

A HDL-Independent Modeling Methodology for Heterogeneous System Designs

Suad Kajtazovic
Graz University of Technology,
Institute for Technical
Informatics
Inffeldgasse 16/1
Graz, Austria
kajtazovic@iti.tugraz.at

Christian Steger
Graz University of Technology,
Institute for Technical
Informatics
Inffeldgasse 16/1
Graz, Austria
steger@iti.tugraz.at

Markus Pistauer
CISC Semiconductor
Design+Consulting GmbH
Lakeside B07
Klagenfurt, Austria
m.pistauer@cisc.at

ABSTRACT

This paper introduces a description methodology to be used in heterogeneous, multi-HDL (Hardware Description Language) system designs. Complex microelectronic embedded systems contain more and more concurrently designed subsystems, which are mostly coded in different HDLs to get best model performances. Verification of all subsystems in one environment represents a difficult task. This paper focuses on a HDL-independent description methodology, which enables a description of models coded in different HDLs using the same language semantic. Moreover, it supports a verification methodology for heterogeneous systems based on a cosimulation using standard EDA tools. The proposed description methodology has been applied on an application framework for heterogeneous system verification and later on evaluated by an example taken from the automotive industry.

1. INTRODUCTION

The continuously growing of system complexity requires a system integration with concurrent designed subsystems (e.g. analog, digital, mechanical and optical subsystems). In order to provide best model performances these subsystems are mostly described in different HDLs. To verify the whole system, they need to be integrated into one system. However, it is difficult to describe a system, which contains subsystems described in different HDLs. There is no common description language, which covers specifications of all used HDLs together. Nowadays EDA tools vendors provide simulators capable to simulate designs written in different HDLs such as VHDL/AMS, Verilog, SystemC, SaberMAST etc. Due to complex systems contain mixed-technology sub-components mostly written in different HDLs, this methodology is not adequate for a system verification. In recent years use of cosimulation techniques becomes more and more an interesting solution for verification of heterogeneous system designs.

The paper presents an advanced HDL-independent description methodology to be used in heterogeneous system designs. The proposed system description uses models without translating of their behavior, which ensures a more reliable system verification.

The proposed description methodology represents an orthogonal layer to other HDLs. This layer enables a system design using models coded in different HDLs together. Fig-

ure 1 depicts an HDL-independent description layer inserted between subsystem level and system level.

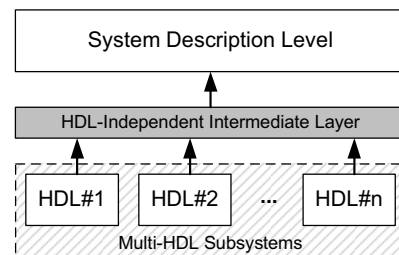


Figure 1. A HDL-independent description layer.

The remainder of this paper is organized as follows. Section 2 presents some of previous published works related to this paper. Section 3 describes the proposed design methodology to be used in heterogeneous system designs. In the section 4 the proposed description methodology has been presented more detailed. The proposed description methodology is followed by section 5, which presents the methodology capabilities based on an example taken from automotive industry. Section 5 is followed by section 6, which concludes this work.

2. RELATED WORK

In the past years HDL independent modeling methodologies have been investigated intensively. Many different solutions have been found. A short overview about used methodologies and tools is given in this section.

Unified Modeling Language (UML)[2] is a meta-language conceived for describing system-structures, process-flows and system-relations at a very abstract level. An UML system design consists of several diagrams and specifications of a system that describes architecture, object-sequences, structures (classes) as well as process-flows (state-machines) of a system. It requires a graphical editing tool to manage system diagrams. UML becomes *de facto* standard for object-oriented system descriptions. It is mostly used in software development but recently it is used in embedded system design, too. In the recent years UML and UML-tools become more and more important for the microelectronic industry,

too ([5],[3]). System designs using UML are language independent and simulator independent.

Systems Modeling Language (SysML)[7], [9] is a new visual modeling language developed from UML for systems engineering applications. It supports the specification, analysis, design, verification and validation of a wide range of systems. These systems may include hardware, software, information, processes, personnel and facilities-systems engineering applications.

