SAUCIS *Research Article*

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 5, NO. 2, AUGUST 2022
DOI: 10.35377/saucis.05.02.1133435

Transfer of Analogies in Traditional Programming Languages to Teaching VHDL

Halit ÖZTEKİN[1], Ali Gülbağ[2]

[1]Corresponding Author; Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, TURKEY, halitoztekin@subu.edu.tr; +90 533 547 06 18

[2]Department of Computer Engineering, Sakarya University, Sakarya, TURKEY, agulbag@sakarya.edu.tr; +90 533 344 34 99

Received 6 June 2022; Revised 25 October 2019; Accepted 7 July 2022; Published online 31 August 2022

Abstract

One of the languages available to describe a digital system in FPGA is the VHDL language. Since programming in hardware requires a different way of thinking than developing software, the students face some difficulties when trying to design in VHDL language with the previous and long experiences kept in mind in the learning of software imperative programming. These are its concurrency, parallel and sequential model. Due to the insufficient understanding of these topics, it is difficult for students to master the VHDL language. Analogies change the conceptual system of existing knowledge by linking the known to the unknown and by changing and strengthening their relationships. This study contributes to overcoming the problems that students encounter in the coding of the above-mentioned topics in VHDL language by using their experiences in traditional programming languages through analogies. Analogies were used in an undergraduate embedded systems course to explain complex concepts such as those related to signals, concurrent/parallel process; and to encourage comprehensive projects in digital circuit design. Computer science and similar degrees include courses focused mainly on high-level programming languages. This has a strong case for shortening the learning curve in teaching hardware description languages of the skills acquired in programming courses. In feedback from students, the discussion and negotiation of analogies seems to minimize confusion and from using inappropriate expressions in using VHDL language.

Keywords: VHDL, FPGA, Teaching, Adaptive learning, Programming languages, Analogy, Think Hardware

1. Introduction

The traditional approach to knowledge transfer is maintained with ideas from the early 20th century. In the current psychological theories, there are particularly seminal ideas of Thorndike regarding the importance of overlapping characteristics between learning and transfer situations [1, 2]. However, in most psychological research on transfer, especially analogical reasoning theories have been used [3-5].

An analogy is similar in some ways between unlike things, and a comparison is based on this similarity. Analogies connect the known to the unknown and change the conceptual system of existing knowledge by changing and strengthening their relationships [6]. Well-defined metaphors are important for students to establish a connection with what they already know and to offer order to the chaos of unfamiliar concepts [7]. There are not many studies on the use of analogies as an active teaching strategy in engineering education. A study was conducted to determine what kind of analogies students use to explain the basic circuit concepts, and what properties the structural analogies have, which are the most common in students' discussions about circuit concepts [8]. However, there are studies on the use of the concept of analogy in other scientific fields such as medical science [9]. In this study, analogies were used in an undergraduate embedded systems course for electrical and electronics students for two purposes: a) to explain complex concepts such as those related to signals, concurrent/parallel and process; and b) to provide practice for students to increase interest in VHDL and encourage comprehensive projects in digital circuit design using these strategies [10].

VHDL is a description language for digital electronic circuits to accelerate the design process. As stated in its acronym (VHDL- Very High Speed Integrated Circuits Hardware Description Language), it is used to accelerate the design process. Performing this process in high-level programming language such as C++ could be an option. However implementing about the hardware aspects of computer systems can not be an easy task. To this end, there are studies emphasizing the importance of hardware description languages [11]. The fact that VHDL is not a programming language is often forgotten. Therefore, knowing syntax as in a standard programming language does not mean being able to design digital circuits with it. A program in VHDL language is not executed sequentially as in a standard programming language such C++. Due to the behaviour of a real hardware, VHDL allows concurrent statements and operations triggered by events. It would be very difficult to explain this to someone who creates the program in a classical programming language [12-14]. These issues tend to distract them from the understanding of digital components. The number of studies using hardware description languages is increasing in the literature. This situation reveals the importance of learning these languages. There are studies in the literature about teaching these languages. Detailed information about this is given in section 2. In this study, it is aimed to facilitate hardware thinking by showing the equivalents of the expressions in VHDL in a programming language. The purpose of teaching methodology is trying to help the transfer of the concepts that students have difficulty in understanding to the familiar environment. Thus, they learn the importance of HDL-based digital design by better understanding the complexities of HDLs.

