Methods and Misconceptions in Teaching VHDL

Shimon Mizrahi
Electronics Department
Jerusalem College of Technology
and
York University
School of Computer Science

Abstract

The teaching of Hardware Description Languages (HDLs), such as VHDL or VERILOG, has become a part of the curriculum in departments of electrical engineering and in programs that train electronics technicians. Additionally, there is a trend to teach these languages in technically-oriented high schools.

Many lecturers talk and write about the difficulties and misconceptions in the understanding of HDL's, but only a few of them suggest ways of improving the quality of learning and understanding[1]. This article describes the results of our researches into the reasons for misconceptions about HDLs and ways to avoid having our students develop these misconceptions.

1. Introduction

1.1 What is VHDL and what is difference between VHDL and other programming languages?

VHDL and VERILOG are **H**ardware **D**escription **L**anguages; they are used to **describe** the **structure and the functionality** of electronic circuits and systems. The syntax of VHDL and VERILOG is very similar to programming languages such as C and PASCAL.

In programming languages used to describe sequential processes, teachers of computer science often concentrate on teaching that the computer processes

[1] Havarvar (2005), (Zwolinski, 2000), see chapter 2.1

instructions one after the other. In HDLs line order is not important, and in order to force a process to be executed sequentially, one must make use of special directives. Consider, for example:

VHDL	C
1) A<=B;	1) A=B;
2) C<=D;	2) C=D;

In sequential programming language, line 1 is executed before line 2. In VHDL the two lines are executed at the **same** time. If one wants to be accurate in one's description of HDLs, there is no "execution" of an HDL "program." Every line causes hardware to be created, and the hardware will then work in parallel.

HDLs look like standard programming languages, but they are hardware description languages. We now present two more examples of VHDL and similar C code, and we see how different the meaning of the code segments is.

When we program the following C function, this function will change the value of O only upon being called by another function.

```
C_Function()
{
if (S==1) O=A;
        else O=B;
}
```

But when we write VHDL code with similar syntax, the result will be totally different. We create hardware! (See Figure 1.)

```
process (S)
begin
if (S='1') then O<=A;
                else O<=B;
        end if;
end process;
```

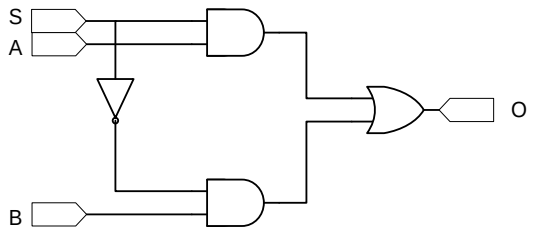

Figure 1: MUX

We can see that with the hardware that was created by the VHDL code O will change value continuously.

For our second example, consider the following code segments.

```
C_Function2(char A)
{
for ( I=0 ;I<3;I++)
        {
        if (A == I)
                O(I) = 1;
          else          O(I) = 0;
        }
}
```

In this C function, only after calling the function will the values in the array O be changed.

The following VHDL snippet is similar, but the meaning is totally different, Figure 2.

```
process (A)
  begin
    for  I  in  0 to 3  loop
      if (A = I) then
          O(I) <= '1';
          else  O(I) <= '0';
      end if;
    end loop;
 end process;
```

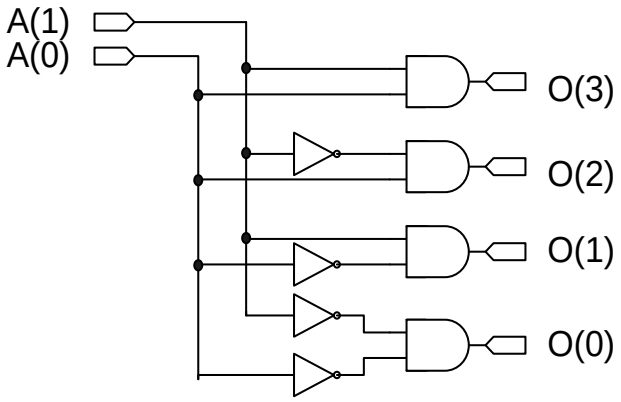

Figure 2: Decoder

The hardware created from this code produces multiple outputs with different conditions. Each output has its own condition, and it operates in parallel with all other outputs.

