Modelling Tile-Based Run-Time Reconfigurable Systems Using SystemC

Carsten Albrecht, Thilo Pionteck, Roman Koch, and Erik Maehle
Institute of Computer Engineering
University of Lübeck
Ratzeburger Allee 160
23538 Lübeck
Germany
{albrecht,pionteck,koch,maehle}@iti.uni-luebeck.de

Abstract— **In the area of hardware design, there is a noticeable trend towards the use of run-time reconfigurable elements as parts of System-on-Chips (SoCs), SoCs themselves are frequently targeted to reconfigurable platforms such as field programmable gate arrays. This development is a challenge to established high-level modelling and simulation methods which assume a static structure of the simulated system. The present paper describes the extension of the SystemC framework with support for the simulation of run-time reconfiguration of tile-based architectures. The proposed solution is set out in detail and its application to an exemplary reconfigurable SoC design is described.**

Keywords— **SystemC Modelling, Run-Time Reconfiguration**

I. Introduction

Over recent years, the demand for flexible and adaptable System-on-Chips (SoCs) increased as more and more features per system became required. Additionally, the hardware development cycles must compete with the rapid changes of application-specific requirements such as protocols in the network domain, or video and audio formats in the multimedia section.

Current FPGA (Field Programmable Gate Arrays) devices such as members of the Xilinx Virtex series [Xilinx, 2005a], [Xilinx, 2005b] provide features for partial run-time reconfiguration. SoCs based on these devices may exchange certain modules at run-time without interruption of unaffected system parts. The reconfiguration can be performed within the system so that there is no need for external administration instances. There are many examples proving that certain applications are significantly accelerated by applying the capabilities of a run-time reconfigurable device [Compton and Hauck, 2002].

As the approach of merging spatial and temporal computing [DeHon and Warzynek, 1999] strongly differs from classical hardware and software design current developer tool suites only provide limited support for run-time reconfigurability. Thus, simulation and high-level models play a more important role to keep up with the tremendous growth of complexity in SoC design. The application of simulation in hardware design is a state-of-the-art technique for testing, verification, and profiling purposes. It generally provides exhaustive views and allows in-depth analysis of crucial, otherwise inaccessible system parts. Using higher-level methods for modelling and simulation especially for hybrid hardware/software systems, SystemC [SystemC, 2006] brought out by a pool of companies is frequently applied. SystemC is a C++ class library that allows simulation of hardware/software systems modelled on varying abstraction levels.

Models may benefit of pure C++, the SystemC library including core language and data types, methodology and technology-specific libraries on top of native SystemC as well as any C++ library to implement module functionalities such as cryptographic cores.

Furthermore, SystemC backs the top-down design methodology so that each module can be iteratively redesigned which leads to a hierarchical design. Nowadays, an increasing number of tools also use SystemC as a hardware description language to support modelling and simulation on different levels.

Unfortunately, SystemC does not provide abilities for partial run-time reconfiguration. Simulated models are completely instantiated before the simulation process starts and cannot be exchanged, removed, or added during simulation, i.e. at run-time from the simulated system's point of view. Dividing the device in a coarse-grained array structure is one of the design strategies proposed in reconfigurable SoC design. The tile-based model for SystemC presented here applies this approach for high-level modelling and design.

This article is organised as follows: In Section II, an overview of related projects covering modelling and simulation is given. Furthermore, Section III presents state-of-the-art devices applied for run-time reconfigurable SoCs. Section IV sets its focus on the overcoming of the shortly described SystemC limitations to run-time reconfigurability and shows the proposed modelling technique. Finally, Section V gets a picture of the simulation-model application across and in Section VI a conclusion is drawn.

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

II. Related Work

SystemC is widely accepted in the field of SoC design. Nevertheless, it is mandatory to overcome the limitations of the basic modelling and simulation principles in order to apply SystemC to partially run-time reconfigurable systems. Currently, there are several projects developing or extending SystemC frameworks and models with run-time reconfigurability. Due to the fact that all modules must be instantiated at the beginning of each simulation process, [de Brito et al., 2006] extended the SystemC kernel with a mechanism to switch modules on and off at run-time.

