

An HW/SW Co-design Environment based on UML and SystemC

Elvinia Riccobene
Università di Milano
Dip. Di Tec. dell'Inf.
Bramante 65, Crema, Italy
riccobene@dti.unimi.it

Patrizia Scandurra
Università di Catania
Dip. Di Mat. e Inf.
A. Doria 6, 95024, Catania, Italy
scandurra@dmi.unict.it

Alberto Rosti and Sara Bocchio
STMicroelectronics Lab R&I
C.d.Colleoni 20041 Agrate, Italy
{alberto.rosti,sara.bocchio}@st.com

ABSTRACT

This paper outlines some fundamental concepts for the development of a system design framework based on standard notations and common CASE tools. We describe an environment for HW/SW co-design of embedded systems based on the Unified Modeling Language (UML) and SystemC. Taking advantage from the capabilities provided by widely used UML tools, this environment provides code generation for co-design of hardware and software.

1. INTRODUCTION AND RELATED WORKS

System level design flow for System on Chip (SoC) starts by writing the system specifications and developing a functional model from them. The system is refined through a set of abstraction levels, towards a final implementation in hardware and software. Nowadays it is an emerging practice to develop the functional model and refine it with the SystemC language [7]. The hardware part of the system goes down to the RTL level for the synthesis flow, while the software part, can be either simulated at high level (functional or transactional) or it can be compiled for an Instruction Set Simulator (ISS).

In our opinion a further improvement to this design flow can be achieved by extending lightweight software modeling techniques as UML [9] to describe the system specification and to generate from it an equivalent description in SystemC. In this paper we present an UML profile for SystemC that, integrated in an UML tool, provides a schematic entry for SystemC by UML diagrams. Moreover, we included it in a whole framework that allows to model hardware, which is implemented in SystemC, and software, which can be simulated by an ISS within the GEZEL co-simulation environment.

A lot of research activity has been focused on analyzing the possibility to extend UML to embedded system design: [10] and [6] define an UML profile and analyze the existing features from the UML standard that can be useful in order to model embedded real time applications. In [2] a specialization of UML is presented to express embedded real-time applications in an abstract way. [1] defines an UML Profile for SystemC, but no code generation capabilities for behavioral information are considered. Moreover all of these proposals are based on the old versions of UML (the so called UML 1.x).

In [5] the authors provide an analysis about the structural modeling concepts in UML 2.0, which has added structural information such as class and structured class. In August 2004 IBM, Fujitsu and NEC submitted to the Object Management Group a proposal of standardization for an UML profile for SoC based on UML 2.0 and on SystemC as a target language. There are evident commonalities between this proposal and our approach, but there are also significant differences because our profile is isomorphic to SystemC while the IBM/Fujitsu/NEC proposal introduces new constructs for SoC modeling that are not present in SystemC. Moreover, this profile leaves the SystemC methods empty, while our approach uses UML state diagrams to describe those behavioral parts.

In section 2 we describe the main concepts of our design framework, section 3 and 4 gives respectively more details about the UML profile for SystemC and the encapsulation of GEZEL environment and section 5 draws some conclusions about the fundamental concepts that inspired this work.

2. THE DESIGN ENVIRONMENT

In order to develop our system design environment, we decided to rely on tools supporting UML 2.0, since only these tools provide the features needed to model the system structure, such as classes with ports communicating through interfaces and composite structure diagrams. Using such UML tools adds the capability to model the system using a diagrammatic UML representation and to perform code generation. Our current implementation is based on Enterprise Architect [3] but any other tool supporting UML 2.0 can also be easily used.

To allow using UML for HW/SW co-design, we start extending UML by a profile for SystemC [13] that allows expressing a SystemC model in UML. A profile is a standard mechanism for extending UML by adding a collection of domain specific notation made of stereotypes, tagged values and constraints, all with the proper semantic. We added the capability to generate SystemC code from the UML model. This feature can be obtained either by customizing the scripting generation capabilities of the UML tool or by exporting the model in XMI format and generating the SystemC code from it. We choose the last approach since it is more general. A SystemC code generator specific for Enterprise Architect has been developed in a reasonable effort. An important note about our environment is that it allows the complete description of the behavior by a proper action semantics that allows to express SystemC operation within the state and arcs of the enhanced state machine of diagrams.

We need also to map hardware and software parts to different implementation paths. Starting from the UML description, the hardware parts are described in SystemC profile, while the software parts can be described either in SystemC profile at functional or transactional level, or in C that can be executed on an ISS for cycle or instruction accurate simulation. We have chosen the GEZEL [4] tool as a cycle accurate cosimulation framework in our design framework because it allows to encapsulate different ISSs and to cosimulate in the SystemC environment. Since Gezel is a multiprocessor simulation environment, this tool allows us to build a unique interface between Gezel (that maps the software part on different processors) and the SystemC description of the hardware part rather than building a SystemC interface for every ISS that is used in the design.

