Transformation of SysML Structure Diagrams to VHDL-AMS

Fabrice Bouquet, Jean-Marie Gauthier, Ahmed Hammad and Fabien Peureux
University of Franche-Comté - Femto-ST
Department of Computer Sciences
Besançon, France
Email: {fbouquet, jmgauthier, ahammad, fpeureux}@femto-st.fr

***Abstract*—In this paper, we propose an approach to translate the `SysML` language to `VHDL-AMS` code. This approach is the first step to the generation of the `VHDL-AMS` code from the structural diagrams `SysML`. In this step, we address the Block Definition Diagram and the Internal Block Diagram. The translation uses Model Driven Engineer (MDE) methods as the transformation of model to another model `(M2M)` with `ATL` Atlas Transformation Language and the code generation from models `(M2T)` using `Xpand`. We provide the translation rules between the two elements. Implementation and methodology are illustrated on a micro-system case study: the `Smart surface` system.**

***Keywords*-`SysML`; `VHDL-AMS`; transformation; metamodel; model;**

I. Problematic

Micro-systems design is complex because such systems are often composed of mechanic, electronic or software components. This complexity makes difficult to validate the system in regards of his requirements. So, it is essential to set up a process for verification and simulation before or during the realization of the system.

Specification modeling is the main step to ensure the success or the failure of the design. This step has three roles:

- to clarify the specification (extract the requirements)
- to analyse the specification and to understand the goal of the system (classify the requirements)
- to communicate between the partners involved in the project as users, analysts and domain experts.

The SysML language is an UML profile which is one of the most popular graphical modeling language. SysML is a semi-formal language which provides the documentation and the graphical specification of all aspects of a system (hardware and software components). SysML enjoys unprecedented popularity both in industry and academia because it has been recently adopted by the OMG as a standard in Systems Engineering. It is used to gather the different contributions of the partners for the achievement of system, and to ensure coherence and quality of design. SysML is a language well suited for micro-systems design thanks to the hierarchical description of hardware and software blocks. It can graphically model the mathematical equations defining the physical system behaviors. In this context, SysML allows formulating the requirements (functional or non-functional) with the *requirements diagram*. Nevertheless, despite the various advantages of SysML, it remains a semi-formal language. As a consequence, the simulation of SysML model is not possible without defining interpretative semantics or transforming it into a formal language.

Our goal is to transfer SysML models into simulation environments. It would be interesting to map SysML models to simulation languages in order to do formal verification. In the context of micro-systems, VHDL-AMS is one of the most used hardware description language which provides several simulation environments.

Therefore, our objective is to define a methodological approach based on the following formalism for SysML:

- the hierarchical description in the Block Definition Diagram and in the Internal Block Diagram (hardware and/or software),
- modeling the behavior for each block and the description of physical constraints in *parametric diagrams*,
- generate the corresponding VHDL-AMS code using techniques of MDE[1] as model transformations.

In this paper, we present the first part of our work. This part is the transformation of the SysML structural view composed by Block Definition Diagram (BDD) and Internal Block Diagram (IBD) into VHDL-AMS code. The paper is composed as follows: first, we give a reminder about languages SysML and VHDL-AMS, and the technical elements about TopCased, Xpand and ATL languages.

In section III, we illustrate our approach with micro-system case study called Smart Surface. We present, in particular the SysML structural diagrams modelling this case study.

In section IV, we give some translation rules between SysML structural diagrams and VHDL-AMS. We apply these rules to the model described previously and we use ATL and Xpand to generate the VHDL-AMS code. Finally, we present some related works in this area and we conclude by presenting perspectives of our work.

[1]Model Driven Engineering

II. Preliminaries

A. SysML

The Systems Modeling Language (SysML) has been standardized by the Object Management Group (OMG) [1] with the participation of the International Council on Systems Engineering (INCOSE) [2] and the Application Protocol 233 (AP233 consortium) [3]. SysML is a profile of UML 2.0 [4] which is the actual standard for software engineering. SysML consists of several diagrams which are requirement, use case, sequence, activity, block, internal block, constraint blocks, parametric, state machine and allocation. SysML is a graphical modelling language for Systems Engineering (SE). It supports the specification, analysis, design, verification and the validation of a broad range of systems.

In section III, we clarify block and internal block diagrams by applying them to the case study.

