HDL Model Verification Based on Visualization and Simulation

Dominik Macko, and Katarína Jelemenská, *Member, IAENG*

***Abstract*— The role of hardware description languages (HDLs) in a current digital systems development process is essential. Their great contribution is undeniable, but they also bring about several disadvantages. The textual form of HDL models is less illustrative for a novice designer than a schematic representation of the model structure. Next, the simulation of such models is most commonly displayed in a waveform representation, even though sufficient for verification, but hard-to-identify design errors. In this paper, we present our progress in developing a visualization environment, which is able to display the simulation results in the structural sphere of the model, allowing the designer to switch between individual hierarchic levels of the structure and watching the signal changes directly in a verified component.**

***Index Terms*— digital system, hardware design, simulation results visualization, model verification**

I. Introduction

THE standardization of VHDL (Very-High-Speed Integrated Circuits HDL) in 1987 initiated the massive development of supporting EDA (Electronic Design Automation) tools. The HDLs in general brought a lot of significant advantages into the digital system design process including clearer design with fewer mistakes, verification by simulation in early stages, and technology independence. However, HDLs brought about some disadvantages as well. The textual form of the structural HDL model is less illustrative for human being than a schematic representation. In a graphical representation, it is usually easier to detect the possible design errors – such as incorrect ports interconnection, or the inappropriate component usage. Consequently, many of the complex HDL design development environments support the conversion of an HDL model to its schematic representation (see Table I). However, for simulation-based design verification a waveform representation of simulation results is the only one available in most of these tools. Although, this representation has a high verification power, especially the inexperienced designers find it hard-to-read and difficult to reveal the potential errors.

These problems could be addressed by a tool that can display the simulation results directly in the schematic representation of an HDL model, and at the same time provides the possibility to move among the levels of the hierarchy. If the designer could observe the data flow through the individual components of the design, the model verification would be simple and more effective.

The comparative analysis of several currently available related tools is summarized in Table I. The simple controllability was evaluated based on the intuitiveness of the environment. The commercial property is based on the price of the tool – free of charge or not. Other aspects concern the supported functionality of the analyzed tools.

Based on the analysis, several possibilities of simulation flow visualization are offered. Connection color based logical value display is used in most of the logic circuit simulators (e.g. LOGiX [11]). Another possibility is to change a color of the port itself (used in TINA Design Suite [8]). These two possibilities work perfectly for one-bit signals, but when transmitting multi-bit signals one connection/port color cannot represent the logic value of multiple bits. However, these possibilities cannot be resolutely refused, because if each signal bit is transmitted through separate connection/port, this problem would be solved. In HDL simulator ModelSim [5] a connection label is used for displaying the signal value which is working well for multi-bit signals as well. A combination of the mentioned methods to display the simulation results have been used in [12] but only SystemC language is supported by this tool. What is more, the analyzed commercial tools are very complex and quite expensive to be used by beginners. Therefore, a simpler, intuitive and affordably priced tool is needed especially for education and novice designers.

In this paper, we present several extensions to the HDL design visualization environment, called VHDLVisualizer. The paper is organized as follows. In the section II, the requirements for the developing environment (resulting from analysis of several existing tools) are described. The next one describes some of the important design partial problems and finally, a simple example is shown before conclusion.

II. VHDLVisualizer Design

The developing environment should take the model

Manuscript received March 22, 2012; revised April 11, 2012. This work was supported in part by the Slovak Science Grant Agency (VEGA 1/1008/12 "Optimization of low-power design of digital and mixed integrated systems").

D. Macko is with the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 3, 842 16 Bratislava, Slovakia (corresponding author e-mail: macko@fiit.stuba.sk).

K. Jelemenská is with the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 3, 842 16 Bratislava, Slovakia (e-mail: jelemenska@fiit.stuba.sk).