Extension of the profile to include the SystemC Verification Library is also under development. Another ongoing activity is the development of the reverse engineering part from SystemC to UML.

3. UML PROFILE FOR SYSTEMC

This section introduces a model-driven SoC design flow based on the UML 2.0 [9] and SystemC [7] as system-level design languages, exploiting the Model Driven Architecture (MDA) [8] framework. Through metamodels, the MDA framework provides a mechanism to define modeling languages in an unambiguous way and to make the languages and the models written in these languages understandable to transformation tools. A metamodel is a precise definition of the constructs and rules needed for creating semantic models in a given language. The transformation rules, conforming to a (not always trivial) transformation definition, describe how a model in a source language can be transformed into a model in a target language by the metamodels.

A UML profile is a set of stereotypes each of which defines how the syntax and the semantics of an existing UML metaclass are extended for a specific target domain. A stereotype can define additional constraints expressed as formula in the Object Constraint Language (OCL) over its base metaclass as well as tags to state additional properties.

3.1 The UML Profile for SystemC

The UML profile for the SystemC language is based on the UML 2.0 specification [9] and on the SystemC 2.0 specification [7]. The profile definition is organized in four parts which reflect the SystemC language architecture:

1. The **SystemC CORE layer: structure and communication** defines stereotypes for the primitive building blocks of the core layer (or layer 0) of SystemC. They are used in various UML structural

diagrams (such as UML class diagrams and composite structure diagrams) to represent hierarchical structures and communication blocks made of modules, interfaces, ports and channels.

2. The **SystemC CORE layer: data types** defines a UML class library for representing the set of SystemC data types.
3. The **SystemC CORE layer: behavior and synchronization** part defines stereotypes which lead to a variation of the UML method state machines, the *SystemC Process State Machines* [12], to allow high level specification of the behavior of SystemC processes (methods and threads) within modules and channels.
4. The **SystemC layer of predefined channels, ports and interfaces** provides concepts for the layer 1 of SystemC. These concepts are implemented both as a class library, built with the basic group of stereotypes of the SystemC core layer, and as a group of standalone stereotypes – the extended SystemC profile – which specialize those defined for the SystemC core layer.


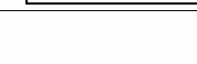

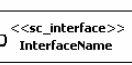





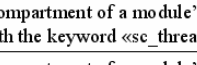
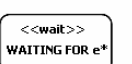
STRUCTURE AND COMMUNICATION	
MODULE	 
PORT	
INTERFACE	  
PRIMITIVE CHANNEL	 
HIERARCHICAL CHANNEL	 
THREAD PROCESS	Within the operation compartment of a module's class or of a channel's class with the keyword <code><<sc_thread>></code> .
EVENT	Within the operation compartment of a module's class or of a channel's class with the keyword <code><<sc_event>></code> .
BEHAVIOR AND SYNCHRONIZATION	
THREAD PROCESS	As a UML <i>method state machine</i> .
EVENT	As a label for a signal trigger on a state machine transition.
WAIT STATEMENT	

Figure 1 UML notation for SystemC concepts

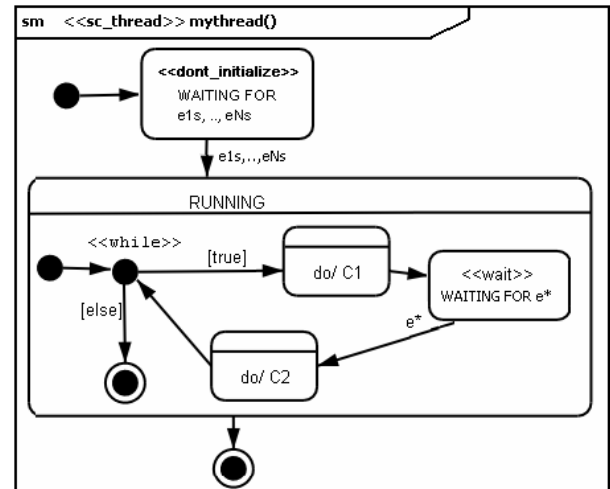


Figure 2 A Thread Process pattern

The complete UML profile definition for SystemC can be found in [11]. In Fig. 1 we report the most significant stereotypes elements of the SystemC core profile. The figure is split in two parts: the first part shows the core layer stereotypes used in UML structural diagrams (like class diagrams and composite structure diagrams) to represent the hierarchical structure of a SystemC specification. The second part shows the core layer stereotypes used in UML behavioral diagrams (such as UML method state machines) to model the functionality expressed by processes in a SystemC specification. The processes are the basic unit of execution in SystemC, and they provide the mechanism to simulate concurrent behaviour. Methods, threads and clocked threads are the processes available in SystemC and they have different behavior, but basically all processes: (i) run concurrently; (ii) are sequential, and not hierarchical, i.e. no process can call another process; (iii) are activated on the base of their own static sensitivity, which consists of a list of some designated events and can dynamically change at run time realizing the so called dynamic sensitivity mechanism.

