THE ANDRES PROJECT: ANALYSIS AND DESIGN OF RUN-TIME RECONFIGURABLE, HETEROGENEOUS SYSTEMS

A. Herrholz[1], F. Oppenheimer[1], P. A. Hartmann[1], A. Schallenberg[2], W. Nebel[2], C. Grimm[3], M. Damm[3], J. Haase[3], F. Brame[3], F. Herrera[4], E. Villar[4], I. Sander[5], A. Jantsch[5], A.-M. Fouilliart[6], M. Martinez[7]

[1]OFFIS Institute, [2]CvO University of Oldenburg, [3]Technical University of Vienna, [4]University of Cantabria, [5]KTH Stockholm, [6]Thales Communications, [7]DS2
email: andreas.herrholz@offis.de

ABSTRACT

Today's heterogeneous embedded systems combine components from different domains, such as software, analogue hardware and digital hardware. The design and implementation of these systems is still a complex and error-prone task due to the different Models of Computations (MoCs), design languages and tools associated with each of the domains. Though making such systems adaptive is technologically feasible, most of the current design methodologies do not explicitely support adaptive architectures. This paper present the ANDRES project. The main objective of ANDRES is the development of a seamless design flow for adaptive heterogeneous embedded systems (AHES) based on the modelling language SystemC. Using domain-specific modelling extensions and libraries, ANDRES will provide means to efficiently use and exploit adaptivity in embedded system design. The design flow is completed by a methodology and tools for automatic hardware and software synthesis for adaptive architectures.

1. INTRODUCTION

Highly integrated embedded systems are usually heterogeneous by including up to three different domains: software, analogue hardware and digital hardware. Extending these systems with abilities to adapt to different operating conditions and functional requirements has become an interesting and motivating goal for industry and research, in particular since the availability of run-time reconfigurable FPGAs.

However, up to now, there is no methodology that allows to seamlessly specify, simulate, synthesise and verify such adaptive heterogeneous embedded systems (AHES), because each domain comes with its own models of computation, languages and tools. In particular, the expression of adaptivity within these domains has either not been thoroughly investigated so far or is not supported at all.

The overall goal of the ANDRES project is to resolve these issues by developing an integrated design flow for AHES. This design flow builds on the open-source modelling language SystemC already adopted by industry and research. As a result, ANDRES will provide a SystemC modelling framework for designing embedded hardware/software systems on a high level of abstraction emphasising in particular the integration of adaptivity. This modelling framework is complemented with concepts and tools for automatic hardware/software synthesis from adaptive system models. At the end of the project, ANDRES will provide a complete design flow for AHES that covers specification, modelling and implementation. The design flow will be evaluated by using state-of-the-art OFDM based communication systems as use-cases.

ANDRES is a specific targeted research project (STREP) and is co-funded by the European Commission within the Sixth Framework Programme. It has started in June 2006 and will last for three years. The project consortium consists of four research partners: OFFIS, Technical University of Vienna, KTH Stockholm and University of Cantabria, and two industrial partners: Thales Communications and DS2. Additional and future information about ANDRES can be found on the project website [1].

The rest of the paper is structured as follows: We will first present the industrial motivation for the project arising from the expectations towards the usage of adaptive architectures. Following a classification of adaptivity and a description of the planned design flow, we we will briefly introduce the domain specific modelling libraries including a formal approach for specification of adaptivity. The paper concludes with a look at the concept for automatic synthesis of run-time reconfigurable hardware.

2. INDUSTRIAL MOTIVATION

The design of today's embedded systems has to deal with the complexity implied by the combination of various technologies (hardware and software) and the increasing need for (re)configurability (e.g. communication systems have to

support several protocols using the same hardware). Modern mobile phones combine for example a PDA, a GPS based navigation system, entertainment and multi-mode communication with all kinds of external devices. Furthermore, these systems have to manage performance and concurrent access to resources. Dynamically reconfigurable systems have the potential of realising efficient systems as well as providing adaptability to changing system requirements. Another motivation for reconfiguration is resource optimisation of the physical layer; one of the crucial issues today is power consumption.

One of the most difficult tasks in the design of modern complex embedded home networking systems, like powerline communication modems, is the validation phase. Here, the use of FPGA technology has the potential to allow early analysis of the behaviour of a system design under almost real conditions while having flexibility for adding or modifying the design, making them the ideal tool for system validation. Dynamic reconfiguration of FPGAs might provide even more capabilities for the validation of this kind of systems: Different peripherals can be included on the FPGA on demand. This way, specific interfaces and modules for information extraction and debugging can be configured and removed on the fly, effectively saving logic and I/O ressources on the FPGA.