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
COMPARATIVE ANALYSIS OF THE TOOLS SUPPORTING MODEL STRUCTURE AND/OR SIMULATION VISUALIZATION

Tool (Author)	simple control	model structure visualization	simulation	simulation visualization in the structure display	interactive simulation	hierarchic structure	simulation flow display	commercial
HDL Author (Mentor Graphics) [1]	yes	yes	no	no	no	yes	no	yes
HDL Designer (Mentor Graphics) [2]	no	yes	no	no	no	yes	no	yes
Leonardo Spectrum (Mentor Graphics) [3]	yes	yes	no	no	no	yes	no	yes
Visual Elite HDL (Mentor Graphics) [4]	yes	yes	no	no	no	yes	no	yes
ModelSim (Mentor Graphics) [5]	no	yes	yes	yes	no	yes	waveform, connection label	yes
Active-HDL (Aldec, Inc.) [6]	no	yes	yes	no	no	yes	waveform	yes
EASE (HDL Works) [7]	no	yes	no	no	no	yes	no	yes
TINA Design Suite (DesignSoft, Inc.) [8]	no	yes	yes	yes	yes	yes	waveform, port color	yes
DirectVHDL (Green Mount. Comp. Sys., Inc.) [9]	yes	no	yes	no	no	yes	waveform	yes
VHDL Simili (Symphony EDA) [10]	yes	no	yes	no	no	yes	waveform	yes
LOGiX (CommTec Soft. Eng.) [11]	yes	yes	yes	yes	only	no	connection color	yes
SystemC + Visualizer (Turoň) [12]	no	yes	yes	yes	no	yes	connection color and label, port label	no
HDL Visualizator (Nosáľ) [13]	yes	yes	yes	no	no	yes	waveform	no
VHDL Visualizator (Zubal) [14]	yes	yes	no	no	no	yes	no	no
VHDL Visualizer (Petráš, Macko) [15, 16]	yes	yes	no	no	no	yes	no	no
FreeHDL (The FreeHDL Project) [17]	yes	no	yes	no	no	no	waveform	no
GHDL (Gingold)[18]	yes	no	yes	no	no	no	no	no
GTKWave [19]	yes	no	no	no	no	yes	waveform	no

structure description in VHDL format as an input and transform it into its schematic structure representation. The individual hierarchical levels should have separated views. Visualized objects layout modification at the selected level of hierarchy should be enabled, but the function of the model cannot change. The environment should offer simulation of the entity representing the selected level of hierarchy – either test-bench entity (top entity) or another lower level entity. The simulation results should be displayed in the visualized model structure. Export of the currently displayed entity architecture into an image format should be possible. The tool should also enable saving the modified layout into a file for further usage. The user interface has to be simple and intuitive, offering just enough functionality to serve the mentioned purposes, in order to have an advantage over existing complex environments.

To visualize and simulate the VHDL model, it has to be syntactically analyzed first. The context-free parser then transforms the VHDL description into an intermediate format. The parser generator ANTLRv3 (ANother Tool for Language Recognition) [20] was selected for analyzing the VHDL model. The VHDL analysis is based on the input grammar and the parser generator generates a group of classes (in C# or other selected programming language). It brings an advantage of supporting different HDLs. For further use of the analyzed VHDL description it is suitable to create an object model that will keep the hierarchical structure of the VHDL model.

Intermediate representation of the VHDL model, visualization, simulation and simulation results visualization of the extended VHDLVisualizer are shortly described below.

A. Intermediate Representation of VHDL Model

The analyzed information can be saved to an intermediate representation, suitable for visualization. An XML (eXtensible Markup Language) representation has the appropriate properties for this purpose since it can preserve the hierarchical structure. The XML representation is widely used and allows easy analyzing of the contained information. The similar approaches were used in [15] and [21]. In terms of visualization, there are 3 types of objects defined in the XML file: Entity instance, Port, and Connection.

The *entity instance* is the only object in visualization that can have its own structure representing the lower level of hierarchy. For each entity instance, the following parameters are extracted from the VHDL description: Entity instance name, Entity declaration name, Architecture name, and Ports list.

The *port* illustrates the interface by means of which the entity instance in the real design communicates with the other ports. Each port has the following parameters: Port name, Port mode, Port type and Port value – the formula to obtain port value in case of the lowest level entity instance output port.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