2. Related Work

There are various methods developed in the literature to overcome the difficulties of using the HDL languages. The main purpose of the studies is to make digital design easier by reducing the complexity of hardware description languages [15]. In one of these studies, developed the tutorial showing students how to design digital circuit systems first, and then introducing them to the language [16]. In other words, it is the transition to language teaching by placing the idea of "think hardware" in students. In the reference [17], a set of design guidelines for HDL-based design explain how to avoid the common conceptual error of synchronizing HDL code with procedural software and instead think in terms of hardware. Technology is a tool that should be used in the teaching-learning process [18]. VerilogTown that is a video game used to control the world has been developed to better understand Verilog, and for the experience to be less jarring in their learning process [19]. Similarly, a language called Abstract Verilog has been defined in order to reduce the cognitive load for new students [20]. The spiral approach method was incorporated into the digital systems course in undergraduate electrical and computer engineering programs at Ohio Northern University (ONU), as HDLs can be comprehensive and very difficult to understand in such an early course as digital systems [21]. A lightweight and cross-platform open-source environment has been developed that is suitable for use as an introductory tool for students learning HDLs [22]. In some studies in the literature, the problems encountered while teaching HDL language [23] and suggestions to instructors wishing to implement the class are gave [24-26]. Literature [27-29] indicates an approach that would enable students to focus more on the design and less on the individual bits instead of HDL in order to perform bit-level operations.

3. Materials and Methods

In this section, the syntax of the HDL languages will not be discussed, only the equivalents of some of its components in one of the classical programming languages (preferred C++ in this work) will be shown and used VHDL as hardware description language. The general structure of this language is given in Figure 1.

```
architecture name of entity is
-- architecture declarations
-- signals
-- components
-- types
begin
-- concurrent statements
-- conditional statements
process(sensitivity list) begin
-- sequential codes
end process;
end name;
```

Figure 1 The general architecture of VHDL language

3.1 Entity Component

Entity is the part where the input and output ports of a digital system are defined. In other words, it accepts data from external units through these ports and transmits the result to external units via these ports. Since only one entity can be defined in a VHDL file, a declaration of a function (function prototype or function interface) can be given in C ++ programming language. The idea that entity components with different names can be created in the same VHDL file may come to mind here. This problem would be eliminated by considering the entity component as a main function in C++. The example below (Code 1) shows the C++ equivalent of the entity component in VHDL. The number and types of parameters in the function signature in the function prototype and the return type of the function represent the port block of the entity component. If the entity component has more than one output port, then the return type of the function will need to be represented by a struct.

Code 1 The entity declaration in VHDL language

VHDL

```
entity example is
port ( A,B,C : in std_logic;
       D : out std_logic );
end example;
```

C++

```
double example ( double, int, int );
```

3.2 Components

While giving information about the functioning of this component, mentioning the subroutine subject in programming languages will help the student to remain in her/his mind. In particular, calling a sub procedure from another procedure is a good example to illustrate this one (Code 2). Unlike an entity component in a program snippet written in a VHDL language, the components can have more than one. The operation of this component in a high-level programming language is the same as that of the entity component.