3. CLASSIFICATION OF ADAPTIVITY

While the heterogeneity of embedded systems has already been extensively investigated in the past [2, 3], considering adaptivity is rather new to embedded system design. In particular through the availability of very flexible programmable logic devices, like partially run-time reconfigurable FPGAs, making systems adaptive has become very attractive by offering a whole new range of possible applications.

3.1. Adaptive Architectures

In general, adaptivity is not limited to one special domain, e.g. the software or digital hardware domain, but may be expressed and implemented using a range of different adaptive architectures. In ANDRES we consider the following architectures exhibiting various forms of adaptivity:

Microprocessor The microprocessor is highly adaptive due to the possibility to load and execute different programs.

FPGA Some FPGA families support both full and partial reconfigurability during run-time.

Analogue circuits The operation of an analogue circuit can be adapted by changing parameters of analogue components.

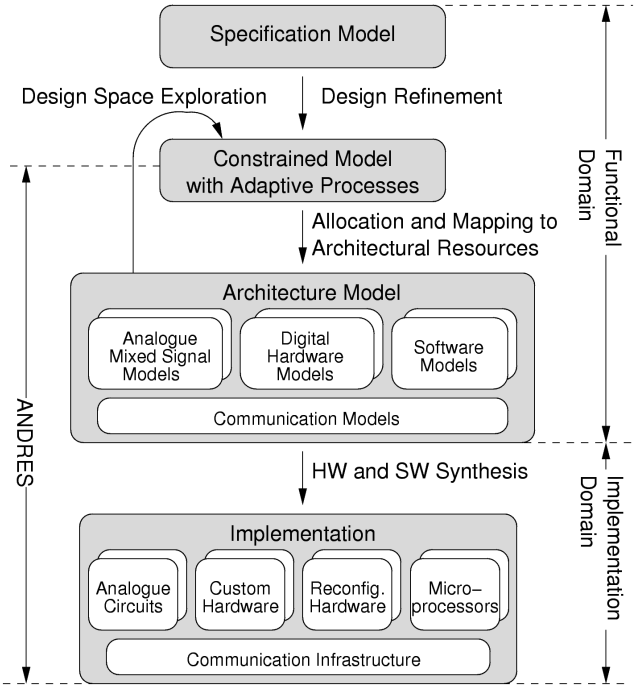

Fig. 1. The ANDRES design flow

Custom hardware Custom hardware can provide some degree of adaptivity, if a component can be set to different modes.

3.2. Types of Adaptivity

Depending on the chosen architecture and application adaptivity comes in different flavours. In ANDRES we cover the full degree of adaptivity, from setting a few parameters up to reconfiguring the whole device. We mainly focus on *dynamic adaptivity*, where the adaptation of the system is done during run-time. Components can either be self-adaptive or their adaptation is controlled externally. We also consider channel adaptation, where a communication channel adapts to different interfaces of an adaptive component.

4. ANDRES DESIGN FLOW

The ANDRES design flow, as illustrated in Figure 1, starts with a constrained system model, already modelling adaptivity and including functional and non-functional properties. This constrained model is a SystemC model, whose design rules and guidelines are based on a formal approach (Section 5.1). This model can then be refined to different target domains using one of the three domain-specific modelling libraries: SystemC-AMS (Section 5.2), HetSC (Section 5.3) and OSSS+R (Section 5.4).

Each of the libraries provides means to create executable specifications to simulate system components. All libraries

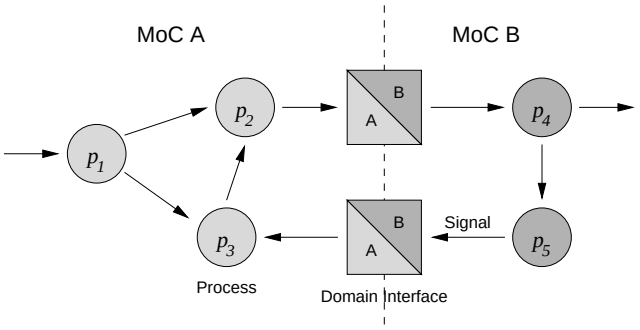

Fig. 2. Processes of different Models of Computation can communicate with each other via domain interfaces