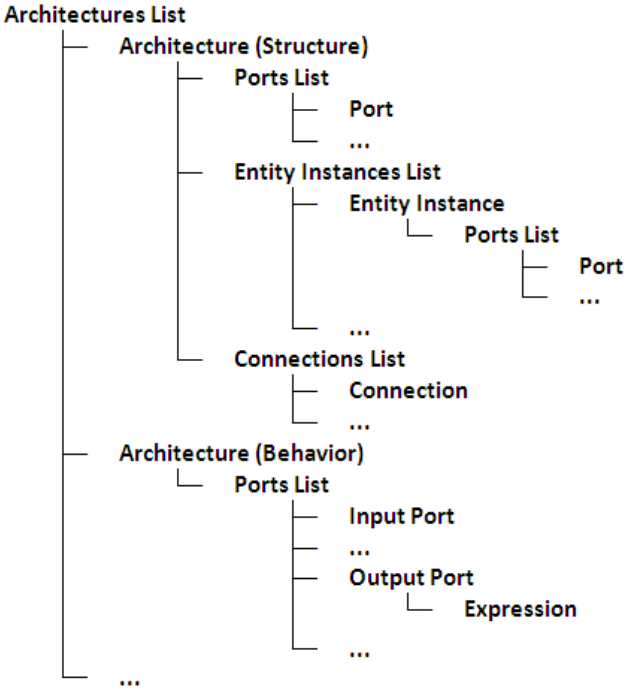

Fig. 1. Proposed XML representation schema

The *connection* joins two ports and presents their ability to communicate to each other by means of sending signals. The connections list of an entity instance includes the interconnections of its lower level structure. Each connection has the following main parameters: Connection name, Source port name and entity instance name, Destination port name and entity instance name, Source port range name, and Destination port range name.

To preserve the hierarchical structure of the original VHDL description for visualization purpose, it is sufficient to keep the information about the architectures in the VHDL description, along with their ports, connections and the entity instances inside. However, in order to simulate the model, we need to capture the behavior of the lowest level entities. For each output port of the entity describing behavior the formula for getting the output value is extracted and saved as a property of that port in the XML representation. Four types of main nodes have been defined in the proposed XML schema (see Fig. 1): Architecture (structure/behavior), Port, Entity instance, and Connection. The information in such an XML representation is an input to the visualization part.

B. *Visualization of XML Representation*

The architecture in the XML file represents one hierarchical level. In fact, it is an entity instance at the upper hierarchical level. Its description includes the ports of the given entity instance (upper level ports), the entity instances of the given level, their ports, and the interconnections among the ports. To visualize the data it is necessary to design graphical representation of each object of the architecture.

1) *Entity Instance*

An entity instance is visualized as "black box", which functionality is not known. It is represented by a rectangle (see Fig. 2a), displaying the entity name, the name of the respective instance, and the input and output ports.

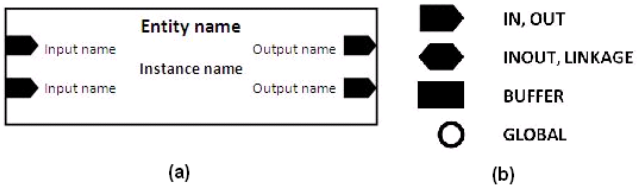

Fig. 2. The graphical representation of (a) an entity instance; (b) a port – it dependents on the port mode.

2) *Port*

A port can be specified in modes: in, out, inout, buffer, or linkage. According to the port mode, the shape of port is drawn (see Fig. 2b). The direction of the port shape (arrow) symbolizes the direction of the data flow. The ports can belong directly to an entity instance. In that case, they allow the entity instance to communicate with other instances. If the port does not belong to any instance, it represents input from or output to the upper level of hierarchy. It is also necessary to consider the special ports, which can represent the constant value sources or the globally defined signals.

3) *Connection*

A connection is represented as a line, which interconnects two ports (see Fig. 3a). In case of connection among the ports of internal instances, the signal name is displayed as well. The connections can pass from one port to several ports or vice-versa and can be single or multi-bits. Therefore branching and splitting of the connection has to be defined as well. The connection branching is the case when the same signal passes to several ports (see Fig. 3b). The connection splitting/joining is the case when the multi-bit signal splits into several one-bit signals (see Fig. 3c), or several one-bit signals join into one multi-bit signal (see Fig. 3d).

4) *Behavior architecture*

For the architectures describing behavior of an entity instead of a structure, the visual difference has to be obvious. Therefore, in case an entity instance of the lowest level is selected, an object with no instance name and no entity name will be displayed (see Fig. 4).

