# PLACING FUNCTIONALITY IN FAULT-TOLERANT HARDWARE/SOFTWARE RECONFIGURABLE NETWORKS

*Thilo Streichert*

Department of Computer Science 12; University Erlangen-Nuremberg
Am Weichselgarten 3; D-91058 Erlangen
email: streichert@cs.fau.de

## ABSTRACT

A novel framework shows the potential of FPGA-based systems for increasing fault-tolerance and flexibility by placing functionality onto free hardware (HW) or software (SW) resources at runtime. Based on Field-Programmable Gate Arrays (FPGAs) in combination with CPUs, tasks implemented in HW or SW will migrate between computational nodes in a network. Moreover, if not enough HW/SW resources are available, the migration of functionality from HW to SW or vice versa is provided. By integrating this resource management, also called online HW/SW partitioning, in a distributed operating system for networked systems, a substantial step towards self-adaptive systems has been done. Besides a short motivation for this project and the presentation of related work, this paper gives an idea about a novel approach and a powerful experimental setup.

## 1. INTRODUCTION

Current networked embedded systems, e.g., in automotive electronics or body-area-networks bind functionality statically on certain electronic control units (ECUs). Thus, if one node fails the functionality hosted by the ECU will be lost and due to data dependencies other functions on working ECUs may not operate either. The same holds true for erroneous communication links which may lead to an isolation of a node or changing routes in point-to-point networks. In order to tolerate defects of computational nodes, it is necessary to replicate computational nodes and links such that a certain degree of redundancy is available. Obviously, introducing redundancy into embedded networks has some essential drawback concerning monetary costs, power consumption, size, weight, etc.. Therefore, an overview about a novel problem instance is presented that requires a separation of functionality from the physical HW and leads to a reduced degree of redundancy. By using reconfigurable devices such as FPGAs together with CPUs, it will be possible to bind functionality implemented in either HW or SW, dynamically to the resources in the network.

Binding functionality onto computational nodes has been investigated in many research fields. The offline approach towards HW/SW partitioning has been considered by many researchers [7, 9, 4]. For example Blickle [4] synthesizes Pareto-optimal systems out of many design alternatives with the help of Evolutionary Algorithm. Thus, this approach helps a system designer for an unbiased decision making.

However, so-called load balancing algorithms have received a considerable interest and solve the problem of task binding with the objective of homogeneously distributing software tasks onto CPUs at runtime. The motivation for load balancing are a) the reduction of latency or average response time, b) to provide fairness and c) reduction of overheads due to many context switches on highly utilized nodes. Load balancing approaches like *Diffusion* [5, 6], *Token Distribution* [11, 3], and so-called *Balancing Circuits* [2] are distributed algorithms and are thus applicable for fault-tolerant networked systems.

Another field of binding functionality on resources has been opened by Vahid et al.. Based on a platform consisting of reconfigurable hardware and a CPU [10] a profiler extracts critical code regions, decompiles them and synthesizes them to hardware. Achieving an average speedup of 2.6 [12] for different benchmarks, this approach to dynamic HW/SW partitioning shows the potential of dynamically assigning tasks to SW of HW resources.

Unfortunately, the presented approaches either do not consider HW/SW reconfigurability at all or provide no extension to reconfigurable networks.

## 2. PROBLEM STATEMENT AND CONCEPTS

Online HW/SW partitioning aims at binding tasks to free HW or SW resources at runtime. Typically, the HW/SW partitioning is executed during the design phase of an embedded system. But since dynamic effects like node or link defects as well as new arriving tasks corrupt an optimal binding, it will be inevitable to determine a new binding online. In Fig. 1 the two basic steps of online HW/SW partitioning are shown in an exemplifying scenario. The presented network topology consists of four computational nodes $c_1, \ldots, c_4$, a sensor $s_1$ and an actuator $a_4$. The controller tasks $t_2^c, t_3^c$ are bound onto the computational nodes $c_1, c_4$, sensor task $t_1^s$ is bound onto sensor $s_1$ and actuator task $t_4^a$ is bound onto $a_4$. Additional to the tasks $t_i^c$, replicas $t_i'^c$ are bound onto the computational nodes $c_1, c_2 \in C$. A requirement to this replica binding is that a task $t_i^c$ and its replica $t_i'^c$ must not be bound onto the same computational node $c_j$. Then, if node $c_1$ fails, all tasks bound onto this node will be lost either. During the *fast-repair* phase, the replicated tasks $t_i'^c$ become the main task $t_i^c$ and new routes for the task-to-task communication have to be established. Obviously, the tasks are sub-optimally bound after the fast-repair phase which will

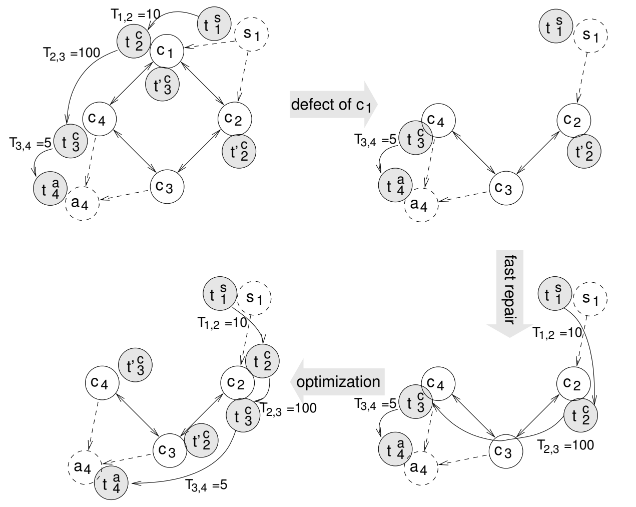

**Fig. 1**. A network is shown after a node defect, a repair phase and an optimization step

be improved during the optimization phase. The optimization phase tries to find a binding of tasks $t_i^c$ to free resources such that the data traffic on the communication links will be minimized and constraints to the CPU-utilization or the usage of hardware resources, resp. will not be violated. In order to tolerate another node defect, replicated tasks $t_i'^c$ need to be created and bound onto the computational nodes $c_j$.

