Reconfiguration and Hardware Agents in Testing and Repair of Distributed Systems

G. Moiş, I.Ştefan, Sz. Enyedi, L. Miclea
Technical University of Cluj-Napoca
{George.Mois, Iulia.Stefan, Szilard.Enyedi, Liviu.Miclea}@aut.utcluj.ro

Abstract

This paper proposes a new approach for distributed system testing and repairing using mobile hardware agents. This way, we obtain a networked reconfigurable system which does not need human intervention for maintenance and testing. The proposed architecture uses two FPGA boards and a microprocessor (agent host) as components and is flexible and re-programmable.

1. Introduction

Our times see the continuation of an exponential development in computer science and microelectronics, constant growth of systems' integration level, miniaturization and the development of increasingly complex and larger scale integrated digital systems. The maintenance and testing of such large systems is possible only by using two relatively new technologies: Mobile Data Agents and Real-Time Reconfigurable Systems. Therefore, it is very likely that in the future we will be surrounded by digitally-aware heterogeneous devices which will communicate to each other through distributed and wireless interfaces.

These emerging technologies, as well as the development of electronics used in heterogeneous systems, require new testing and fault elimination methods.

For these systems, classic and local testing and repair does not produce results at overall system level, due to the very large number of elementary subsystems and their heterogeneous nature. On the other hand, a *distributed, decentralized* solution is much easier to design, implement, maintain and utilize.

The growth of microelectronics integration level has brought, besides benefits, also the multiplication of transient and permanent faults, a fact that led to a decrease in reliability. Of course, there are manyoldand acknowledged solutions for counteracting this problem, but most of them require human intervention, or at least the intervention of an external factor. The solution discussed here accomplishes testing and fault removal without the need of these factors.

2. Agents

An *agent* is a piece of software capable of independent existence within an environment provided for it, is able to communicate with other entities, to unaidedly accomplish the work assigned to it and also to travel between geographically separate locations in its environment. It is an independent, mobile program capable of functioning for task execution in a flexible manner and in continuously changing surroundings.

Any entity that perceives its environment using sensors and takes action modifying those surroundings using actuators can be considered an agent.

A seemingly complete definition was given by J. Ferber (1989): "A real or abstract entity that is able to act on itself and its environment; which has partial representation of its environment; which can, in a multi-agent universe, communicate with other agents; and who's behavior is a result of its observations, its knowledge, and its interactions with the other agents." [1].

Distributed systems, especially ones consisting of a large number of modules or those distributed over a wide geographical area, are easier to monitor, test and maintain with the help of such agents.

It is desired that the testing and repairing actions to be brought to the hardware level by using the *Mobile Hardware Agent* paradigm [2]. Mobile Hardware Agents are software agents which can move together with their program executable code, data, and hardware components.

The mobile agent is not a new concept and can be viewed as a specific instance of the wider concept of *code mobility* [3]. The notion of the agent as a special software engineering paradigm has been somehow borrowed from the artificial intelligence domain, where high-level, theoretical aspects of autonomous agents has been deeply studied and discussed since the mid-Eighties [4]. There, the main aim was to model rational and autonomous entities that are able to recreate some aspects of the human intelligence. In the specific field of software engineering, agents are computer programs that can act on behalf of humans [5] and have the attributes of *autonomous behavior*, *proactivity*, *reactivity* and *adaptivity*[4]. Coupled with the newly emerged idea of code mobility, the notion of agents has lead to the now well-established concept of mobile agents defined as software components possessing the attribute of mobility and the ability to "move from host to host *of their ownvolition*" [6].

In practical scenarios, agents have been useful in the area of distributed and mobile computing [7] and have significantly improved performance of several distributed applications such as distributed data mining [8], information retrieval [9] and web services [10]. The use of *Mobile Data Agentstechnology* in a system appears as the most innovative, powerful, less expensive, most flexible and scalable choice. Mobile Data Agents guarantee the possibility of observing the system behavior, evaluate the constraints imposed by the external environment, and apply appropriate corrections, with the minimum overhead.

The most appealing and breakthrough advantages of a system that uses reconfiguration and mobile agents for testing and repairing will include: lower global costs, reduced on board hardware and software, knowledge-based, extension to different classes of systems or abstraction level.

In our case, the agents travel from device to device, try to detect and repair errors. They can also gather "experience" through their work.

3. Reconfigurable System

This article also focuses on the use of reconfigurable systems and tries to find a viable and cost efficient solution for the implementation of such systems. Hardware self-maintenance is a relatively new method and implies that the defective device or the device that is about to be out of order "takes notice" and recovers from this state. Usually, this can be realized through a surplus or a duplication of the present functions, in hardware, and through the use of local software or the use of decision electronics for testing and/or deactivating the defective part and reallocating the tasks to the spare part of the device.

There are many solutions for the decision logic and the replacement of the tasks. If the functionality is implemented in an FPGA, in case of a fault, an unused part of the matrix can be reconfigured, taking over the functionality of the defective part [11,12]. For better results, one can use a method for detecting the faulty parts which is similar to the human immune system: the cells found to be defective are replaced with spare ones that take over their functions [13].