Code 2 The component instantiation in VHDL language

VHDL

```
component sub_program
port ( A,B : in BIT;
        Y : out BIT );
end component;
```

C++

```
...............
Y=sub_program(A, B)
...............
```

3.3 Execution Modes

The statements in VHDL can be executed in the following three modes. These modes are namely:

i) Sequential mode

ii) Concurrent mode

iii) Parallel mode

Sequential mode is an environment where commands are executed sequentially from top to bottom as in all programming languages and the user knows this environment very well. The commands in this mode can only be written in the process block in VHDL that are scheduled for execution by events. In order to understand the operation of the process block, there are 2 issues to consider: event and scheduling. The first one is a mechanism called an event (sensitivity list) that activates the execution of sequential commands in the process block. The sequential commands in the process are executed when changes in any signal specified in the sensitivity list. In other words, the process will be executed in an infinite loop. The events such as timer, click etc. used in classical programming languages can be used to explain the functioning of the concept of process. Secondly, the instructions in the process block are executed with the current values of the signals and their updated values occur after the end of *end process* clause. This situation can be associated with a queue structure that stores the current value at the top of the queue and the updated value at the bottom. The pop command is configured to leave at least one element in the queue. In other words, pop command will not perform data retrieval if there is only one element in the queue. Otherwise, there may be no data in the queue. The following example (Code 3) describes these one. In this example, let's assume that three queue structures that work according to the FIFO rule are created for the variables. Their names are q_a, q_b and q_c respectively. If we accept that there are value1, value2 and value3 values in the queues at the beginning, that is, before the block of process runs, after line 3 in the VHDL code snippet, the value(value2) at the top of the b queue is pushes to the queue named q_a. After the piece of code in line 4 runs, the value(value1) from the queue named q_a is pushed to the queue named q_c. At the end of the process block, the elements below the queues are transferred to the top.

Code 3 The equivalent of process block in C++

VHDL

```
1 process(e)
2 begin
3 q_a<=q_b;//(1)
4 q_c<=q_a;//(2)
5 end process;
6
7
8
9
```

C++

```
1 private void button1_click(object sender, EventArgs
2 e){
3    q_a.push(q_b.pop());
4    q_c.push(q_a.pop());
5    return update_queues;}
6 void update_queues(){
7    q_a.pop();
8    q_b.pop();
9     q_c.pop();}
```

The changes in the example above are given step by step below(Figure 2).

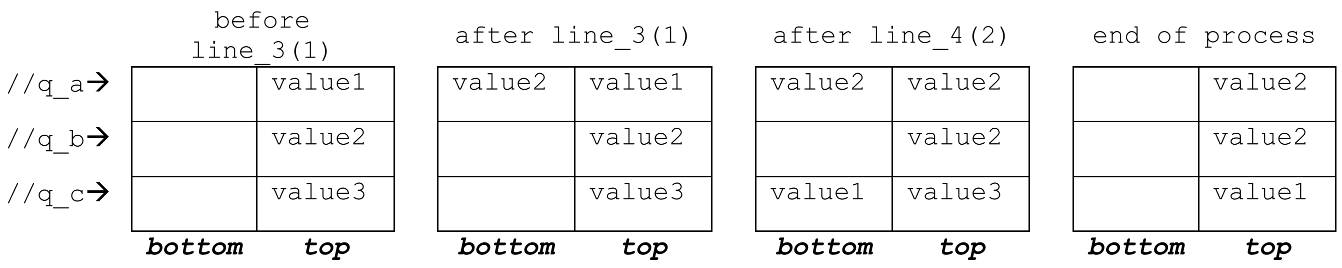

Figure 2 The changes of signals in the process

One of the most important reasons why FPGAs are popular is that their structure is suitable for parallelism. The term that is often confused with the term parallelism is concurrency. Considering that the VHDL language is a hardware description language, it is useful to specify what these two terms correspond to in terms of hardware. If there is cooperation between hardware units, the term to be mentioned is concurrency, if not parallelism. Code 4 explains how two different kinds of execution method were compared.