Here we can start to understand the reasons for this research. After years of learning sequential programming and after much effort have been expended to teach the students to think sequentially, they start to learn a language that looks like a

programming language. The student tries to use his old knowledge (knowledge transfer) and (sometimes) it doesn't works.

1.2 Research objective

Our research objective was to construct an effective model of teaching VHDL, a model that utilizes the previous courses in the Department of Electronics Engineering. To achieve the research objective it was necessary to achieve a number of intermediate goals:

A. To discover the points of difficulty in the learning of the VHDL language.
B. To focus on the prerequisites for learning a VHDL language.
C. To identify the primary emphases that should be addressed when teaching the VHDL language.
D. To search for ways and to build models for teaching the language according to the needs of the target population.

When we come to research misconceptions and the ways to avoid them, we need to examine the learning environment and the learning process. The learning environment is mainly the lecturer and the tools he uses. The learning process is the way the student with his knowledge and background gets and understand the learning environment.

In the framework of the present research, we examined four components of the research questions.

A. The collection and analysis of information in regards to teaching methods and approaches in universities around the world, at the Technion in Israel, at JCT in Israel, and in junior colleges in Israel.
B. The review of a considerable number of textbooks used to teach VHDL. We performed analyses of the material presented in the books from different aspects such as the order of instruction, contents, and teaching approaches and whether it starts from a hardware perspective or whether the HDL language is taught as a programming language.

C. The analysis of the achievements of students in the VHDL course in JCT. (There were 18 students.)
D. The analysis of achievements and the difficulties encountered by the students who performed final projects in the VHDL are at JCT. (There were 7 such projects, each project tow students.)

2. Research and Conclusions

2.1 The Points of Difficulty in the Study of the VHDL Language

We made use of several sources of information that allowed us to study the difficulties associated with the learning of the VHDL language.

2.1.1 Articles and learning materials.

The first sources were the articles and learning materials used by the students in VHDL courses from several universities around the world. Most of the materials begin with the presentation of the difficulty of learning of HDLs. Here are some of the phrases used to describe the difficulties involved in learning VHDL:

Dr. Adnan Shaout of the University of Michigan – Dearborn**: For all you C people -- forget everything you know. (7)**

A Short Description of the VHDL Language (Hebrew), Technion IL: **Think in parallel about the possible implementation in hardware**.(8)

Refresh the knowledge in Digital Systems and Design before engaging in the study of the VHDL language (Zwolinski, 2000).

In the preface to his book, Havarvar (2005) cites the difficulty that the learners face in the coping with the VHDL language but does not focus it on any particular difficulty. We found that many of the instructional books present the problems encountered when learning VHDL but do not present the focus of the problem and the difficulty in the learning of the language.

2.1.2 Final test in VHDL course.

The second sources of information were the results of a test held for the 18 students who took the VHDL course.

The Test Questionnaire

The course under consideration is “a laboratory in advanced digital design.” The duration of the examination was 2 hours, and the test consisted of five questions, all of which were compulsory. Each question was worth 20 points.

Question 1 engages in the design of a system in VHDL. The system is a calendar - counter of days and weeks and month. The student is required to write in VHDL engaged in synchronous system design. The questions is textually. Use is made of tables describing the states of the calendar, writing that makes it easier for the student in writing the entity and defining the Ports.

The question examines the students’ ability to perform a basic design, in all its components, the definition of the entity, Port, variables, in a correct manner, the writing of the architecture, and its link to the entity. The question requires of the student knowledge of the writing structure and implementation ability. To a certain extent, the student is required to have a hardware view of the problem. There is no need to understand the nuances of VHDL.

Question 2 and **3** were based on the design of traffic light behavior according the schedule defined in the question. The students were required to implement the response using a state machine of the Moor type.

Question 2 addresses hierarchical design. The question asks the student to create a block diagram that describes the hierarchy of the project components. After the drawing of the block diagram the student is required to convert it to VHDL code. The question examines the student’s ability to design according to the top down approach, to display hardware thinking, to convert to VHDL code, and to correctly use components. The understanding required of the student is a hardware view of the problem and the ability to implement what is required in the question. It is necessary to understand the use of repetitive blocks.

Question 3 Engages in the design of a state machine of the Moor type. The performance stages are the drawing states chart, converting to VHDL code, defending variables, the declaration of the machine states, and writing architecture of state machines. These focus on the student's knowledge in the design of state machines.