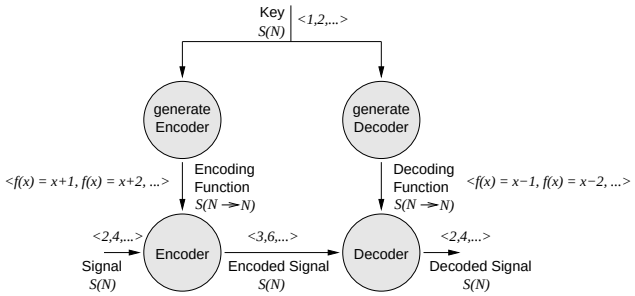

Fig. 3. The Encoder/Decoder is a typical example for adaptivity.

are based on the discrete-event kernel of SystemC, so in principle they can be coupled to simulate the complete system. Simulation coupling is supported by an overall framework and so called *polymorphic signals* (Section 5.5). These signals support exchanging Models of Computation of system components without having to change the connecting communication infrastructure.

The general adaptivity concept already expressed in the constrained model is specialised within each of the libraries using domain specific techniques. Because not all target architectures provide the same types of adaptivity, e.g. analogue circuits only allow parametrisation, exploration of different types and architectures may be limited.

Finally the refined SystemC system model is the entry point for automatic synthesis. ANDRES will provide tools for automatic synthesis of digital reconfigurable hardware, software and communication infrastructure. However, ANDRES does not cover automatic synthesis of analogue circuits.

5. SPECIFICATION OF ADAPTIVE SYSTEMS

5.1. Formal Specification

The modelling framework in ANDRES is based on the existing ForSyDe [2, 4] framework developed at KTH. ANDRES extends ForSyDe by integration of adaptivity into the modelling framework. This is done by the concept of an *adaptive process*, which changes its behaviour depending on special input signals from the environment. The values carried by these input signals can be data values, but also functions or complete processes. Thus adaptation can be modelled at a varying degree of complexity.

ANDRES uses a formally defined, hierarchical heterogeneous MoC, which is illustrated in Figure 2. Processes communicate via signals. There are so-called domain interfaces to formally define the interaction between processes of different computational models. The following MoCs are used in ANDRES: *untimed model*, *synchronous model*, *discrete time model* and *continuous time model*.

Since the designer uses SystemC as a modelling language, the ANDRES project will define modelling rules and guidelines, which guarantee the SystemC models to be compliant with the ANDRES modelling formalism. Thus methods developed for this framework, e.g. for property analysis, verification or transformation, will also be applicable to the ANDRES SystemC models.

Adaptivity is modelled by means of formally defined adaptive processes. The functionality that the adaptive process computes can be changed from the environment depending on the value of an input signal.

Figure 3 shows a simple, but typical example for the modelling of adaptivity. The processes *Encoder* and *Decoder* are both adaptive processes and are fed with signals carrying functions. These functions change the behaviour of the encoder. In the first cycle the encoding function adds one, while in the second cycle two is added to the incoming value. The same adaptivity mechanism is used for the decoder. To yield an implementation this model is refined and non-functional characteristics, like reconfiguration time, are taken into account in subsequent design phases and the associated model.

5.2. SystemC-AMS

The OSCI working group SystemC-AMS is currently working on a prototype allowing designers to simulate systems that combine data-flow modelling (using the Synchronous Data Flow (SDF) MoC), analogue circuits (using the continuous time network (CT-NET) MoC), and rather control oriented digital circuits (using the discrete-event (DE) MoC). Compared with circuit simulators such as SPICE, the focus of SystemC-AMS is on executable specification, design space exploration and virtual prototyping of signal processing and analogue/mixed-signal systems [5]. All these use-cases require high simulation performance while less accurate simulation is acceptable.

Another interesting feature of SystemC-AMS is its extensibility by other models of computation by integrating

new *solvers* for alternative MoCs into the synchronisation layer. However, this feature plays a minor role in the ANDRES project since various MoCs are already provided by HetSC.

In ANDRES the means for specifying and modelling adaptivity will be encapsulated within a class that is inherited from an abstract adaptive object (AAO) class specified in the original system model. Different sets of possible functional behaviour can be expressed by overloading virtual methods. Parametrisation and other kinds of adaptivity can be modelled by parameters, and be made more explicit by attributes. To support system level design of adaptive communication systems, a library of building blocks is being developed. These building blocks focus on the field of communication, radio frequency systems, in particular on signal sources and modulation/demodulation parts (e.g. BASK or QAM), signal analysis (eye diagrams, scatter plots) and bit error rate calculation. Thereby it is explored for which parts dynamic reconfiguration could be advantageous. Such adaptive features will then be implemented by means of the AAO class mentioned above.

