# A Reconfigurable Hardware-Monitor for Communication Analysis in Distributed Real-Time Systems

Andreas Kirschbaum and Jürgen Becker and Manfred Glesner

Darmstadt University of Technology, Institute of Microelectronic Systems
D-64283 Darmstadt, Germany

**Abstract.** The importance of interprocess communication increases in recent embedded system designs. In particular this is true for distributed real-time systems, where mainly communication and synchronization cause violations of timing constraints. In this paper we present *HarMonIC*[1] – a reconfigurable hardware monitoring system for the real-time observation of interprocess communication architectures. We will show that reconfigurability is essential for the use within a rapid prototyping environment like *REPLICA*[2]. It will be demonstrated how *HarMonIC* can be used to significantly improve debugging, performance evaluation, and design space exploration of distributed real-time systems.

## 1 Introduction

Nowadays complex systems are often not built from scratch but are assembled by reusing previously designed modules or off-the-shelf components such as processors, memories or I/O circuitry in order to cope with more aggressive time-to-market constraints. A lot of effort in the design process is spent on interfacing existing system modules because communication becomes a bottleneck in many embedded real-time systems. Appropriate interface circuits are either hand-crafted or synthesized with automatic communication-synthesis algorithms [4] [14] [12] [3]. Those circuits often have to be fine-tuned with respect to parameters, e.g. bus width, buffer depth, synchronization/arbitration schemes etc., or even different implementation alternatives have to be evaluated. Thus, in order to achieve a good communication architecture the designers have to rapidly explore many more points in the design space than current prototyping systems allow.

Additionally, real-time systems differ from non-real-time systems because of the timing constraints imposed on them, i.e. they have to meet logic *and* timing requirements. The most likely cause of missing timing constraints is overhead due to poor or improper interprocess communication and synchronization [11]. Since debugging of real-time systems has to take into account the behavior of the target system as well as it's environment, runtime information is extremely important. Therefore, static analysis with simulation methods is too slow and not

[1] Hardware Monitor for Interprocess Communication
[2] Realtime Execution Platform for Intermodule Communication Architectures

sufficient. A prototyping system focusing on *realistic* emulation of intermodule communication with an integrated hardware monitoring facility will significantly improve the design and debugging process, as we will describe in this paper.

A monitor observes a target system during its normal operation and collects runtime information through different forms of instrumentation techniques [10]. The obtained data provides accurate information for e.g. system testing and debugging, dynamic task scheduling, performance analysis, or architecture optimization. The special constraint of monitoring distributed *real-time* systems is to keep the interference with the target system so small, that the change in system performance due to the monitor will neither effect the order nor the timing of events. Monitors may be classified into hardware [1], hybrid [5], and pure software monitoring approaches [2]. The interference of the monitor with the target system is increasing analogous to the order of approaches just mentioned.

In section 2 we describe the architecture of our reconfigurable prototyping system *REPLICA* in which our proposed hardware monitor is integrated. The concept and implementation of the hardware monitor *HarMonIC* is presented in section 3. Finally, some conlusions are drawn in section 4.

## 2 REPLICA – The Prototyping System

*REPLICA* [8] is a Rapid-Prototyping System focusing on realistic inter/-processor-module communication emulation. The system is integrated in a design environment for embedded mixed hardware/software systems (*DICE*) [4]. *REPLICA* facilitates design space exploration and validation of communication links. The reprogrammable system architecture allows prototyping of different topologies, communication types and protocols and is supported by a powerful toolset for automatic system configuration. The integrated hardware monitor (*HarMonIC*) presented in section 3 extracts real-time data about I/O-channel activities for performance evaluation and enhancement of arbitrary communication channels within the system.

*REPLICA* is based on a scalable and reconfigurable system architecture. As depicted in Figure 1, a minimal system, which is also referred to as a cluster, consists of a backplane *BP*, up to six processing modules *PM*, and optionally an interface module *IM* between each processing module and the backplane. Each port of the backplane can also be used as a gateway *GW*. Several clusters may be combined via gateways if the amount of modules to be connected exceeds the capacity of one cluster.

The backplane can be configured for different interconnect topologies [13]. A non-blocking switch matrix has been chosen as interconnection device. During the prototyping step this device imposes no restrictions on the topologies which can be implemented and elaborated. The final application system will of course only comprise the optimized communication architecture and therefore will not suffer from the high interconnection costs introduced by the switch matrix. Each backplane provide six 106 bit wide symmetric ports which are interconnected with two SRAM-based bit-oriented FPID[3] switches [7]. All pro-

[3] Field Programmable Interconnection Devices

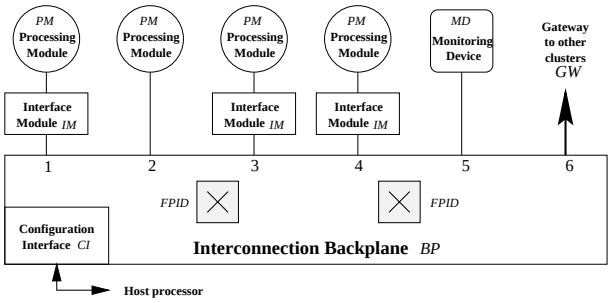


**Fig. 1.** Configuration Example of *REPLICA*.

grammable devices in *REPLICA* located on the backplane and the interface modules are configured via a unique host processor interface (*CI*).