To model the behaviour of reactive processes of SystemC, we exploit the UML 2.0 method state machine, i.e. a state machine that specifies the procedure or algorithm for a behavioural feature (such as a class's operation) and reacts to specific events occurrences. We defined a new graphical formalism [12], called SystemC Process State Machine, which allows high level specification of the functionality of SystemC processes, and generation

of efficient and compact executable SystemC code from the UML model. Moreover, for a SystemC process we distinguished different behavior cases on the base of the process sensitivity, the process initialization, and the process termination. Every SystemC process matches a specific abstract behavior pattern, where “abstract” denotes that the pattern, and therefore the state machine associated to the pattern, may be refined to add further details depending on the specific functionality of the process. Fig. 2 depicts one of these behavior patterns. It corresponds to a thread process which: (i) has both a static (the event list e_{1s}, \dots, e_{Ns}) and a dynamic sensitivity (the state WAITING FOR e^*), (ii) runs continuously (the infinite while loop), and (iii) is not initialized.

4. GEZEL ENCAPSULATION

GEZEL is a language and open environment developed in UCLA [4, 14] for exploration, simulation and implementation of multiprocessor system on chip and embedded hardware.

A specialized language allows the representation of the micro architecture of domain-specific processors. GEZEL uses cycle-true semantics with dedicated modeling of control structures (FSMD). The simulation back-end is an open C++ library that enables easy integration of GEZEL into different host environments. GEZEL has two different cases of use: as a standalone environment, it works as a hardware exploration environment, and, when linked with an instruction-set simulator, it becomes a co-design environment.

Several co-simulation interfaces are available to different instruction-set simulators as well as to SystemC: this is the main feature that makes GEZEL suitable as component in our flow. GEZEL can simulate a C-compiled code on the ISS independently of the rest of the application which can be written in a pure SystemC code. This means that the UML generation of C/SystemC will be independent from GEZEL environment. A SystemC interface has been developed by Gezel’s authors on our specific request so that it can be used in our environment.

5. CONCLUSIONS

We think that the design of a CAD environment should be based on standard notations so to exploit existing tools and frameworks. The Soc design approach that we present in this paper combines together a lightweight method (UML), a coding language (SystemC) and a HW/SW cosimulation environment (GEZEL) and allows building up a seamless environment exploiting the capabilities of all these notations synergically.

6. REFERENCES

- [1] F. Bruschi and D. Sciuto. A SystemC based Design Flow starting from UML Model. In *Proc. of ESCUG '02*.
- [2] R. Chen, M. Sgroi, L. Lavagno, A. S. Vincentelli and J. Rabaey. UML And Platform Based Design. In *UML for Real design of Real-Time System*. Kluwer Academic Publisher, 2003.
- [3] The Enterprise Architect Tool. <http://www.sparxsystems.com.au/>
- [4] The GEZEL Design Environment. <http://www.ee.ucla.edu/~schaum/gezel/>
- [5] S. Gerard and F. Terrier. UML for Real-Time In *UML for Real design of Real-Time System*. Kluwer Academic Publisher, 2003.
- [6] O. Haugen, B. Moller-Pedersen and T. Weigert. Structural Modeling with UML 2.0. In *UML for Real design of Real-Time System*. Kluwer Academic Publisher, 2003.
- [7] G. Martin, S. Swan, T. Grötter and S. Liao. *System Design with SystemC*. Kluwer Academic Publisher, 2002.
- [8] OMG. Model Driven Architecture. <http://www.omg.org/mda/>
- [9] OMG. UML 2.0 Superstructure, ptc/04-10-02. <http://www.omg.org/>
- [10] OMG. UML Profile for Schedulability, Performance and Time, ptc/02-03-02. <http://www.omg.org/>
- [11] E. Riccobene, P. Scandurra, A. Rosti and S. Bocchio. A UML 2.0 Profile for SystemC. ST Microelectronics Technical Report AST-AGR-2005-3.
- [12] E. Riccobene, P. Scandurra. Modelling SystemC Process Behaviour by the UML Method State Machines. In *Proc. of RISE'04*. Springer.
- [13] E. Riccobene, P. Scandurra, A. Rosti, S. Bocchio. A SoC Design Methodology Based on a UML 2.0 Profile for SystemC. In *Proc. of DATE'05*.
- [14] P. Schaumont and I. Verbauwhede. Interactive Cosimulation with Partial Evaluation. In *Proc. of DATE'04*.