Automatic generation of SysML diagrams from VHDL code

Fateh Boutekkouk
Research Laboratory on Computer Science's Complex Systems (ReLa(CS)^2), University of Oum El Bouaghi
Oum El Bouaghi, Algeria
Fateh_boutekkouk@yahoo.fr

Sofiane Zaidi
Dept. of Mathematics and Computer Science
Mohamed Cherif Messaadia University
Souk Ahras, Algeria
sofiane_zaidi@yahoo.fr

Abstract—**In the last years, the SysML standard is attracting more attention from hardware designers. As UML, SysML has been used to automatically generate an HDL code written in SystemC, Verilog and VHDL. Contrarily to most existing works, we propose in this paper, a new reverse engineering approach to generate SysML definition bloc and internal bloc diagrams from VHDL code. Code generation is done on the basis of a set of well defined mapping rules between SysML and VHDL concepts. The benefit of our work is to enable both hardware and software designers to maintain and comprehend VHDL programs.**

Keywords—VHDL; SysML; Reverse Engineering; Bloc Definition Diagram ; Internal bloc diagram

I. INTRODUCTION

Reverse Engineering (RE) can be defined as the process of abstract graphical models generation from textual low level models or programming language code. This process is generally passed by several steps and can employ models transformation technology by explicitly specifying Meta-models of the source and target languages and transformation rules in a certain language. The big advantage of RE is to enhance the maintenance and comprehensibility of legacy textual code by transforming it into more readable and comprehensive models.

As in software domain, RE is becoming more interesting in hardware domain especially when new graphical standard languages are issued such as UML and their extensions. Among UML extensions, we find SysML [2], the new OMG standard for Systems modeling and documentation. SysML brings some modifications on UML diagrams and new diagrams more suitable for hardware and systems in the large sense including mechanical, electronic and physical systems.

On the other hand, VHDL [10] is a well known IEEE standard for electronic systems simulation and synthesis. VHDL supports hierarchy, reuse and some software related concepts. VHDL code is textual and often hard to maintain especially for complex systems where the need for a more readable notation is mandatory.

In this work, we try to present our approach for bloc definition and internal bloc SysML diagrams generation from VHDL code. Code generation is done on the basis of a set of corresponding rules between SysML and VHDL concepts. The rest of paper is organized as follows: section two is devoted to related works. Section three presents the main concepts of VHDL. In section four, we put the light on SysML. Our approach is discussed in section five and the developed tool with an illustrative example is presented in section six before the conclusion.

II. RELATED WORK

As UML, SysML have been used as a front language to generate hardware description languages code in SystemC [7], VHDL or Verilog [9]. The works in [1, 3, 6] employed SysML bloc definition, internal bloc and activity diagrams to automatically generate VHDL-AMS code. The transformation rules are defined in ATL language. An other technique to generate VHDL code is to generate an XML format from SysML diagrams then generate VHDL code from XML [5].

The work in [4] generates SystemC code from SysML diagrams.

According to literature, we remark a scarcity in works targeting SysML diagrams generation from VHDL code. We think that such a reverse engineering process is very important because it will enable both hardware and software designers to maintain and comprehend legacy VHDL code. The focus of the presented work is on generation of static SysML diagrams from VHDL code.

III. VHDL CONCEPTS

The main VHDL design unity is called an entity. The latter gives the external view of the system as a black box. The entity exhibits the list of ports, signals and may be some assertions and global variables, constants and declarations. The entity implementation part is called architecture. The latter specifies the function of the system and can be expressed as behavioral using processes, data flow or structural using preconceived components or internal blocks. Processes are viewed as infinite loops which are sensitive to some signals (sensitivity list) blocked using wait statements and resumed whenever some events occur on its sensitivity list. We can attach to an entity many architectures (configuration). Beyond processes and signals, VHDL use some software related concepts like variables, files, procedures, functions, control, sequential statements and packages. The main entity in VHDL is the test bench without ports.