The communication type and the protocol of the communication links to be prototyped may require additional hardware resources such as memory, glue logic, synchronization circuit etc. These components will be mapped on the interface modules (Fig. 2).

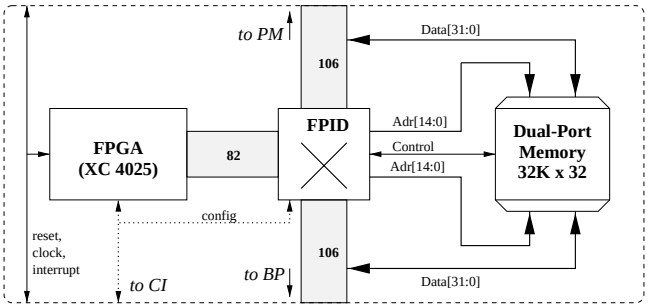


**Fig. 2.** Architecture of the Interface Module.

*REPLICA* comprises the following *PMs*: a 32-bit floating point signal processor PC-board with two TMS320C40 signalprocessors, a 32-bit microcontroller board based on a PowerPC505, and an FPGA-based hardware emulator with a capacity of 50K gates.

## 3 A Reconfigurable Hardware-Monitor

In this section we discuss *HarMonIC* – the reconfigurable hardware-monitor concept used in *REPLICA*. *HarMonIC* comprises an electronic device physically

connected to specific points of the target system which is currently prototyped in the *REPLICA* environment. As shown in section 2, all signals belonging to the interprocess communication of the target system will be available at the routing devices within *REPLICA*. *HarMonIC* can access the signals via these crossbar switches at a physical level without changing the target systems behavior and performance (non-intrusive). Additionally, the monitoring system has to be reconfigurable because the points of observation as well as the target system itself may vary. This approach overcomes the lack of flexibility which is normally inherent in hardware monitors, and thus improves the usability of *HarMonIC* significantly. The main tasks of *HarMonIC* are the observation of the target system during its normal operation, as well as the collection and the analysis of runtime data.

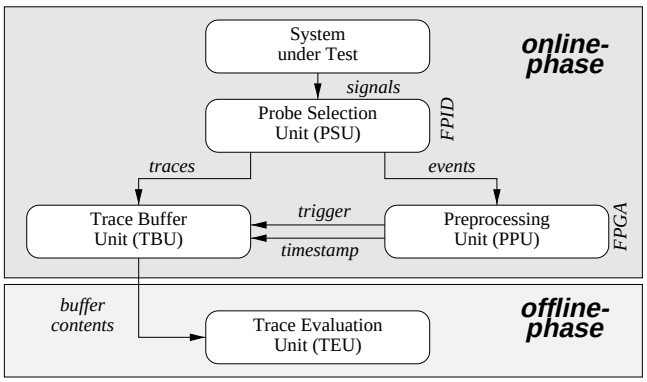


**Fig. 3.** Basic Building Blocks of *HarMonIC*.

Our monitoring concept consists of two main phases. In a first phase named *online-phase HarMonIC* collects and stores data of the target system in real-time, i.e. synchronously to the systems clock frequency. All components used in this phase have to fulfill stringent timing constraints that no real-time event is missed. Therefore they are completely designed in hardware but still reconfigurable to adopt to different target systems. In the second phase the sampled data is analyzed and rated. There is no need for real-time in that phase because the input data has already been captured in a buffer. As a consequence, all components of this *offline-phase* are purely implemented in software what also increases the flexibility of the analysis. Figure 3 shows the basic building blocks of the reconfigurable *HarMonIC* system.

The target system which is to be monitored interfaces with *HarMonIC* via the probe selection unit *(PSU)*. As target systems change in a prototyping system like *REPLICA*, the *PSU* is reconfigurable to adopt to arbitrary system architectures. The reconfigurability of the *PSU* is based on programmable FPID devices [7] which are used to physically connect to the (real-time) target system

without causing additional delays. This conceptional approach allows a non-intrusive monitoring of arbitrary target architectures, which is not possible with common prototyping systems. Thanks reconfigurability *HarMonIC* still provides flexibility at the high speed of a hardware-monitoring device.

Probed signals are either interpreted as real-time traces or used for event detection. Real-time traces characterize the target system state during a time-frame. Given the sequence of sampled signals of the target system as well as their chronological order, the system state can be replayed deterministically after execution. Real-time traces need no further online processing and are therefore directly forwarded to the *Trace Buffer Unit (TBU)* which also stores detected events, event counters, and the global timestamp.

The memory for storing real-time traces must be large enough and fast enough to store all generated information. However, such data collection would demand excessive storage capacity. Since buffer size is limited, methods for data reduction have to be employed. To reduce the amount of generated data, *HarMonIC* records only significant signals at dedicated timepoints. Events are used to determine when data is sampled. In this context an event denotes entering or leaving a target system state. Data is sampled either in between two events or only at the occurrence of an event.

