OSSS+R: SIMULATION AND SYNTHESIS OF SELF-ADAPTIVE SYSTEMS

Philipp A. Hartmann⋆, Andreas Schallenberg†, Frank Oppenheimer⋆, Wolfgang Nebel†
⋆OFFIS Institute – †Carl v. Ossietzky University – Oldenburg, Germany

philipp.hartmann@offis.de http://andres.offis.de

Abstract

We present the modelling of (self-)adaptive systems with the OSSS+R library, which is based on SystemC^{TM}. Additionally, an FPGA-based reconfigurable demonstrator shows first synthesis results.

1. Introduction

The inclusion of (self-)adaptivity into today's system design is gaining importance for the development of flexible and efficient systems. Due to the heterogeneous nature of these systems, adaptivity has to be considered in different domains: digital hardware, analogue hardware and even software, each of it coming with its own computational models, languages and design tools. The main objective of the ANDRES project (*ANalysis and Design of run-time Reconfigurable heterogeneous Embedded Systems*, [1]) is to develop an integrated modelling approach that allows to seamlessly specify, simulate, synthesise, and verify such adaptive heterogeneous embedded systems (AHES). A tool to translate adaptive models into RTL descriptions is currently being developed, targeting run-time reconfigurable architectures such as FPGAs.

This presentation focuses on the modelling, simulation, and synthesis of adaptive *digital* hardware components. Since today's common system description languages do not support the modelling of the specific aspects of reconfigurable systems, the ANDRES project uses and extends the OSSS+R library [5], initially developed during the POLYDYN project [4]. The OSSS+R approach (*Oldenburg System Synthesis Subset + Reconfiguration*) presented here extends OSSS with capabilities to model, simulate and synthesise reconfigurable architectures. OSSS [2] itself is a library for modelling, simulation, and synthesis of object-oriented descriptions based on SystemC^{TM} [3].

2. OSSS+R library

The OSSS+R library provides high-level language constructs to model (self-)reconfigurable systems. Therefore, the designer can focus on the modelling of the application, without having to cope with the low-level management of different configurations.

Following the object-oriented paradigm of OSSS, the reconfigurable components are modelled as objects in OSSS+R. The objects in OSSS+R largely resemble the concept of polymorphism mapped onto run-time reconfigurable hardware. This is due to the fact that polymorphic objects in the modelling world and reconfigurable areas on an FPGA share important properties:

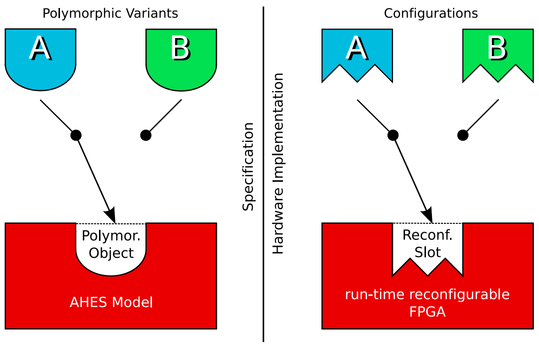

Figure 1: Polymorphism and configurations

Using polymorphism means having a typed reference to an object, whose type is not necessarily exactly known. The object instance and object type may be exchanged during runtime but the type of the reference does never change. The reference is the *interface* to the object's contents, just like the interface of the reconfigured area, which provides access to the different configurations.

On the other hand, reconfiguring a part of an FPGA during runtime requires that the signal-level interface to the non-changing part must be static. This common, or at least compatible, interface allows the communication between those parts, although the functional core of the reconfigured part has been modified and provides new functionalities.

The above mentioned references are called *Named Contexts* in OSSS+R. These contexts are bound to a *Recon-Object* – a shared ressource that represents the reconfigurable area after synthesis. Access to contexts corresponds to access to the *Recon-Object*, which handles the reconfiguration to provide the requested functionality transparently, if needed. Conflicting accesses to a *Recon-Object* are serialised and therefore scheduled, if necessary (see Figure 3, and Section 4.). Additional properties such as reconfiguration times can be specified, for details see [5]. The OSSS+R simulation library will be publicly available just as the OSSS simulation environment already is [2].

