COMPONENT CASE STUDY OF A SELF-OPTIMIZING RCOS/RTOS SYSTEM

*A reconfigurable network service**

Björn Griese, Simon Oberthür, Mario Porrmann
Heinz Nixdorf Institute, University Paderborn
Fürstenallee 11, 33102 Paderborn, Germany
bgriese@hni.upb.de, simon@oberthuer.net, mario@hni.upb.de

Abstract In highly dynamic scenarios a real-time communication/real-time operating system (RCOS/RTOS), which can fulfill all upcoming demands of the application, is normally very extensive. These RCOS/RTOS systems are heavy-weighted and produce much overhead. System resources for an application or a system service are often reserved for worst-case scenarios and are not usable for other applications. We present a self-optimizing RCOS/RTOS with an integrated flexible resource management. Our RCOS/RTOS adapts its services to the application demands and redistributes temporarily unused resources to other applications under hard real-time conditions. The benefit of our system is shown by means of a self-optimizing communication service. The main building block of this communication service is a reconfigurable dual-port Ethernet switch. Using dynamically reconfigurable hardware to implement the switch enables an adaption of the switch to changing requirements during run-time.

Keywords: RCOS, RTOS, self-optimization, resource management, Ethernet switch, dynamic reconfiguration

1. Introduction

Increasing complexity is one of the major problems in today's automotive industry. To deal with this complexity an approach is to build mechatronic systems in a self-reflecting, self-adaptive and self-optimizing way. Such self-optimizing applications have highly dynamic resource requirements and versatile demands to system services. To fulfill every upcoming demand a standard operating and communication system would have to be very extensive

*This work was developed in the course of the Collaborative Research Center 614 - Self-Optimizing Concepts and Structures in Mechanical Engineering - Paderborn University, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

and would have a high overhead. We have developed a dynamic real-time communication/real-time operating system (RCOS/RTOS) for such systems to fulfill the upcoming demands efficiently. The RCOS/RTOS optimizes the infrastructure of the services and redistributes the released resources to the applications. Even resources of an application, which are temporarily unused but guaranteed (e.g. for error recovery), can be temporarily provided to other application under hard real-time condistions.

In this paper we give an overview of the self-optimizing concept in our RCOS/RTOS in section 2. This includes the mechanism of getting necessary information from the applications about their upcoming resource requirements, the modelling of the reconfigurable system services and the optimization of these services and of the system resources. Subsequently, the concepts are illustrated considering as an example a concrete reconfigurable system service: A reconfigurable RCOS service.

The basis of our reconfigurable RCOS are dual-port network nodes. Each node consists of two external network interfaces that connect the node to its neighbors and an internal interface to an embedded processor. In order be able to adapt the network nodes to changes in protocols and interface requirements, which can not be foreseen, we use reconfigurable hardware for the implementation of the network interfaces.

The nodes handle two different types of data streams: data originated from or terminated at the processor and streams that are simply passed through. If network traffic is rather small or if real-time requirements are low or even nonexistent, comparatively simple network interfaces are sufficient, which occupy only a few resources. In this case, data packets are forwarded from one port to another by a software implementation on the embedded processor. This causes a high load for the processor, the internal bus and the memory while the FPGA resources can be utilized by other applications. If the software implementation is not able to deliver the required performance, e.g., due to increasing bandwidth or real-time requirements, the two separate interfaces are substituted by a single integrated hardware switch during run-time. This switch is able to forward data packets autonomously and, as a consequence, manages a much higher amount of traffic. However, the structure of this switch is more complex and requires additional FPGA resources, which are no longer available for other applications.

A prototypical implementation of the dynamic RCOS/RTOS has been realized on the RAPTOR2000 System, a Rapid Prototyping System developed at the System and Circuit Technology research group at the University of Paderborn [5]. Xilinx Virtex-II Pro FPGAs are used, which comprise two embedded PowerPC processors. The Ethernet switch is implemented either in software or in hardware, as detailed in section 3.

Then we show an example scenario (section 4) in which the benefit of a self-optimizing RCOS/RTOS system with this RCOS component is shown.

