Abstract
Throughout the world, many VHDL (Very-high-speed integrated circuit Hardware Description Language) simulators exist, but only a few of them support the visualization of VHDL model simulation. The simulation is most commonly displayed as a waveform. This representation of the visualization is sufficient for the verification of the model, however it is hard-to-read for novice designers and it is difficult to identify the potential errors. In this paper, we present our progress in developing a visualization environment, which is able to display the simulation in the structural sphere of the model. During the simulation visualization, the designer is able to switch between hierarchic levels of the structure and watch how the signal changes directly in the component that he/she wishes to verify.

1. Introduction
With the advent of the VHDL standard, various design tools began to be developed to supporting this standard. Digital system design based on HDL (Hardware Description Language) brings about several advantages (clearer design, fewer mistakes, verification by simulation, and efficiency), however it also generates several disadvantages. HDL design is less illustrative for a human being, especially in the case where it represents a digital system structure. In general, it is easier to follow the circuit structure in a graphical form than in a textual one, therefore it is easier to detect the possible mistakes in a design (e.g. caused by an incorrect interconnection). Consequently, some design tools support some form of visualization of the model structure under design (see Table 1).

For the visualization of the simulation results, most of these tools (if supporting simulation) still use only a waveform representation. For complex structural models this display is hard-to-read and during such model verification it is difficult to reveal the source of the error – especially for inexperienced designers. Therefore it would be more useful to display the simulation results directly in the schematic representation of the HDL model, and at the same time provide the possibility to move among the levels of the hierarchy. Such visualization of the simulation would simplify, clarify and speed-up the process of model verification. The designer would be able to
Table 1. Comparative analysis of the tools supporting model structure and/or simulation visualization

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

accurately identify the erroneous components and more effectively eliminate the error.

Since only few existing VHDL simulators support simulation visualization in the structural sphere, those tools with related problems are used for inspiration. In the Table 1, some of the existing design tools are shown. Some of them support a kind of simulation visualization, others possess inspirational functionality.

2. Solution possibilities

The existing tools offer several possibilities for displaying the simulation flow of the VHDL model. Most commonly, the waveform is used for representing the flow and results of the simulation. This representation is insufficient for the needs of our visualization. It can only serve as an additional display of a simulation flow. In priority visualization, we should be able to observe the simulation flow in individual hierarchic levels as a signal flow through VHDL model structure. Therefore, the type of signal value representation required needed to be chosen. The logic value, which the signal is carrying, can be displayed by different colored connections (the logic
But a problem can occur if the signal is multi-bit (more bits are transmitted through one connection) – since the colors of each bit cannot be displayed in one connection, so the designer would not see the transmitted signal value. However, this solution can not be resolutely refused, because if each signal bit is transmitted through separate connections, this problem would be solved. A different way of showing the transmitted logic value of the signal is in the connection label (the HDL simulator ModelSim [13]). This solution is also clear for multi-bit connections. It can be improved by changing the color of the label at the signal state change. The last visualization possibility (from existing solutions) is by different colored ports (the design suite TINA [14]). The same problem can occur, but also the same solution, as in the case of different colored connections. Some combinations of the mentioned methods to display the simulation results have been used in [7]. However, that solution is too unnecessarily complicated.

3. Visualization environment

The visualization environment was already proposed in [11] and can be described by several basic actions: analysis of the loaded VHDL model, binding of the analyzed information to the right objects, saving useful information into transient XML representation, and visualizing the objects from the XML representation. We now have to add other actions to these, to enable both the simulation and simulation visualization. VHDL model simulation can be provided by integrating the simulator into our visualization environment, or by using an existing freeware simulator, which can provide the simulation results to our tool. The environment requirements state that the visualization environment should be supplemented by the following:

- ability to simulate the visualized model
- ability to only simulate the required hierarchic level
- possibility to display the simulation flow in the structural sphere of the individual hierarchic level

The creation of a new internal simulator is a difficult task, which requires the need to analyze the whole VHDL model, not only those parts needed for visualization purposes. The next step is an extension of the transient XML representation by adding the information that is important for simulation purposes. Subsequently, it is necessary to load this additional information, along with the information for model structure visualization. The simulation module is created for simulating individual hierarchic levels, using the VHDL behavioral description loaded from the XML representation. An internal simulator can be used for the interactive simulation, where the designer establishes the input port values for the simulated hierarchical level. These values are spread through the design hierarchy, and subsequently the output port values are computed. In case of a test-bench entity, waveform constructions are analyzed and the simulation is created using the internal simulator. So the designer can look through the simulation results. For the chosen simulation time, all the changes that should occur till that time are executed, and the results of simulation are visualized.

