Transformation of SysML Structure Diagrams to VHDL-AMS

Abstract—In this paper, we propose an approach to translate the SysML language to VHDL-AMS code. This approach is based on model transformations from SysML into VHDL-AMS. We first address the block definition and internal block diagrams. The translation uses Model Driven Engineering (MDE) methods as the transformation of model to another model (M2M) with 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 micro-system case study: the Smart surface system.

Index Terms—SysML, VHDL-AMS, transformation, Model.

I. PROBLEMATIC

Micro-system design is complex because it is often composed by heterogeneous components (mechanic, electronic, software...). This complexity makes difficult to validate the system in regards of the (critical) requirements. So, it is essential to set up a process for verification and simulation before the realization and/or deployment in vehicle, bank or hospital.

Modeling of the specification is main step to ensure the success or the failure of the design. This step has three roles: i) to clarify the specification (extract the requirements) ii) to analyse the specification and to understand goal of the system (classify the requirements) iii) to communicate between the parties involved on project as users, analysts and domain experts. So the graphical model are most adapted.

SysML language is a UML profile, one of most popular graphical modeling language. It is a language to document and to graphically specify all aspects of a system (hardware and software components). SysML enjoys unprecedented popularity both in industry and academy because it is recently adopted by the OMG as a standard in Systems Engineering. It is used to gather the different actors contribution to the achievement of system, and to ensure coherence and quality of design. It is a language well suited to micro-systems, in addition to the hierarchical model of hardware and software blocks. It can graphically model the mathematical equations defining the physical system behaviors. In this context, SysML allows to formulate the requirements (functional or non-functional) with dedicated diagram: requirements diagram. Nevertheless, despite the various advantages of SysML, it remains a graphical language. So, it is not possible to simulate the model directly without define interpretative semantics or to transform it into a formal language.

Our goal is to transfer SysML models into simulation environments and verification. It would be interesting to map SysML models to languages of simulation and/or formal verification. In the context of micro-systems, a hardware description languages most used is the VHDL-AMS.

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

- The hierarchical description in the 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 using the techniques of MDE model transformations.

In this paper, we present the first part of our work. It is the transformation of the SysML structural view composed by Block Definition Diagram (BDD) and Internal Block Diagram (IBD) into VHDL-AMS. The paper is composed as follows: First, we give a reminder about languages SysML and VHDL-AMS, and the technical elements about TopCase, Xpand and ATL languages. In section 3, we illustrate our approach with micro-system case study called Smart Surface. We present, in particular the SysML structural diagrams modeling this case study. In section 4, 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 techniques to generate the VHDL-AMS code. Finally, we present some related works in this area and we conclude by presenting perspectives of our work.

II. PRELIMINARIES

A. SysML

The Systems Modeling Language SysML has been standardized by the Object Management Group (OMG) [1] with participation of the International Council on Systems Engineering (INCOSE) [2] and the AP233 consortium [3]. It is based on the actual standard for software engineering, the Unified Modeling Language (UML) [4] version 2.0, with some extensions. 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, it supports the specification, analysis, design, verification and validation of a broad range of systems.

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

1 Model Driven Engineering
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, 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, 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 adhere to 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 and, in addition, supports both structural composition and behavioural descriptions of analogue systems. VHDL-AMS can be used to model mixed signal systems, which means both event driven techniques and differential/algebraic equations are supported. Systems to be modelled using VHDL-AMS are lumped systems, that can be described by differential and algebraic equations; the solution 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 an acronym of (Toolkit In OPen source for Critical Applications & SystEms Development) is a project started in 2005 from the French “Aerospace Valley” cluster, dedicated to aeronautics, automotive and space embedded systems. It contains an IDE based on the framework of the Eclipse development platform, which adds features that provides methods and tools to develop a safety software systems or mixed software hardware systems [7].

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

D. The ATL Language

ATL, the Atlas Transformation Language, is the ATLAS INRIA & LINA research group. ATL is a model transformation language and toolkit. It was 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 an 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 information such as the module name (it is the transformation name : it must match the file name), the input and output metamodel 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 is used in 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 that you define your EMF metamodel and one or more templates which will translate the model into text. Once this definition is done you can run the code generator by defining a EMF model and by running the generator.

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 structural view. It is the Block Definition Diagram (BDD) and The internal Block Diagram (IBD) used in our approach.