The didactic ability required of the student in a **slight** degree is a hardware view of the problem and the ability to implement what is required in the question. Understanding state machines is necessary.

Question 4 addresses the topic of Xilinx Constraints. The question examines the student's ability to use the data sheets of an FPGA and through them to connect and adjust code to the existing hardware. The didactic ability required of the student is mostly technical--reading the technical text and knowing how to program. The level of the question is knowledge and some understanding.

Question 5 addresses the design of ROM model and the writing of a test bench for the model. In addition, the question examines the student's ability to choose the options in front of the student in the design process. The understanding ability required of the student is the use of advanced tools learned in the class.

Table Number 1: Average and standard deviation

5	4	3	2	1	
13.91667	15.25	15.19444	13.33333	14.55556	AVG
6.511208	5.73791	4.22341	7.133645	5.387457	STDEV

The researcher conducted a correlation analysis of the achievements among the questions using Pearson correlations.

Table Number 2: Pearson Correlations of the Achievements

	1	2	3	4
2	0.41439			
3	0.539737	0.752953		

4	0.133889	0.322349	0.532721	
5	0.334696	0.448524	0.637469	0.815808

From the analysis of the test results and the correlation between the different abilities tested. The question that tested the ability of hardware view had the lowest mean and the highest standard deviation. [2]). This indicates significantly that the students have difficulty with the hardware view and in the case of the test, primarily the students from the weaker group.

2.1.3 Follow up after the project performance.

The third source was the analysis of interviews conducted with students performing final projects, 7 projects. From the summary of the interviews with the students and follow up after the project performance, it was determined that the students have a fundamental problem taking a hardware view, unlike the researcher's first assumption that the problem in the understanding of the language derives from the understanding of the concepts of synchronous and asynchronous systems. In addition, the problem is more prominent among the low-performing students, when the lack of linkage between the written code and the understanding of the hardware sometimes induces students to view the VHDL code as software.

The follow up after the students shows that there is difficulty differentiating between synchronous writing and asynchronous writing. In addition, there is difficulty writing state machines, but it appears that it derives from a more entrenched problem of internalization that VHDL is a hardware description language that looks like a programming language.

2.2 What Are the Pre-Conditions for the Learning of the VHDL Language?

There are some attempts to teach VHDL as a programming language. However, it appears that the people who go in this direction encounter many difficulties, as noted

[2] Question 5 was almost as bad as Question 2 because it was on advanced tool that few of the students know how to use them well

by many including Shaout[3] in his introduction to the course. In addition, our research among students shows that the primary difficulty is the development of the ability to view VHDL code as code that creates (asynchronous) hardware. Hence, the Digital Systems Design course is an essential prerequisite of the VHDL course. In addition, it is important that in the prerequisite course the students learn state machines well, as they are part of the study material in the VHDL course.

It should be noted that traditional programming courses are a **disruptive factor** in the understanding and perception of the VHDL language, which by nature is a parallel language, since the hardware works in parallel, in contrast to other programming languages, which are generally sequential. In addition, the code in VHDL creates hardware or relationships in the hardware that perform the process. In contrast, in programming languages the hardware does not experience any change in the programming process. These differences in the understanding of the sequential languages on the one hand and HDLs on the other hand make it difficult for the students to fully understand VHDL, and this fact must be taken into account. Therefore, the emphases that appear in the following passage are essential to the teaching of a hardware description language.

2.3 Primary Emphases in the Teaching of the VHDL Language

Our research shows that there indeed is a problem of the erroneous perception of what is learned in the VHDL lessons. This mistaken perception is evinced primarily in situations in which the student is required to realize systems that he or she has not encountered in the course framework and the system needs to combine sequential and parallel processes. Our research shows that the problem in the erroneous perception derives from the approach to VHDL as a software and not as a hardware description language. Therefore, in teaching the language we need to perform actions that turn the learner's attention to the fact that he is addressing hardware.

The interviews with the students who performed projects that utilized HDLs showed that running a test bench and presenting a waveform were not sufficient to

[3] www-personal.engin.umd.umich.edu/~**shaout**/Introduction-to-**VHDL**.ppt

clarify what happens inside the component. The lower the student's level was, the more important it was to use a schematic view (RTL) and block diagrams.

