Microprocessors and Microsystems 32 (2008) 447–459

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

FPGA based tester tool for hybrid real-time systems

Jan Krákora *, Zdeněk Hanzálek

Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, Karlovo náměstí 13, Prague 2, 121 35, Czech Republic

ARTICLE INFO

Article history:
Available online 11 August 2008

Keywords:
Hardware-in-the-Loop
Real-time system testing
Model checking
Timed automata
FPGA
Hybrid system

ABSTRACT

This paper presents a design methodology for a hybrid Hardware-in-the-Loop (HIL) tester tool, based on both discrete event system theory, given by timed automata, and continuous systems theory, given by difference equations. It is implemented using an FPGA platform that guarantees speed enhancement, time accuracy and extensibility with no performance loss. We have focused on the implementation of a discrete event system, specifically timed automata into FPGA, and we have linked them with continuous systems implemented as filters in fixed point arithmetic. The paper shows a methodology, which employs widely used tools (Matlab, UPPAAL) as a user interface, and which implements the FPGA based tester tool.

1. Introduction

Hardware-in-the-Loop (HIL) applications are used by design and test engineers to evaluate and validate, e.g. vehicle components (electronic control units, etc.), during the development of new systems. Rather than testing these components in complete system setups, HIL allows the testing of new components and prototypes, so called Implementation Under Test (IUT).

Replacing the rest of the system by a model implemented in a computer (Tester tool) increases the flexibility and the rate of test scenarios. The physical components being tested respond to the simulated signals as if they were operating in a real environment. Therefore, they can not distinguish between the signals sent by other physical components and signals provided by models running on a computer.

This paper presents the design methodology of a tester tool. The objective of the tool is to check the behavior of the IUT while simulating the behavior of the controlled system. The tool has to be able to automatically analyze the behavior of the IUT and to vary the parameters of the system so that the IUT is forced to operate in different conditions. In most applications, the controlled system incorporates complex dynamics of physical nature, usually captured as a continuous change of continuous states on one hand and as complex dynamics of logic nature conveniently modeled by discrete states and events on other hand. Therefore, our tester tool is a hybrid, i.e. it is based on both the discrete event system theory, given by timed automata [3], and continuous systems theory, given by difference equations [23,24]. In order to avoid implementation errors, a high level of specification is required, so that the application expert can easily implement the system model, test cases and their analyses. The choice of such high level specifications, which are widely supported, enables us to execute a preliminary analysis (using Matlab/Simulink or UPPAAL) without incorporating any specific hardware, which simplifies the implementation of the tester tool.

Fig. 1 shows a setup of the tester tool. The system model block emulates environment interacting with IUT. It is given by the timed automata and by the difference equations. The tester block includes test cases and checks the behavior of the IUT. It is collection of timed automata executing a test and monitoring specified properties.

Simulation tools for continuous systems (like Matlab-Simulink) and model checking tools for discrete event systems (like UPPAAL) are often used during the analysis and design phases of hardware and software designs (e.g. [18,17]). Such model based designs usually lead to a modular structure, and the behavior of the modules is often analyzed separately in different tools, especially in the case of complex hybrid systems that are not tractable in polynomial time.

On the other hand, the testing phase of the model based design requires a compact solution in order to describe the complete system behavior. Therefore, in this article, we have followed this practice: we have assumed separate modules of the system, which are described and analyzed by appropriate tools (widely treated in the literature [7,16]) and we have used these models as parts of the hybrid tester tool.

Our tester tool is implemented by using the FPGA platform that guarantees, not only speed performance, but also time accuracy, and has quite good extensibility with no performance loss as well. Compared to the operating system based platforms, the FPGA platform is able to achieve a much faster sampling frequency.

* Corresponding author. Tel.: +420 224355710.
E-mail addresses: krakorj@fel.cvut.cz (J. Krákora), hanzalek@fel.cvut.cz (Z. Hanzálek).

0141-9331/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2008.07.003

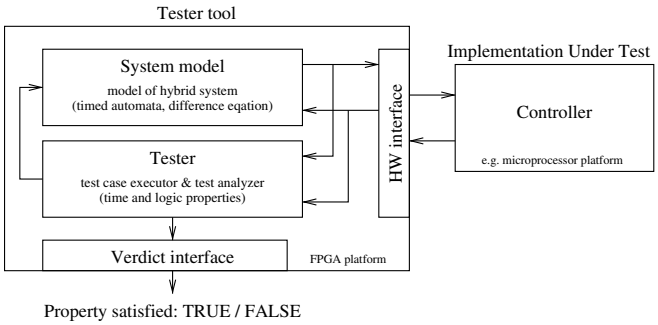

Fig. 1. Hardware in the loop conception.