Code 4 The examples for concurrent and parallel operations in VHDL

Concurrent

```
architecture   example   of
exp_concurrency is
begin
e <= c or d;
c <= a and b;
end  exp_concurrency;
```

Parallel

```
architecture example of exp_parallel is
begin
c <= a and b;
d <= a or b;
end  exp_parallel ;
```

While commands are executed as they are written in classical programming languages, this does not matter in VHDL language. The order of execution is depended only by events on the signals. In addition, the commands are executed repeatedly in every change of the signal. Concurrent or parallel operations in VHDL language are similar to thread structure in traditional programming languages. While there is no need for an extra command in VHDL language to do these operations, software description languages require an extra instruction (Code 5). In other words, each line (block) in VHDL language can be evaluated as a thread. The order of execution is defined only by events occurring on the signals that the assignments are sensitive to. First of all, the order of the command lines with the "<=" operator in the VHDL language is not important. Even changing only one of the variables to the right of the "<=" operator in these lines is sufficient to execute the relevant line. In this example, two separate threads are created for concurrent execution of two lines. However, while these threads are running once, the lines in the VHDL language will run multiple times. To realize this situation in high-level programming language, threads must be written inside an event. If the variable/s to the left of the "<=" operator are not included in the variables to the right of the "<=" operator in other lines, it means that these lines will be executed in parallel.

Code 5 The examples for concurrent and parallel operations in C++

Concurrent

```
void task1(int m){
for(int i=1;i<=m;i++)
cout << “Hello”;}
void task2(int n){
for(int i=1;i<=n;i++)
cout << “World”;}
int main(){
thread t1(task1, 3);
thread t2(task2, 2);}
//Output(machine dependent)
//HelloHelloWorldHelloWorld
```

Parallel

```
#pragma omp parallel
{
cout<<”Hi!”
}
//For dual-core system,twice text “Hi!”
//It may output: HiH!i!,
//printing is //parallel
```

4. Results

Feedback is an important part of learning effectively and improving students' learning experiences. Student feedback has become a widely used method for evaluating and improving teaching effectiveness, as even small changes in the classroom can make a big difference. Students' opinions are asked to evaluate the effectiveness of this teaching approach through a survey. All of the students enrolled in this course have successfully completed Algorithms and Computer Programming courses in which C ++ is taught. Therefore, they are familiar with the programming language concepts mentioned in the study. The participants enrolled in the Embedded System Course were divided into two groups, one experimental and another one control. For participants in the control group, the instructor used PowerPoint slides and delivered in the traditional manner of the lecture style. On the contrary, in the experimental group participants were engaged in the lecture format plus use of analogy in classical programming languages. At the end of the semester, a questionnaire consisting of 5 questions was made to both groups and the answers given are given in the Table 1.

Table 1 Student Feedback on Embedded Systems Course

	Control Group(%)			Experimental Group(%)		
Statements	**Disagree**	**Neutral**	**Agree**	**Disagree**	**Neutral**	**Agree**
1. I realized that the VHDL language is a hardware description language.	12.5	37.5	50	0	10	90
2. Decreased complexity in understanding features such as Concurrent / Parallel.	37.5	25	37.5	10	10	80
3. Increased in understanding how the process block performs schedule.	62.5	0	37.5	0	30	70
4. I realized the operating characteristics of Signal and variables (in computer programming languages)	0	25	75	0	10	90
5. Increased interest in VHDL language and encouraging more comprehensive projects in digital circuit design	25	50	25	0	30	70

The comparison of midterm and final exam averages of both groups is given in the graphic below (Figure 3). The midterm exam averages of the two groups are very close to each other. This situation can be explained as follows: It is due to the use of the analogy approach presented in this article on the subjects mentioned after the midterm exam. This situation is reflected in the final exam averages of the groups.