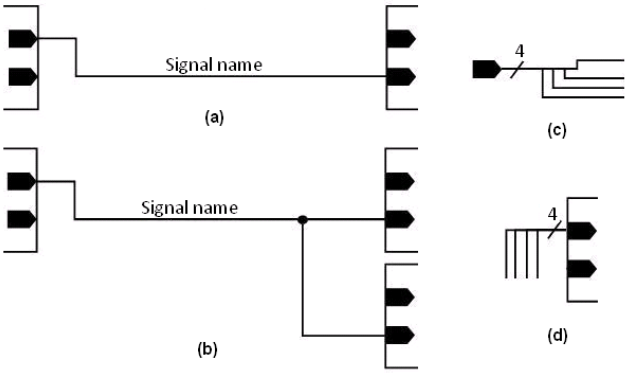

Fig. 3. The graphical representation of (a) connection, (b) branching, (c) splitting, and (d) joining.

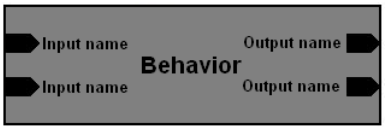

Fig. 4. The graphical representation of a behavioral description.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

C. Objects Layout Algorithm

For structural model visualization a layout optimization algorithm is used that positions the entities in two rows, sequentially, according to their occurrence in the VHDL source code. The entities are drawn in the order up down and left to right.

In the displayed hierarchical level, the ports are divided into external and internal. The internal ports are situated inside the entity instances and automatically aligned to the instance borders. The input ports are situated on the left hand side and the rest on the right hand side. The external ports belong to the upper level entity. They are displayed near the left and right border of the display surface and aligned to the connections position.

All the connections are positioned into a so called virtual bus located between the entities rows. Each connection has its own part of virtual bus reserved which ensures that the connections will not overlap in horizontal direction, therefore it is not necessary to check up on it allowing thus an algorithm speed-up. However, the vertical parts of the connections are checked up for overlapping. If there is an overlap, the connection is shifted.

D. VHDL Model Simulation

There are two ways to simulate a visualized VHDL model. The designed internal simulator allows the interactive simulation, where the designer can set up the input ports values for the selected hierarchical level. The values are spread through the design hierarchy, and subsequently the output ports values are calculated using the VHDL behavioral description loaded from the XML representation. When simulating test-bench entity, the input ports values are loaded also from XML. For the chosen simulation time, all the changes that should occur till that time are executed. The results of simulation are displayed directly in the schematic view.

Another designed possibility is to simulate the VHDL model using an external simulator that will provide the simulation results to the VHDLVisualizer. The GHDL (Gcc-based HDL) simulator [18] was chosen for this purpose due to a sufficient functionality and non-commercial characteristics (to preserve non-commercial use of VHDLVisualizer). This simulator enables to check the VHDL code syntax, to analyze it and to run the simulation saving the simulation results into the standardized VCD (Value Change Dump) format [22] for which we developed an analyzer. After the simulation is finished, the results are loaded from the generated temporary VCD file. Thus, for the visualized hierarchical level it is possible to display the value of the chosen port at the selected time. In the case lower level entity (not test-bench entity) is simulated using the external simulator, it is necessary to generate a new test-bench entity. In the architecture of this new entity, the values set up by the designer are assigned to the input ports. In this way it is possible to run interactive simulation using the external simulator as well. With a small modification of the VHDLVisualizer source code it is possible to use another external simulator generating simulation results in the VCD format.

The internal simulator is suitable mostly for the logic gate level design interactive simulation of combinatorial circuits, but it handles also the test-bench entity simulation in this level of abstraction. The external simulator is suitable mostly for the test-bench entity simulation. It handles both the combinatorial and sequential digital systems designs in any level of abstraction. However, it has several restrictions described in [18]. Both simulators have limitations concerning the VHDL support, so their combination helps to increase this support.

E. Simulation Flow Visualization

Based on the analysis of available solutions summarized in the Introduction, we decided to display the simulation flow by means of connections labelling. Moreover, the colour of the label changes when the signal state alters to improve the visibility. We compared this solution to the one where the signal value is represented by a connection colour, or a port colour. But the other possibilities had problem with the multi-bit signal displaying.