Moreover, the FPGA platform is not affected by the rather complex behavior of the operating system services, interrupt handling, etc. In contrast to the typical sampling period of 10 μs achieved in real-time operating systems like RTLinux [11] or OCERA [25], a sampling period of less than 100 ns can be achieved on the FPGA platform. More importantly, is the fact that, the FPGA platform has zero jitter since it is synchronous HW, and the separate parts do not influence each other. On the other hand, the operating system based platforms are well supported by widely used development tools.

1.1. Background

The hybrid tester tool presented in this article combines a discrete event system, in the form of timed automata, and a continuous system in the form of difference equations.

Timed automata [3] are finite state automata, consisting of states (locations) and transitions, among these states. The transitions are extended by clocks that are used to specify quantitative time. Clocks are variables having non-negative real values, that can be reset. The difference between two clocks can be compared to a constant. In the initial system state, all clock values are zero, then all evolve at the same speed, synchronously with time.

The main feature of the timed automata approach, is that it can be verified using temporal logic [9]. It allows a user to check if any specification property of the model is satisfied or not. It is suitable in order to inspect the model due to deadlock, logical and timed conditions, for example.

The UPPAAL tool [5] used in this paper allows the user to design timed automata and to verify the automata using the mentioned temporal logic. Moreover, it offers an XML API for model export as well. The tool has been used as a design tool for timed automata that have been implemented into hardware.

Timed automata implementation into FPGA is inspired by [2]. The paper shows a way how to transform a timed automata (TA) into a program. The global time is represented by one running clock. For each clock of a timed automaton there is one integer variable (called digitized interpretation of TA). Such a variable is set to the value of the global time whenever the clock is reset. Consequently, the difference between this variable and global time represents the clock value. Each channel, used to synchronize the evolution of two timed automata, is replaced by one logic variable which is triggered synchronously with all other channels in the model. A zeno behavior (a model has zeno behavior if it can take an infinite amount of actions in finite time) is not supported in our FPGA implementation.

The continuous part [24] of the tester tool is realized using difference equations that are able to represent a continuous system with periodic sampling (called discrete-time systems). The transfer function $G(z)$ of such a system influences the physical layout of the arithmetic and storage units in the FPGA.

The continuous part may have the following influence on the discrete part: when the continuous state variable reaches a certain threshold, a state in discrete part changes. The discrete part may have the following influence on the continuous part: when the state in discrete part changes, the parameters of continuous part are changed.

The tester tool was implemented in high performance FPGA, where several arithmetic operations can be executed within a period close to tens of ns (Xilinx Virtex 4 XC4VFX12-F668-10C). As presented, a discrete event system, as a set of timed automata, is designed in UPPAAL. A continuous system is designed as a transfer function using the Matlab control toolbox. Both a discrete event and a continuous system are transformed into FPGA using the Xilinx System Generator, Matlab/Simulink toolbox (XSG). The toolbox can generate the final VHDL code for a target FPGA platform.

1.2. Related work

The related tools are briefly described in the following paragraphs, including comparison with our approach.

The TTG tool [14] is an off-line test generator (complete test scenarios and property results are computed a-priory and before test execution). The tool is based on the IF modeling language [8] and is able to generate a set of test cases for an IUT represented by a timed automata. Compared to our approach the TTG tool does not cover continuous systems.

The UPPAAL-TRON [20,15,12] is a tool for on-line testing (a single test primitive is generated from the model and at the time it is then immediately executed on the IUT) based on the discrete event system description using the timed automata. The tool both generates test cases to the IUT and checks test properties to be verified at the same time. The tool is executed on a UNIX platform, however, it can not guarantee the system time resolution of the FPGA. Moreover, it is not able to test an IUT using a hybrid system approach.

On the other hand, our tool only executes the test cases that are manually constructed, whereas TTG, UPPAAL-TRON and UPPAAL-COVER [12] generate the test cases automatically. In particular we assume the test cases to be based on the system requirements and to be written in the form of TA including fixed point representation of continuous variables.

SoftCom [26] is an industrial HIL application that simulates industrial environment activities to react with an IUT, i.e. a programmable logic controller (called PLC). The application is based on discrete event systems based on state machines. However, SoftCom is an application developed for a Windows OS or a non real-time Unix OS. The time resolution of these systems is approximately 100 ms, thus not very high. Moreover, it does not support hybrid system testing.

LabView FPGA [22] is a HIL application that allows a user to test the IUT running at a very high time resolution. The user can design their own HIL using the LabView environment supporting both discrete event and continuous systems. Such a HIL can be directly set into the FPGA. However, compared to our approach the tool misses the discrete event system timed automata representation, and for that reason, an interconnection with model checking tool is not straightforward.

Another tool, CarMaker [13] is used for testing vehicle embedded control systems. The system is used for the development of vehicle embedded control systems interacting with a discrete event and a continuous system environment as well. Nevertheless, CarMaker, similarly to LabView FPGA, does not support timed automata.