Our research shows that to achieve understanding it is necessary and important to present to the students the schematic diagram results of the synthesis and how it is related to the code written. It is recommended not to illustrate this by using large code segments so that the schema created will be understandable. In addition, it is possible to perform the synthesis and place and route for simple target components as small CPLD and not FPGA, in which the results of the synthesis will be clearer and more understood.

The study of the topic only through computer and simulations increases the chance for erroneous perception and, therefore, it is essential that the exercises on true hardware, including downloading the code to a real component, be performed a number of times during the course to strengthen the hardware view that must be adopted when using HDLs.

Our research is reinforced by the results found by other authors. Hussein, Gruenbacher, and Noureddin (1999)**:**
Creating an effective learning environment for the VHDL class was accomplished using several techniques. These techniques included the use of classroom examples, homework problems, internet resources, and classes held in a computing laboratory. Classroom discussions were utilized extensively as they were found to be very productive. This was especially true with this class since it had less than 10 students.

Beyond the aforementioned statements, the recommendations made by Ben Ari (2000) for curriculum design in computer science are also valid for VHDL. However, the target components are not CPU processors but CPLDs or FPGAs. Here are Ben Ari recommendations changed by the author to fit course teaching HDLs.

1. It is necessary to teach a model of the target hardware that explicitly addresses the structure of the hardware and the flow of signals. The question is what the required level of detail is and what component to use CPLD or FPGA. There are

several perspectives on this issue. This type of model will support the understanding of how a VHDL code is place & routed on a component.

2. A bare-bones model of the CPLD / FPGA should be taught before the abstraction.

3. One must refrain from allowing the use of actual components such as CPLDs or FPGAs to take the place of building an effective *model* of the components to be used as part of the learning process.

4. The fact that there are different approaches for design and programming must be acknowledged, from top down programming to bottom up and bricolage. However, the use of bricolage without an appropriate model and a good understanding of the concepts and practice can disrupt beginning students.

5. Laboratory assignments allow the student to be active and to construct knowledge on the basis of his experience. Therefore, it is preferable to combine assignments with extremely well defined goals and the type of assignments that are more open-ended and encourage cognitive processes such as analysis, examination of possibilities, and decision making.

6. Attention should be given to the method of evaluation. In contrast to the tests in which the results are evaluated, evaluation in computer science should address not only the product but also, primarily, the process of problem solving.

3. Summary and Conclusions

We have studied the primary factors leading to erroneous perceptions among students who learn the VHDL language. The development of a hardware point-of-view will significantly reduce the deficiencies evinced in the study of the HDLs. In this research, we present additional sources that reinforce the research results.

In the discussion of the research results, we present a number of practical conclusions as to how HDLs should be taught. The primary recommendations revolve around causing the students to internalize the fact that their code “creates” hardware. This focus is supposed to be maintained by the correct use of the development tools

and by downloading the code to real components at least a number of times during the course.

Bibliography

1. Ben-Ari, M., (2001). “Constructivism in Computer Science Education”, *Journal of Computers in Mathematics and Science Teaching*, 20(1), pp. 45-73.

2. Havarvar, E. (2004). *VHDL Hardware Description Language*, Shoresh. (Hebrew)

3. Hussein, A. I., Gruenbacher, D. M., and Ibrahim, N. M. (1999). “Design and Verification Techniques Used in a Graduate Level VHDL Course”, *Frontiers in Education Conference*, 1999. FIE apos; 99. 29th Annual, Volume 2, Issue, pp. 13A4/28 - 13A4/31.

4. Pea, R. D. (1986). “Language-Independent Conceptual “Bugs” in Novice Programming”, *Journal of Educational Computing Research*, Vol. 2:1, pp. 25-36.

5. Zwolinski, M. (2000) "Digital System Design and VHDL". Prentice Hall; 1 edition (October 18, 2000)

6. Richard E. Haskell and Darrin M. Hanna, *Learning By Example Using VHDL – Advanced Digital Design*, LBE Books, Rochester, MI, 2007.

7. http://www-personal.engin.umd.umich.edu/~**shaout**/Introduction-to-**VHDL**.ppt

8. http://www.ee.technion.ac.il/vlsi/Projects/Manuals/vhdlexpl_sci_04.pdf

Web sites on project oriented learning

9. http://www.samford.edu/ctls/problem_based_learning.html