Other approaches do not need any kernel modifications. [Amicucci et al., 2006] uses a sort of template class that contains the usual sensitive functions of the module. These function bodies scheduled for execution by the kernel consist of a function pointer that is re-targeted by the reconfiguration. The proposed model includes a reconfiguration controller attached to the run-time reconfigurable module.

OSSS+R [Schallenberg et al., 2006] also bases on polymorphic functions and objects. It introduces run-time reconfigurability to SystemC with the goal to support the complete design flow from a high-level description down to a synthesised bit stream for an FPGA. [Schallenberg et al., 2006] uses a fixed base design with reconfigurable modules containing reconfiguration controllers tightly-coupled to the reconfigurable area part.

The ADRIATIC project [Pelkonen et al., 2003], [Qu et al., 2005] enhances SystemC by a framework for system analysis and even resource estimation. Based on these tools a complete design flow is proposed. The basic architecture is bus-based and includes multiple hardware accelerators assisting a general-purpose processor. At least one accelerator is reconfigurable utilising its own configuration scheduler and memory.

All in all, the capabilities of these approaches are often limited by basic architecture models or tightly-coupled reconfiguration controller and area.

Barros presents a general formal approach of modelling and simulating run-time reconfigurable systems [Barros, 1997]. Here, the systems, called dynamic structure discrete event systems, are basically described by networks of modules representing internal relations and interactions of the system modules. Beside defining input and output sets and functions for output and time progress, the description comprises states of the structure and a state transition function. Additionally, an abstract simulator for system models utilising this formalism is provided so that the models can directly be executed [Barros, 1998].

III. System-on-Chip Design

Modern System-on-Chip designs usually comprise configurable components or are realized on configurable hardware platforms. This eases the adaptation of a single SoC design for different application areas and, thus, helps spreading the non-recurring engineering (NRE) costs over larger quantities. As configurable devices continuously evolve in the direction of partial reconfigurable devices, these techniques now become available for SoC designs as well. Utilising partial reconfigurable devices as SoC components or as a hardware platform for SoCs, system designs cannot only be adapted to different application areas during design time, but can be adapted to changing operation conditions during run-time, too. This additionally increases the range of applications of a single SoC design.

Typical representatives of partially reconfigurable devices are the Xilinx Virtex-II, Virtex-4, and Virtex-5 FPGA series. The basic structure of these FPGA series consists of an array of configurable logic blocks (CLBs) and hierarchically organised routing resources. In addition, dual-ported SRAM modules (Block-RAM), multipliers, and, depending on the device, application specific components such as processor cores or DSP components are regularly spread over the device. The configuration data is stored into SRAM which can either be accessed by external interfaces or internally by using an internal configuration access port (ICAP). All mentioned Virtex FPGAs series provide the possibility to selectively change parts of the configuration during run-time. This feature is referred to by Xilinx as active partial reconfiguration [Xilinx, 2005c] and allows configurable elements of the FPGA to be reconfigured without interrupting the operation of logic circuits configured into other parts the FPGA. The smallest unit that can be reconfigured on the hardware level is referred to as a frame. A frame spans a fraction of the width of one CLB column. Its height depends on the Virtex series it is part of. For the Virtex-II series, a frame spans the entire height of a device [Xilinx, 2005a]. This results in a so called column-based reconfiguration, as the smallest unit that it makes sense to exchange is a CLB column with the width of one CLB and the height of the device. With regard to the exemplary SoC shown in Figure 1, this is equivalent to exchanging a complete column at once. The columns are visualised by combining tiles in grey and white columns. The Virtex-4 and Virtex-5 series offer a finer reconfiguration granularity. For these architectures, the height of a frame is a multiple of 16 or 20 CLBs, respectively, [Xilinx, 2007a], [Xilinx, 2007b]. Thus, the smallest unit that can be reconfigured consists of a tile with a width of one CLB and a height of 16 or 20 CLBs. Applied to the SoC in Figure 1, it is equivalent to exchanging only a single square at once without any effect to the column or row.

