Teaching Design-Oriented VHDL

Volnei A. Pedroni
Federal Center of Technological Education of Paraná, CEFET/PR – Curitiba, PR – Brazil
pedroni@CEFETPR.br, www.pessoal.CEFETPR.br/pedroni

Abstract

The teaching of state-of-the-art design techniques (an HDL, for example) requires also the learning of a variety of CAD tools. However, though learning such tools is indispensable, it should not come at a sacrifice of fundamental principles. To attain a balance between the practices and fundamental design concepts, we have adopted a “design-oriented” approach in the teaching of VHDL at CEFET-PR. In it, the focus is partially shifted from the language itself to the designs: basic design principles are reviewed and commented, the examples are rich in details, the code is synthesized into physical devices, and simulations are always performed and examined.

1. Introduction

In his excellent article [1] presented at MSE ’2001, B. S. Carlson argues that the new hire should have a strong knowledge of current CAD tools. However, he argues, this should not compromise the learning of fundamental principles. The article also states that teaching an HDL (Hardware Description Language; e.g., VHDL or Verilog) is necessary, but in advanced courses, for the introductory ones should focus mainly on fundamental concepts.

In this article, we describe the teaching of Design-Oriented VHDL at CEFET-PR. The “design-oriented” phrase serves precisely to express our preoccupation with such balance between the teaching of VHDL (the language itself plus the related EDA tools) and the basic concepts involved in the designs. Indeed, our approach with respect to VHDL has basically the same motivation as the design-oriented approach (e.g., [2]) that Prof. R. D. Middlebrook, of CALTECH, has so effectively adopted in his circuit design courses (despite the latter being not VHDL related).

2. Design-Oriented VHDL

VHDL (VHSIC Hardware Description Language) is a powerful circuit synthesis (and simulation) tool. Consequently, the teaching of VHDL (or Verilog) is now indispensable in any EE program. However, besides the large amount of time needed to learn the associated tools, VHDL books (e.g., [3], [4]) concentrate almost exclusively on the language rather than on design using the language. Here, we describe how VHDL is taught at CEFET-PR, and the actions taken to overcome such limitations.

When to teach VHDL: At CEFET-PR, VHDL is taught as part of the Programmable Devices (PLDs and FPGAs) undergraduate course. Such devices provide an invaluable means for real-world practices. At this point, the student is already familiar with basic digital electronics, computer architecture, and microprocessor-based design.

Tools needed: In order to learn VHDL and actually convert it into physical designs, at least the following tools are needed: a compiler, a place-and-route software, a simulator, plus a PLD/FPGA development kit (if the code is to be synthesized into a physical device). At CEFET-PR, we have used Leonardo Spectrum (from Exemplar Logic) for compilation, along with either Altera’s Quartus II place-and-route/simulation package plus their development kits (for their CPLD/FPGA devices) or Xilinx’s ISE design suite along with their development kits (for their CPLD/FPGA devices). A popular tool for simulation is ModelSim (from Mentor Graphics).

How to teach VHDL: This is the key issue. Due to the complexity of the language and due to the several tools that one must learn, it is indeed risky that the fundamental aspects of the designs end up overlooked. To avoid this problem, two actions are taken. First, simplified tutorials of each tool are given in order to speed up the learning process and avoid learning features that are not immediately needed. Second (and more important), the lecture notes are well documented and try to always connect the language constructs with the designs they are intended for. Examples are accompanied by reviews of fundamental concepts, circuit diagrams, and simulation results (which is never the case in VHDL books; e.g., [3], [4]). A summary of the main characteristics of the course is presented in Table I.

Table I: Course characteristics.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>VHDL books</th>
<th>Design-oriented VHDL course</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Focus</td>
<td>on the language</td>
<td>on design using the language</td>
</tr>
<tr>
<td>2) Review of fundamental design principles</td>
<td>never</td>
<td>often</td>
</tr>
<tr>
<td>3) Circuits w/ diagrams</td>
<td>rarely</td>
<td>always</td>
</tr>
<tr>
<td>4) Physical synthesis</td>
<td>---</td>
<td>often</td>
</tr>
<tr>
<td>5) Simulation results</td>
<td>never or rarely</td>
<td>always</td>
</tr>
</tbody>
</table>
To exemplify the characteristics above, two figures are presented, which were extracted from the lecture notes. Fig. 1 shows the introduction of one of the chapters of the course (Chapter 5, which deals with concurrent code), and serves to give an idea regarding how basic principles can be inserted into the course and reviewed. Fig. 2, on the other hand, contains a section of Chapter 6 (sequential code), showing the description of one VHDL statement (LOOP) and a related example. As can be seen, the example is accompanied by diagrams, the corresponding code (two solutions), plus simulation results. Additional comments were omitted in Fig. 2 due to limited space.

**Project examples:** Process/time controller; Digital-to-PWM converter; Memory banks; Digital filters; Signal generator; PCM time slot selector.

---

**Chapter 5: CONCURRENT CODE**

### 5.1 INTRODUCTION

**Combinational versus Sequential Logic**

By definition, combinational logic is that in which the output of the circuit depends solely on the current inputs (Fig. 5.1(a)). It is then clear that, in principle, the system requires no memory and can be implemented using conventional logic gates.

In contrast, sequential logic is defined as that in which the output does depend on previous inputs (Fig. 5.1(b)). Here, storage elements are required, which are connected to the combinational logic block through a feedback loop, such that now the stored states (created by previous inputs) will also affect the output of the circuit.

---

**Fig. 5.1: Combinational versus sequential logic.**

A common mistake is to think that any circuit that possesses storage elements (flip-flops, for examples) are sequential. Let us consider the case of a RAM (random access memory) as an example. A RAM can be modeled as in Fig. 5.2. Notice that the storage elements appear in a forward path rather than in a feedback loop. The memory-read operation depends only on the address vector presented to the RAM input, with the retrieved value having nothing to do with previous memory accesses.

---

**Fig. 5.2: RAM model.**

**Concurrent versus Sequential Code**

VHDL code is inherently concurrent (parallel). Only statements placed inside a PROCESS, FUNCTION, or PROCEDURE are sequential. Still, though within these blocks the execution is sequential, the block, as a whole, is concurrent with any other (external) statements. Concurrent code is also called dataflow code.

---

### 6.4 The LOOP statement

**LOOP** is useful when the same instance has to be repeated several times. There are several ways of using **LOOP** with FOR, WITH, NEXT, with EXIT, or as an infinite loop. **All of these forms will be studied below.**

(a) **FOR / LOOP:**

```vhdl
LABEL: FOR identifier IN range LOOP
(sequential statements)
END LOOP [label];
```

Example: `FOR i IN RANGE (7 DOWNTO 2) LOOP
x(i) <= enable AND w(i-2);
END LOOP;` ("")

Example: **8-Bit Carry Ripple Adder**

An 8-bit unsigned carry ripple adder is shown below. Two solutions are presented: one generic (that is, for any number of bits) and using vectors, and another specific for 8-bit numbers and using integers.

---

**Fig. 2. Basic structure of the examples (from lec. notes).**

---

### 3. Conclusion

The teaching of new design techniques includes extensive use of EDA tools. However, one must not lose basic design principles from view. To maintain an adequate balance in

---

### References


---


0-7695-1973-3/03 $17.00 © 2003 IEEE