Requirements to algorithms that bind functionality onto free resources are: a) they have to be executed in a distributed manner and b) they require only local knowledge. Therefore, we presented a first approach to online HW/SW partitioning for reconfigurable networks [14, 13] which is based on a combination of diffusion algorithms and a bi-partitioning. In a first step the diffusion algorithm balances the load between the resources and thus, maximizes the free resources on each node. In the next step the bi-partitioning assigns the functionality to free HW or SW resources on each single node. With this strategy the likelihood that the load of defect computational nodes or new tasks may be adopted by every node is increased. Of course, the presented approach may run into local-minima, but in our experimental results the binding has been drastically improved. Moreover, the algorithm fulfills the previously mentioned constraints and scales very well with the network size.

## 3. IMPLEMENTATION

In order to evaluate online HW/SW partitioning approaches not only by simulation, a network of four reconfigurable FPGA-based boards incorporating a RISC-CPU and additional logic for implementing hardware has been set up. The operating system *microC-OS II* has been extended such that node and link defects are automatically detected. Additionally, a *task manager* has been designed and implemented which gathers information about the task placement and provides the task migration functionality.

On top of this network infrastructure, a driver assistance application has been implemented which demonstrates the capability of the network infrastructure. With the help of pattern recognition algorithms, the application tracks the lane and in case of an unintended lane change, the assistant sets off an acoustic warning. The entire driver assistant runs on the network in a distributed manner. Thus, if one node fails, the tasks have to be dynamically reassigned to free resources in the network. For further information on this experimental setup, please refer to [1, 8].

## 4. CONCLUSION AND FUTURE WORK

In this contribution, a new problem class (online HW/SW partitioning) is presented which places functionality onto free hardware or software resources in a reconfigurable network. A first approach has been investigated which solves this problem at runtime, in a distributed manner, and only with local knowledge. With the help of a powerful experimental setup, it will be now possible to prove the applicability, to analyze the timing behavior and to investigate further approaches. Next steps, in the context of online HW/SW partitioning will target the minimization of data traffic and the placement of replicated tasks.

## 5. REFERENCES

[1] *ReCoNets-Demonstrator*. www.reconets.de.
[2] J. Aspens, M. Herlihy, and N. Shavit. Couting networks. *Journal of ACM*, 41:1020–1048, 1994.
[3] F. M. auf der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. *Algorithmica*, 15:413–427, 1996.
[4] T. Blickle, J. Teich, and L. Thiele. System-Level Synthesis Using Evolutionary Algorithms. In R. Gupta, editor, *Design Automation for Embedded Systems*, 3, pages 23–62. Kluwer Academic Publishers, Boston, Jan. 1998.
[5] J. E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Practice and Experience*, 2:289–313, 1990.
[6] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.
[7] V. Kianzad and S. S. Bhattacharyya. CHARMED: A Multi-Objective Co-Synthesis Framework for Multi-Mode Embedded Systems. In *Proc. of the 15th IEEE Int. Conf. on Application-Specific Systems, Architectures and Processors (ASAP'04)*, pages 28–40, Galveston, U.S.A., Sept. 2004.
[8] D. Koch, T. Streichert, S. Dittrich, C. Strengert, C. Haubelt, and J. Teich. An Operating System Infrastructure for Fault-Tolerant Reconfigurable Networks. In *Proc. of ARCS06*, Frankfurt (Main), Germany, Mar. 2006. Springer.
[9] M. López-Vallejo and J. C. López. On the Hardware-Software Partitioning Problem: System Modeling and Partitioning Techniques. *ACM Trans. on Design Automation of Electronic Systems*, 8(3):269–297, July 2003.
[10] R. Lysecky and F. Vahid. A configurable logic architecture for dynamic hardware/software partitioning. In *Proc. of DATE04*, pages 480–485. IEEE Computer Society, 2004.
[11] D. Peleg and E. Upfal. The token distribution problem. *ORSA Journal on Computing*, 18:229–243, 1989.
[12] G. Sitt, R. Lysecky, and F. Vahid. Dynamic Hardware/Software Partitioning: A First Approach. In *Proc. of DAC03*, Anaheim, California, Germany, June 2003.
[13] T. Streichert, C. Haubelt, and J. Teich. Distributed HW/SW-Partitioning for Embedded Reconfigurable Systems. In *Proc. of DATE05*, Munich, Germany, Mar. 2005.
[14] T. Streichert, C. Haubelt, and J. Teich. Online Hardware/Software Partitioning in Networked Embedded Systems. In *Proc. of ASP-DAC05.*, pages 982–985, Shanghai, China, Jan. 2005.