The reduced and adjustable height of a frame for the Virtex-4 and Virtex-5 series eases the design of dynamically reconfigurable systems. Logic resources of the FPGA can be exploited more efficiently, as the area allocated for a hardware module can be adjusted more precisely to the actual required amount of logic resources for one module. Depending on the size of a currently configured hardware module, a dynamically reconfigurable SoC may comprise of a varying number of modules. To guarantee communication among a varying number of hardware modules with different sizes and locations, special communication units with

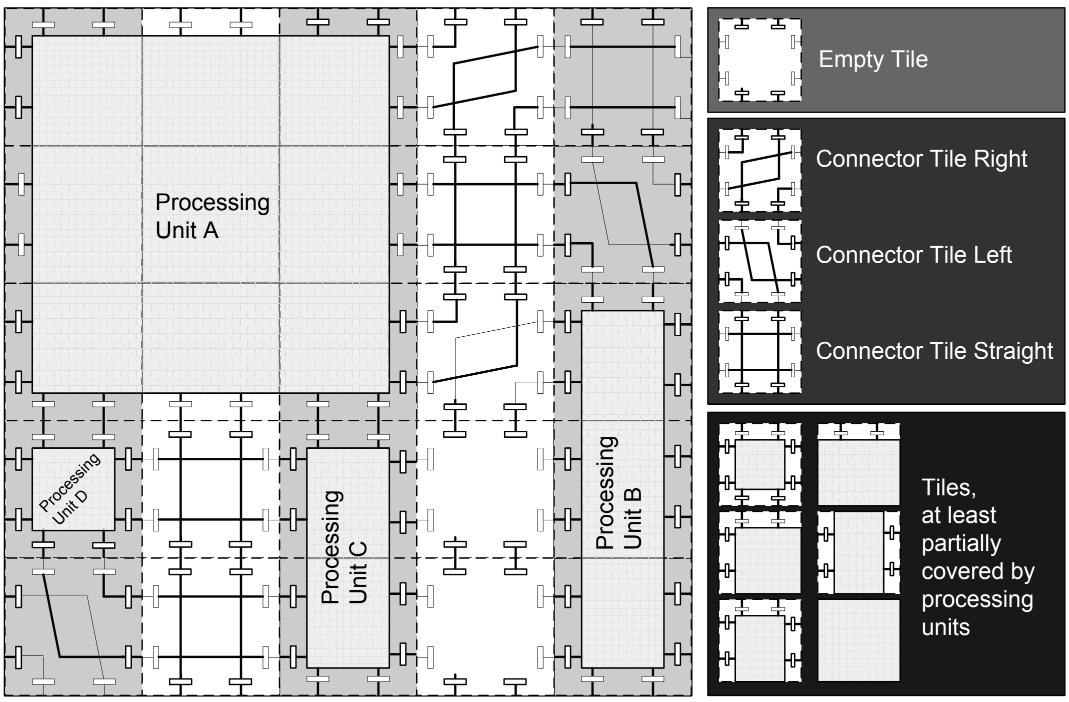


Fig. 1. Column/tile-based design of an examplary System-on-Chip.

fixed positions at the edges of the tiles have to be provided. These units, which are called bus macros, consists of communication lines crossing the border of adjacent tiles, with starting points and endpoints routed to LUTs (Look-Up Tables). Dynamically exchanged hardware modules can connect to these units and thereby get access to predefined communication channels. In partial reconfigurable systems these predefined communication channels can itself be reconfigured, allowing an adaptation of the communication resources of a design to the actually instantiated hardware modules.

An example for such a SoC design is given in Figure 1. Four hardware modules are mapped onto a reconfigurable area which is partitioned into a grid of rectangular tiles. Each tile can either be used for the realization of a hardware module, for the communication network or it can be left blank. Besides the case that a tile is completely used for the realisation of a hardware module, each tile provides bus macros for the connection to adjacent tiles, shown in Figure 1 as small boxes within a tile. Depending on the position of the actually instantiated hardware modules and their communication requirements, communication channels can be created using predefined communication tiles.