In complex architectures, where a large amount of connections is displayed, the preferred solution would not be clear enough, because the designer would have to search the connection (which might be quite long) that is connected to the monitored port in order to determine the signal value placed above it. For the designer it is more useful to see the signals' values at an entity instance inputs and outputs. Therefore, the simulation results are displayed in the labels belonging not to the connection itself, but to the ports. The labels are located above the connections, but close to the ports. The signal value actually replaces the number of bits (see Fig. 3) which is not necessary to display any longer since it is evident from the signal value.

The simulation results can also be displayed in the commonly used waveform. For this purpose the external tool GTKWave (GTK+-based Wave viewer) [19] is used. This tool reads data from the VCD file and displays the simulation results in a waveform. This form of simulation results visualization is an additional one, which can be used when needed. The internal simulator does not generate the VCD file, so the waveform visualization is only possible for the external simulation.

The internal visualization module enables to also show the three nearest signal values of the mouse-hovered port at the selected time-point. It is displayed in the form of waves, but only the previous, current and following values are shown (see Fig. 5).

III. VHDLVisualizer Evaluation

The visualization environment VHDLVizualizer was designed for Microsoft Windows operating system with the component of .NET Framework 3.5. Regarding the

Fig. 5. The three nearest signal value changes of the port c in the time-point between 15 ns and 20 ns.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
...
  <EntityInstance x="180" y="50" width="150" height="60">
    <InstanceName>ha1</InstanceName>
    <EntityName PackageName="" LibraryName="work">
half_adder</EntityName>
    <ArchitectureName PackageName="" LibraryName="work">
structure</ArchitectureName>
    <VHDLPorts>
      ...
      <Port anchor="right" x="310" y="80" width="20" height="10">
        <PortName>s</PortName>
        <PortMode>OUT</PortMode>
        <PortType>bit</PortType>
        <PortPinCount>1</PortPinCount>
        <PortValue> ha1.xor1.out1 </PortValue>
      </Port>
    </VHDLPorts>
  </EntityInstance>
...
```

Fig. 6. Fragment of VHDL model XML representation

hardware requirements, it only needs a small amount of hard disc space (2 MB), but requires higher computing power for simulation purpose (at least 500 MHz CPU and 256 MB RAM). The visualization environment has been tested using a set of VHDL models designed in the frame of school assignments that represent different designs of the combinational and sequential circuits with one and multi-bit signals. Thus VHDLVizualizer was evaluated in education process at the Slovak University of Technology but it can be used also for designers as a helpful tool for verification.

As an illustrative example we took VHDL structural description of a simple one-bit full adder. This description was analyzed, translated into XML representation and visualized using our tool. A fragment of the XML representation corresponding to the instance “ha1” and port “s” is shown in Fig. 6. It contains information about the location, size and names of the objects. Fig. 7 illustrates the simulation visualization in VHDLVisualizer for this adder. In this case the interactive mode based on the internal simulation module was used.

IV. Conclusion

The paper is devoted to the problem of visualization and simulation of digital system models described in VHDL. The core of the paper forms the design and implementation of the tool usable in digital systems design process. For a human being, the graphical representation of the structural model, generated by this tool, is more understandable and easier and faster to detect the errors made during the VHDL structural model creation. The VHDL model visualization is useful not only for verification purposes, but also for design documentation. Using the graphic schematics of the designed model in the documentation, other developers would easier and faster understand the design structure. The proposed and implemented environment includes two kinds of simulation – interactive and simulation based on a test-bench entity. For simulation results visualization in the schematic representation the solution possibility and algorithm were chosen and integrated into the VHDLVisualizer which bring the simplest, fastest, the most definite and the illustrative presentation of the simulation flow. The traditional waveform representation was added as well to allow for dual representation.