Open Model Interface (OMI) [6] was first introduced in the year 1997 as a language independent and simulator independent interface between simulators and models of intellectual property (IP). Dunlop and McKinley [1] present OMI as a standard model interface for IP delivery. The purpose of the OMI interface is to protect IP embedded in models. That is often the primary balm to the availability of component models. OMI interface provides model interoperability while protecting the IP in the models. Moreover it provides a standard method for interfacing and managing complex electronic models to design automation tools. This method is aimed at providing efficient, accurate, and tool independent interfaces suitable for large designs such as systems on a chip (SoC).

Paragon [4] is an analog/RF and mixed-signal IC design environment with emphasis on interoperability and support for common data interfaces. Paragon uses an Extensible Markup Language (XML) model for the system-description, which includes the Mathematical Markup Language (MathML) [8] for description of the model behavior. A model representation in a paragon format includes a symbol formatted in Scalable Vector Graphics (SVG). Further, the Paragon tool contains code generators for VHDL-AMS, Verilog-AMS and SaberMAST HDLs.

2.1 Summary

UML designs are language independent and simulator independent but an UML-based system design is too complex and it requires definition of several diagrams. UML provides a good overview about system architecture and its structure at a very abstract level. Moreover, it supports a HDL-independent system design. However, it is impossible to verify a system without translating the system into the target language(s) for the specific simulator(s).

Similar to an UML-model a SysML-model consists of structural and behavioral system description (e.g. diagrams). SysML as the UML-enhancement for system design provides better design possibilities, but it is still too complex and the verification requires the system translation into the target language(s) for the specific simulator(s).

OMI provides a description methodology with an ability to protect the IP in the models. Since the OMI interface is language and simulator independent, it can be used for modeling of multi-HDL systems. In a comparison to other related methodologies, OMI models are represented in a compiled form as object code and they support programming interface, which is targeted on the use of the model in simulation. However, the OMI interface is based on Synopsys SWIFT model interface and Verilog HDL Programming Language Interface (PLI). Therefore the verification of OMI models is covered only with simulators that support these interfaces.

Table 1. Language independent methodologies - a comparison.

Modeling language	Design level	Model description		IP support	Graphical representation	Limitation
		Interface	Behavior			
UML	Very abstract	YES	YES	NO	YES	Very abstract design-level. For FSMs (finite-state machines) adequate.
SysML	Very abstract	YES	YES	NO	YES	An enhancement of the UML.
OMI	RTL → Physical	YES	YES	YES	NO	Designed for RTL level and below.
Paragon	Abstract	YES	YES	NO	YES	Requires model-behavioral translation into own structure. Possible loss of model performances.
Presented solution	Abstract → RTL	YES	NO	YES	YES	Coverage and design level depend on provided IP models. Direct translation into a foreign HDL.

Paragon uses VHDL definitions in its DTD description. This reduces the language-independency of the used description methodology. However, it describes model structure as well as model behavior using MathML syntax.

All solutions related in this paper describe a system structure together with the system behavior. Due to a system behavior is also described using language independent semantic, the system cannot be validated before it is not translated into the target simulator-language. This can cause a loss of model performances or even it can change the model behavior, too. Therefore the reliability of IP models cannot be hold. In order to avoid such problems the proposed description methodology handles models *as they are*. Furthermore, instead of translating models into *target languages*, a cosimulation verification platform has been used to verify a multi-HDL system. Table 1 depicts a comparison between related solutions and presented methodology.

3. SYSTEM DESIGN FLOW

The approach proposed in this paper supports a system verification for concurrent developed subsystems based on a heterogeneous co-design methodology. Based on top-down system design using IP models, the proposed design methodology subdivides the system design into three different levels: (1) the *System Design Level*, (2) the *Language Level* and (3) the *Simulator Level*.

At the system design level a heterogeneous system is described using the proposed language-independent description semantic, that enables integration of subsystems designed in different HDLs. In order to meet the requirements of the target simulation environment, a heterogeneous system has been enhanced at the language level with special cosimulation blocks and the subsystems are grouped by its HDLs.

The performed modification does not have any influence on the system behavior. Moreover, it prepares the system for the next design step specified as the simulator level. At the simulator level the modified system description serves as an outline for code generation and setup of the cosimulation platform. Involved simulators communicate via integrated cosimulation interfaces.

An application framework has been developed to support system design based on the proposed design methodology. Language level and simulator level described above are design steps in a heterogeneous system design that are per-

formed in this framework fully automatically.