2. Self-optimizing RCOS/RTOS system

For building an RCOS/RTOS system, which optimizes itself and provides the released resources optimally to the applications, we had to answer the three following questions:

1. How can we get information about the dynamic resource requirements during run-time of an application? A suitable interface between the application and the operating system is required.
2. How can we describe the design space for reconfiguration of the RCOS/RTOS system? A model is required that describes the dependencies between the system services.
3. Based on this information: Which is an adequate system configuration? A resource management system is required to activate/deactivate the system services and to release resources for the applications.

Our self-optimizing RCOS/RTOS system consists of three components (illustrated in figure 1), which deal with the questions mentioned above: Profile Framework, Online TEReCS and Flexible Resource Management (FRM).

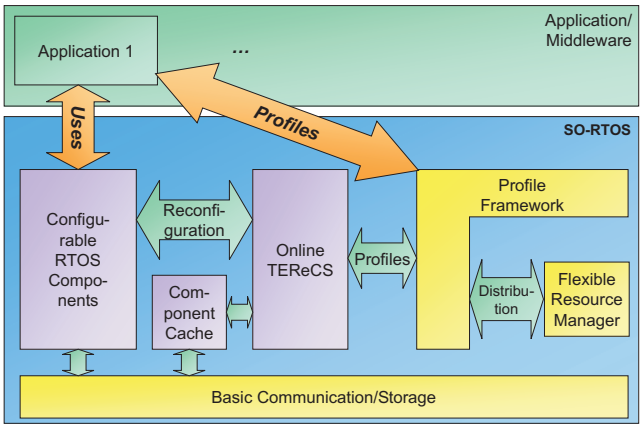

Figure 1. Overview of the self-optimizing RCOS/RTOS system

2.1 Profile Framework

The Profile Framework, described in detail in [9], has two main purposes: The first purpose is to get information about the dynamic resource requirements of the application and of the system services. The knowledge about

the resource requirements of the applications is necessary to optimize the system by deactivating services that are currently not required. The resource requirements of the system services are necessary to determine the amount of resources, which can be released and to guarantee that applications and system services get their required resources under hard real-time condistions. The second purpose is to provide released resources from deactivated system services to the applications.

By means of this framework the developer can define a set of profiles per application. Profiles describe different *service levels* of the application, including different quality and different resource requirements. Furthermore, the states of deactivated/activated system services are mapped to profiles by Online TEReCS.

A single profile contains the following information:

Resource requirements: The *maximum* and *minimum resource usage* per resource of the application if the profile is active.

Maximal assignment delay: All resource allocations of an application require an announcement to the FRM. The maximal delay is the worst case time the assignment of the resource can be delayed by the FRM from the announcement until its allocation.

Switching conditions: The information, between which profiles a switching is possible, and the worst-case execution times (WCET) for activating and deactivating the profiles.

Profile quality: By means of this value the profiles of an application can be ordered according to their quality. Thus, the FRM knows, which profile to prefer when trying to increase the system quality by selecting a profile for activation.

2.2 Online TEReCS

On the basis of the offline configurator TEReCS (**T**ools for **E**mbedded **Re**al-Time **C**ommunication **S**ystems) [1] an online configurator Online TEReCS has been developed for our customizable component based RCOS/RTOS library DREAMS (**D**istributed **R**eal-time **E**xtensible **A**pplication **M**anagement **S**ystem) [3].

The Online-TEReCS component has knowledge of the design space of the customizable components, maps system services into profiles, and is responsible for the low-level reconfiguration of the system services. The knowledge about the design space of the RCOS/RTOS components is stored by the RCOS/RTOS system developer in an AND/OR service dependency graph [2]. For the online case of the configuration a coarse grained hierarchy of this graph is used on the system service level – one service can be composed of many

components. Thus, the overall decision space is more restricted and the number of generated profiles is easy to manage.

Each service that is provided in the system is treated as a resource of the system and is managed by the FRM. For a system service, which can be in a deactivated or in an activated state, two profiles are created. One profile for the activated and one profile for the deactivated state. In the profile for the activated state the service specifies the required resources to fulfill its task. In the profile for the deactivated state the service occupies the resources it provides in the activated state. Hereby, it is guaranteed that the services can only be deactivated if the provided services are not required.

2.3 Flexible Resource Manager

The major goal of the FRM [9] is to optimize the resource utilization and the over-all system quality by selecting profiles for activation under the actual conditions. To maximize the utilization the FRM puts the resources, which are held back for worst case resource requirements of an application, at other application's disposal. Normally, an application acquires as much resources as it requires for worst-case scenarios. Thus, the application has always enough resources for its tasks and can fulfill its service at any time. These resources are only required when the worst-case scenario occurs. In our approach the FRM tries to minimize this internal waste of resources, by making these resources temporarily available to other applications under hard real-time condistions.