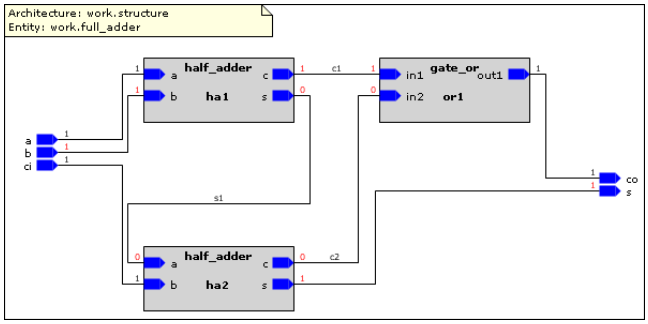

Fig. 7. Structural VHDL model visualization with simulation results displayed

Due to the unusual simulation and its visualization, this environment becomes a strong design and verification tool. The VHDLVisualizer nature makes it especially suitable for beginners in VHDL design - therefore education is the most natural area of application. However, it can become the useful verification tool for simple professional designs as well.

The tool is ready to be extended for other HDLs support such as Verilog and SystemC. Another possible extension is elimination of restrictions in the support of VHDL constructions, or object layout algorithm optimization. These extensions represent our further effort and future work.

References

[1] Mentor Graphics, “HDL Author – Manage Design Data and Flows,” Mentor Graphics's products, Online, January 2012. www.mentor.com/products/fpga/hdl_design/hdl_author
[2] Mentor Graphics, “HDL Designer,” Mentor Graphics's products, Online, January 2012. www.mentor.com/products/fpga/hdl_design/hdl_designer_series
[3] Mentor Graphics, “Leonardo Spectrum,” Mentor Graphics's products, Online, January 2012. www.mentor.com/products/fpga/synthesis/leonardo_spectrum
[4] Mentor Graphics, “Continuous design flow from TLM to RTL - Visual Elite HDL,“ Mentor Graphics's products, Online, December 2011. http://www.mentor.com/products/fpga/hdl_design/visual-elite-hdl
[5] Mentor Graphics, “ModelSim,” Mentor Graphics's products, Online, December 2011. http://www.mentor.com/products/fpga/simulation/modelsim
[6] Aldec, Inc., “Active-HDL,” Aldec's products, Online, December 2011. http://www.aldec.com/products/active-hdl/
[7] HDL Works, “EASE Graphical HDL Entry,” HDL Works's products, Online, January 2012. www.hdlworks.com/products/ease/index.html
[8] DesignSoft, “Analog, digital, symbolic, RF, VHDL, MCU and mixed-mode circuit simulation & PCB design,” Tina Design Suite, Online, December 2011. http://www.tina.com/English/tina/start.php
[9] Green Mountain Computing Systems, “DirectVHDL for Windows,” Online, January 2012. www.gmvhdl.com/directVHDL.html
[10] Symphony EDA, “VHDL Simili,” Symphony EDA's products, Online, January 2012. www.symphonyeda.com/products.htm
[11] CommTec - Software Engineering, “LOGiX - Simulation of logic circuits,” Simtel products, Online, December 2011. http://www.simtel.net/product/view/id/90288
[12] J. Turoň, K. Jelemenská, “Contribution to graphical representation of SystemC structural model simulation,” in Proc. of the 7th FPGAword Conference, L. Lindh, V.J. Mooney, S. de Pablo, J. Öberg, Eds. Copenhagen (Denmark), September 2010, pp. 42–48.
[13] K. Jelemenská, M. Nosáľ, P. Čičák, “Visualization of verilog digital systems models,” in CISSE 2010, Bridgeport, Connecticut (USA), December 2010, in press.
[14] M. Zubal, “VHDL model visualization,” master theses, FIIT STU Bratislava (Slovakia), 2008, 80 p.
[15] J. Petráš, “VHDL model visualization,” master theses, FIIT STU Bratislava (Slovakia), 2008, 85 p.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[16] D. Macko, K. Jelemenská, "VHDL structural model visualization," in EUROCON 2011, Lisbon (Portugal), April 2011.
[17] The FreeHDL Project, "The FreeHDL Compiler/Simulator System," Online, January 2012. www.freehdl.seul.org/code/code.html
[18] T. Gingold, "GHDL – Where VHDL meets gcc," Online, December 2011. http://ghdl.free.fr/
[19] GTKWave Project, "Welcome to GTKWave," sourceforge's projects, Online, December 2011. http://gtkwave.sourceforge.net/
[20] R. M. Volkmann, "ANTLR 3," Online, December 2011. http://jnb.ociweb.com/jnb/jnbJun2008.html.
[21] M. H. Reshadi, B. Goji-Ara, Z. Navabi,"HDML: compiled VHDL in XML," in VHDL International Users Forum Fall Workshop, Tehran Univ., 2000, pp. 69-74.
[22] IEEE, "IEEE standard Verilog hardware description language," IEEE Standards (IEEE Std 1364-2001), September 2001.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)