The Port mapping concept permits us to connect design components using signals. Of course there are a set of design rules in VHDL and all designers must respect them. VHDL is used for functional/temporal simulations (due to its event based nature) and synchronous/asynchronous hardware synthesis. Recently, a new version of VHDL called VHDL-AMS to model analogical/mixed parts was occurred.

IV. SysML

SysML is the new OMG [8] standard for systems modeling, it extends UML by modifying some UML2 diagrams more suitable to model hardware related concepts such as composite and activity diagrams and introducing some new diagrams such as requirement and parametric. The SysML diagram taxonomy is shown in Figure 1.

The focus of this paper is on definition bloc (DBD) and internal bloc (IBD) diagrams. These two diagrams with the parametric diagram represent the structural view of the system under design.

DBD defines features of blocks and relationships between blocks such as associations, generalizations, and dependencies. It captures the definition of blocks in terms of properties and operations, and relationships such as a system hierarchy or a system classification tree. IBD captures the internal structure of a block in terms of properties and connectors between properties. A block can include properties to specify its values, parts, and references to other blocks. Internal blocs are connected via ports. A flow port is a new concept introduced by SysML. Flow properties specify the kinds of items that might flow between a block and its environment, whether it is data, material, or energy. Contrary to standard ports, Flow ports are more suitable to model hardware ports.

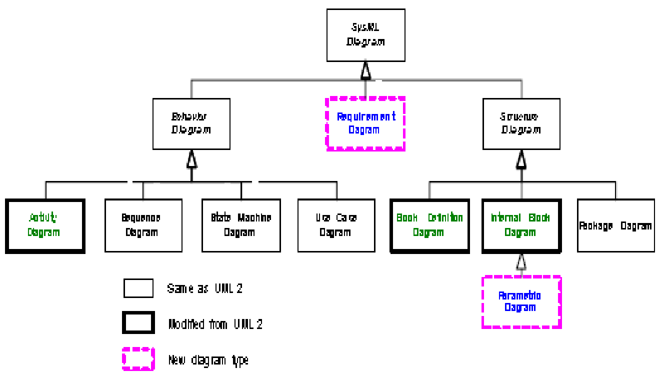


Fig. 1. SysML taxonomy [8]

V. Our approach

As mentioned before, our objective is to develop a tool for automatic SysML diagrams generation from VHDL code. Figure 2 shows our proposed approach. It consists mainly of three steps:

A. *VHDL code simulation*

This step is very important to produce correct SysML diagrams. We can for instance resort to *ModelSim* tool for functional/temporal simulations.

B. *Lexical analysis*

The aim of this step is to collect VHDL constructs and identifying them. A VHDL design may contain many files .vhd.

C. *Elements structuring*

In this step, we organize the VHDL elements in a hierarchal fashion. To facilitate this, we can use XML.

D. *SysML diagrams generation*

This is the last step for diagrams generation. This step requires VHDL/SysML mapping rules.

VHDL/SysML Mapping rules are presented in table 1. These mapping rules are defined according to the semantic attached to each VHDL/SysML concept.

Each VHDL entity is mapped to SysML bloc. The test bench in VHDL is the top level entity without any ports; it corresponds to SysML top level bloc. In order to manage complexity, VHDL decomposes its design to many files and packages containing procedures/functions, types, constants, global variables and signals definition. The USE instruction permits to the design to import all objects defined in a certain standard or user package. This instruction is close to heritage in object oriented paradigm. All procedures and functions represent classes' methods in SysML. Components reuse in VHDL is mapped to an aggregation relation in SysML. VHDL components are defined in separate files. VHDL blocs are defined in a design and can not be reused outside the design, so VHDL blocs definition is transformed to a composition relation. Number of component and/or bloc instances represents cardinality in SysML. VHDL processes and signals are stereotyped in SysML. Each VHDL port is mapped to a SysML flow port and each VHDL port mapping is interpreted as a SysML link. VHDL Assertions can be interpreted as SysML constraints and VHDL architecture can be implemented as a SysML Statechart if the behavior is state-based or Activity Diagram if the behavior is data-based (data flow).

