system. Furtheron, system synthesis, i.e. the optimized decision on which parts to map to hardware and on which parts to map to software, is only possible with a system-level specification language like, for example SDL [5], as it is used in Ref. [14]. However, SDL as well as other system-level specification languages, does not allow the specification of complex temporal behaviour, like reaction time or actions triggered at a specific point in time.

Temporal behaviour can be expressed by the use of events where an event marks a point in the specification. A simple requirement is that the interval of time between event e_1 (the signal from the crash sensor) and event e_2 (the activation of the airbag) must be less then 30 ms under any circumstances. Such a timing behaviour only uses a so-called *timing bound*. More complex requirements consider an additional *precondition*, e.g. send a data packet every 10 ms, but *only if a data packet is available*.

In real-time systems, the specified timing requirements have to be met. This requires a mechanism to quantify the progress of time which usually is carried out with a clock.

[☆] This work is supported by the DFG research program 'Rapid Prototyping of Embedded Systems with Hard-Time Constraints' under grant HE 1408/4-3.

[*] Corresponding author. Tel.: +49-441-9722-247; fax: +49-441-9722-282.

E-mail addresses: frank.slomka@informatik.uni-oldenburg.de (F. Slomka); rfmuenze@informatik.uni-erlangen.de (R. Münzenberger); doerfel@informatik.uni-erlangen.de (M. Dörfel); rhofmann@informatik.uni-erlangen.de (R. Hofmann).

[1] Tel.: +49-9131-85-27026; fax: +49-9131-85-27409.

0026-2692/03/$ - see front matter © 2003 Published by Elsevier Ltd.
doi:10.1016/S0026-2692(03)00167-8

As with all physical measuring instruments, the parameters of a clock deviate from their ideal values. The most important ones are drift, granularity and offset, and may vary with time. So, a specification technique should allow for specifying the parameters of the clock device, especially in a real-time environment.

Modern communication systems like GSM or UMTS even amplify time related design problems. Since they consist of several devices (the mobiles and the base station) they also implement several clocks. Even if these clocks are driven by crystal oscillators from the same production charge, they are not exactly the same. A specification technique that claims to be usable for real-time systems must allow to specify all necessary parameters of the clocks used. An alternative would be to support the designer in deciding how good the synchronization scheme needs to be in order to keep the different mobiles synchronized.

Reliable system design requires reproducible results for the major system properties. Thus, the properties of clocks for measuring time has to be specified with the same precision as the quality of voltage meters. All the related issues can only be handled with an adequate model of time that we present in Section 3. An overview of the possibilities of specification techniques currently used in the area of real-time systems with respect to their model of time is given in Section 2. Specific examples of how to integrate the presented time model in these specification techniques is shown in Section 4. Section 5 finally presents an application example of a large DECT fixed part to prove the usability of the formal time model.

2. Time in specification techniques

Several languages are being used for system modelling; some of them even have been extended for real-time systems. Though most of them are able to *implement* a real-time system, not all of them are really useful to *specify* the complex timing behaviour of the system.

Statecharts [6] allows the functional behaviour to be specified in a hierarchical order of concurrent finite state machines with synchronous communication. With the language construct *timeout* that controls state transitions, a simple temporal behaviour can be realized, but the specification of a timing requirement is not possible. Even the maximum allowed delay for a chain of state transitions cannot be specified in a formal way within the notation.

In the *Specification and Description Language* SDL [5], a system is composed of extended finite state machines that *asynchronously* communicate with each other via so-called *signals*. Activities related to time can be expressed by the timer construct. This is sufficient for timeout mechanisms, used in communication protocols. However, due to timer signals being treated like any other signals, real-time requirements cannot be formulated. SDL* [9], an extension of SDL, was designed in order to express further non-functional aspects like resource restrictions, cost constraints, mapping directives or timing requirements. In SDL*, only simple timing requirements can be given using events in the execution of the system.

Beside the specification languages mentioned before, techniques for specifying constraints on the temporal behaviour of systems exist. The *time constraint language TimeC* [8] is targeted at the control of code generation from C code by use of instruction level parallelism. With the extensions introduced by TimeC, timing requirements can be specified based on events. In Ref. [12] the TimeC concept was integrated into the specification language Esterel where it is used for automated validation. The *time constrained event language* TCEL [3] allows the specification of timing constraints in a structured manner, i.e. it is closely related with the structural constructs of modern programming languages. The expressiveness of both, TimeC and TCEL, is restricted to timing requirements that we call *optional*: timing requirements are formulated between pairs of events, if one of them is missing the timing requirement still appears as valid although actually the error of a missing event obscured the error of a timing violation.

The *real time logic* introduced by Mok et al. [7] can express a superset of the timing requirements possible with TimeC and TCEL, i.e. it also allows to specify requirements that we call mandatory: it is possible to specify that an event has to occur and when it has to occur.

All of the techniques discussed before lack an important feature: they cannot express complex preconditions like sequences of events that have to occur before a time constraint is considered relevant. None of them also considers the properties of existing clocks like drift and offset or the existence of multiple clocks as needed for a system comprising several distributed subsystems, e.g. a wireless communication system like GSM or UMTS.

3. Time model

Adding time to SDL as well as to any other event-oriented approach requires the association of time values with events. This section addresses the basic semantics of time and clocks related to the events happening in the system specified. The semantics is based on a mathematical model that describes the relation between time and real physical clocks in an abstract and formal way. This model will later be used for the extentions of specification languages like SDL.

3.1. Real-time and clocks

The measured value $C(t)$ of an event happening at time t is called the reading of the clock. In the following, mathematical models of clocks with different properties shown in Fig. 1 are discussed in detail.

An ideal clock $C_{\mathrm{i}}(t) \in \mathbb{R}$ shows the physical value without any deviation. It is defined by a linear function of