B. VHDL-AMS

VHDL-AMS is the IEEE standard modeling language (standard 1076.1) created to provide compatibility and capability in an open language for modern analog, mixed-signal and multi-domain designs. VHDL-AMS provides both continuous-time and event-driven modeling semantics, and is suitable for analog, digital, and mixed analog/digital circuits. VHDL-AMS also facilitates modeling of multi-domain systems that can include (among others) a combination of electrical, mechanical, thermal, hydraulics, and magnetic models. This allows the entire system now to be modeled, simulated, analyzed, verified and optimized. Not only does VHDL-AMS provide powerful modeling capabilities, but also provides compatibility where models can be exchanged between different simulation tools that implements the VHDL-AMS standard [5].

VHDL-AMS supports the description of a system of differential and algebraic equations where the solution varies continuously with time. In addition, VHDL-AMS supports both structural composition and behavioural descriptions of analog systems. Solutions of the equations describing the behaviour of the system may include discontinuities. Interactions between the discrete part of a model and its continuous part are supported in an adaptable and efficient way [6].

C. The TopCased Project

TopCased is the acronym for Toolkit In OPen source for Critical Applications & SystEms Development. This project started in 2005 from the French "Aerospace Valley" cluster. TopCased is dedicated to aeronautics, automotive and space embedded systems. It contains an Integrated Development Environment (IDE) based on the Eclipse framework which adds features that provides methods and tools to develop a safety software systems or mixed software and hardware systems [7].

Relying primarily on standardized languages for modeling software (such as SysML/UML, AADL, EAST-ADL, SDL, etc.) and associated tools (graphical and text editors, documentation and code generators, validation through model animation, verification through model checking, version management, traceability, etc), TopCased modeler is one of the most complete free solutions which respects the current standards.

D. The ATL Language

ATL is a model transformation language and toolkit initiated by the AtlanMod team (previously called ATLAS Group) [8]. In the field of Model-Driven Engineering (MDE), ATL provides ways to produce a set of target models from a set of source models. Released under the terms of the Eclipse Public License, ATL is a M2M (Eclipse) component, inside of the Eclipse Modeling Project (EMP). An ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target models [8]. An ATL transformation can be decomposed into three parts: a header, helpers and rules. The header is used to declare general informations such as the module name (it is the transformation name: it must match the file name), the input and output meta-model and potential import of needed libraries. Helpers are subroutines that are used to avoid code redundancy. Rules are the heart of ATL transformations because they describe how output elements (based on the output meta-model) are produced from input elements (based on the input meta-model). They are made up of bindings, each one expressing a mapping between an input element and an output element.

E. Xpand

Xpand is a language specialized on code generation based on EMF (Eclipse Modeling Framework) models. The Xpand language defines templates to control the output generation. Xpand allows to create text output from EMF model. The text output can be a programming language or something else. Xpand requires the definition of EMF meta-model and one or more templates which will translate the model into text. Once this definition is done, the generator requires the definition of an EMF source model to be translated.

III. SysML Case Study: Smart Surface

In this section, we present the Smart Surface case study proposed in [9]. We use only the elements associated to the structural view. These elements are defined in the Block Definition Diagram (BDD) and in the internal Block Diagram (IBD).

The Smart Surface consists of a rectangular grid of rectangular *cells*. Each *cell* is made of a *microActuator*, a *microSensor* and a *microController*. The purpose of the Smart Surface is to sort microscopic objects according

to parameters such as their shape or colour. The following Smart Surface characteristics are extracted from the specification document conjointly written during the "Smart Surface" project.

1) **The grid**
   - The grid is divided in 24 lines of 24 cells each, a cell having a size of about 2 mm.
   - There is no centralized control. Cells communicate step by step through their direct neighbours.
2) **Objects to sort**
   - We assume that an object should be found from a small number of options (2, 3 or 4).
   - Objects to convey are typically included in a square of less than 4 μm a side.
3) **Microactuator**
   - A microActuator can communicate with its 4 neighbours via its cells controllers.
   - A microActuator acts on the objects through an air-flow (should not be specified at this level).
4) **Sorting**
   - At a given moment, at most one object have to be present on the smart surface.
   - The sorting will be done according to the shape of the object.

After this informal description, we present the model associated to SysML structural diagrams.

A. Modeling structure

The major structural extension in SysML is the **block** which extends the UML structured Class. The block is the main element for model the system's structure in SysML.