IV. Simulation Model

A common approach of modelling SoCs or generally hardware is the decomposition into independent modules of different functionality with well-defined interfaces. An important issue of reconfigurable SoCs is the hardware/software co-design of these systems. Runtime reconfigurable systems are often managed or controlled by a software instance running on an embedded, hard [Koch et al., 2006] or soft-core [Ullmann et al., 2004], or attached processor [Majer et al., 2006] combined with dedicated hardware modules. So, runtime reconfigurable SoCs demand hardware-software co-development and, of course, a combined model to avoid gaps between hardware and software. An environment that fulfils the requirements for hybrid systems is SystemC, a C++ class library for system and hardware design.

A. Tile Model

Modelling SoCs realised on run-time reconfigurable platforms demand that the platform reconfiguration capabilities are included as well. For example, leaving out the constraints of the platform could lead to wrong simulation results because of assuming wrong timing models.

The derivation of a SystemC simulation model from a run-time reconfigurable hardware design can be achieved by transferring the basic entities presented in Section III to SystemC modules. The area is divided into a one dimensional sequence of columns or even a two dimensional grid of tiles. The latter is more general because a column can be seen as a row of tiles, cf. Figure 1. A tile is used as a container encapsulating all needed functionalities.

Figure 2 shows the abstract model of a tile. The inner process of a SystemC module is implemented using a synchronous sequential circuit (SSC), asynchronous sequential circuit (ASC) or combinatorial circuit (CC) represented by `SC_CTHREAD`, `SC_THREAD` and `SC_METHOD`. Each tile activates all of these methods and sets their sensitivity to any incoming port including the system clock. The method bodies only consist of a function pointer that indicates the currently configured functionality. These pointers are represented by arrows in Figure 2. The complete set of functionalities consists of

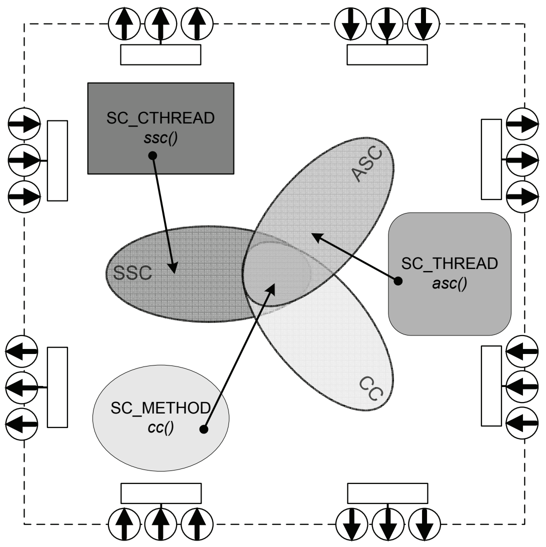


Fig. 2. SystemC simulation model of a tile.

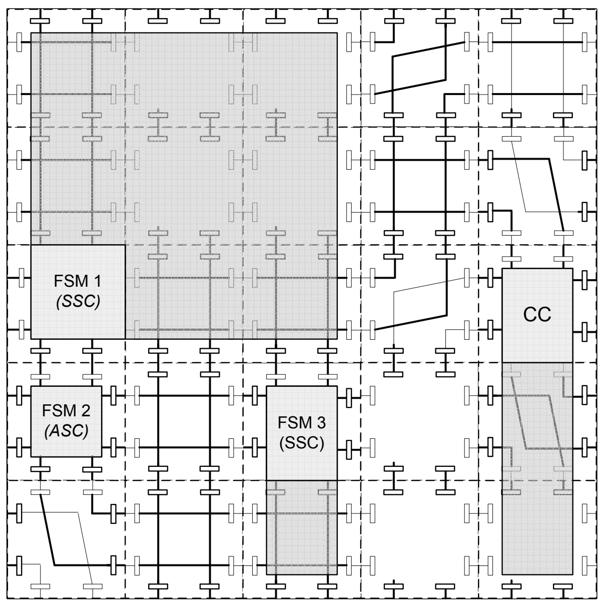


Fig. 3. Tile-based simulation model of an examplary SoC.