4. MODEL REPRESENTATION

The proposed description methodology considers system designs at the system level that uses models provided from a library. At this level the functional descriptions of models are not necessary since the system has been designed using provided IP models. Concerning that, a system description becomes language independent. It opens a possibility to use the same description semantic for models written in different hardware description languages.

4.1 HDL-Independency

Figure 2 depicts the relationship between the level of the language independency of a system description and the HDL-refinement. It is obvious, that the HDL-refinement reduces

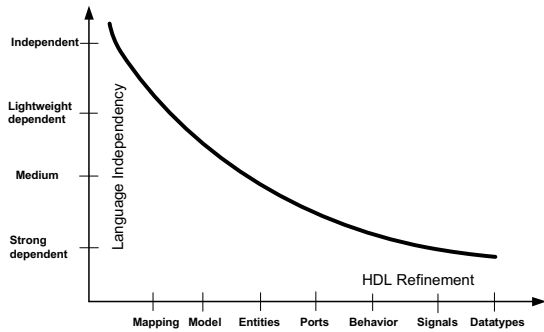


Figure 2. Language independency and HDL refinement.

the language independency in a system description. In order to keep a system description HDL-independent, a system design contains information about block interfaces and port mapping only. A system description does not contain the behavior information of the system components (e.g. blocks). Because the behavior of the system components is not translated into a meta-format, the proposed methodology holds models reliable. Moreover, it simplifies the system description and makes it more HDL-independent.

However, the models are not changed during the design steps and a lost of their performances cannot occur because of the used co-verification methodology.

4.2 Model Structure

A system consists of a number of *blocks* and *nets*, which describe connections between blocks. A *block* is a hierarchical structure that describes an interface (e.g. entity) of a model. Main parts of a *block* are *ports* and *parameters* used to connect the block at a higher-level and specify its behavior. Figure 3.a depicts a block interface with ports and parameter definition and figure 3.b shows the system structure.

4.2.1 Hierarchy

The block-based system design allows the creation of hierarchical structures on a very simple way. The hierarchy of the system is achieved by the definition of sub-blocks and

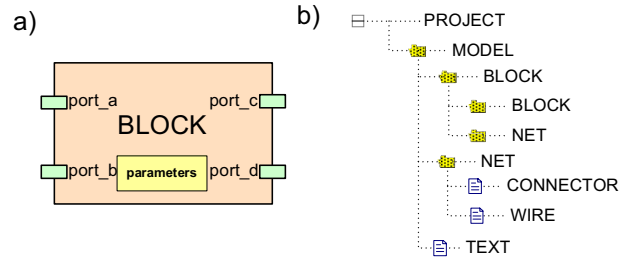


Figure 3. Block interface (a) and model hierarchy tree (b).

sub-nets in a block structure. Figure 3.b depicts a hierarchical structure of a model description. A block can contain sub-blocks and sub-nets that define its internal structure.

4.3 Language Semantic

One of the most used languages for describing meta-data is XML. The essential benefits of XML are:

- language independency,
- legibility,
- compactness,
- support of hierarchical structures.

The proposed description methodology is based on XML. XML is supported by the Document Type Definition (DTD). It is the foundation of a task-specific language, which specifies rules and types used in a XML-model specification.

4.3.1 XML Schema

The language semantic of proposed description methodology can best be explained using DTD definition of the used XML. Listing 1 depicts a part of a model structure represented in a DTD specification.

Listing 1. A part of the model structure represented in a DTD.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE xhdl SYSTEM [
<!ELEMENT project (model,(simulation,author)*)>
<!ELEMENT model (block+,net+,(text|description)*)>
<!ELEMENT net (connector+,wire*)>
<!ELEMENT connector EMPTY >
<!ATTLIST connector
  name CDATA #REQUIRED
  block CDATA #REQUIRED
  port CDATA #REQUIRED
>
<!ELEMENT wire EMPTY >
<!ATTLIST wire
  srcblock CDATA #IMPLIED
  srcport CDATA #IMPLIED
  destblock CDATA #IMPLIED
  destport CDATA #IMPLIED
  points CDATA #IMPLIED
>
<!ELEMENT block (position,symbol,port+,parameter*,src*,
  description*,(block+,net+,text*)*)>
<!ATTLIST block
  name CDATA #REQUIRED
  src CDATA #REQUIRED
  lagid CDATA #REQUIRED
  type (external_block|file|ANY) #REQUIRED
  catid CDATA #IMPLIED
  warning CDATA #IMPLIED
  simulator CDATA #IMPLIED
  description CDATA #IMPLIED
>
```