In general purpose, the hierarchical structuring mechanism allows a better design because each level defines itself details. Blocks can represent any level of the system hierarchy, including the top level system and logical or physical components of the system or environment. A SysML block describes a system as a collection of parts and connections between them. Connections is used to communicate and to interact with blocks. SysML provides standard ports which support client-server communication (e.g., required and provided interfaces) and Flow Ports that define flows in or out of a block. Ports are discussed in more detail below. In the following, we shall present the BDD and IBD :

- The Block Definition Diagram (BDD) defines the characteristics and relationships between blocks such as composition/aggregation, association, generalization and connector.
- The Internal Block Diagram (IBD) allows to refine the structural aspect of the model. The IBD is the equivalent in SysML of the composite structure diagram in UML.

B. Block Definition Diagram (BDD)

The first level modeling of the Smart Surface is a BDD. Fig. 1 shows this BDD with eight blocks. The block named Smart Surface represents the micro-system. It is decomposed into three sub blocks (*Surface*, *Interface* and *Object*) and is linked to them by the following relationships:

- composition to the *Surface* and *Interface* blocks,
- aggregation to the *Object* block.

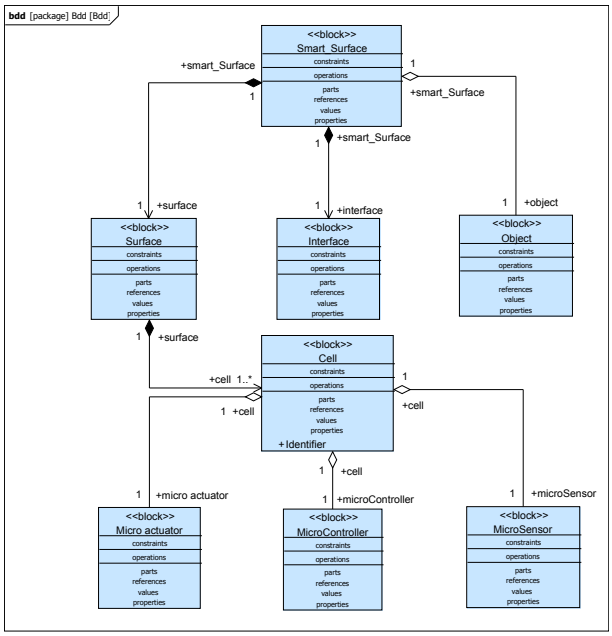

Figure 1. Smart surface BDD

The block named *Object* represents a microscopic object to sort by the Smart Surface environment. The block named *Interface* represents all the interactions between *Surface* and *Object*. The block named *Surface* represents the distributed MEMS under design. This block Surface is linked by composition to a new block named *Cell*. The composition relation with the block *Cell* is labeled with the multiplicity *1..** and expresses that the block *Surface* is composed of many *cells*. The block *Cell* is itself composed of three parts, namely a *microActuator*, a *microSensor* and a *microController*. Each of them is represented by a block.

The two blocks *Surface* and *Cell* represent physical components and together constitute a physical model of the Smart Surface. The details of this components must be describe in Internal Block Diagram.

C. Internal Block Diagram (IBD)

In the IBD, parts are basic elements assembled to define how they collaborate to realize the block structure and/or behavior. In the IBD, the designer can refine the definition of interactions between blocks by defining flow ports. Two types of ports are available in SysML:

- **standard ports** handling requests and invocations of services with other blocks
- **flow ports** which let blocks exchange flows of information.

Fig. 2 shows two flow ports: the flow port *Direction enable* continuously passes the direction of the *object*. Through the flow port *Object detection*, the *microSensor* sends to the *microController* a signal to indicate the detection of an object.

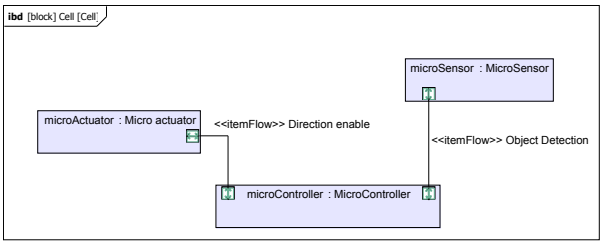

Figure 2. Cell IBD

IV. Implementation: Transformation of Model and Code Generation

A. Model-Driven Engineering (MDE)

The MDE is an approach of design which aims to keep the models as reference point throughout the process of software development. The MDE provides two main approaches for transforming models: the first one is called M2M (Model To Model) and the second is called M2T (Model To Text). The M2M transformation offers the possibility of transforming models or a part of a model into another one. Whereas the M2T transformation provides a way for transforming a model mostly into source code. The model transformation takes an input model that is conform to the source meta-model and produces an output model, which is conform to the target meta-model. The meta-model is the model of the model, so we use "meta" to indicate this concept.