5.3. HetSC

HetSC [6] is a methodology enabling heterogeneous specification of complex embedded systems in SystemC. Computational models supported include untimed MoCs, synchronous MoCs and the timed MoCs already supported by SystemC. In ANDRES, HetSC will be used for modelling and generation of embedded software (see Figure 4). Providing several abstract MoCs (e.g. KPN, PN, CSP, SR, etc.), HetSC enables a more intuitive and safer design of concurrent software systems adjusting to different specification needs of the designer.

The HetSC methodology defines a set of specification rules and coding guidelines for each specific MoC making the design task more systematic. The fulfillment of MoC specification rules provides useful properties for concurrent software, such as determinism, deadlock protection, etc. The HetSC library, associated to the HetSC methodology, provides a set of facilities to cover the deficiencies of the SystemC core language for heterogeneous specification. The support of some MoCs requires new specification facilities with specific semantics and abstraction levels. In addition, some facilities of the HetSC library help to detect and locate MoC rule violations and assist in debugging concurrent specifications.

The HetSC based software design flow is called SWGen [7]. SWGen is a methodology for automatic software generation of embedded software from SystemC. Target platform is an embedded system including an embedded real time operating system (RTOS). The development platform has to provide a C++ cross-compiler and a programmer API for the RTOS. The methodology takes as input HetSC code

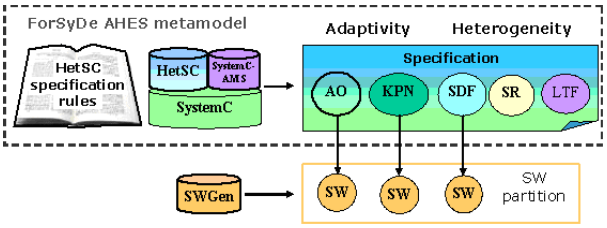

Fig. 4. HetSC in ANDRES

and produces preprocessed C++ code including RTOS system calls. Additionally, the target binary file can be generated. The SWGen flow uses the SWGen library. The library substitutes the code of the SystemC core and HetSC libraries, in charge of supporting the simulation engine and other system-level features, by an efficient implementation, suited to the target platform.

Several tasks are being carried out for integrating the HetSC methodology into the ANDRES design flow. Guidelines and features for connecting HetSC with SystemC-AMS are being developed. This will enable a methodology supporting a wide spectrum of MoCs. This task involves the connection of MoCs, rising several syntactical and semantical issues appearing at the interface between the methodology/MoC interfaces. In addition, it is being studied how the different types of adaptivity defined in ANDRES can be supported by the HetSC methodology. This will provide the ANDRES flow with the ability to support adaptivity in software. A prerequisite is providing an interpretation about what adaptivity is in software. From a general point of view, adaptation is already present in embedded software in the shape of different coding techniques. However, a conceptualisation of its use could actually help to a clean and efficient design and maintenance of the software code demanding such adaptivity features. Finally, the ForSyDe metamodel is being used to formalize the ANDRES methodology, and by extension, the HetSC specification methodology. The latter is being done through an interpretation of the HetSC specification (basically, a set of processes and channels) in terms of the ForSyDe metamodeling facilities (basically, a set of signals and process constructors).

5.4. OSSS+R

OSSS+R is a SystemC based modelling library providing high-level language constructs enabling application-driven modelling (self-)reconfigurable hardware systems. As all OSSS+R elements have a well-defined synthesis semantics, designs can automatically be mapped to platforms supporting dynamic partial reconfiguration (see Section 6).

Based on OSSS [8], in OSSS+R object-orientation is used as an adequate abstraction mechanism for dynamically reconfigurable hardware. The concept is based on the as-