```

<ELEMENT position EMPTY >
<ELEMENT port EMPTY >
<!ATTLIST port
  name          CDATA #REQUIRED
  type          (IN|OUT|BIDIR) #REQUIRED
  datatype     CDATA #REQUIRED
  default      CDATA #IMPLIED
  pos          (LEFT|RIGHT|TOP|BOTTOM) #REQUIRED
  LSB         CDATA #IMPLIED
  MSB         CDATA #IMPLIED
>
<ELEMENT symbol (line|circle|rectangle|polygon)+ >
<ELEMENT description (#PCDATA) >
...
]>

```

4.3.2 Model-Interface Definition

The DTD-specification of a block covers different types of blocks. Due to the proposed design methodology uses models from a provided IP library, one XML block contains only references about the real model stored in an IP library. Listing 2 presents the entity definition of a comparator model coded in VHDL-AMS.

Listing 2. Entity definition of a comparator model coded in VHDL-AMS.

```

LIBRARY IEEE;
USE IEEE.ELECTRICALSYSTEMS.ALL;

-- entity declaration for analog comparator
ENTITY comparator IS
  PORT (TERMINAL inn:ELECTRICAL;
        TERMINAL inp:ELECTRICAL;
        SIGNAL outp:OUT BIT);
END ENTITY comparator;

```

The XML description of the comparator model is shown in listing 3. The XML-*block* definition contains interface information of the comparator model. The same XML structure contains an additional component used for the graphical representation of the model (<symbol>...</symbol>) in a schematic editor.

Listing 3. An XML description of the comparator model.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<block name="CMP" src="67" langid="2" type="" >
  <position posX="10" posY="10" width="100" height="80" />
  <symbol>
    <line p1X="0" p1Y="0" p2X="0" p2Y="80" color="0,0,0"
      size="1" />
    <line p1X="0" p1Y="80" p2X="100" p2Y="40" color="0,0,0"
      size="1" />
    <line p1X="100" p1Y="40" p2X="0" p2Y="0" color="0,0,0"
      size="1" />
  </symbol>
  <port name="inp" type="IN" pos="LEFT" datatype="
    ELECTRICAL" pX="0" pY="20" />
  <port name="inn" type="IN" pos="LEFT" datatype="
    ELECTRICAL" pX="0" pY="60" />
  <port name="outp" type="OUT" pos="RIGHT" datatype="
    ELECTRICAL" pX="100" pY="40" />
</block>

```

The model-interface definition uses HDL-specific constructs such as parameter's and port's datatype definitions. However, it does not cause a limitation in the language independency. Moreover, it is required for proper model instancing and mapping of its ports.

4.3.3 Mapping

Mapping of models (e.g. sub-systems) designed in different HDLs using one language is one of the challenges in this work. The proposed description methodology provides a common mapping semantic used for net structures in a multi-HDL design.