Figure 3 shows a generic transformation framework. A model transformation manipulates concepts that are specified in the source and target meta-models (which can be different). These meta-models describe the static structure of models manipulated by the transformation. We have developed these meta-models which are conform to the MOF (Meta Object Facility). In the following, we consider transformations that take a single input model and produce a single output model. The source model is specified using a specific format based on XML-based Metadata Interchange (XMI) that supports the interchange of any kind of metadata and that can be expressed using the MOF specification, including both model and meta-model informations.

In our work, we use the SysML meta-model as a source meta-model and the VHDL-AMS meta-model as a target meta-model. The goal is the transformation of the input model (conforms to input meta-model) and to produce as result an output model of VHDL-AMS (conforms to the target meta-model). We choose to use the ATL language for this transformation. Moreover, we choose to use the Xpand language to produce VHDL-AMS code.

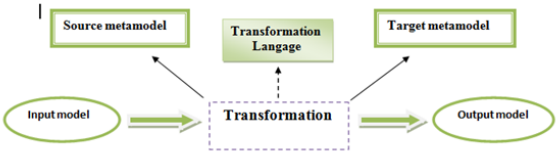

Figure 3. generic model transformation

B. Rules of transformation

1) Block Definition Diagram:

1) The block SysML constitutes the basic brick to model a system. It represents a support for various elements which characterized the system, as the operations, constraints, and parts. The ENTITY in VHDL-AMS represents the external view of a component with its connections. This semantic mapping allows to translate a block SysML into VHDL-AMS ENTITY, which takes the name of the block.
$\{SysML : Block\} \rightarrow \{VHDL - AMS : ENTITY\}$
$\{Block : name\} \rightarrow \{ENTITY : name\}$

2) *Flow port* represents the input and output items that may flow between a block and its environment. A *Flow port* is an interaction point through which data, material or energy can enter or leave a block. VHDL-AMS Port got relatively the same semantic, so this semantic mapping allows us to translate SysML *Flow port* into VHDL-AMS Port. Moreover, we consider the name, the direction of the Port (IN/OUT) and the type of the Port.
$\{Block : FlowPort\} \rightarrow \{ENTITY : Port\} :$
$\{FlowPort : name\} \rightarrow \{Port : name\}$
$\{FlowPort : direction\} \rightarrow \{Port : direction\}$

3) Operations defined in a block can be translated into functions which represent the behavioral aspect of the system. Parameters of the operation will be translated into parameters of the VHDL-AMS function. Implementation of functions is made in the body of the ENTITY's ARCHITECTURE.
$\{Block : operation\} \rightarrow \{ARCHITECTURE : operation\}$

4) *Parts* or *References* properties represent components which constitute the system. They serve to describe the composition and the internal structure of a block. Components in VHDL-AMS also constitute a structural description of the circuit. So we can have

a semantic link between parts and components. The declaration of components is made in the body of the ARCHITECTURE of the VHDL-AMS ENTITY .
$\{Block : Property : Part/Reference\} \rightarrow \{VHDL-AMS : Component\}$

5) The property of type *Value* defines a quantifiable value with its unit, its dimension and a particular type. It provides a way to specifie an *Item Flow*. An *Item Flow* represents the thing that flow through *Flow Port*. Moreover, VHDL-AMS provides the definition of six objects which allow to transport the information in models: CONSTANT, VARIABLE, SIGNAL, TERMINAL, QUANTITY, FILE. We shall use one of these types according to the use and the nature of the circuit.
$\{Property : value\} \rightarrow \{VHDL-AMS : variable\}$

6) A constraint can specify mathematical formulas or relationships between datas that flow through the system. These constraints can be specified in a block to be translated in the ARCHITECTURE of the ENTITY as VHDL-AMS instructions.
$\{Block : constraints\} \rightarrow \{VHDL-AMS : instructions\}$