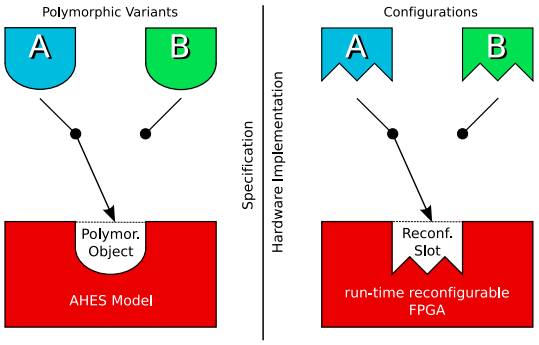

Fig. 5. Polymorphism and configurations

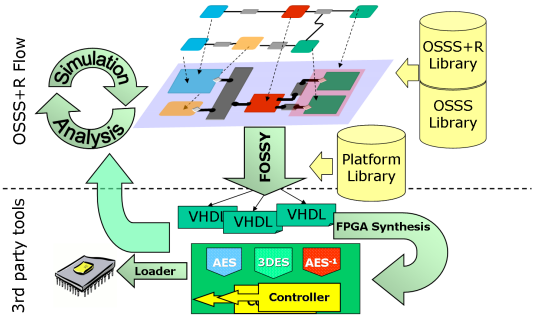

Fig. 6. OSSS+R synthesis flow

sumption that changing functions of parts of a hardware system largely resembles the use of polymorphism in object-oriented software design [9].

Polymorphism, as used in object-oriented programming, enables calling methods on an object, whose exact type is unknown to the caller. The only known reference to the object is its *interface*. Depending on the actual class of the object, the corresponding implementation of a method is executed. This enables changing parts of the software at run-time without modifying the static part of the code.

Considering a digital hardware system consisting of a static and a dynamically reconfigurable part, it is obvious that the interface between the two parts needs to be fixed. However, the functionality of the reconfigurable hardware may change. Hence, the key idea of OSSS+R is to model the reconfigurable area of a hardware system as an adaptive (polymorphic) object with a fixed interface. This *interface* is defined by a base class, while its possible variants belong to different subclasses. During run-time, different variants of the adaptive object can be configured and used (see Figure 5).

To handle the management of different object configurations and to ensure persistance OSSS+R introduces *Named Contexts*. A context represents all relevant information of an object, including its current type and state. From the designer's point of view, a context is used similar to a pointer in C++, however automatically instantiated infrastructure ensures that a context is enabled (i.e. configured) if it is accessed by a user process. Its state is automatically saved and restored during consecutive reconfigurations. Because a context can be accessed concurrently, incoming request are serialised using a built-in scheduler. Contexts hide the complexity of configuration management and state preservation and enable the designer to use adaptivity transparently.

5.5. Polymorphic Signals

When modelling heterogeneous systems, typically several different MoCs are used for different subsystems. Moreover, the MoC used for a specific subsystem may change during system development, e.g. when passing over (due to a top-down refinement) from an abstract description of a low pass filter using a transfer function within the SDF-MoC to an explicit one using an RC circuit within a continuous time MoC.

Therefore, two problems arise: to use appropriate converter modules when connecting systems components described within different MoCs, and to exchange these when changing the MoC for a specific subsystem. This has to be done manually and is therefore time consuming and potentially error-prone. To overcome these problems, the concept of *polymorphic signals* was developed and prototypically implemented in SystemC-AMS [10].

Polymorphic signals are able to convert value types (e.g. real to bit-vector) and semantics (e.g. SDF-MoC to DE-MoC) automatically. This is achieved by instantiating appropriate converter modules depending on the types of the ports the signal is connected to. Polymorphic signals will be used within ANDRES to support the coupling of system components using different modelling libraries and MoCs and to support the interactive performance analysis of heterogeneous systems by mixed-level simulation.

6. SYNTHESIS OF RUN-TIME RECONFIGURABLE DIGITAL HARDWARE

One of the major goals of the ANDRES project is the automated hardware synthesis for dynamic reconfigurable architectures. The starting point for synthesis is OSSS+R. In this case *synthesis* covers the translation of a given OSSS+R model to RT-level VHDL, which in turn serves as input for third-party backend tools, e.g. Xilinx' *Early Access Partial Reconfiguration Design Flow* [11] (see Figure 6). The synthesis tool FOSSY (Functional Oldenburg System SYnthesizer) for OSSS will be extended to support the language constructs for reconfigurable components introduced with OSSS+R.

The major transformation step towards a pure RTL design from an OSSS+R model consists of the generation of various management structures. In addition to the applica-

tion and annotations given by the designer, different arbitration mechanisms, structural information (e.g. FPGA types) etc. need to be considered. The generated infrastructure consists of a set of hierarchically organised controllers. A set of distributed controllers for each reconfigurable area handles access requests by the static design parts. Each of these access controllers uses a central reconfiguration controller per device to accomplish reconfigurations. That unit resolves conflicts between different distributed access controllers and provides an interface to the FPGA's configuration port.