The FRM is responsible for switching between the profiles of the applications and system services under the defined switching conditions. The decision is made based on a quality function, which considers the actual resource requirements, the importance of an application, and the quality of the profiles. To guarantee hard real-time conditions, an acceptance test for profile activation is integrated in the optimization process.

3. Reconfigurable communication service

The dynamically reconfigurable Ethernet switch consists of a software and a hardware switch. The software switch comprises two Ethernet Media Access Controllers (MACs), which are connected to the system bus of the network node. For each port of the switch, one Ethernet MAC is required. The switching decision is made by the processor. The whole Ethernet traffic is transferred over the system bus, which results in a high processor and system bus load.

In contrast to the software switch, the switching decision of the hardware switch is carried out in the internal logic of the switch. Every packet that is not destined for the processor is processed and forwarded to the calculated output port by the hardware switch. Therefore, the processor and the system bus are released from performing network infrastructure tasks.

The hardware switch consists of two independent cross-connected sub switches, which are connected to the system bus. The sub switch component is an extension of the Ethernet MAC mentioned above. It integrates an additional buffer for receiving and forwarding the packets that are not destined for the processor. One sub switch component is able to forward packets in one direction. Because of this hierarchical structure of our switch, the Ethernet software driver for both the Ethernet MAC and the hardware switch are the same.

To comply with the real-time requirements of the network service it is important to maintain the connection to the neighboring nodes without loosing packets, while the network interface is reconfigured. Hence, we have introduced a method to reconfigure network interfaces without loosing packets [12]. The reconfiguration strategies are based on a parallel instantiation of both switch implementations, which allows a fast transition from the software switch to the hardware switch and vice versa. To hand over interface control from the current configuration to the new configuration we use the Inter Frame Gaps (IFG) of the applied Ethernet protocol. Because the IFG does not appear simultaneously on both directions the transmit process and the receive process can be switched separately by the hardware multiplexer, which controls the access to the Media Independent Interface (MII).

The reconfiguration from a software switch to a hardware switch is shown in figure 2. The dashed arrows represent copy operations of the processor, and transmitted packets respectively. The solid arrows represent new arriving Ethernet packets. The figure illustrates the state after the hardware switch is loaded. At the first IFG the multiplexer can handover the receive signals from the software implementation to the hardware implementation. Moreover, the processor has to copy the packets that still reside in the receive buffer of the software switch to the transmit buffer of the hardware switch. If the transmit buffer of the software switch is empty, the multiplexer can handover the transmit signals to the new configuration.

3.1 Profiles of the Ethernet switch

The two implementations of the Ethernet switch are mapped into one profile each by Online-TEReCS. Additionally, a third profile is created, in which the reconfiguration between both switch variants is done. We assume that the node will be in a network in which transit traffic goes through the Ethernet switch component. This means it can not be disabled completely. In table 1 we show the defined settings of the three profiles.

The first profile includes the software switch, while the second profile includes the hardware switch. The quality of both profiles is zero, because in the quality function of the FRM only applications should be considered. The goal of the FRM is to maximize the quality of applications, not of the system