2) Internal Block Diagram: The Internal Block Diagram describes the internal view of a block. It is based on BDD to assemble parts that make up the main block. This diagram consists of two units: parts and ports. In a hardware modeling, we will use only flow port, and therefore there will be two mapping rules between IBD and VHDL-AMS. The first links parts and components as in the BDD and the second links SysML ports with VHDL-AMS ports.
$\{IBD : Part\} \rightarrow \{ARCHITECTURE : Component\}$
$\{IBD : Reference\} \rightarrow \{ARCHITECTURE : Component\}$
$\{IBD : FlowPort\} \rightarrow \{ARCHITECTURE : Port\}$

Figures 4 and 5 show the relationship between the two kinds of block (BDD and IBD) with the corresponding VHDL-AMS code.

C. Steps of the implementation

The transformation process takes place in several steps, starting from a SysML BDD diagram to VHDL-AMS code generation. The first is using ATL to transform a SysML model to a VHDL-AMS model (meta-model to meta-model). The second uses Xpand for generating code from the VHDL-AMS model. The Figure 6 depicts the several steps of our approach.

D. ATL transformation

First, we built meta-model for BDD and IBD diagrams as meta-model source and the second for VHDL-AMS as

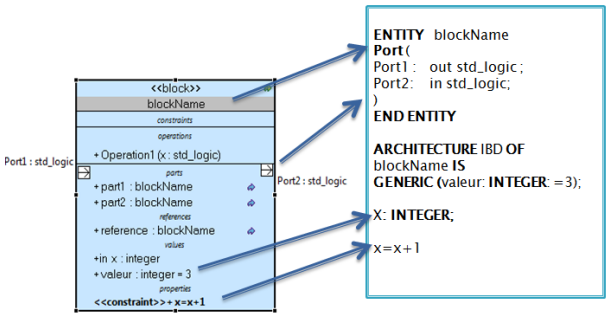

Figure 4. Rules of BDD transformation

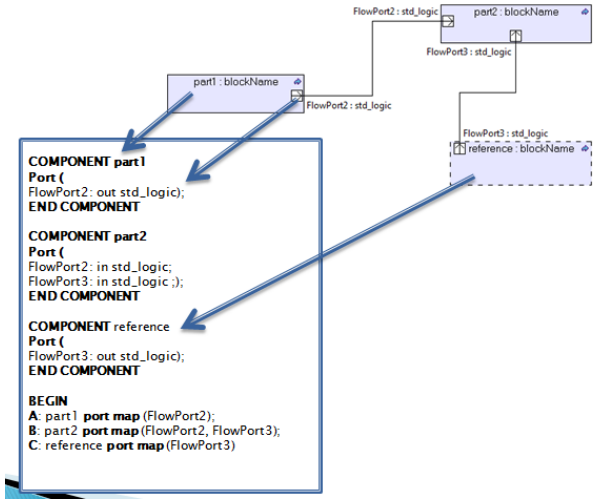

Figure 5. Rules of IBD transformation

meta-model target. For this, we will base on the framework EMF and the Ecore meta-model. We take the BDD and IBD from the Section III and we model them with Topcased. We extract the information from TopCased with the exported XMI. We parse this XMI file with JDOM (API for processing XML documents with Java) to extract informations about the BDD and IBD diagram to populate our own SysML meta-model. Finally, we use ATL to translate it into VHDL-AMS model.

E. Code generation

Xpand is a template engine which allows to create text output from EMF models. The text output can be a programming language or something else. Xpand requires to define an EMF meta-model and one or more templates which will translate the model into text. Once this definition is done, the code generator can be run [10].

In our example, we have the VHDL-AMS meta-model and an instance of it. To generate code, we must create a Xpand project and configure the workflow for the code generation. Finally, we create the code of the template. On the first