A smart surface consists of a rectangular grid of rectangular cells. Each cell consists of a microActuator, a microSensor and a microController. The purpose of a Smart Surface is to sort microscopic objects according to parameters such as their shape and/or colour. The following Smart Surface characteristics are extracted from a 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 a 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 cell controller.
• A microActuator acts on the objects through an airflow (should not be specified at this level).

4. Sorting
• At a given moment on the smart surface, there is at most one object that it may be present,
• 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 main element for SysML model, which allows to structure the design.

In general purpose, the hierarchical structuring mechanism allows a better design because each level define itself details. Blocks can represent any level of the system hierarchy including the top level system, and logical or physical components of system or environment. An 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 what follows, we shall present the BDD and IBD:

- The Block Definition Diagram (BDD) defines the characteristics and relationships of 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.

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.

IV. Implementation: Transformation of Model and Code Generation

A. Model-Driven Engineering (MDE)

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.
the possibility of transforming model, of a part of model into another one. The model transformation takes an input model that conforms to the source meta-model and produces an output model, which conforms to the target meta-model. The meta-model is the model of the model, so we use “meta” to indicate this concept. When it is about a transformation of a model towards another model at the same level of abstraction, we speak then about the M2M (Model to Model). On the other hand if it is about a transformation of a model towards the other one with a different level of abstraction, then it is about a M2T (Model to Text).

Figure 3 shows a generic transformation framework that provides the context for discussion in this section. 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 the models that are manipulated by the transformation. In the transformations we have developed, these meta-models conform to the MOF\textsuperscript{2}. 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 information.

![Fig. 2. Cell IBD](image)

**B. Rules of transformation**

**B.1 Block Definition Diagram**

1. The block SysML constitutes the basic brick for the modeling of a system and which represents a support for the various elements which can characterized the system as the operations and the constraints, and of the other part. The ENTITY in VHDL-AMS represents the external view of the model and the support of various connections what allows to translate a block SysML into an entity VHDL-AMS which will take the name of the block.

\[
\{\text{SysML}:\text{Bloc} \rightarrow \{\text{VHDL-AMS} : \text{ENTITY}\}\}
\]

\[
\text{Bloc} : \text{name} \rightarrow \text{ENTITY} : \text{name}
\]

2. Flow ports represent what can circulate in entry and/or in exit of a block (item flow) and what will correspond to the material flow what allows to translate them into Port of the entity VHDL-AMS and considering the name, the direction of the port (in/out) and the type of the port.

\[
\{\text{Bloc} : \text{FlowPort} \rightarrow \{\text{ENTITY} : \text{Port}\} : \text{FlowPort} : \text{name} \rightarrow \text{Port} : \text{name}
\]

3. FlowPort : direction \rightarrow Port : direction

3. The operations that a block can contain can be translated into function and the parameters of the operation will be used as parameters of the function VHDL-AMS. The implementation of the functions is made in the body of the architecture of the entity and which represent the behavioral aspect.

\[
\{\text{bloc} : \text{operation} \rightarrow \{\text{Architecture} : \text{operation}\}\}
\]

4. Part or references properties represent the integral or not integral parts, constitute the system and serve to describe the composition and the structural architecture of the model, while components in VHDL-AMS also constitute a structural description of the circuit what allows to have a link between parts and components. They represent the same aspect, the declaration of components is made in the body of the architecture of the entity VHDL-AMS.

\[
\{\text{bloc} : \text{property : part/reference} \rightarrow \{\text{VHDL-AMS} : \text{component}\}\}
\]

5. The properties of type "Value" which define a quantifiable value with its unit, its dimension and a particular type and will be the values treated by the system. In the material case of the modelling, we speaks about Item Flow, the values which circulate by means of Flow Port and of other part one in six objects which allow to transport the information in VHDL-AMS models: CONSTANT, VARIABLE,
SIGNAL, TERMINAL, QUANTITY, FILE and we shall use one of the 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 data that flow through the system, these constraints can be specified in a block and then taken and used in the architecture of the entity as basic formulas for implementing VHDL-AMS instructions.

\{block : constraints\} \rightarrow \{VHDL - AMS : instructions\}

B.2 Internal Block Diagram

The internal block diagram describes the internal view of a block, and based on BDD to assemble parts that make up the main block. This diagram consists of two units : parts and ports. In a modeling material, we will be used 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 SysML ports with VHDL-AMS ports.