Listing 4. An example of the model mapping in XML.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<project>
  <model name="xmlExample" lang="" src="325" description="
    = "An example for model mapping.">
    <block name="OP1" langid="2" src="67" description="
      Operational amplifier" showname="1" type="" catid="
        " warning="" >...</block>
    <block name="SINUS" langid="2" src="94" ... >...</block>
    <block name="Rload" langid="5" src="101" ... >...</
      block>
    <block name="R2" langid="5" src="101" ... >...</block>
    <block name="R1" langid="5" src="101" ... >...</block>
    <block name="GND" langid="7" src="119" ... >...</block>
    <block name="GND_1" langid="7" src="119" ... >...</
      block>
    <block name="GND_2" langid="7" src="119" ... >...</
      block>
    <net name="net" vias="380-180" >
      <connector name="net_R2_1_n" block="R1" port="n" />
      <connector name="net_OP1_inn" block="OP1" port="inn"
        />
      <connector name="net_R2_p" block="R2" port="p" />
      <wire srcblock="R1" srcport="n" destblock=""
        destport="" points="350-180,350-180,380-180" />
      <wire srcblock="" srcport="" destblock="OP1"
        destport="inn" points="380-180,410-180,410-180"
        />
      <wire srcblock="R2" srcport="p" destblock=""
        destport="" points="
          410-250,380-250,380-250,380-180,380-180" />
    </net>
    <net name="net_1" vias="590-160" >...</net>
    <net name="net_2" vias="" >...</net>
    <net name="gnd_" vias="" >...</net>
    <net name="gnd_1" vias="" >...</net>
    <net name="gnd_2" vias="" >...</net>
  </model>
</project>

```

Net nodes contain mapping information between block-ports. Which ports of which blocks are connected with a net is specified with the *connector* sub-node. For the graphical representation of a connection the *wire* sub-nodes have been defined. A wire node contains routing information of a wire in a schematic design. Listing 4 shows a part of an XML

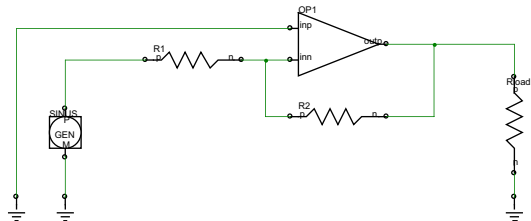


Figure 4. An example of the graphical representation in a schematic editor.

system description with four blocks coded in different HDLs (*sinus*, *OP1*, *R1*, *R2*, *Rload* and three *GND* blocks) and mapping information between ports, whereby figure 4 depicts the same model represented in a schematic editor.

5. AN EXPERIMENTAL EXAMPLE

To illustrate the language independent model creation described in section 4 one example of a system description of a heterogeneous system using the proposed methodology is presented in this section. Figure 5 depicts a system overview of an automotive power management system (APMS). It ought to support the automotive industry with an energy management system that contains algorithms for controlling the power consumption of automotive electrical systems in modern vehicles. The system controls the power needs of the automotive electromechanical loads and the charging of the battery. It prevents a complete discharging of the battery in any system condition.

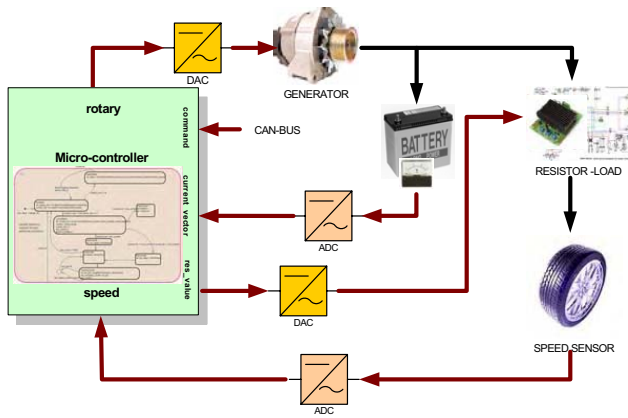


Figure 5. A system overview of the automotive power management system.

The system contains analog and digital components as well as software components. The microcontroller unit (MCU) developed in SystemC as a state-machine represents a software part of the system. All other components such as ADC/DAC, Generator, Speed-Sensor, Battery etc. are coded in VHDL-AMS.

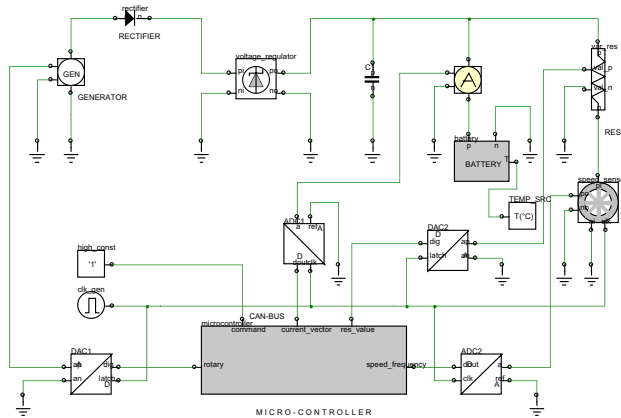


Figure 6. APMS at the system design level in a schematic editor.

A test-bench has been designed whereby all components have been inserted at top-level (Fig. 6). Concurrently devel-

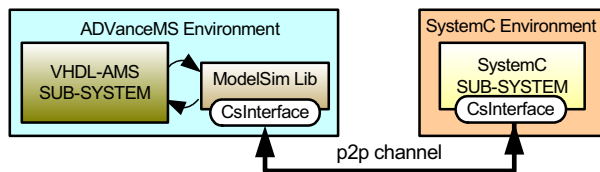


Figure 7. Configuration of the cosimulation platform.

oped subsystems are integrated using the schematic editor integrated in our application framework. The ADVanceMS simulator from MentorGraphics has been used to simulate

the VHDL-AMS subsystem and SystemC simulator has been used to simulate the SystemC side of the system. Both simulators are connected via a cosimulation interface as depicted in figure 7. It creates a simple peer-to-peer (p2p) connection between involved simulators. Each side contains one cosimulation module (*CsInterface*), which creates a connection to the simulator and to the cosimulation network. The *CsInterface* module uses simulator open API (Application Programming Interface) for the data-exchange with the simulator. Moreover, it synchronizes the connected simulator with the cosimulation network.

