A Concept of Visual Programming Tool for Learning VHDL

Aneliya Ivanova
University of Ruse, Faculty of Electrical Engineering, Electronics and Automation, Department of Computing, 8 Studentska Str., 7017, Ruse, Bulgaria

e-mail: aivanova@uni-ruse.bg

Abstract. Due to the COVID-19 pandemic, distance education starts playing a crucial role in higher education and the need for development of educational tools, helping the students learn better at home cannot be ignored. Teaching programming languages online is a complicated task and when the course subject is programmable logic design through Hardware Description Languages (HDLs), online teaching becomes a complex challenge. In this paper is presented a concept of a training environment that uses the visual programming technique to help the students create VHDL models of various digital devices. The students construct VHDL models by combining simple visual objects while the environment is providing guidance in real time and is preventing a wrong match of VHDL operators and signals. The strategy for building a model of a complex digital circuit by moving and connecting visual objects will help the students with visual kinesthetic learning style to internalize the concept and the structure of the VHDL model. The training environment will benefit mostly the students, learning in distance mode, but it is also useful for face-to-face students, who find it difficult to assimilate the specifics of VHDL modelling.

1. Introduction
Educating the digital generation has been widely discussed as one of the most significant challenges the higher education meets in the digital age. The COVID-19 pandemic made this challenge even more complicated. Distance education starts playing a crucial role in higher education and the need for development of educational tools that will help the students learn better at home is something that cannot be ignored. Teaching programming languages online is a challenging experience and when the course subject is programmable logic design through Hardware Description Languages (HDLs), the online teaching becomes a very complex challenge.

One of the reasons is that the development environments for HDLs are usually designed for highly trained professionals and could be complicated as an introductory tool for students learning HDLs, so the students need more support and guidance from the lecturer. Another challenge is providing online feedback and assessment of students’ HDL projects.

In [1] is proposed an open-source environment for developing and simulating VHDL (Very High Speed Integrated Circuit Hardware Description Language) programs providing automated checking of submitted students' projects and feedback about the functional correctness of the projects. In other case, a VHDL-based Network Interface Card simulation model is used as a pedagogical tool that helps the students understand better the principles of computer architectures [2]. An approach of e-assessment of VHDL code integrated in Moodle platform is presented in [3]. The implemented module checks whether the student's code can be successfully compiled and whether the simulation output of student's code is the same as the reference code simulation output.
2. Motivation
The “Design Technology” course from the curricula of “Computer Systems and Technologies” degree course introduces to the students the concepts of VHDL and the principles of programmable logic design.

The first challenge to the students is to start thinking in terms of hardware when they write VHDL code, and to assimilate the idea that each VHDL structure models a digital device. The second challenge is to get used to think in terms of signals, ports and processes running parallel to each other, rather than of variables and functions. VHDL requires strict declaration of data types and matching the types in each operation over the signals in a project. The VHDL code is compiled to a RTL (Register-Transfer Level) structure, which requires a logically justified construction of the code, which describes the behavior of the modelled device. And finally, the students have to become familiar with the structure and hierarchy of the various types of VHDL models.

In this paper is presented a concept of a training environment that uses the visual programing technique to help the students create VHDL models of various digital devices. Using the features, provided by the environment the students construct VHDL models by combining simple visual objects, while the environment is providing guidance in real time and is preventing wrong matches of VHDL operators and signals.

The visual programming approach has been discussed in several works as a good technique to enhance learning of programming languages’ principles [4, 5, 6, 7, 8, 9]. In this work it is considered mostly, because the digital students as a whole have visual-kinesthetic learning style and further, because of the fact that visual programming tools, such as Scratch and Kudo are widely used in school education for basic introduction to the principles of programming and are proved to be engaging learning tools [10]. A lot of the students taking “Design Technology” course are familiar with these environments from IT classes at school.

3. Outlining the context of the training environment
The primary objective of the environment is to facilitate the process of learning the structure of the VHDL model, as well as the basic operators and syntax of the language and the way the digital devices are modelled. The output file, generated by the environment should be in .vhd format, in order to be directly associated to a new or an existing VHDL project. The functionality of the tool should correspond to the content of the course and the specifics of the projects developed by the students not only in the lab, but also as course assignments.

VHDL supports three modeling styles: dataflow, structural and behavioral, as the latter is used for both combinational and sequential circuits modelling. For this reason, it is mostly used by the students when developing their projects. There are two approaches to model complex digital devices – through a hierarchy of .vhd files, defined in a top file with a structural architecture and through a single .vhd file with behavioral architecture, as each module of the complex device is modelled as a separate process within the architecture. Since the second approach outputs a single .vhd file, it is the most reasonable choice for the VHDL model that is to be generated by the learning tool.