The required interfaces to the reconfigurable areas can be determined during synthesis by analysing the interfaces of the corresponding *Named Contexts* that are bound to the area. This even allows the synthesis of static signal-level interfaces for unrelated method interfaces bound to a single reconfigurable area on the application layer.

For each possible functional content of a *Named Context*, a VHDL implementation of the behaviour is generated separately. In the later steps of the synthesis flow each of these functional blocks can be used for the generation of the required partial bitstreams.

One of the crucial aspects of system-on-chip design, is the generation of efficient communication infrastructure between system components. ANDRES will extend the approaches for communication refinement and synthesis developed during the ICODES project [12]. In particular, the communication between adaptive objects will be studied. One primary goal is the automatic generation of hardware/software interfaces, which will enable, for example, to access and to control adaptive hardware objects from software and vice versa. Also the interaction between the digital and the analogue components is investigated.

7. CONCLUSION

This paper presented motivation, goals and ongoing work of the ANDRES project. While the heterogeneous nature of embedded systems still yields many problems in current design flows due to different computational models, languages and tools, these methodologies particularly lack support for adaptive architectures. To resolve these issues, ANDRES is developing a seamless design flow for such adaptive heterogeneous embedded systems and corresponding tools including automatic synthesis of software and runtime reconfigurable digital hardware, like FPGAs.

Based on a formalism to express and analyse adaptivity in different Models of Computation, three SystemC based modelling libraries are used for specification, simulation and analysis of AHES designs: SystemC-AMS for mixed-signal components, HetSC for software and OSSS+R for run-time reconfigurable digital hardware. An overall framework includes and connects these libraries using *polymorphic signals*. This modelling framework provides entry points for automatic synthesis of hardware, software and communication infrastructure. The synthesis concept particularly considers different types of adaptivity and adaptive architectures and will be implemented in corresponding synthesis tools.

The concepts and tools of ANDRES are evaluated using industrial use-cases. As a result, ANDRES will provide a complete design flow and tools based on SystemC. At the end of the project, the modelling framework is planned to be released to the public.

8. REFERENCES

[1] ANDRES *project*, Website, http://andres.offis.de.

[2] A. Jantsch, *Modelling Embedded Systems and SoCs*. Morgan Kaufmann, June 2003.

[3] E. A. Lee and A. Sangiovanni-Vincentelli, "A Framework for comparing Models of Computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 12, December 1998.

[4] I. Sander and A. Jantsch, "System Modeling and Transformational Design Refinement in ForSyDe," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 17–32, January 2004.

[5] A. Vachoux, C. Grimm, and K. Einwich, "Towards Analog and Mixed-Signal SoC Design with SystemC-AMS," in *IEEE International Workshop on Electronic Design, Test and Applications (DELTA'04)*, Perth, Australia, 2004.

[6] F. Herrera and E. Villar, "A framework for embedded system specification under different models of computation in SystemC," in *Proceedings of the Design Automation Conference*, 2006. [Online]. Available: http://www.teisa.unican.es/HetSC

[7] V. Fernandez, F. Herrera, P. Sanchez, and E. Villar, *Embedded Software Generation from SystemC*. Kluwer, March 2003, ch. 9, pp. 247 – 272. [Online]. Available: http://www.teisa.unican.es/HetSC

[8] H. Kleen, T. Schubert, and C. Grabbe, "A tutorial for OSSS," OFFIS Institute, Oldenburg, Germany, Tech. Rep., January 2006, http://icodes.offis.de. [Online]. Available: http://icodes.offis.de

[9] A. Schallenberg, F. Oppenheimer, and W. Nebel, "OSSS+R: Modelling and Simulating Self-Reconfigurable Systems," in *Proceedings - 2006 International Conference on Field Programmable Logic and Applications*, Aug. 2006, pp. 177–182.

[10] R. Scholl, C. Grimm, and K. Waldschmidt, "Heaven: A Framework for the Refinement of Heterogeneous Systems." in *Proceeding of the Forum on Specification and Design Languages (FDL '04), Lille, France*, Sept. 2004.

[11] *Early Access Partial Reconfiguration User Guide (UG208)*, Xilinx, Inc., Mar. 2006. [Online]. Available: http://www.xilinx.com/xlnx/xil_entry2.jsp?sMode=login&group=prealounge

[12] ICODES *project*, Website, http://icodes.offis.de.