Due to the fact that the current version of the ADVanceMS simulator does not provide required API functions for the used cosimulation technique, the ModelSim library has been used to get access to the ADVanceMS simulator. Therefore, the *CsInterface* module on the VHDL-AMS side has been simulated using ModelSim simulator and the rest of the VHDL-AMS side has been simulated using ADVanceMS simulator. The microcontroller unit has been simulated using SystemC simulator.

5.1 Evaluation Results

The proposed description methodology has been evaluated by the used heterogeneous example. A distributed cosimulation has been created and the system has been simulated successfully. Furthermore, the proposed description methodology has been applied effective on the example presented in this work. Used at the system level, the XML system description addresses the HDL-independent model-instancing and mapping information. According to the XML system-description the model parameters have been modified successful. Apart from a VHDL-AMS-SystemC system description, other heterogeneous systems using models coded in different HDLs such as SaberMAST, Matlab/Simulink and Modelica have been evaluated. Moreover, the proposed description methodology has been used successful for the graphical representation of the system. However, describing more complex systems using XML without a schematic editor, which generates the XML-code automatically from a schematic design, could be difficult.

6. CONCLUSION

A HDL-independent system description methodology has been presented in this paper. It has been shown that the proposed description methodology can be used effectively in design of heterogeneous systems.

One of the key benefits of the proposed description methodology is the seamless description of multi-HDL systems using one description semantic without translating the functional behavior of the subsystems. The subsystems are used in their *original* HDL that ensures reliable system verification. Without any restriction and modifications the proposed description methodology has been used for description of heterogeneous system using models coded in different HDLs.

Further work in this project will be the enhancement of the XML specification to support the parameter passing for models with hierarchical structures described in XML, too. Another important issue is to support the simulation-settings such as the signal-tracing information for certain HDLs (e.g. SystemC), whereby this information must be available at the compilation time in order to trace the wave forms of signals specified in a system description.

7. REFERENCES

- [1] D. Dunlop and K. McKinley. *OMI - A Standard Model Interface for IP Delivery*. Proceedings of the IEEE International Verilog HDL Conference, Santa Clara, California, USA, June 1997.
- [2] P. Kukkala, J. Riihimäki, M. Hännikäinen, T. D. Hämmäläinen, , and K. Kronlöf. *UML 2.0 Profile for Embedded System Design*. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE05), Munich, Germany, March 2005.
- [3] M. Lajlo, A. S. Basu, and M. Prevostini. *UML Specifications Toward a Codesign Environment*. Proceedings of the FDL'03 Forum on Specification and Design Languages, Lille, France, September 2004.
- [4] P. Mallick, M. Francis, V. Chandrasekhar, A. Austin, and H. A. Mantooth. *Achieving Language Independence with Paragon*. Proceedings of the International Workshop on Behavioral Modeling and Simulation (BMAS'2003), San José, California, USA, October 2003.
- [5] M. Marchetti and I. Oliver. *A Methodology for Implementing Highly Concurrent Data Objects*. Proceedings of the FDL'03 Forum on Specification and Design Languages, Frankfurt, Germany, September 2003.
- [6] Object Management Group. <http://www.eda.org/omf/>, June 2005.
- [7] SysML partners. *Systems Modeling Language (SysML) Specification*. Object Management Group, <http://www.sysml.org/artifacts.htm>, January 2005.
- [8] W3C. *Mathematical Markup Language (MathML)*. W3C, <http://www.w3.org/TR/MathML2/>, version 2.0 second edition, October 2003.
- [9] W. D. Yves Vanderperren. *UML 2 and SysML: an Approach to Deal with Complexity in SoC/NoC Design*. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE05), Munich, Germany, March 2005.