4. Concept of the training environment
The students’ projects usually model devices comprising structures of sequential and combinational circuits such as registers, counters, decoders, multiplexers, comparators, etc. and some more complex devices as FSM (Finite State Machines), memory blocks, ALU (Arithmetic and Logic Unit), etc. It is appropriate in the initial stage, for the student to be offered to select a template (Figure 1) from a template library, thus avoiding inconsistent structures and combination of devices in the project. Furthermore, when the student is more confident and experienced, he/she could choose to work without a template.

For the given template (Figure1), the following visual structure is expected to be assembled by the student (Figure 2).
**Figure 1.** Exemplary design template, selectable from a template library.

**Figure 2.** A visual structure, assembled by the student (architecture segment is partially presented).
When a template is selected, the “Visual Components” panel of the environment displays only the appropriate visual components, that correspond to the template. The elements, included in the visual components library of the training environment are generally classified as follows:

- visual objects, representing design units;
- visual objects, representing declarations;
- visual objects, representing concurrent statements;
- visual objects, representing sequential statements;
- visual blocks, representing logic circuits;
- visual blocks, representing complex logic circuits.

According to their „Edit” property, the visual objects and blocks are classified as editable and non-editable meaning, that the students are able to enter names and parameters of port declarations, signals, processes, etc. when using the corresponding visual objects, or to name the inputs and outputs of a visual block representing a logic circuit. For example, on figure 2 the “Signal” object has 3 editable fields – “name”, “MSB” (most significant bit), “LSB” (least significant bit).

On Figure 2 the “architecture” segment of the VHDL model is partially shown – only Counter and Comparator elements of the template are represented. The “Decimal Counter” and “Comparator” are editable visual blocks and it is visualized what the student will see if the “Edit” button is clicked (edit mode). Since VHDL provides a flexibility to synthesize any device that could be modelled via the syntax constructs of the language, in this particular project the comparator compares two 4-bit signals and outputs the larger one.

After the visual structure is assembled correctly, the student is offered to start its’ conversion to VHDL code. The output VHDL code that is to be generated for the discussed exemplary structure is given below (architecture segment is partially presented):