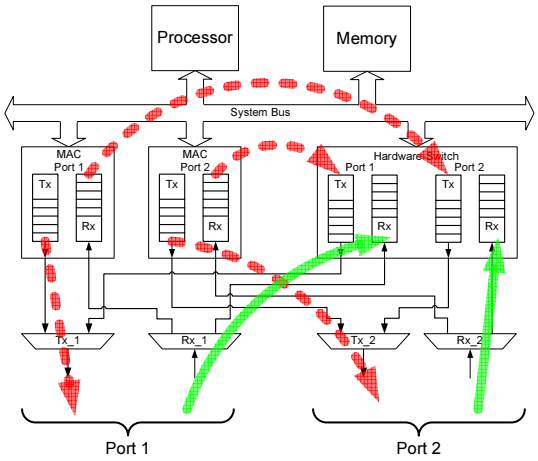

Figure 2. Packet flow during reconfiguration

System service or application	Importance	Profile							
		Profile name	Profile quality	Resource requirements				WCET	
				FPGA		Bandwidth			
				in Slices	Delay	MBit/s	Delay	Enter	Leave
Ethernet switch	0.0	SW	0	1981	0 ms	86	0 ms	1 ms	1 ms
		HW	0	4573	0 ms	0	0 ms	1 ms	1 ms
		reconf.	0	6554	0 ms	0	0 ms	10 ms	1 ms
BW manager	1.0	standard	0	0	0 ms	1-100	15 ms	10 ms	1 ms
Self-opt. application	1.0	normal	0	0	0 ms	0	0 ms	1 ms	1 ms
		accel.	0	3000	0 ms	0	0 ms	10 ms	1 ms

Table 1. The profiles of the example scenario

services. System services are indirectly optimized by deactivating them and releasing their acquired resourses, thus allowing the FRM to activate a profile of an application with higher quality.

To simplify the example only the requirements for FPGA resource and for bandwidth are shown in the table, additional resource requirements like memory or CPU are hidden. The minimum and maximum required space on the FPGA is constant in each profile. Once activated the profiles do not allocate additional space on the FPGA. Both profiles allocate the required space on the FPGA in their initialization phase. This is the reason why the maximal assignment delay is zero.

The resource 'bandwidth' has a special significance for this communication system service: It is the resource the service provides to the system. The maximum bandwidth provided is 100 MBit/s, which can only be provided by the

hardware profile of the Ethernet switch. The hardware/software variant can only provide 14 MBit/s. Therefore, the Ethernet switch blocks the unavailable bandwidth by allocating 86 MBit/s in the software profile. This guarantees that the FRM does not activate this variant if a bandwidth larger than 14 Mbit/s is required. This kind of modeling is necessary because the FRM operates on a fixed quantity of resources.

As mentioned above, handing over the communication service without packet loss from the software switch to the hardware switch requires a coexistence of both switch implementations. In the short period of reconfiguration the third profile provides the FPGA resources that are necessary for both switch implementations. Only if packet loss and an interruption of the connection is tolerable it is possible to just deactivate the current configuration and replace it with the new one. In this case the third profile is dispensable.

4. Case study: Optimization example

To show the benefit of our self-optimizing RCOS/RTOS we present an example scenario. The scenario consists of three components, which define profiles. The presented Ethernet switch component, a bandwidth manager, and a self-optimizing application. In table 1 we show the defined settings of the profiles the components define.

Bandwidth manager. The Ethernet switch is responsible for delivering packets to components on the node and for transit traffic through the node. To guarantee communication under hard real-time condistions bandwidth reservation is applied. By means of the bandwidth manager applications from other nodes can reserve bandwidth in the Ethernet component of this node. The manager allocates the amount of bandwidth at the local FRM for the remote application.

The Bandwidth manager defines only one profile because it cannot be deactivated nor has any implementation alternatives. It is implemented in software, thus no FPGA resources are required. The minimum and maximum bandwidth requirements range from 1 MBit/s to 100 Mbit/s with an assignment delay of 15 ms. This delay allows the activation of the Ethernet switch's software implementaion in order to release FPGA resources. If more bandwidth is required enough time is available to load the hardware implementation. 1 MBit/s is reserved for the communication between the bandwidth managers of the whole communication system (e.g. for the reservation protocol). The remaining bandwidth is allocated if reservations are made by the bandwidth manager.

Self-optimizing application. The self-optimizing application is a component that tries to optimize another application component (e.g., a mechatronic

feedback controller). The optimization process can be accelerated, e.g., by a floating point unit (FPU) implemented in the FPGA. The PowerPC processor of our system does not provide an integrated FPU, so if no FPU is loaded into the FPGA floating point calculations are performed in software with a large overhead. The FPU requires 3000 slices of the FPGA resource and accelerates a 32-bit floating point multiplication by the factor of 44. For system services and application 7.000 slices of the FPGA are available in the system. This means, only the hardware Ethernet switch or the FPU can be executed on the FPGA, not both at the same time. Our self-optimizing resource management system deactivates the hardware Ethernet switch by activating the software profile of the switch if a bandwidth less than 14 MBit/s is required in the system. If a higher bandwidth is required, the floating point acceleration of the application is deactivated and the hardware switch is activated.