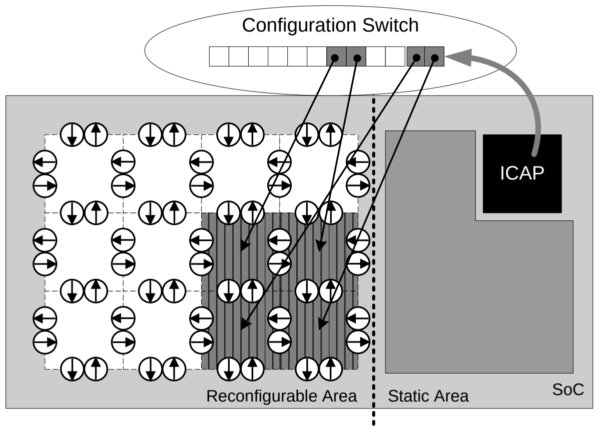


Fig. 4. Reconfiguration process for part of the run-time reconfigurable area activated by ICAP utilising the global management level of the simulation model.

functions of all three classes. The classes or subsets are not disjoint. At least the empty or idle function as well as functions implementing simple signal transfer from incoming to outgoing ports for the communication network are members of all subsets shown in Figure 2. Although it is possible to implement any processing unit using only an `SC_THREAD`, the other module methods are kept to easily allow combinations of individual functions. So, there is no need to have separated and combined implementations. Furthermore, the implementation of an SSC becomes more difficult by avoiding triggers raised by an arbitrary port, for example. Of course, a growing array of tiles increases simulation time by processing unused methods several times per simulated cycle per tile. The bus macros used as connectors in the FPGA design are represented in Figure 2 by multiple ports. The set of ports per edge is given by the superset of ports required by all implemented functionalities so that each function might be connected to any neighbour. Thus, all ports are rarely used simultaneously, the number of ports can be reduced by modelling bus macros by ports for a complex, self-defined record that includes any incidence of occurring data types.

B. System Model

In complete system models based on reconfigurable tiles area consumption of different modules must be taken into account. The tile size is determined by the smallest module and the size of the bus macros needed to ensure connectivity. Modules that consume more area than provided by a single tile allocate multiple adjacent tiles.

Figure 3 shows a SoC model with modules of varying sizes. Due to the fact that a tile always performs the complete functionality of a simulated hardware module, modules are compounded from tiles so that more than a single tile is consumed. In the simulation model the tiles are not equally used. There is one tile, called master tile, of the compound that executes the task while the others are functionally switched off or have to route signals so that the module is connected to its neighbours. In Figure 3, the master tile is emphasised by a short description in a solid rectangular shape whereas the compound is marked by a transparent full-sized module. The shown simulation model belongs to the design of Figure 1. Here, the background functionality of compound tiles is sketched as well. In general, they are idle but some route signals to establish the complete communication network.

C. Run-Time Reconfiguration

Run-time reconfiguration demands access to the configuration memory of the device. As mentioned before the access is gained by writing the new configuration to the ICAP. This technique for reconfiguration is reused in the SystemC model. After retrieving the data, the ICAP simulation module enables the new configuration after receiving a switch command. In contrast to other models presented in Section II, the ICAP or reconfiguration port does not have direct port access to the tiles reconfigured in this process. This process is performed using the capabilities of standard C++ by utilising global or parent-class data structures. The configuration switch indicates that the current configuration must be at least (partially) exchanged and, thus, the reconfiguration method per tile is called via these data structures of higher levels to achieve a new configuration. Since standard methods are executed within a simulated cycle the new configuration takes over instantly. Figure 4 depicts this process: the ICAP signals upwards to the configuration switch layer that a switch has to be performed. The configuration switch layer processes the signal and executes the reconfiguration methods of the affected tiles. The reconfiguration of a tile functionality itself is performed by re-targeting the function pointer of any repeatedly executed method.

