AC 2007-95: LESSONS AND EXPERIENCES OF TEACHING VHDL

Guoping Wang, Indiana University-Purdue University-Fort Wayne (Eng)

GUOPING WANG is currently Assistant Professor in the Department of Engineering, Indiana University Purdue University Fort Wayne. He teaches courses in digital system design, VLSI Design Lab, and computer architecture.

Lessons and Experiences of Teaching VHDL

Guoping Wang

Department of Engineering
Indiana University Purdue University Fort Wayne

Abstract

VHDL has become an industrial standard language in digital system design. This paper discusses the author's experience of teaching VHDL to undergraduate engineering students at IPFW. Logic synthesis is focused in the educational activities instead of the complex features of VHDL. Projects which involved synthesis, simulation, implementation and verification using FPGA board were assigned. The pits and falls of teaching and learning of VHDL were discussed. The author's teaching methodology of VHDL is presented, which is followed by some of the problems that students faced when they were trying to design digital systems using VHDL.

1. Introduction

The VHSIC (Very High Speed Integrated Circuits) Hardware Description Language (VHDL) is a very powerful hardware description language for digital system design. It has become indispensable in electrical and computer engineering programs.

This paper summarizes the main issues that cause learning problems for the students when they are learning to use VHDL to design digital systems. The author's teaching methodology of VHDL is described in this paper. Instead of introducing the complex features and various statements of VHDL, the synthesizable VHDL models from simple to complex systems were introduced to give the students a good understanding of VHDL.

Most VHDL books use models developed for simulation only and they are armed at practicing engineers. They frequently use language features not supported in synthesized circuit and they are not easy for beginners to read. They seem to confuse students more than help them and end up mixing constructs that are only suitable for synthesis with other VHDL features that should only be used for simulation. Having taught VHDL for several years and used VHDL on several research projects, the author adopted a teaching methodology which is easy for the students to follow. The purpose is trying to help students design synthesizable digital systems instead of some fancy models only for simulation.

2. VHDL Teaching Methodology

When introducing VHDL to the students, it is very important to point out that VHDL is NOT a programming language, it is used for describing the required digital systems. During the teaching activities, the author always drew the relationships between VHDL codes and the

corresponding synthesized circuits. A clear distinction was also made between the synthesized statements and simulation-only VHDL statements. While many VHDL textbooks start out with the description of many VHDL features including adding in time related information, etc, the author adopted another approach, which only introduced the minimum subset of VHDL suitable for synthesis, and then accompanied with synthesizable VHDL combinational and sequential models such as decoder, encoder, multiplexer, tri-state, comparator, D flip-flop, counters, shift register, and finite state machine. It should be pointed out that the purpose of teaching VHDL is for the students to design a digital system on FPGA/CPLD or ASIC, instead of designing some fantastic systems only for simulation purpose.

After VHDL for synthesis was covered, then the additional keywords, constructs and VHDL statements were briefly introduced, thus students could refer to them in case they may need in their future work. The frustrations and confusions of students when they are learning VHDL will be alleviated. This teaching methodology is further explained in the following.

In the beginning of the VHDL course, two lecture sessions were spent to review combinational and sequential circuits. It is very important and essential for the students to understand the basic principles of digital circuits before VHDL is introduced because even in VHDL design, the schematic design is still the blueprint for digital systems. Thus the review sections will help students refresh what they have learned in the past and anticipate what will happen in the VHDL world.

After that, the first VHDL example was introduced to the students to let them have a taste and this example in Fig. 1 is just a simple AND gate VHDL description:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity and2 is
port (A,B: in std_logic;
C: out std_logic);
end entity and2;
architecture ex1 of and2 is
begin
C<= A and B;
end architecture ex1;
```

Fig. 1. AND gate VHDL Module

To teaching the different VHDL styles, the dataflow and structural VHDL descriptions in Fig. 2 and Fig. 3 to realize a simple logic function F = A'B+AC (where ' stands for logic complement) were presented thereafter.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity comb_function is
  port (A,B,C : in std_logic; Z: out std_logic);
end entity comb_function;
architecture expression of comb_function is
begin
  Z<= (not A and B) or (A and C);
end architecture expression;
```