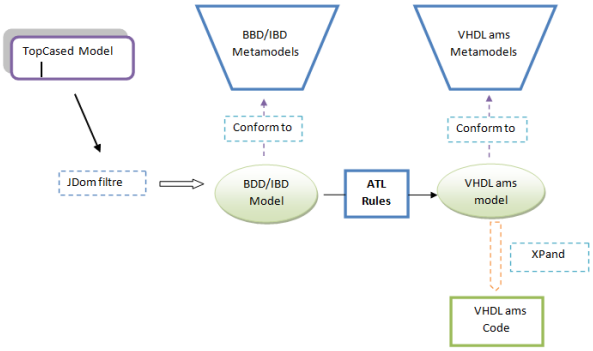

Figure 6. Transformation process

```
module m2m;
create OUT : VHDLamsMetamodel from IN : BDDMetamodel;

rule BDD2DEntity{

     from s :BDDMetamodel!BlockDefinitionDiagram
     to t : VHDLamsMetamodel!DesignEntity (
     name <- s.name,
     entite<-s.block )
}
rule Block2VHDLentity   {
    from
    s:BDDMetamodel!Block
    to t:VHDLamsMetamodel!Entity(
    name<-s.name,
    port<-s.port,
    archi<-s.blockElement)
}
rule Block-port2VHDL-port{
  from
  s:BDDMetamodel!Port
  to t:VHDLamsMetamodel!Port(
  name<-s.name,
  Type<-s.Type,
  direction<-s.direction)
```

Figure 7. ATL Rules

line in Figure 8, we import the meta-model (IMPORT vhdl) so that the generator (and the editor as well) knows about the structure of our model. The main concept of Xpand is the DEFINE keyword, also called a template. This is the smallest identifiable treatment unit. It is used to identify the model elements in order to generate the VHDL-AMS code. Fig 9 shows the result of the generation.

```
«IMPORT vhdl»
«DEFINE main FOR vhdl::Model»
«EXPAND definition FOREACH entite»
«ENDDEFINE»
«DEFINE definition  FOR vhdl::Entity »
«FILE "fileName.vhd"»
ENTITE «name» is
PORT(
    «FOREACH this.port AS p»
    «p.name» : «p.Type»
«ENDFOREACH»)
«EXPAND architecture FOREACH  this.archi»
«ENDFILE»
«ENDDEFINE»
«DEFINE architecture FOR vhdl::Architecture»
ARCHITECTURE «this.name» OF «entiteA» IS
BEGIN
«FOREACH this.contraints AS c»
«c.name»
«ENDFOREACH»
«FOREACH this.composants AS composant»
COMPONENT «composant.name» IS
  PORT(
  «FOREACH composant.portInterne AS portI»
  «portI.name»: «portI.type»
  «ENDFOREACH»
      )
END COMPONENT «composant.name»;
«ENDFOREACH»
«FOREACH this.composants AS composant»
S«composant.name»:«composant.name» PORT MAP(
«FOREACH composant.portInterne AS portI» P«portI.name»,«ENDFOREACH»)
«ENDFOREACH»
«ENDDEFINE»
```

Figure 8. Xpand code generation

V. Related work

There are some works on the transformation of SysML model. We can cite MetaSyn [11] product that synthesizes SystemC models from SysML, establishing a link from software and system models to the hardware Electronic Design Automation (EDA) flow.

In [12], the authors present the Fiacre language which is designed both as the target language of model transformation engines from various models such as SDL, UML, AADL, and as the source language of compilers into the targeted verification toolboxes, namely CADP and Tina.

Several studies on UML transformations [6], [13], [14], [15] into VHDL-AMS have been made. In [6], the authors presented an algorithm for mapping the class diagram to VHDL-AMS. The work described in [13], presents a translation of class diagrams, objects diagram and UML state transitions into VHDL. Works described in [14] [15], present

```
ENTITE cell is
PORT (

     )

ARCHITECTURE AR OF cell IS


COMPONENT microActuator IS
  PORT (port1: in bit);
END COMPONENT microActuator;

COMPONENT microSensor IS
  PORT (port2: in bit) ;
END COMPONENT microSensor;

COMPONENT microController IS
  PORT (port3: in bit ; port4: out bit) ;
END COMPONENT microController;

BEGIN

ARmicroActuator: microActuator PORT MAP (ARport1);

ARmicroSensor: microSensor PORT MAP (ARport2);

ARmicroController: microController PORT MAP (ARport3, ARport4);

END AR ;
```

Figure 9. code generated

the translation of state machines to VHDL. These last two years, we find works on the passage of SysML to VHDL-AMS [16], [17], for purposes of simulation, particularly the transformation of the Block Definition Diagram (BDD) to VHDL-AMS.

These transformations usually generate a skeleton of VHDL-AMS code. The designer is forced to complete the model with VHDL-AMS code to exploit it. In this case, the designer can choose to directly describe its architecture with VHDL-AMS without using SysML. Our project is to generate the complete VHDL-AMS code corresponding to a SysML model. In addition to this generation, we have extracted PSL [18] properties (Properties Specification Language) from requirements diagram. These properties will be incorporated in the VHDL-AMS code for static formal verification, dynamic temporal verification, reliability, performance and energy saving verification.