\{IBD : part\} \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 (metamodel to metamodel). 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 BBD and IBD diagrams as meta-model source and the second for VHDL-AMS as a 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 with Topcased. We extract the information from TopCased with the XMI exported. We parse this XMI file with JDOM (API for processing XML documents with Java) to extract the information about the BBD and IBD diagram and put it on SysML Meta-model. This step can be done directly (without XMI file, but not in current version of tool because Topcase meta-model for SysML is not compliant to OMG). We can use ATL to translate it into VHDL-AMS meta-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 that you define your EMF meta-model and one or more templates which will translate the model into text. Once this definition is done you can run the code generator.[10]
In our example, we have the meta-model and the model of the VHDL-AMS, to generating code, we must create a Xpand project and configure the workflow necessary for the code generation and finally creating the code of the template. On the first line in figure 8, we import the metamodel (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, also called a template. This is the smallest identifiable treatment unit. It used to define the main reference to the workflow to identify the model elements to generate the VHDL-AMS code. Fig 9 shows the result of generation.

V. RELATED WORK

There are some works for 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 model transformations of graphical languages [6], [13], [14], [15] into VHDL have been made and are being made through the transformation of UML models to language VHDL-AMS. In [6], the authors presented an algorithm for mapping the class diagram to VHDL-AMS. The work described in [13], presents a translation class diagrams, objects and UML state transitions to VHDL. Works described in [14], [15], presents the passage 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 code VHDL-AMS. The designer is forced to complete the VHDL-AMS can 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 code VHDL-AMS corresponding to a SysML model, in addition to this generation, we have extracted, diagrams from requirements, properties rewritten in PSL [18] that will be incorporated in the code VHDL-AMS for static formal verification and/or dynamic temporal properties, reliability, performance and energy saving.
SysML enjoys unprecedented popularity both in industry and academic usages. In this paper, we were presented an approach for modeling a smart surface micro-system at a high level using SysML TopCased and transforms these diagrams into a VHDL-AMS code. Basing on the ATL for the M2M transformation and Xpand for the M2T transformation. Previous research has demonstrated that a mapping from UML to VHDL in the digital domain is feasible, because SysML is a UML profile allows us to set up a process for mapping SysML to VHDL-AMS. SysML is chosen because it allows a mixed modeling software and hardware at high level of abstraction and help the developer of different discipline to share the same specifications and to reduce the gaps between software and hardware. This first step allows a skeleton of VHDL-AMS code based on structural modeling, outcome from BDD and IBD diagrams. To have a complete VHDL-AMS code, we propose as future work the transformation of other SysML diagrams such as the SysML activity diagram that allows us to model the behavioral aspect of the micro-system and to complete the VHDL-AMS with concurrences or simultaneous instructions, and the parametric diagram to add the constraints 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, formalization and structuring of a system requirements outcome from the requirement diagram with the possibility to use it to describe invariant/functional/non functionnal properties. Because the SysML is a heigh level language, we can’t validate all properties of the system directly. For that, we think the PSL language is used to express these properties as assertions in VHDL-AMS and we hope to verify these properties in this lower level.

VI. Conclusion and future work

SysML enjoys unprecedented popularity both in industry and academic usages. In this paper, we were presented an approach for modeling a smart surface micro-system at a high level using SysML TopCased and transforms these diagrams into a VHDL-AMS code. Basing on the ATL for the M2M transformation and Xpand for the M2T transformation. Previous research has demonstrated that a mapping from UML to VHDL in the digital domain is feasible, because SysML is a UML profile allows us to set up a process for mapping SysML to VHDL-AMS. SysML is chosen because it allows a mixed modeling software and hardware at high level of abstraction and help the developer of different discipline to share the same specifications and to reduce the gaps between software and hardware. This first step allows a skeleton of VHDL-AMS code based on structural modeling, outcome from BDD and IBD diagrams. To have a complete VHDL-AMS code, we propose as future work the transformation of other SysML diagrams such as the SysML activity diagram that allows us to model the behavioral aspect of the micro-system and to complete the VHDL-AMS with concurrences or simultaneous instructions, and the parametric diagram to add the constraints 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, formalization and structuring of a system requirements outcome from the requirement diagram with the possibility to use it to describe invariant/functional/non functionnal properties. Because the SysML is a heigh level language, we can’t validate all properties of the system directly. For that, we think the PSL language is used to express these properties as assertions in VHDL-AMS and we hope to verify these properties in this lower level.

References