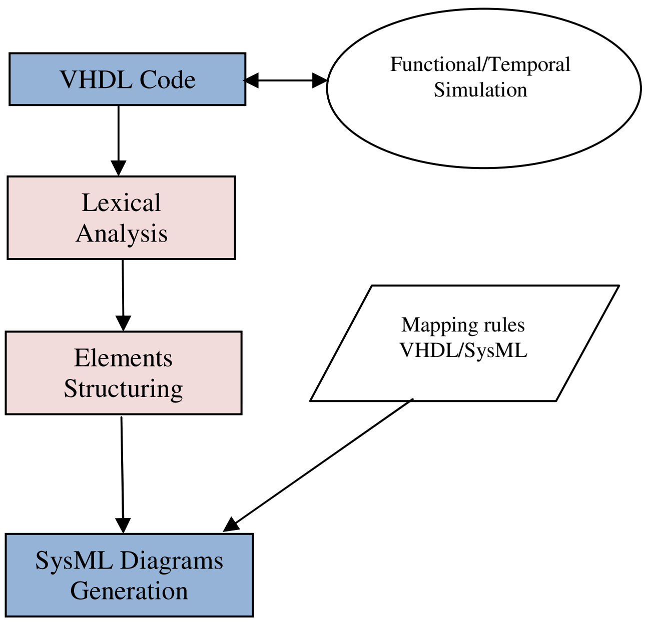


Fig. 2. Our flow

TABLE I
VHDL/SYSML MAPPING RULES

VHDL concept	SysML concept
Entity	Bloc
Test bench	Top level Bloc
Use	Heritage relation
Component reuse	Aggregation relation
Port	Flow port
Bloc definition	Composition relation
Process	Stereotype « Process »
Signal	Stereotype « signal »
Variable	Variable
Component / bloc instance	Class instance
Procedure/function	Class method
Port map	Link between two Flow ports
Number of instances	Cardinality
Architecture	StateChart /Activity diagrams
Assertion	Constraint