Fig. 2. Dataflow VHDL Style Example

```
architecture netlist of comb_function is
```

```
  component and2 is
    port (X, Y : in std_logic; Z: out std_logic);
  end component and2;
  component or2 is
    port (X, Y : in std_logic; Z: out std_logic);
  end component or2;
  component not1 is
    port (X : in std_logic; Z: out std_logic);
  end component not1;
signal P,Q,R : std_logic;
begin
  G1: not1 port map (A,P);
  G2: and2 port map (P,B,Q);
  G3: and2 port map (A,C,R);
  G4: or2 port map (Q,R,Z);
end architecture netlist;
```

Fig. 3. Structural VHDL Style Example

While presenting the students the VHDL structural style, the instructor compared the VHDL design to a circuit on a breadboard and signals were compared to the physical wires on the breadboard wire connections. In the end, a simple testbench VHDL code was introduced thus students could begin their projects from VHDL captures, to synthesize, to simulation right after the first VHDL lecture.

After the introduction of the first VHDL example, the author followed the materials of a regular introductory digital system course instead of the introduction of complex VHDL features. VHDL modules of basic combinational circuits such as decoder, encoder, multiplexer, comparator, adder etc were introduced, in the meantime, the synthesized circuits were presented with different styles VHDL code. For example, Figs. 4, 5 and 6 show three different VHDL styles of a simple 4 to 1 multiplexer.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity mux is
port (A,B, C, D: in std_logic;
        S: in std_logic_vector(1 downto 0);
        Y: out std_logic);
end entity mux;
architecture mux1 of mux is
begin
  with S select
     Y <= A when "00",
               B when "01",
               C when "10",
               D when others;
end architecture mux1;
```

Fig. 4. VHDL Code 1 of 4-to-1 MUX

```
…..
architecture mux2 of mux is
begin
  Y <= A when S="00" else
          B when S="01" else
          C when S="10" else
          D;
```

```
end architecture mux2;
```

Fig. 5. VHDL Code 2 of 4-to-1 MUX

```
.......
architecture three_state of mux is
begin
  Y <= A when S="00" else 'Z';
  Y <= B when S="01" else 'Z';
  Y <= C when S="10" else 'Z';
  Y <= D when S="11" else 'Z';
end architecture three_state;
```

Fig. 6. VHDL Code 3 of 4-to-1 MUX

The synthesized multiplexer circuits from these VHDL codes were also described. VHDL style code 1 and 2 are synthesized into the circuit as Fig. 7, while VHDL code 3 is synthesized into a different one with tri-state bus as Fig. 8.

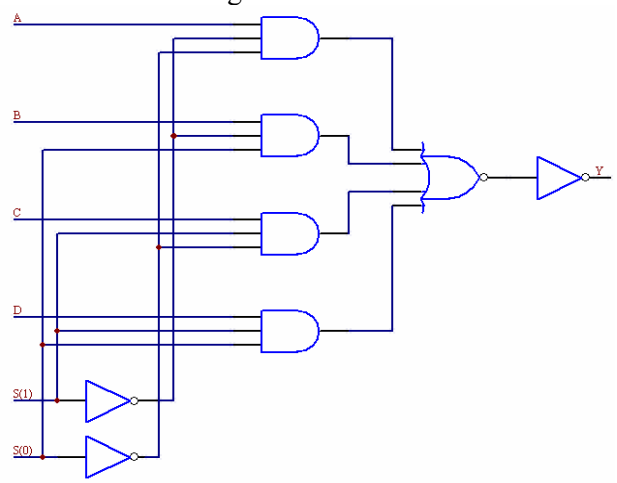

Fig. 7. Synthesized MUX Circuit from Code 1 and 2

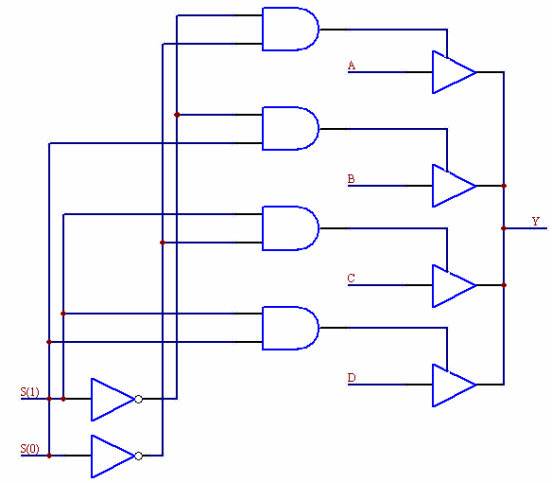

Fig. 8. Synthesized MUX Circuits from Code 3

The similar approach was followed in the sequential circuit design procedures. Here the instructor introduced the VHDL codes first, and then described the synthesized circuits for basic sequential circuits such as D latch, D latch with output enable, D flip-flop with Q output, D FF with Q, QN output, DFF with synchronous RESET/SET, with asynchronous RESET/SET, D FF with Clock enable, with output enable, etc. In these lecture sessions, the author sometimes made a little modification of the VHDL codes and then asked the students to tell what kind of the circuit could be synthesized. For example, Fig. 9 describes the VHDL module of a simple D flip-flop with Q, Qbar output:

```
entity DFF_Q_QBAR_2 is
    Port ( D,CLK : in std_logic;
           Q,QBAR : out std_logic);
end DFF_Q_QBAR_2;
architecture Behavioral of DFF_Q_QBAR_2 is
begin
  process(D,CLK)
          variable QQ:std_logic;
       begin
            if (CLK'event and CLK='1') then
             QQ:=D;
            end if;
             Q<=QQ;
             QBAR<= not QQ;
   end process;
end behavioral;
```

Fig. 9. VHDL Module of a Simple D Flip-Flop

The instructor changed this code into to Fig. 10 VHDL module and asked the students to figure out the synthesized circuit. Through this even simple design example, students could get a very good understanding of the VHDL statement of clk'event and clk='1' statements. They would also learn that the effects of the synthesized circuit for different locations of a simple VHDL statement. It also helped them to get a clear picture of the concepts of register outputs and combinational outputs.

```
architecture Behavioral of DFF_Q_QBAR_2 is
begin
  process(D,CLK)
          variable QQ:std_logic;
       begin
            if (CLK'event and CLK='1') then
                       QQ:=D;
                       Q<=QQ;
                       QBAR<= not QQ;
            end if;
   end process;
end behavioral;
```

Fig. 10. Another VHDL Module of a Simple D Flip-Flop

Another difficult concept in VHDL is the differences between signals and variables. The instructor tried to explain that using design examples from both of simulation steps and

synthesized circuits. For example, Fig. 11 shows the parity generator VHDL module using signals and variables respectively.

```
library ieee;
use ieee.std_logic_1164.all;
entity parity is
  port (A : in std_logic_vector(7 downto 0);
          Y : out std_logic);
end entity parity;
architecture XOR1 of PARITY is
  signal X:std_logic_vector(5 downto 0);
begin
  process (A,X) is
   begin
    X(0)<=A(0) xor A(1);
    X(1)<=X(0) xor A(2);
    X(2)<=X(1) xor A(3);
    X(3)<=X(2) xor A(4);
    X(4)<=X(3) xor A(5);
    X(5)<=X(4) xor A(6);
    Y     <=X(5) xor A(7);
  end process;
end architecture XOR1;
  ……

architecture XOR2 of PARITY is
      begin
         process(A)
            variable XX:STD_LOGIC;
         begin
          XX:=A(0) xor A(1);
          XX:=XX xor A(2);
          XX:=XX xor A(3);
          XX:=XX xor A(4);
          XX:=XX xor A(5);
          XX:=XX xor A(6);
          Y     <=XX xor A(7);
        end process;
end architecture XOR2;
```

Fig. 11. Parity Generator VHDL Modules using Signals and Variables

For the example in Fig. 11, the instructor asked the students to list the simulation steps, values of signals and variables at different simulation delta times, the synthesized circuits. Another example in Fig. 12 and Fig. 13 was also used to help to explain the concepts of signals and variables. Thus through these examples, students could understand the concepts of VHDL signals and variables very well.