Implementing a dynamic design may require mechanisms for the state-preservation of configurations, the detection of necessary adaptations, and the resolution of conflicts due to parallel requests to exclusive resources. Such infrastructures are complex and their manual design is an error-prone and time-consuming task. The OSSS+R modelling library relieves this burden from the designer, leading to shorter development times of adaptive systems.

3. Example application

At the booth we present both an OSSS+R model and an actual hardware implementation of a cryptographic co-processor. In the example application, the different encryption and decryption algorithms (Triple-DES, AES, AES^{-1}) are used mutually exclusive. Therefore, the corresponding objects are bound to a *Recon-Object*. The model allows a cycle-accurate simulation of the application including the influence of the dynamic reconfiguration on the timing behaviour and demonstrates various language features of OSSS+R.

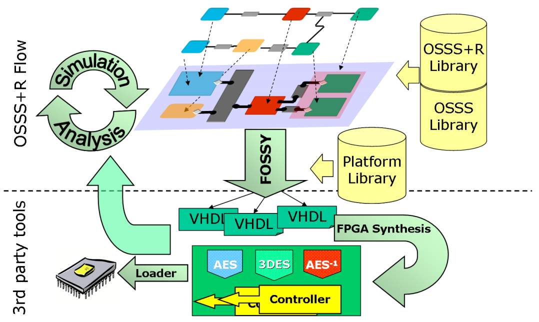

Figure 2: OSSS+R Design Flow

As a first step towards an automated synthesis, this example has been manually translated to VHDL and implemented on a Xilinx ML401 evaluation board, using standard backend tools. The resulting architecture on the FPGA follows the synthesis concepts of OSSS+R closely and exemplifies the feasibility of our approach. The proposed design flow is shown in Figure 2.

4. Synthesis

One of the major goals of the ANDRES project concerning digital hardware is the automated synthesis of *Adaptive Objects*. "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* [6]. The synthesis tool FOSSY (Functional Oldenburg System SYnthesizer) for OSSS is currently being extended to support reconfigurable components and their special properties.

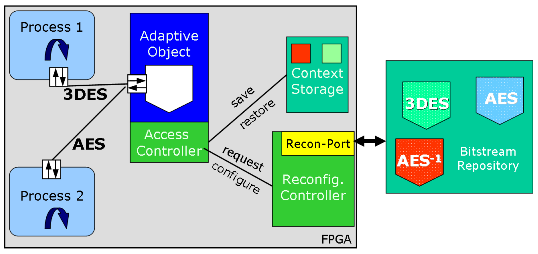

Figure 3: OSSS+R architecture of crypto-application

The major transformation step towards a pure RTL design from the OSSS+R model consists of the generation of various management structures. Additional to the application 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 (see Figure 3). A set of distributed controllers for each reconfigurable area handles access requests by the static design parts. Requests are delayed when the access requires a reconfiguration or other accesses are currently processed. Each access controller uses a central reconfiguration controller per device to accomplish reconfigurations. That unit resolves conflicts between different distributed 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 *Recon-Object*. This even allows the synthesis of static signal-level interfaces for unrelated interfaces bound to a single reconfigurable area on the application layer.

For each possible functional content of a *Named Context*, resp. *Recon-Object*, 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.

5. Conclusion

In this work we have presented the modelling, simulation, and synthesis of self-adaptive digital hardware components with OSSS+R. A simple demonstrator, which can dynamically exchange different cryptographic algorithms during run-time, has been developed as an OSSS+R model with accompanying, cycle-accurate simulation and as a manually synthesised FPGA prototype implementation.

The implementation of the demonstrator follows the synthesis concepts (see [5]), that serve as the basis for the currently ongoing development of automated synthesis tools for OSSS+R models.

References

[1] ANDRES *project*. `http://andres.offis.de`.
[2] ICODES *project*. `http://icodes.offis.de`.
[3] Open SystemC Initiative. *SystemC*^{TM}. `http://www.systemc.org`.
[4] POLYDYN *project*. `http://ehs.informatik.uni-oldenburg.de/en/research/projects/polydyn`.
[5] 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*, pages 177–182, Aug. 2006.
[6] Xilinx, Inc. *Early Access Partial Reconfiguration User Guide (UG208)*, Mar. 2006.

Acknowledgements

The ANDRES project is co-funded by the European Commission within the Sixth Framework Programme (IST-5-033511). The POLYDYN project is funded by the DFG under grant No. NE 629/7.