VI. Conclusion and future work

SysML enjoys unprecedented popularity both in industry and academic usages. In this paper, we have presented an approach for modeling a Smart Surface micro-system at a high level of abstraction using SysML TopCased. Models are then transformed into a VHDL-AMS code thanks to ATL (for the M2M transformation) and to Xpand (for the M2T transformation).

Previous researches have demonstrated that a mapping from UML to VHDL in the digital domain is feasible. Moreover, because SysML is an UML profile, it allows us to set up a process for mapping SysML to VHDL-AMS. SysML is chosen because it allows to model mixed software and hardware at a high level of abstraction and it helps the developer of different discipline to share the same specification and to reduce the gaps between software and hardware. This first step allows the generation of a VHDL-AMS code skeleton based on structural modeling, outcome from BDD and IBD diagrams.

In order to have a complete VHDL-AMS code, we propose as future work the transformation of other SysML diagrams such as the SysML activity diagram. Activity diagram allows to model the behavioral aspect of the micro-system with concurrences or simultaneous instructions. Moreover, the parametric diagram specifies equations of the micro-system. We think to verify the full algorithm of transformation of the different diagrams with a technique of model checking to guarantee a correct code generation without ambiguities. The identification and formalization of requirements could be done by using PSL (Properties Specification Language) to describe invariant/functional/non functionnal properties. Because SysML is a high level language, we can't validate all properties of the system directly. For that, we think that PSL language could be used to express these properties as assertions in VHDL-AMS. Finally, we hope to verify these properties in this lower level.

VII. Acknowledgement

References

[1] http://www.omg.org.

[2] http://www.incose.org.

[3] http://ap233.eurostep.com.

[4] http://www.uml.org.

[5] http://www.ansoft.com/products/em/simplorer/vhdl-ams.cfm.

[6] C.T. Carr, T.M. McGinnity, and L.J. McDaid. Integration of *UML* and *VHDL-AMS* for analogue system modelling. *Formal Aspects of Computing*, 16:80,94, 2004.

[7] http://www.topcased.org.

[8] *ATL Documentation*. http://en.wikipedia.org/wiki/ATLASTransformationLanguage.

[9] Alain Giorgetti, Ahmed Hammad, and Bruno Tatibouet. Using *SysML* for smart surface modeling. In *"dMEMS'10, 1st workshop on design, control and software implementation for distributed MEMS(2010) 100–107"*.

[10] http://wiki.eclipse.org/Xpand.

[11] *ExperMeta*. http://www.expermeta.com.

[12] B. Berthomieu, S. Dal Zilio J.-P. Bodeveix and, P. Dissaux, M. Filali, S. Heim, P. Gaufillet, and F. Vernadat. Formal verification of aadl models with fiacre and tina. In *ERTSS 2010, 5th International Congress and Exhibition on Embedded Real-Time Software and Systems*, 2010.

[13] Medard Rieder, Rico Steiner, Cathy Berthouzoz, Francois Corthay, and Thomas Sterren. Synthesized *UML*, a practical approach to map *UML* to *VHDL*. In springer verlag, editor, *RISE 2005 - Rapid Integration of Software Engineering techniques, LNCS*, volume LNCS 3943, pages 203,217, 2005.

[14] Dag Bjorklund and Johan Lilius. From *UML* behavioral descriptions to efficient synthesizable *VHDL*. 2002.

[15] D. H. Akehurst, O. Uzenkov, W. G. Howells, K. D. Mcdonald-Maier, and B. Bordbar. Compiling *UML* state diagrams into *VHDL*: An experiment in using model driven development. In *Using Model Driven Development. Forum on Specification and Design Languages (FDL'07)*, 2007.

[16] David Guihal. *Modélisation en langage* VHDL-AMS *des systèmes pluridisciplinaires*. PhD thesis, Université Toulouse III, 2007.

[17] Jean Verries. *Approche pour la conception de systèmes aéronautiques innovants en vue d'optimiser l'architecture : Application au système portes passagers*. PhD thesis, Université Toulouse III, 2010.

[18] http://www.eda.org/vfv/docs/PSL-v1.1.pdf.