The reconfigurable *Preprocessing Unit (PPU)* generates a trigger signal for the *TBU* and updates an internal counter whenever an event occurs. The trigger signal is derived from edge detection circuitries within the *PPU*. Up to 30 different user-defined events can be detected and counted with the *PPU*. The monitoring system has to be fast enough to detect all events. Therefore the target system clock is derived by downsampling of the monitoring clock. A programmable clock generation unit in the *PPU* scales an arbitrary clock source and generates the monitoring clock as well as the target system clock. Because of the non-continuously sampling of data we additionally need a global time information if we are not only interested in the occurrence of events but also in the duration of system states. The *PPU* generates a global 32-bit timestamp which allows long-term measurements at the resolution of the monitoring clock. The timestamp is forwarded to the *TBU* and is always stored together with the real-time trace. The *PPU* is completely implemented in an FPGA and is always adopted to the target system and the respective monitoring task.

The *Trace Buffer Unit (TBU)* stores the real-time traces and the timestamps during the execution of the target system. The main building block of the *TBU* is a logic analyzer. It provides a sufficient buffer depth and sampling rate to be used within *HarMonIC*. Built-in functions like pattern recognition and state sequencing allow the definition of complex trigger conditions. This trigger conditions complement the event detection capabilities of the *PPU* and further reduce the necessary storage capacity of the monitor.

The *Trace Evaluation Unit (TEU)* is a software package for the analysis and rating of sampled data. The modular software package comprises a set of tools for data filtering and different analysis methods [9]. Included tools allow the statistical analysis of the real-time trace e.g. counting of event occurrences for the calculation of absolute/relative frequency of events. Also the violation of real-time constraints may be checked.

## 4 Conclusions

In this paper we have presented a hardware monitoring approach for interprocess communication analysis in distributed real-time systems [4]. The importance of process communication and synchronization, especially for real-time systems with timing constraints, has been sketched. A reconfigurable hardware monitoring system (*HarMonIC*) has been presented in detail. *HarMonIC* imposes minimal intrusion on the target system, i.e. neither function nor timing will be changed, while still providing enough flexibility to adopt to arbitrary target systems. It has been shown how the monitor is integrated into the rapid prototyping environment *REPLICA* which focus on the *realistic* emulation of communication architectures. The combination of prototyping system and integrated hardware monitor allows the real-time observation, analysis and rating of interprocess communication architectures, which is not feasible with a simulation approach due to it's limited speed. This improves debugging capabilities, performance evaluation and design space exploration significantly, what in turn will result in better product quality and shorter design time.

## References


1. W.C. Brantley, K.P. McAuliffe, and T.A. Ngo. *RP3 Performance Monitoring Hardware*, pages 35–47. ACM Press, New York, 1989.
2. S.E. Chodrow, F. Jahanian, and M. Donner. Run-Time Monitoring of Real-Time-Systems. In *Real-Time Systems Symposium*, pages 74–83, Los Alamitos, USA, 1991. IEEE CS Press.
3. R. Ernst and T. Benner. Communication, constraints, and user-directives in cosyma. Technical report, Technical University of Braunschweig, 1994.
4. M. Gasteier et al. An Interactive Approach to Hardware/Software Co-Design. In *International Workshop on Logic and Architecture Synthesis*, pages 211–218, Grenoble, France, December 1996.
5. D. Haban and D. Wybranietz. A hybrid monitor for behavior and performance analysis. *IEEE Trans. Software Eng.*, 16(2):197–211, Feb. 1990.
6. H.-J. Herpel et al. Real-Time System Prototyping Based on a Heterogeneous Multi-Processor Environment. In *5th Euromicro Workshop on Real Time Systems*, pages 62–67, Oulu, 1993.
7. I-Cube Inc. *IQX Family Data Sheet*, 1996.
8. A. Kirschbaum and M. Glesner. Rapid Prototyping of Communication Architectures. In *IEEE Workshop on Rapid System Prototyping*, pages 136–141, Chapel Hill, USA, June 1997.
9. B. Mohr. *SIMPLE – User's Guide Version 5.3.* University of Erlangen, 1992.
10. B.A. Schroeder. On-Line Monitoring: A Tutorial. *IEEE Computer*, pages 72–78, June 1995.
11. J. Tsai and S. Yang. *Monitoring and Debugging of Distributed Real-Time Systems.* IEEE Computer Society Press, 1995.
12. B.P. Upender and P.J. Koopman Jr. Communication protocols for embedded systems. *Embedded Systems Programming*, 7(11):46–58, November 1994.
13. A. Varma and C.S. Raghavendra, editors. *Interconnection Networks for Multiprocessors and Multicomputers – Theory and Praxis.* IEEE Computer Society Press, 1994.
14. T.-Y. Yen and W. Wolf. Communication Synthesis for Distributed Embedded Systems. In *International Conference on Computer Aided Design*, pages 288–294. IEEE Computer Society Press, 1995.



[4] This research is supported by the german research foundation *Deutsche Forschungsgemeinschaft (DFG)* under grant Gl 144/14-1