Hardware Design and Synthesis in ForSyDe

Ingo Sander, Alfonso Acosta, and Axel Jantsch
Royal Institute of Technology, Stockholm, Sweden

The ForSyDe (Formal System Design) methodology has been developed with the objective to move system-on-chip design to a higher level of abstraction. ForSyDe provides libraries for several models of computations (MoCs), which allow to model and simulate heterogeneous systems consisting of analog hardware, digital hardware, and software. ForSyDe is implemented as an EDSL (Embedded Domain Specific Language) on top of the Haskell programming language. A system is described as a set of concurrent processes and domain interfaces, which communicate with each other via signals. All processes are implemented by process constructors. A process constructor is a higher-order function taking functions and values as input and producing a combinational or sequential process as output. Process constructors separate communication from computation and are the base for a formal model that allows for design transformation [5]. In addition process constructors have a structural interpretation, which we use as basis for hardware synthesis.

Historically, the development of ForSyDe has been dominated by the modelling purpose [4]. Systems have been modelled as streams of data isomorphic to lists. This shallow-embedded version of ForSyDe allows to model heterogeneous systems, but its use is restricted to simulation. Currently the following models of computation with their own set of process constructors are supported: continuous time MoC, synchronous MoC, and untimed MoC.

Recent work has focused on on an additional deep-embedded implementation of ForSyDe in order to offer compilation and model-transformation features to the designer. The inclusion of a hardware-synthesis backend has been the main motivation for this development. Deep-embedded signals are used in a similar way as shallow-embedded signals. However, strongly inspired by Lava [2] [3], deep-embedded signals are modelled as an abstract data type, which, transparently to the end user, keeps track of the system structure. ForSyDe's deep-embedded implementation uses Template Haskell, which allows to express computations in plain Haskell without the need to design yet another DSL for that purpose. Based on the structural information, ForSyDe's embedded compiler [1] can perform different analysis and transformations such as simulating the system or translating it to other target languages (e.g. synthesizable VHDL and GraphML). So far, the deep-embedded ForSyDe implementation can only be used to design systems belonging to the synchronous model of computation. This limitation is, however, likely to change in the future.

In contrast to other methods, a strong point of ForSyDe is the possibility to combine shallow-embedded and deep-embedded system models and to simulate them as one integrated heterogeneous system model. This allows to validate the deep-embedded synthesizable model inside a complex and heterogeneous environment, which may even include models of analog circuits and analog-digital converters.

Our presentation will illustrate hardware design in ForSyDe from the perspective of the designer. Guided by a concrete example, we will briefly introduce ForSyDe's shallow-embedded heterogeneous modelling framework, which the designer will use for both rapid-prototyping and design validation. The main focus of the presentation will be on the deep-embedded implementation of ForSyDe and its use for hardware synthesis. For this discussion we will use a subsystem of the initial example and its adaptation to the deep-embedded implementation of ForSyDe. We will not only emphasize synthesis-related ForSyDe features, such as fixed-size vectors, customized data types and reuse through components, but will also discuss to what extent the use of Template Haskell has affected the modelling environment of ForSyDe. We will evalute the generated VHDL-code with respect to quality and suitability for logic synthesis using Altera Quartus II as synthesis tool and an Altera FPGA as target architecture. Finally, limitations and directions for future work will be discussed.

References

[1] ForSyDe: Formal System Design. `http://www.ict.kth.se/org/ict/ecs/sam/projects/forsyde/www/`.

[2] Per Bjesse, Koen Claessen, Mary Sheeran, and Satnam Singh. Lava: Hardware design in Haskell. In *International Conference on Functional Programming*, pages 174–184, Baltimore, Maryland, USA, September 1998.

[3] Koen Claessen and David Sands. Observable sharing for functional circuit description. In *Proceedings of the 5th Asian Computing Science Conference on Advances in Computing Science (ASIAN'99)*, pages 62–73, 1999.

[4] Axel Jantsch. *Modeling Embedded Systems and SoCs*. Morgan Kaufmann, 2004.

[5] Ingo Sander and Axel Jantsch. System modeling and transformational design refinement in ForSyDe. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(1):17–32, January 2004.