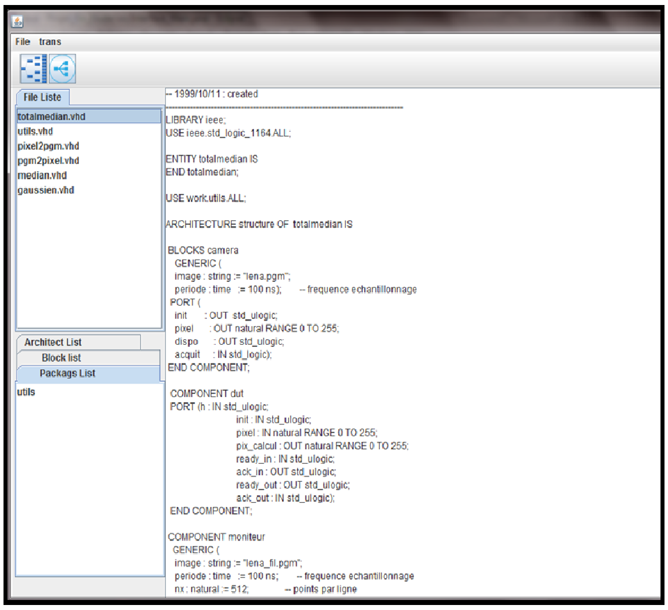
VI. OUR TOOL

We implemented our tool in JAVA under the Eclipse environment. The tool enables designers to import VHDL files (.vhd) and then generate SysML bloc definition and internal bloc diagrams. All design components and packages are placed in separate files. The code bellow represents a VHDL code of an entity called *totalmedian*. The latter is composed of an instance of the 'camera' bloc and of two reusable components 'dut' and 'monitor'.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY totalmedian IS
END totalmedian;
USE work.utils.ALL;
ARCHITECTURE structure OF  totalmedian IS
BLOCKS camera
GENERIC (
image : string := "lena.pgm";
periode : time   := 100 ns);
PORT (
init       : OUT  std_ulogic;
pixel       : OUT natural RANGE 0 TO 255;
dispo       : OUT std_ulogic;
acquit      : IN std_logic);
END COMPONENT;
COMPONENT dut
PORT (h : IN std_ulogic;
      init : IN std_ulogic;
      pixel : IN natural RANGE 0 TO 255;
      pix_calcul : OUT natural RANGE 0 TO 255;
      ready_in : IN std_ulogic;
      ack_in : OUT std_ulogic;
      ready_out : OUT std_ulogic;
      ack_out : IN std_ulogic);
END COMPONENT;
COMPONENT moniteur
GENERIC (
image : string := "lena_fil.pgm";
periode : time   := 100 ns;
nx : natural := 512;
ny : natural := 512);
PORT (
init       : IN  std_ulogic;
pixel       : IN  natural RANGE 0 TO 255;
dispo       : IN  std_ulogic;
acquit      : OUT std_ulogic);
END COMPONENT;
SIGNAL   h,   init,   ready_in,   ack_in,   ready_out,   ack_out
:std_ulogic;
SIGNAL pixel, sortie : integer;
```

```
FOR C1 : camera
USE ENTITY work.pgm2pixel(pour_cci);
FOR d1 :dut
USE ENTITY work.filtre(median);
FOR M1 :moniteur
USE ENTITY work.pixel2pgm(ecriture);
TYPE tableau IS ARRAY (positive  RANGE<>) OF natural
RANGE 0 TO 16*255;
BEGIN  --par_constantes
C1 : camera GENERIC MAP (
image => "/net/ens/nouel/pub/filtrage/lenabruit.pgm")
PORT MAP (
init   =>init,
pixel  => pixel,
dispo  =>ready_in,
acquit =>ack_IN );
d1: dut
PORT MAP (h, init, pixel, sortie, ready_in,ack_in,ready_out,
ack_out);
M1 :moniteur
GENERIC MAP (
image => "median.pgm")
PORT MAP (
init   =>init,
pixel  => sortie,
dispo  =>ready_out,
acquit =>ack_out);
pour_circ : horloge(h, 50 ns , 50 ns);
END structure;
```

Figure 3 presents our tool interface. Figures 4 and 5 show respectively the generated DBD and IBD for the example.



```
-- 1999/10/11 : created
--------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY totalmedian IS
END totalmedian;

USE work.utils.ALL;

ARCHITECTURE structure OF  totalmedian IS

 BLOCKS camera
   GENERIC (
   image : string := "lena.pgm";
   periode : time   := 100 ns);        -- frequence echantillonnage
  PORT (
   init       : OUT  std_ulogic;
   pixel      : OUT natural RANGE 0 TO 255;
   dispo      : OUT std_ulogic;
   acquit     : IN std_logic);
 END COMPONENT;

  COMPONENT dut
  PORT (h : IN std_ulogic;
                  init : IN std_ulogic;
                  pixel : IN natural RANGE 0 TO 255;
                  pix_calcul : OUT natural RANGE 0 TO 255;
                  ready_in : IN std_ulogic;
                  ack_in : OUT std_ulogic;
                  ready_out : OUT std_ulogic;
                  ack_out : IN std_ulogic);
  END COMPONENT;

