Predictability of a RTX2000-based Implementation

CHARLES ANDRÉ
Laboratoire Informatique, Signaux et Systèmes (13S)
Université de Nice-Sophia Antipolis / CNRS
41 Bd Napoléon III
F-06041 NICE cedex

MARIE-AGNÈS PÉRALDI
13S Lab., and also, Swiss Federal Institute of Technology, Lausanne
Industrial Computer Engineering Laboratory
EPFL-DI-LIT
CH-1015 Lausanne, Switzerland

andré@alto.unice.fr
peraldi@di.epfl.ch

Abstract. Any implementation should be proven to meet its specification. In this paper, we cope with the issue of checking the temporal correctness of a real-time program, implemented on a RTX microcontroller. Our real-time programs are first written in a high-level synchronous language (ESTEREL) and then, automatically translated into T-FORTH. Under these assumptions, the temporal behaviour of the generated RTX program can be predicted, at compile-time. In this paper, we present algorithms that compute realistic durations of system reactions. These algorithms use an abstract representation of the RTX program based on a behavioural semantics given to T-FORTH.

1. Introduction

Real-Time controllers are often used in critical environment, so they must perform their reactions to incoming requests without failure and on time. More generally, real-time applications require

- dependability
- predictability
- safety.

A modern approach to the design of real-time system must provide high-level “real-time” programming languages, safe and efficient compilers, and verification tools. Verification tools are necessary to prove that the implementation meets its specification. Both the logical correctness and the temporal correctness must be established. The semantics of the programming language and the quality of its compiler are determining factors in the logical correctness which is highly implementation-independent. On the contrary, the temporal correctness is implementation-dependent: The actual performances of the processor must be taken into account.

Recently, a book has been dedicated to predictability (Halang and Stoyenko, 1991). To guarantee predictable behaviour is a difficult task because it is a matter of both logical and
temporal correctness. In this paper, we address the issue of predictability for a restricted class of real-time system: Applications are almost pure reactive systems (mostly control, little data-processing) and microcontrollers are target processors. Several points in our experience are novel:

- The programming language: a synchronous one,
- The target processor: a non-conventional micro-controller,
- The formal characterization of the temporal behaviour of the implementation.

The microcontroller we use is the Harris' RTX2000 (Harris, 1988). It is a fast stack-oriented processor whose programming language is FORTH (more precisely a dialect: T-FORTH). We adopt the synchronous approach to real-time programming. This new approach has been advocated in an issue of the "proceedings of the IEEE" (IEEE, 1991). The introductory paper (Benveniste and Berry, 1991) of this issue explains why the synchronous approach improves real-time programming. Facilities for proving the logical correctness of synchronous programs are presented, too. Among synchronous languages, we have chosen ESTEREL (Boussinot and de Simone, 1991), an imperative synchronous language particularly suitable for programming event-driven applications. The implementation of an ESTEREL program on the processor is done by an automatic translator from ESTEREL into FORTH (André and Péraldi, 1992). In this paper we restrict our attention to one of the requirements of real-time system, which is the predictability. So, we explain how and why it is possible to guarantee the temporal behaviour of an implementation running on RTX2000 and programmed in ESTEREL. The intermediate language (FORTH) plays a central role in the verification process.

Since ESTEREL has a formal semantics, the logical correctness of a program can be proved using tools of the ESTEREL environment (AUTO, XSIMUL (Boussinot and de Simone, 1991)). We strive to preserve this correctness down to the target processor by an automatic code generation. In order to prove the correctness of the translation, the target language (i.e., T-FORTH) must be also endowed with a formal semantics. So, we propose an operational semantics for an abstract FORTH machine.

Besides the logical correctness, the temporal correctness must be checked, too. To ensure that a Real-time application acts in a predictable timely manner, the program must be analyzed at compile-time to determine whether or not it will meet all deadlines. This analysis is based on the proposed semantics. Algorithms are given that compute the duration of each transition (i.e., a reaction to a stimulus).

In order to explain these different aspects, we have organized this paper as follows:

- First, we give an overview of the method from the specification down to the verification of constraints. We present the steps of the compilation and we sketch the principles of our run-time system.
- We then point out the characteristics of our implementations. We explain why the predictability may result from a conjunction of several factors (software and hardware).