From the application's point of view, the synchronisation of the reconfiguration and the application-specific processing within the module is rather crucial. Asynchronous reconfiguration can lead to data loss or system inconsistencies if a module cannot complete its task. Basically, current FPGA devices as mentioned before do not support any mechanisms for synchronisation. This task is completely committed to the system designer. Thus, reconfiguration and module processes are not synchronised by the simulation model inherently. Necessary synchronisation mechanisms have to be included in the application. The configuration switch is performed when the ICAP signals a switch command without paying attention to the state of the module. To avoid trouble during the simulation, module processes with internal states should be implemented in a way that state transitions always takes place at the end of a function execution. The state is stored globally within the module and reset in case of reconfiguration. Therefore, `wait()` statements can be avoided in reconfigurable functions targeted by function pointers described before.

V. Application Example

In the following, an exemplary partially reconfigurable SoC design called DynaCORE [Albrecht et al., 2006] which is modelled using the described technique is presented. DynaCORE is a run-time reconfigurable coprocessor for network processors. It is designed to release the network processor from computational intensive tasks such as encryption/decryption, compression, or network-intrusion detection. One characteristic of this application area is that the composition of network flows and thus the tasks to be performed change during time. Hence, a flexible solution is required where the types and numbers of hardware modules mapped onto the system can be adapted to the characteristics of the network flow.

The structure of the DynaCORE architecture is depicted in Figure 5. The system is divided into a static and a reconfigurable part. The static part is made up of global control and system management logic while the reconfigurable part contains the actual hardware modules or hardware assists in terms of assisting the network processor. The hardware modules and the system control logic are connected by a topology adaptive network-on-chip (NoC) called CoNoChi [Pionteck et al., 2006], which allows the adaptation of the communication structure to the number and location of currently configured hardware modules.

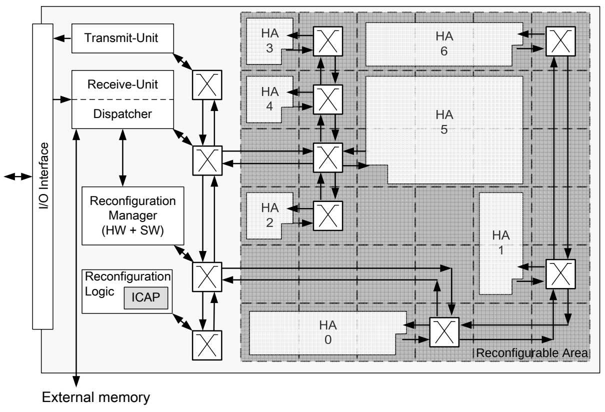


Fig. 5. The architecture of DynaCORE includes a tile-based reconfigurable area for data processing. Hardware assists (HA) perform provided tasks.

In DynaCORE, incoming network packets are evaluated in the receive unit. The dispatcher determines the type of processing and forwards the packets to the appropriated hardware modules. There, the packets are processed by one or multiple hardware modules in a chain and finally forwarded to the transmission unit. In case a currently unsupported function is needed or the reconfiguration manager detects a significant change in the traffic profile the system is adapted utilising partial reconfiguration. In contrast to the hardware modules, the reconfiguration manager is a software program executed on an embedded processor core. Here, the hardware/software co-design features of the SystemC model are utilised.

The simulation model and the high-level SoC architecture of DynaCORE are absolutely compatible. The decoupled implementation of reconfiguration manager and reconfigurable area do not demand the integration of additional software parts. Furthermore, the synchronisation of reconfiguration and data processing can be neglected for this specific application area. Nevertheless, the protocol of CoNoChi provides a synchronisation mechanism to avoid data loss.