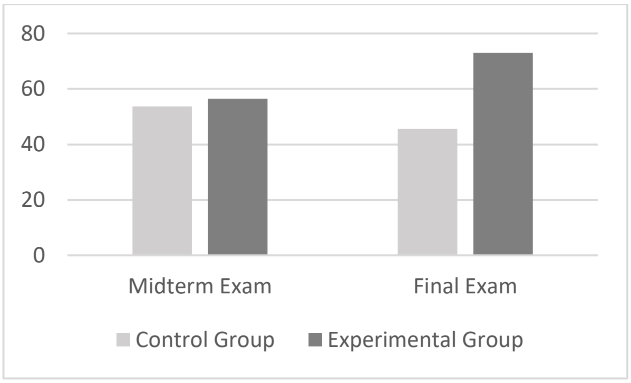

Figure 3 A comparison of student performance on Midterm and Final exams

4. Discussion and Conclusion

In this study, the pits and falls of teaching VHDL language to students having learned the programming languages such as C or C++ are mentioned. There are three main types of problems we encounter in course. The first of these is the confusion created by the concepts of signal and variable. The second is thoughts on how to execute concurrent and parallel processes without an additional instruction. Finally, the details of how the processes in the process block are scheduled.

Transfer of learning is a method of applying the knowledge and skills learned in one situation to a different situation. The transfer is more likely to occur if instructors use transfer strategies that include students' knowledge and skills. Students' knowledge and experience in classical programming languages were used to contribute to the solution of these problems.

In the teaching and learning process of VHDL, the following suggestions can be of great help from the author's experience. At the beginning of the teaching process of each new concept, students should be reminded that the VHDL language is NOT a programming language, and it should be contributed to reduce the effects of the conventional method on minds by providing a clue of what kind of hardware as much as possible. In other words, thinking about hardware will contribute to the teaching process. It should be said that every statement outside of the process block is like calling a sub-function at every signal change, and the calling of operation happens automatically. It should be pointed out that the processes in the process block are executed sequentially as in programming languages as in C++, and only the updating of each signal happens at the end of the process. Therefore, only each signal in the process section has a queue data structure that works according to the FIFO rule. The signals outside process behaves the variables as in the programming languages.

References

[1] E. L. Thorndike, “Mental discipline in high school studies,” *Journal of Educational Psychology*, vol. 15, pp. 83–98, 1924.

[2] E. L. Thorndike and R. S. Woodworth, “The influence of improvement in one mental function upon the efficiency of other functions,” *Psychological Review*, vol. 8, pp. 247–261, 1901.

[3] D. Gentner, “Structure-mapping: A theoretical framework for analogy,” *Cognitive Science*, vol. 7, pp. 155–170, 1983.

[4] J. E. Hummel, and K. J. Holyoak, “Distributed representations of structure: A theory of analogical access and mapping,” *Psychological Review*, vol. 104, pp. 427–466, 1997.

[5] J. E. Hummel, and K. J. Holyoak, “A symbolic-connectionist theory of relational inference and generalization,” *Psychological Review*, vol. 110, pp. 220–264, 2003.

[6] A. Harrison and D. Treagust, “Teaching and Learning with Analogies: friend or foe?”, In: *Metaphor and Analogy in Science Education*, pp. 11-24, Netherlands, Springer, 2006.

[7] U. Teucher, “Metaphor in crisis: The language of suffering,” *Pain and Suffering Interdisciplinary Research Network*, University of British Columbia, 2004.

[8] N. Pitterson, N. Perova-Mello, and R. Streveler, “Engineering students’ use of analogies and metaphors: Implications for educators,” *International Journal of Engineering Education*, vol. 35, pp. 2-14, 2018.

[9] R. Kanthan and S. Mills, “Using Metaphors, Analogies and Similes as Aids in Teaching Pathology to Medical Students,” *The Journal of the International Association of Medical Science Educators*, vol. 16, no. 1, pp. 19-26, 2006.

[10] Ž. Nakutis, and M. Saunoris, “Challenges of Embedded Systems Teaching in Electronic Engineering Studies,” *Elektronika Ir Elektrotechnika*, vol. 102, no. 6, pp. 83-86, 2010.