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity PROJECT_1 is
port (SAMPLE : in STD_LOGIC_VECTOR(3 downto 0);
DATA : in STD_LOGIC_VECTOR(3 downto 0);
RESET : in STD_LOGIC;
LOAD : in STD_LOGIC;
CLOCK : in STD_LOGIC;
7SEGMENT : out STD_LOGIC_VECTOR(7 downto 0);
LED : out STD_LOGIC_VECTOR(3 downto 0);
ENABLE : out STD_LOGIC_VECTOR(7 downto 0));
end PROJECT_1;
architecture Behavioral of PROJECT_1 is
signal COUNT : STD_LOGIC_VECTOR(3 downto 0);
signal CODE : STD_LOGIC_VECTOR(3 downto 0);
begin
process(RESET, CLOCK, LOAD, DATA)
begin
If (RESET = '0') then COUNT <= "0000";
elsif (LOAD = '0') then COUNT <= DATA;
elsif (CLOCK'event and CLOCK = '0') then
COUNT <= COUNT + 1;
end if;
end process;
process(COUNT, SAMPLE)
If SAMPLE > COUNT then CODE <= SAMPLE; else CODE <= COUNT;
end if;
end process;
.......
end Behavioral;
```

5. Taxonomy of the visual components library
A crucial role in the implementation of the visual programming approach plays the library of the visual components used by the students to build a VHDL model. The library content is formalized as two taxonomies, respectively for visual objects and for visual blocks (Figure 3 and Figure 4). As seen on figures 3 and 4, the visual objects substitute declarations, operators, statements, keywords, as the visual blocks substitute complete logic circuits.
A VHDL model could be constructed only by visual objects, but for the students it will be easier and more convenient to use visual blocks for their first attempts with the training environment.

6. Interface wireframe of the training environment
The user interface of the training environment is shown on Figure 5 as a design wireframe. It comprises 6 areas, as 4 of them are zones of active interaction, such as Menu Bar, Project Templates Panel, Visual Components Panel and Construction Panel.

The menu bar has two drop-down menus – the first one providing all the options concerning the project – new, save, save as and close, and the second one giving the user options to customize the working area through showing/hiding the panels currently not in use. The rest of the menu items are buttons giving options to check the consistency of the project, to generate VHDL code and to output the generated code to .vhd file.

The Project Templates Panel allows a choice between two categories – relatively simple project templates and templates of projects with FSM.

The Visual Components Panel follows the taxonomy of the visual components library and offers 4 categories of visual objects and 2 of visual blocks as the subcategories are organized the same way they are represented in the taxonomy. When a specific template is selected, some categories that are not needed for the project are hidden for the user, to prevent confusion and mistakes.

The Visual Components Panel and the Construction Panel are the most actively used interface areas. The student selects visual objects and blocks and places them in the construction area via drag-and-drop operation until the structure of the VHDL model is ready. The further action is to check whether the assembled structure is correct, and after successful check to start VHDL code generation. The VHDL code area is a passive interaction area, since the student is only able to observe the produced code.
The second passive area is the **Feedback Panel**, where the student gets real-time guidance in the form of hints for best match of visual components, or warnings and system messages when an attempt of incorrect operation is performed. The only active action in this panel is selection of the "**Help**" option for more detailed instruction manual.

![User Interface Wireframe](image)

*Figure 5. User Interface Wireframe.*

The interface is organized in a manner that resembles the interface of a typical IDE (Integrated Development Environment), the only difference is the place of the feedback area, which is located in the upper part of the window, because the feedback is an important feature of any educational environment.

**7. Conclusions**

The visual programming approach and the strategy for building a model of a complex digital circuit by moving and connecting visual objects will help the students with visual kinaesthetic learning style to internalize the concept and the structure of the VHDL model.

The guidance, provided by the environment as alerts and recommendations when attempts of connecting non-matching objects are detected, will teach the students how to correctly build a VHDL structure and will form good development skills.

The option to select a design template will help the novice students easily understand the structure of their project assignments and will ease the guidance, provided by the training environment through the construction phase.

The training environment will benefit mostly the students, learning in distance mode, but it is also useful for face-to-face students who find it difficult to assimilate the specifics of VHDL modelling.
Acknowledgments

This work/paper is supported by project 20-FEEA-01 “Methods and tools for multimedia content analysis, and automated document and big data processing”, funded by the Research Fund of the “Angel Kanchev” University of Ruse.

References

[1] Kumar A, R C Panicker, A Kassim 2013 Enhancing VHDL learning through a light-weight integrated environment for development and automated checking Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering pp 570-575 doi: 10.1109/TALE.2013.6654502.

[2] Godofredo R G, A Tchernykh, A Yu Drozdov, S N Garichev, S Nesmachnow, M Torres-Martinez 2019 Visualization of VHDL-based simulations as a pedagogical tool for supporting computer science education Journal of Computational Science Volume 36 doi: 10.1016/j.jocs.2017.04.004.

[3] Jelemenska K, P Cicak, M Gazik 2016 VHDL models e-assessment in Moodle environment International Conference on Emerging eLearning Technologies and Applications (Vysoke Tatry) pp 141-146 doi: 10.1109/ICETA.2016.7802048.

[4] Seraj M, S Autexier, J Janssen 2018 BEESE, a block-based educational programming tool for end users Proceedings of the 10th Nordic Conference on Human-Computer Interaction (New York: USA/ACM) doi: 10.1145/3240167.3240239

[5] Pinto-Llorente A M, S Casillas-Martin, M Cabezas-González 2018 Building, coding and programming 3D models via a visual programming environment (Qual Quant 52, 2455–2468) doi: 10.1007/s11135-017-0509-4

[6] Baldwin L P, J Kuljis 2000 Visualisation techniques for learning and teaching programming Proceedings of the 22nd International Conference on Information Technology Interfaces (Cat. No.00EX411) (Pula: Croatia) pp 83-90 doi: 10.2498/cit.2000.04.03.

[7] Carlisle M C, T A Wilson, J W Humphries, S M Hadfield 2005 RAPTOR: a visual programming environment for teaching algorithmic problem solving. SIGCSE Bull 37 1 pp 176–180 doi: 10.1145/1047124.1047411

[8] Xie C, H Qi, L Ma, J Zhao 2019 DeepVisual: A Visual Programming Tool for Deep Learning Systems 27th International Conference on Program Comprehension (Montreal: Canada/IEEE/ACM) pp 130-134 doi: 10.1109/ICPC.2019.00028.

[9] Al-Tahat K 2019 The Impact of a 3D Visual Programming Tool on Students' Performance and Attitude in Computer Programming: A Case Study in Jordan Journal of Cases on Information Technology (JCIT) 21(1), doi: 10.4018/JCIT.2019010104

[10] Rose S P, J M P Habgood, T Jay 2017 An Exploration of the Role of Visual Programming Tools in the Development of Young Children’s Computational Thinking The Electronic Journal of e-Learning Volume 15 Issue 4 pp 297-309