The use of an external simulator is much easier, with the assumption of finding a suitable simulator. In the introduction chapter the GHDL [18] simulator is also mentioned, which is suitable for use with our visualization environment. This simulator can save the simulation results into the standardized VCD format for which it is easy to create an analyzer. After the simulation is finished, the results are loaded from the generated temporary VCD file. These results are assigned to the objects, which the visualization environment assigns to the individual ports. Thus, for the visualized hierarchical level it is possible to display the value of the chosen port at the selected time.

The GHDL [18] simulator works as a command-line program, so it is very easy to access the program from another tool. This simulator enables checking of the VHDL code syntax, its analysis and running of the simulation. In the case of simulating anything other than the top entity (test-bench entity) 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 also run interactive simulations using the external simulator. Both simulators have limited support of VHDL, so their combination helps increase this support.

4. Simulation visualization design

In the chapter on solution possibilities, we reached the conviction that it is preferable to display the simulation flow by means of connections labeling. This solution was improved by changing the color of the label when the signal state changes (Fig. 1a). We compare this solution to the one where the signal value is represented by a
connection color (Fig. 1b), or a port connector color (Fig. 1c). In the solution in Fig. 1b, we have to transmit each bit of a multi-bit signal through separate connection, in order to see the value of the individual bit (or for each bit, there have to be separate ports or at least separated port connectors as in Fig. 1c).

In complex architectures, where a large amount of connections is displayed, the preferred solution (illustrated in Fig. 1a) 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 preferred solution was improved. 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, as is illustrated in Fig. 1d. The signal value will replace the number of bits label used in [11]. It is not necessary to display the number of bits any longer since it is clear from the signal value.

The simulation results can also be displayed in the commonly used waveform. This is enabled using the external tool GTKWave [19]. This tool reads in the information from the VCD file (generated by the GHDL [18] simulator) and then 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 GHDL [18] simulation.

5. Solution verification

The visualization environment is designed for Microsoft Windows operating system with the component of .NET Framework 3.5. Regarding the hardware requirements, it only needs a small amount of hard disc space (2 MB), but requires high 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 created in school assignments. These models have represented different designs of the combinational and sequential circuits with one-bit, and also multi-bit, signals. The simulation visualization in VHDL Visualizer for the simple example of a full one-bit adder is shown in Fig. 2. The testing has revealed some issues that have been mostly solved later.

Fig. 1 (a) Display of the value in the connection label; (b) Display of the value represented by a connection color; (c) Display of the value represented by port connector color; (d) Display of the value represented by a special port label

Fig. 2 Visualization of the model structure with the simulation results displayed
6. Conclusions

This paper is devoted to the problem of visualization of a VHDL model simulation in its structural sphere. At the beginning, several existing tools were described which support some form of simulation flow visualization. In the main, only waveform representation is supported. Although this representation is the most common it is hard to read especially for the novice designer. Based on existing tools several possibilities of simulation result visualization in the sphere of the structure were analyzed, and their advantages and disadvantages were discussed.

The core of the paper describes the extension of a previously developed tool called VHDL Visualizer by adding simulation possibilities. This extension includes two kinds of simulation – interactive and simulation based on a test-bench entity. In addition, both kinds of simulation can be executed using the internal simulator, as well as by the freeware external simulator GHDL [18]. For simulation result visualization in the structural sphere the solution was chosen and integrated into the VHDL Visualizer which brings the simplest, fastest, most definite and the illustrative presentation of the simulation flow. The traditional waveform representation was added as well to allow for dual representation, thus improving the tool’s verification power.

This work has enriched the VHDL Visualizer environment by the simulation and simulation visualization abilities. Due to that, this environment becomes a strong design and verification tool. The VHDL Visualizer nature makes it especially suitable for beginners in VHDL design, therefore education is the most natural area of application. However, it can become a useful verification tool for professional designs as well.

Acknowledgements

This work was partially supported by the Slovak Science Grant Agency (VEGA 1/0649/09 “Security and reliability in distributed computer systems and mobile computer networks”).

References