Chapter 16
SC² StateCharts to SystemC: Automatic Executable Models Generation

Marcello Mura and Marco Paolieri

Abstract The recent development of embedded systems calls for the necessity of a complete framework for design and simulation of applications that span through all levels of system design. Desirable characteristics of such a framework are rapidity of use, simplicity and reusability. For this purpose we already introduced a generator that converts specifications written with a subset of StateCharts to behavioral SystemC [16, 17]. We present here the new version of our tool: most of the limitations of the previous versions have been overcome, the considered subset of the StateCharts formalism has been extended and the target has been changed from behavioral to Register Transfer Level (RTL) SystemC. A major enhancement of this new version is the possibility of obtaining various module instances starting from a single specification, which is vital in some contexts (e.g. Wireless Sensors Networks simulation). The semantics chosen for our StateCharts diagrams is clearly described. The generation of executable models, as well as the kernel template of the generated code, are discussed in detail.

16.1 Introduction

The possibility of generating customized simulators to model a relevant subset of systems in a very effective way could open interesting scenarios in early design phases (even before Hardware/Software partitioning [12]), especially when intrinsic complexity related to the projects is such that people with different expertise need to cooperate. In fact within this kind of framework it is possible to design, in a very short time, virtual prototypes that can be used for requirements formalization and validation. Moreover systems under development could be extensively tested from the very beginning up to advanced design stages with the same tool, incrementally integrating the model level of definition. Functional and non-functional
properties can be analyzed, e.g. using such kind of instruments we analyzed power consumption of a networking protocol in [14, 15] and of the cache memory of a microprocessor in [16].

In our work the emphasis is on the model: our main contribution is in fact a model-based generator of simulators that – starting from dynamic information about a system expressed with a convenient subset of the StateCharts formalism – generates well structured RTL SystemC code for simulation. The framework is organized in a way that it is possible to iteratively refine models up to a point that the generated code is very near to the synthesizable level. During this process results can be compared, allowing for an easier development process.

In Section 16.2 related work is described. The semantics of the StateCharts dialect we use are presented in Section 16.3 and compared with the most important variants. The methodology for extracting information from UML diagrams and using it to create SystemC models is briefly outlined in Section 16.4. Section 16.5 presents a major innovation of our work: the possibility of performing multi-instance simulation. A small example showing the most noticeable features of our framework as well as the introduction of a shell console is illustrated in Section 16.6. Conclusions and further work are outlined in Section 16.7.

16.2 Related Work

In the past ten years there has been a consistent research effort on this subject, leading also to commercial software products. I-Logix StateMate [1] generates executable models starting from UML diagrams, MATLAB Stateflow [2], does the same starting from a concurrent FSM formalism similar to that of StateCharts. In [4] the translation of StateCharts into Hierarchical Finite State Machines (HFSMs) is explored in order to build test cases for the corresponding VHDL realization. StateCharts formalism is also very appropriate for the formal validation of models. In particular, automatic translation into Promela/SPIN, a language used for automatic model checking, was presented in [5, 10, 13]; recently an interesting approach to this problem was reported in [9]. The present research effort aims at building a framework for the generation of simulators. It differs from the commercial products ([1, 2]) first of all for the choice of SystemC as a target language for the generated models [11] so that they can be inserted in already existing SystemC simulation frameworks. It has simpler semantics allowing for an easier customization. As a result the simulator code is clearly structured and easy to understand and manage. Moreover it is possible to use the generated model as an entry point for successive refinement phases leading possibly to HW synthesis. The output can be reduced to a minimum, therefore simulations are quicker and this greatly extends the range of applicability (i.e. contexts in which simulations for long periods of time are necessary). The use of SystemC is particularly indicated for modeling purposes, e.g. in [18] and in [20] SystemC code is generated starting from UML representations, with the final purpose of creating a hardware synthesis. Differences between these