[11] R. Obeidat, and H. Alzoubi, “Why Are Hardware Description Languages Important for Hardware Design Courses?,” *International Journal of Information and Communication Technology Education (IJICTE)*, vol. 17, no. 2, pp. 1-16, 2021.

[12] R. J. Duckworth, “Embedded system design with FPGA using HDL (lessons learned and pitfalls to be avoided),” *2005 IEEE International Conference on Microelectronic Systems Education (MSE'05)*, pp. 35-36, 2005.

[13] M.D.L.Á. Cifredo-Chacón, Á. Quirós-Olozábal, and J.M. Guerrero-Rodríguez, “Computer architecture and FPGAs: A learning-by-doing methodology for digital-native students,” *Comput Appl Eng Educ*, vol. 23, pp. 464-470, 2015.

[14] C. M. Kellett, “A Project-Based Learning Approach to Programmable Logic Design and Computer Architecture,” *IEEE Transactions on Education*, vol. 55, no. 3, pp. 378-383, 2012.

[15] S. A. Shinde and R. K. Kamat, “FPGA based Improved Hardware Implementation of Booth Wallace Multiplier using Handel C,” *Elektronika Ir Elektrotechnika*, vol.109, no. 3, pp. 71-74, 2011.

[16] F. Vahid and R. Lysecky, *VHDL for Digital Design*, John Wiley & Sons, 2007.

[17] J. A. Nestor, “HDL coding guidelines for student projects,” *2011 IEEE International Conference on Microelectronic Systems Education*, pp. 86-89, 2011.

[18] M. Meral, C. Akuner, and I. Temiz, “Competencies of Teachers’ use of Technology in Learning and Teaching Processes,” *Elektronika Ir Elektrotechnika*, vol. 18, no. 10, pp. 93-97, 2012.

[19] P. Jamieson, “VerilogTown - Improving Students Learning Hardware Description Language Design - Verilog - with a Video Game,” *124th American Society for Engineering Education Annual Conference and Exposition*, vol. 29, 2017.
[20] C. Ebeling and B. French, “Abstract Verilog: A Hardware Description Language for Novice Students,” *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07),* pp. 105-106, 2007.
[21] S. Vemuru *et al.* “A spiral learning approach to hardware description languages,” *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pp. 2759-2762, 2013.
[22] A. Kumar, R. C. Panicker, and A. Kassim, “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 (TALE)*, pp. 570-575, 2013.
[23] V. Bonato, M. M. Fernandes, J. M. P. Cardoso, and Eduardo Marques, “Practical Education Fostered by Research Projects in an Embedded Systems Course,” *International Journal of Reconfigurable Computing*, vol. 2014, pp. 1-12, 2014.
[24] S. Edwards, “Experiences teaching an FPGA-based embedded systems class,” *ACM Sigbed Review*, vol 2, no. 4, pp. 56-62, 2005.
[25] G. Wang, “Lessons And Experiences Of Teaching Vhdl,” *Proceedings of the 2007 American Society for Engineering Education Annual Conference & Exposition*, pp. 12.1015.1-12.1015-10, 2007.
[26] G.Wang, “Bridging the gap between textbook and real applications: A teaching methodology in digital electronics education,” *Comput. Appl. Eng. Educ.*, vol. 19, pp. 268-279, 2011.
[27] W. Balid and M. Abdulwahed, “A novel FPGA educational paradigm using the next generation programming languages case of an embedded FPGA system course,” *2013 IEEE Global Engineering Education Conference (EDUCON)*, pp. 23-31, 2013.
[28] A. Kumar, R. C. Panicker, and A. Kassim, “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 (TALE)*, pp. 570-575, 2013.
[29] A. Kumar, S. Fernando, and R. C. Panicker, “Project-Based Learning in Embedded Systems Education Using an FPGA Platform,” *IEEE Transactions on Education*, vol. 56, no. 4, pp. 407-415, 2013.