VI. Conclusion

Motivated by current run-time reconfigurable devices the tile-based simulation model for SystemC is derived from SoC designs. The transfer of the tile-based modelling approach to SystemC allows the application of run-time, even partial reconfiguration techniques in SoC models for designing, testing, verifying, and evaluating on a high level. In contrast to other models, the architecture does not require a certain reconfiguration controller per reconfigurable unit and, thus, is highly adaptable to the design. The tile size and port definitions can be freely defined as well as it is possible for FPGA designs. Furthermore, the reconfiguration of the device and the reconfiguration of the application are kept basically independent. The simulation model maintains the independence of simulated application and hardware device. This quality is sketched in the given application example. So, the transfer of the coarse-grained, scalable tile-based model to SystemC is supposed to be a further step to close the gap of spatial and temporal computing.

VII. Acknowledgement

This work was funded in part by the German Research Foundation (DFG) within priority programme 1148 under grant reference Ma 1412/5.

References

[Albrecht et al., 2006] Albrecht C., Foag J., Koch R., and Maehle E. 2006. *DynaCORE - A Dynamically Reconfigurable Coprocessor Architecture for Network Processors.* In Proceedings of the 14th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP). IEEE Computer Society. Montbeliard-Sochaux, France.

[Amicucci et al., 2006] Amicucci C., Ferrandi F., Santambrogio M.D., and Sciuto D. 2006. *SyCERS: a SystemC Design Exploration Framework for SoC Reconfigurable Architecture.* In Proceedings of the 2006 International Conference on Engineering of Reconfigurable Systems & Algorithms, ERSA 2006, Las Vegas, Nevada, USA.

[Barros, 1997] Barros, F.J. *Modeling Formalisms of Dynamic Structure Systems.* In ACM Transactions on Modeling and Computer Simulation, Vol. 7, No. 4, ACM Press, New York, NY, USA.

[Barros, 1998] Barros, F.J. *Abstract Simulators for the DSDE Formalism.* In Proceedings of the 30th Winter Simulation Conference. Washington, D.C., USA.

[de Brito et al., 2006] de Brito A. V., Melcher E. U. K., and Rosas W. 2006. *An Open-Source Tool for Simulation of Partially Reconfigurable Systems Using SystemC.* In Proceedings of the 2006 Emerging VLSI Technologies and Architectures. Karlsruhe, Germany.

[Compton and Hauck, 2002] Compton K., Hauck S. 2002. *Reconfigurable Computing: A Survey of Systems and Software.* In ACM Computing Surveys, Vol. 34, No. 2, ACM Press, New York, NY, USA.

[DeHon and Warzynek, 1999] DeHon A., Wawrzynek J. *Reconfigurable Computing: What, Why, and Implications for Design Automation.* 1999. In Proceedings of 36th ACM/IEEE Conference on Design Automation Conference (DAC). New Orleans, Louisiana, USA.

[Koch et al., 2006] Koch R., Pionteck T., Albrecht C., and Maehle E. 2006 *An Adaptive System-on-Chip for Network Applications.* In Proceedings of the 13th Reconfigurable Architectures Workshop (RAW) associated with the 20th International Parallel and Distributed Processing Symposium, IEEE, Rhodes Island, Greece.

[Majer et al., 2006] Majer M., Ahmadinia A., Bobda C., and Teich J. 2006. *A Flexible Reconfiguration Manager for the Erlangen Slot Maschine.* In Proceedings of the Dynamically Reconfigurable Systems Workshop (DRS), Frankfurt (Main), Germany.

[Pelkonen et al., 2003] Pelkonen A., Masselos K., and Cupák M. 2003. *System-Level Modeling of Dynamically Reconfigurable Hardware with SystemC.* In Proceedings of IEEE International Parallel & Distributed Processing Symposium. Nice, France.

[Pionteck et al., 2006] Pionteck T., Koch R., Albrecht, C. 2006. *Applying Partial Reconfiguration to Networks-on-Chips.* In Proceedings of 2006 International Conference on Field Programmable Logic and Applications. Madrid, Spain.