 COMPONENT moniteur
   GENERIC (
   image : string := "lena_fil.pgm";
   periode : time   := 100 ns;       -- frequence echantillonnage
   nx : natural := 512;              -- points par ligne
```

Fig. 3.   Our tool

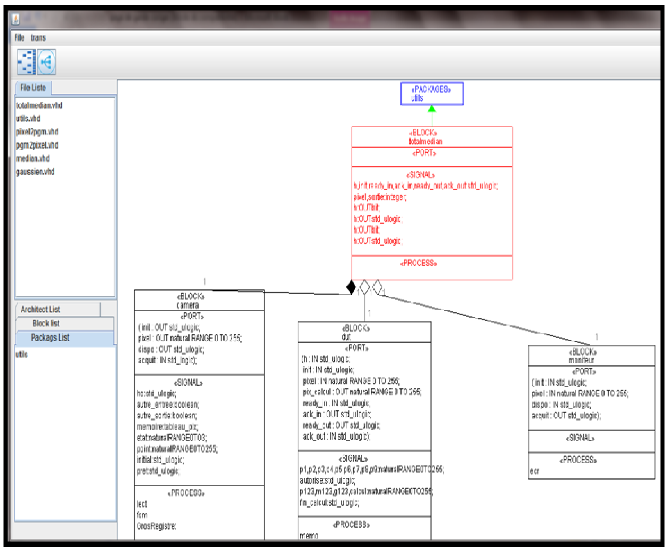


Fig. 4.   DBD generation

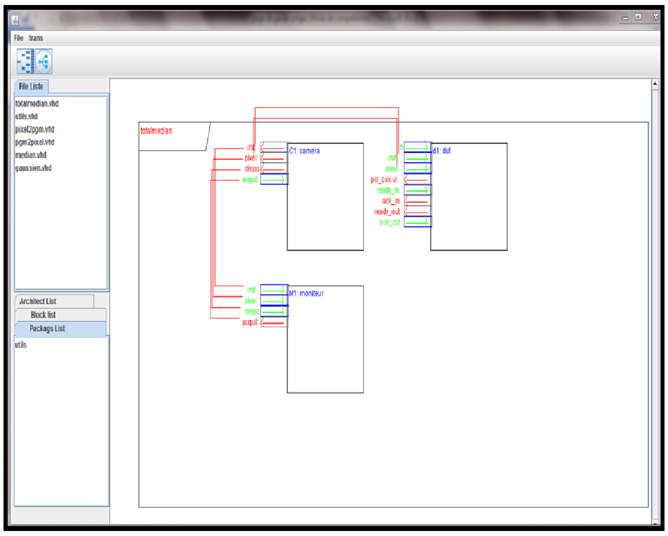


Fig. 5. IBD generation

VII. Conclusion and Perspectives

In this work, we developed a reverse engineering approach and its corresponding tool permitting SysML DBD and IBD automatic generation from VHDL programs following well defined matching rules between VHDL and SysML concepts.

The benefit of our work is to enable hardware and software engineers to comprehend and maintain VHDL legacy code by transforming it to graphical notations in SysML. Since VHDL is a textual language, our tool will minimize the effort to read and understand very long VHDL programs. The focus of this work was on structural aspects of VHDL programs. As a perspective, we plan to test our approach on more complex examples and generate SysML dynamic diagrams such as Statecharts and activity diagrams from VHDL code. Another perspective will be the application of models transformation technology to generate SysML diagrams. In this case, we have to model both meta-models of VHDL and SysML and transformation rules explicitly.

References

[1] J-M. Gauthier, F. Bouquet, A. Hammad, F. Peureux,"Transformation of SysML structure diagrams to VHDL-AMS". In dMEMS - 2nd Workshop on design, control and software implementation for distributed MEMS - 2012 2012 Second Workshop on Design, Control and Software Implementation for Distributed MEMS (dMEMS) ,2012, pp.74-81.

[2] J. Holt and S. Perry, SysML for Systems Engineering. Institution of Engineering and Technology, London, United Kingdom, 2008.

[3] K. OURFELLA, Transformation de modèles SysML vers VHDL-AMS. Mémoire de stage Master 2 recherche. UNIVERSITE DE FRANCHE COMTE S2Lsoutenue le 21 juin 2011.

[4] M. Prevostini and E. Zamsa. "SysML Profile for SoC Design and SystemC Transformation". ALaRI, Faculty of Informatics University of Lugano via G. Buffi, 13, 5 2007.

[5] S. Stancescu, L. Neagoe, R. Marinescu, E. P. Enoiu. "A SysML model for code correction and detection systems". In MIPRO 2010, Opatija, Croatia. 2010.

[6] J. Verries and A. Sahraoui. "Case Study On SYSML and VHDL-AMS for Designing and Validating Systems". Proceedings of the World Congress on Engineering and Computer Science 2013 Vol. I WCECS 2013, 23-25 October, 2013, San Francisco, USA.

[7] www.systemc.org

[8] www.sysml.org

[9] www.verilog.com

[10] www.vhdl.org