A Flexible Specification Framework for Hardware-Software Codesign

José Manuel Moya†, Santiago Domínguez‡, Francisco Moya†, Juan Carlos López†

†Universidad de Castilla-La Mancha
Departamento de Informática
ESI. Pº Universidad 4, 13071 Ciudad Read, Spain
{jmmoya,fmoya,jclopez}@inf-cr.uclm.es

‡Universidad Politécnica de Madrid
Departamento de Ingeniería Electrónica
ETSIT, Ciudad Universitaria, 28040 Madrid, Spain
sdb@die.upm.es

Abstract

In this poster, we present a new specification technique for complex hardware-software systems, based on standard *high-level programming languages, such as C, C++, Java, Scheme, or Ada, without extensions or semantic changes. Unlike previous approaches, the designer may choose the model of computation and the specification language that best suits her needs, while still being able to formally verify the correctness of the specification. The details of the available hardware and software resources, and the implementation of the different models of computation are encapsulated in libraries to maximize reuse in system specifications.*

Figure 1 shows the basic architecture of our specification framework. The actual model of the application is built on top of two libraries: `libarch` and `libMoC`.

The `libarch` library encapsulates the details of the specific hardware and software resources, so that the system specification remains independent of the target system. Architectural exploration of different implementation alternatives is done by modifying this library.

The `libMoC` library encapsulates the details of the specific model of computation. To be verifiable, the application code should use the interface provided by this library. Then, we use external tools to verify different properties of the model of computation. For every pair (specification language, model of computation) we define a new MoC library component, with well-defined interfaces, providing support of the precise semantics of that model of computation from the specification language. `libMoC` currently supports C++, Java, Ada, and Scheme programming languages, and FSMs, Petri Nets and Dataflow models of computation, but it can be easily extended to support other languages and models of computation.

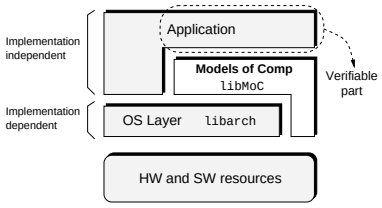

Figure 1. Basic architecture of the specification framework

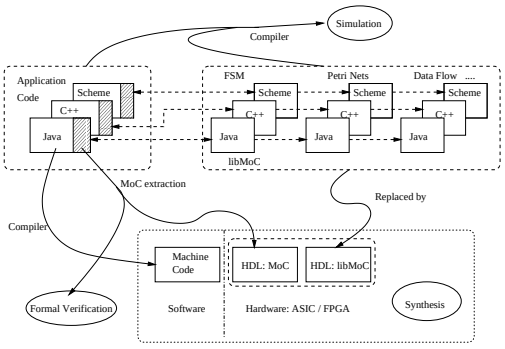

Figure 2. Simulation, formal verification and synthesis using `libMoC`

The complete system can be simulated compiling and executing the application code with software versions of `libarch` and `libMoC`.

To support verification and synthesis, we link the specification with special versions of the MoC libraries. The information required by the verification and synthesis tools is generated when the resultant binary is executed.

The described process is depicted in figure 2. It is important to remark that the same application code is used for simulation, verification and synthesis.

The main benefits obtained from our approach can be summarized as follows:

- The designer is free to choose the system specification language and the underlying model of computation, independently of the final implementation (software or hardware) of the components. New specification languages and MoCs can be added easily.
- Simulation of the whole system is straightforward and available tools for formal verification can be used for the different models of computation implemented in `libMoC`.
- Hardware components can be efficiently synthesized for the part of the system that uses `libMoC`.