The key feature of a reconfigurable system is the ability to perform computations in hardware to increase performance, while retaining much of the flexibility of a software solution. Like software, the mapped circuit is flexible, and can be changed over the lifetime of the system or even at run-time. In this case the performance is boosted even more, since the circuits can be optimized based on run-time conditions. In this manner, the performance of a reconfigurable system can approach or even surpass that of an ASIC.

As a positive fact, reconfigurable computing allows product differentiation, one of the driving forces of embedded system implementations beside time-to-market, performance and costs. In this aspect, reconfigurable computing solutions have a clear advantage over most of the alternatives what could help to establish reconfigurable systems in the future marketplace.

In our research, we tried to achieve system reconfiguration by reprogramming an entire FPGA in case a critical part of the system is found to be defective.

4. Experimental Setup

In this section we present the experimental setup of a simple reconfigurable system. It represents the experimental proof-of-concept of a distributed system that incorporates the *reconfiguration* and *mobile data agents'* technologies. The prototype we built is oriented towards a networked infrastructure and is essentially aimed at exploring a particular architectural pattern based on a centralized agent host (a microprocessor) and two low-end peripheral nodes with reconfigurable capabilities. This organization is representative for a large class of distributed systems and could be used to model, for example, a distributed control application. The nodes are very resource-constrained embedded systems provided with reprogrammable hardware (FPGA). The Mobile Agent System is split into two components, a common software layer included in the central host and some low-level local routines installed on each peripheral node.

We used two XUP V2P Xilinx FPGA boards which communicate with each other and with the computer through a wireless interface using Wi-Fi modules.

One of the boards is programmed to act as a microprocessor in charge of controlling a process. This part has a high importance in our experimental system, because if this microprocessor is defective, the entire system is unable to correctly execute its tasks.

The other FPGA board is programmed to act as a video board for displaying the parameters of the controlled process. This is the part of the system that we can give up in case we need a programmable spare.

The agent host is the supervisor and, collaborating with the agents, verifies periodically if the two other components are working without any faults. In fact, the supervisor waits for a message from the mobile agent that "lets it know" whether the two other parts of the systems work correctly. Also, it is continuously backing up the microprocessor's state. The mobile hardware agent scheme was introduced in the project for pushing the concept of testing and repairing using mobile agents to the hardware level.

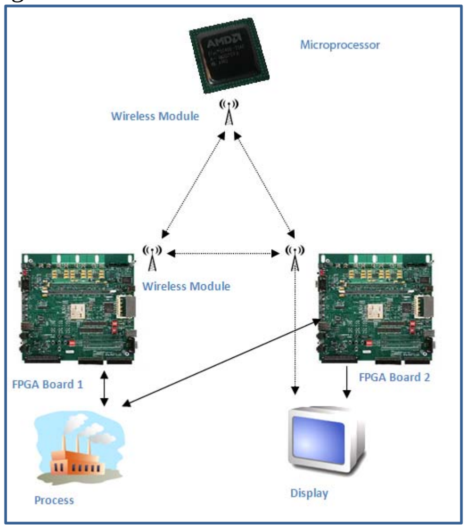

Figure 1. Experimental setup

The program that tests the correct functionality of the two programmable parts of the system is a mobile hardware agent. Through a wrapper, it uploads its code to the microprocessor board and to the display controller board in order to check their behavior. The agent runs a series of test routines to see if the two parts produce the expected data output. If not, it sends a notice message to the supervisor which will send the required bitstream through the wireless modules to reprogram the FPGAs. The FPGAs can be reprogrammed via SlaveSerial or SelectMAP Modes, and depending on the number of general purpose I/O pins that the host microprocessors has, the use of a CPLD can be needed [14]. This embedded processor-based configuration solution reduces board real estate requirements in a distributed system, assuming that the embedded host processor has sufficient memory.

If the display board is not working properly, the entire system can execute its tasks without any immediate intervention, but, if the microprocessor is damaged, the process is endangered.At this point, the entire system is reconfigured,reprogramming the video

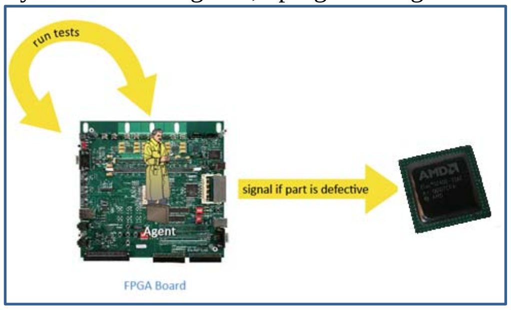

Figure 2. Hardware agent running the tests

board to act as an identical backup microprocessor. This new microprocessor has the last valid values of the old one written in its registers, for controlling the process. The old microprocessor is no longer used and the one the supervisor has just programmed takes its place in the system.