5. Related work

The idea to use reconfigurable logic for the integration of network applications into the network interface has been realized, e.g., by Underwood et al. [11]. Comparable network interfaces have been used for server and network applications, e.g., webservers, firewalls [4], and virus protection [6]. Furthermore, it has been shown that Switched Ethernet is appropriate for low-latency hard real-time communication [7].

Customization and configuration is not a new approach for adapting a system to the requirements of an application. In the field of (real-time) operating systems various approaches for customization have been proposed. SYNTHESIS [8] adapts its code by partially evaluating and recompiling condition statements depending on available input data at run time. This can result in changing compare instructions and conditional jumps by unconditional jumps and vice versa. This eventually leads to the elimination or integration of complete code fragments. Likewise, the operating system K42 [10] includes system support for online reconfiguration by a hot-swap mechanism using C++ virtual function tables.

6. Conclusion

In this paper, we have presented the concept and design of a self-optimizing RCOS/RTOS systems for dynamic, self-optimzing applications. Our RCOS/RTOS system deactivates sytem services or activates slim version of the service, which are restricted in functionality, if the service or the full version is not required. In addition to an efficient use of resources our system includes support for upcoming application demands. This concept of redistributing temporally unused resources is not only applicable for system services but also for the applications itself. Thus, the resources of the system can be utilized much

better, even under hard real-time conditions. Furthermore, the internal waste of resources for worst-case resource consumptions are minimized. To facilitate this, possible system services and applications have to provide additional information about their resource requirements and about corresponding conditions by means of the Profile Framework. The benefit of our system is shown by an example of a communication service, which we have implemented on our platform for self-optimizing systems.

References

[1] C. Böke. *Automatic Configuration of Real-Time Operating Systems and Real Time Communication Systems for Distributed Embedded Applications*. Dissertation, Universität Paderborn, Heinz Nixdorf Institut, Entwurf Paralleler Systeme, 2004.

[2] R. P. Chivukula, C. Bøke, and F. J. Rammig. Customizing the Configuration Process of an Operating System Using Hierarchy and Clustering. In *Proc. of the* 5^{th} *IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC)*, pages 280–287, Crystal City, VA, USA, 29 April – 1 May 2002. IFIP WG 10.5. ISBN 0-7695-1558-4.

[3] C. Ditze. *Towards Operating System Synthesis*. Phd thesis, Department of Computer Science, Paderborn University, Paderborn, Germany, 1999.

[4] D. Friedman and D. Nagle. Building firewalls with intelligent network interface cards. Technical report, 2001.

[5] H. Kalte, M. Porrman, and U. Rückert. A prototyping platform for dynamically reconfigurable system on chip designs. In *Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip (SoC)*, 2002.

[6] J. W. Lockwood, J. Moscola, M. Kulig, D. Reddick, and T. Brooks. Application of hardware accelerated extensible network nodes for internet worm and virus protection. In *Proceedings of the International Working Conference on Active Networks (IWAN)*, 2003.

[7] J. Löser and H. Härtig. Low-latency hard real-time communication over switched ethernet. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS 04)*, 2004.

[8] H. Massalin. *Synthesis: An Efficient Implementation of Fundamental Operating System Services*. Phd thesis, Columbia University, 1992.

[9] S. Oberthür and C. Böke. Flexible resource management - a framework for self-optimizing real-time systems. In B. Kleinjohann, G. R. Gao, H. Kopetz, L. Kleinjohann, and A. Rettberg, editors, *Proceedings of IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES'04)*. Kluwer Academic Publishers, 23 - 26 Aug. 2004.

[10] C. Soules, J. Appavoo, K. Hui, D. Silva, G. Ganger, O. Krieger, M. Stumm, R. Wisniewski, M. Auslander, M. Ostrowski, B. Rosenburg, and J. Xenidis. System support for online reconfiguration, 2003.

[11] K. D. Underwood, R. R. Sass, and W. B. Ligeon. A reconfigurable extension to the network interface of beowulf clusters. In *Proceedings of the IEEE Conference on Cluster Computing (Cluster 2001)*, 2001.

[12] E. Vonnahme, B. Griese, M. Porrmann, and U. Rückert. Dynamic reconfiguration of real-time network interfaces. In *Proceedings of the 4th International Conference on Parallel Computing in Electrical Engineering (PARELEC 2004)*, pages 376–379, Dresden, Germany, 7 - 10 Sept. 2004.