[Qu et al., 2005] Qu Y., Tiensyrja K., and Soininen J.-P. 2005. *SystemC-based Design Methodology for Reconfigurable System-on-Chip.* In Proceedings of the 8th Euromicro Conference on Digital System Design (DSD'05). Porto, Portugal.

[Schallenberg et al., 2006] Schallenberg A., Oppenheimer F., and Nebel W. 2006. *OSSS+R: Modelling and Simulating Self-Reconfigurable Systems.* In Proceedings of 2006 International Conference on Field Programmable Logic and Applications. Madrid, Spain.

[SystemC, 2006] IEEE Computer Society 2006. *IEEE Standard SystemC Language Reference Manual.* IEEE 1666-2005.

[Ullmann et al., 2004] Ullmann M., Hübner M., Grimm B., and Becker J. 2004. *An FPGA Run-Time System for Dynamical On-Demand Reconfiguration.* In Proceedings of the 11th Reconfigurable Architectures Workshop (RAW) associated with the 18th International Parallel and Distributed Processing Symposium (IPDPS). Santa Fe, New Mexico, USA.

[Xilinx, 2005a] Xilinx Inc. 2005. *Virtex-II Pro and Virtex-II Pro X FPGA User Guide.* Technical Report.

[Xilinx, 2005b] Xilinx Inc. 2005. *Virtex-4 Family Overview.* Data Sheet.

[Xilinx, 2005c] Xilinx, Inc. 2005. *Two Flows for Partial Reconfiguration: Module Based or Difference Based.* Xilinx Application Note 290, Version 1.2.

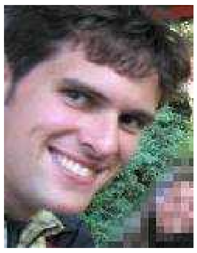
[Xilinx, 2007a] Xilinx, Inc. 2007. *Virtex-II Configuration Guide.* Xilinx User Guide UG071, Version 1.5.

[Xilinx, 2007b] Xilinx, Inc. 2007. *Virtex-4 Configuration Guide.* Xilinx User Guide UG1911, Version 2.2.

Author Biographies

Carsten Albrecht received his Diploma degree in Computer Science from the University of Lübeck in 2002. In the same year, he joined the Institute of Computer Engineering, University of Lübeck. His current research topic is application-specific management of dynamically reconfigurable systems. Further research interests include multithreaded processor design and network processor architectures.

Dr.-Ing. Thilo Pionteck received his Diploma degree in Electrical Engineering in 1999 and his PhD from the Technical University of Darmstadt in 2005. In the same year, he joined the Institute of Computer Engineering, University of Lübeck where he leads the "Reconfigurable Computing" research group. His research interests include dynamically reconfigurable systems, topology adaptive Network-on-Chips and network processors.



Roman Koch received his Diploma degree in Computer Science from the University of Lübeck in 2003. He is currently a PhD student at the Institute of Computer Engineering, University of Lübeck where he holds a position as a research assistant. His current research topic are dynamically reconfigurable systems.

Prof. Dr.-Ing. Erik Maehle received his Diploma and Doctoral degree in computer science from the University of Erlangen-Nürnberg in 1977 and 1982, respectively. After subsequent positions as a postdoc at the IBM Zurich Research Laboratory and the University of Erlangen-Nürnberg he became a Professor at the Universities of Augsburg in 1987 and Paderborn in 1989. Since 1994 he is Director of the Institute of Computer Engineering at the University of Lübeck. He is Vice-Chair of the IFIP Working Group 10.3 Concurrent Systems, steering committee member of the German Special Interest Group on Parallel Processing PARS, member of the Steering Committees of the German Informatics Society (GI) on Computer and System Architecture ARCS as well as Fault-Tolerance and Dependability VerFe and speaker of GI Specialist Area Computer Engineering. His research interests include parallel and fault-tolerant computing, reconfigurable computing, robotics and embedded systems.