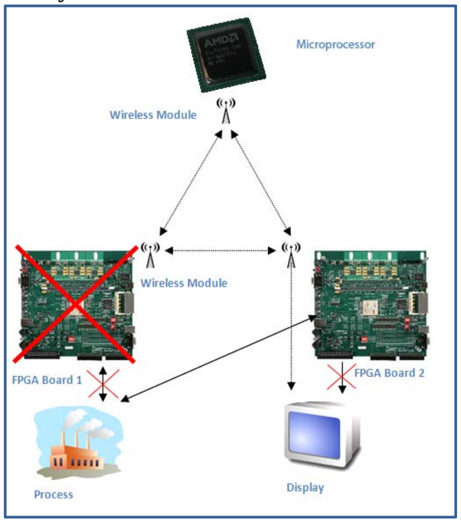

Figure 3. Reconfigured system structure

The experimental setup proves that the concept can be implemented and can be used to solve problem situations without needing human intervention.

5. Conclusions

Modern trends in developing digitally-aware environments will increasingly lead to scenarios where ubiquitous and heterogeneous digital systems interact together through distributed and wireless interfaces, and provide high productivity and great flexibility. The heterogeneity and the ever-growing complexity of such scenarios will raise many new aspects and issues (including, for example, transient faults, unpredictable working contexts, emerging security flaws and threats, interoperability issues) that may affect the system functionality and health, and might become the next toughest challenges for the IT industry and research community. Two technologies are emerging as promising responses to such issues: Mobile Data Agents, to handle the complexity and heterogeneity of networked infrastructures, and Real-Time Reconfiguration, to implement flexible, adaptable, and high performance digital systems.

In this paper, we analyzed how the innovative aspects of reconfigurable systems technology and hardware agents can be exploited to implement efficient and new test and repair strategies.

We also provided an experimental proof-of-concept of a self-healing system with the capability to host and interact with mobile hardware agents. Specifically, we explored a particular architectural pattern based on a centralized agent host and two low-end peripheral nodes. The number of nodes can be increased leading to an architecture which is representative for a large class of distributed systems and could be used to model distributed control applications.

Our experimental proof of concept setup uses an embedded processor as supervisor and reconfiguration manager for the hardware agents and reconfigurable systems, but the use of another FPGA board for this job is planned. This would give the system even more flexibility.

10. References

[1] J. Ferber 1989. *Des Objets aux Agents*. Unpublished Doctoral Dissertation, University of Paris VI.

[2] A.Benso, Sz. Enyedi, L. Miclea, *Intelligent Agents and BIST/BISR – Working Together in Distributed Systems*, IEEE International Test Conference, Baltimore (Maryland), October 2002.

[3] A. Fuggetta, G. P. Pico, G. Vigna, *Understanding Code Mobility*, IEEE Transactions on Software Engineering, vol. 24, n. 5, pp. 342-361, May 1998.

[4] M. Wooldridge and N. R. Jennings, *Agent Theories, Architectures and Languages: A Survey*, Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architecture and Languages, pages 1-32, 1995.

[5] H. S. Nwana, *Software Agents: An Overview*,Knowledge Engineering Review: Intelligent Systems Research, AA&T, BT Laboratories, UK, pages 206-244, 1996.

[6] S. Fischmeister, and W. Lugmayr. *The Supervisor-Worker Pattern*, Pattern Languages of Programs (PLoP'99) (Allerton House, IL, August 15-18, 1999).

[7] D. Kotz, R. Gray, and D. Rus, *Future Directions forMobile-Agent Research*, IEEE Distributed Systems Online, vol. 3, n.8, August 2002.

[8] S. Krishnaswamy, A. Zaslavsky, and S.W. Loke, *Techniques for Estimating the Computation and Communication Costs of Distributed Data Mining*, Procs. of International Conference on Computational Science (ICCS2002) - Part I, Lecture Notes in Computer Science (LNCS) 2331, Springer-Verlag. pp. 603-612, 2002.

[9] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko and D. Rus, *Mobile Agents for Distributed Information Retrieval*. In Matthias Klusch, editor, Intelligent Information Agents, chapter 15, Springer-Verlag, 1999.

[10] A. Padovitz, S. Krishnaswamy, and S.W. Loke,*Towards Efficient and Smart Selection of WebService Providers Before Activation*, the Workshopon Web Services and Agent-based Engineering (WSABE 2003), Melbourne, Australia, July, 2003.

[11] A. Benso, A. Cilardo, N. Mazzocca, L. Miclea, P. Prinetto, Enyedi. Sz., *Reconfigurable Systems Self-Healing using Mobile Hardware Agents*, Proceedings of International Test Conference 2005, Austin, TX, USA, November 8 - 10, 2005.

[12] S. Habermann, R. Kothe, H. T. Vierhaus, *Built-in Self Repair by Reconfiguration of FPGAs*, Proceedings of IEEE International On-Line Testing Symposium, 2006, ISBN 0-7695-2620-9/06.

[13] P. K. Lala, B. Kiran Kumar, *An Architecture for Self-Healing Digital Systems*, Proceedings of IEEE International On-Line Testing Workshop, 2002, ISBN 0-7695-1641-6/02.

[14] Mark Ng and Mike Peattie, *XAPP502 (v1.5)*, www.xilinx.com, December 3, 2007.