```
entity xor_sig is
    port (A,B,C: in STD_LOGIC;
              X,Y: out STD_LOGIC);
end xor_sig;

architecture SIG_ARCH of xor_sig is
```

```
    signal D: STD_LOGIC;
begin
  SIG: process(A,B,C)
   begin
    D<=A;       -- ignore !!
    X<= C xor D;
    D<=  B;     -- overrides!!
    Y<= C xor D;
end process;
End SIG_ARCH;
```

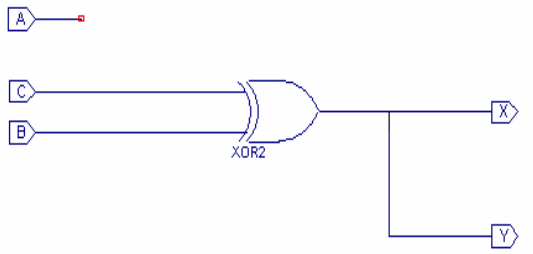

Fig. 12. A Simple VHDL Module and Synthesized Circuit Using Signals

```
entity xor_var is
    port (A,B,C: in STD_LOGIC;
              X,Y: out STD_LOGIC);
end xor_var;

architecture VAR_ARCH of xor_var is
begin
 VAR: process(A,B,C)
      variable: D: STD_LOGIC;
    begin
       D:=A;
       X<= C xor D;
       D:=  B;
      Y<= C xor D;
end process;
End VAR_ARCH;
```

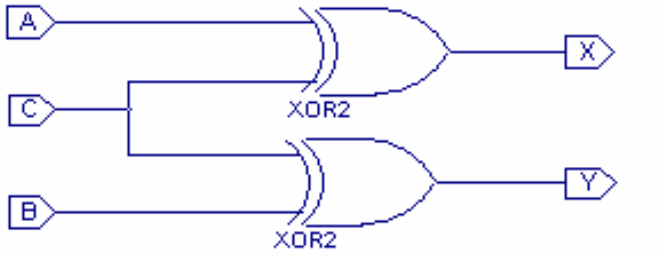

Fig. 13. A Simple VHDL Module and Synthesized Circuit Using Variables

FPGA boards were used extensively in the early stages of VHDL teaching, thus students got exposed to the real hardware instead of only simulations on the computers.

3. Main Issues Students' Encounter in Learning VHDL

One popular question students would like to ask: My VHDL module provides perfect output from testbench simulation, but why it doesn't work on the FPGA board after downloading the bitstrem.

Many students, having learned other programming languages such as C or Matlab, can usually modify a program until it performs the required function. With VHDL this may not be a good option. Students should be kept in mind that VHDL is NOT a programming language, it is used to describe the behavior and property of the digital hardware, thus while writing VHDL module, they should always have a clue of what kind of hardware can be generated or what type of hardware they are designing. Being able to put this statement into practice would require substantial practices and experiences.

There could be many reasons if students' design won't work on FPGA board. For example, if a syntax error is found in C or Matlab program, the compiler usually points at where the error is. This is not the case of VHDL design, because one signal output could be inferred from several VHDL statements concurrently. Thus when a problem is identified, the EDA tool can only provide a general indication of what or where the problem could be. Students may spend many hours trying to find a problem with a statement while the error could be somewhere else. During the teaching and learning process of VHDL, the following suggestions could be very helpful from the author's experience.

- Always ask the students to study the examples. Students tend to write or design their own VHDL modules without spending too much time on studying. A very bad VHDL design code will come from their hands. The design examples are well-written and most are standard VHDL module codes.
- Ask the students to check the synthesized RTL schematic circuit before running the simulation. This could help them to better understanding the relationships between VHDL code and the synthesized circuit and no silly mistakes were made.
- In order to verify their VHDL module, besides the behavioral simulation, a timing simulation should also be made. Sometimes the behavioral simulation may give you the correct results while a correct hardware cannot be synthesized. The timing simulation would pretty much match the outputs of the hardware.
- It should also be pointed out to the students that VHDL does not provide a short-cut to digital systems design. They should understand the basic principles and theories of digital logic design and have a good understanding of the type of digital logic that should be produced.

4. Acknowledgments

The generous supports from Xilinx and Mentor Graphics through their university programs enabled us to educate our students in this very important area. We gratefully appreciate their continuous donations of Xilinx ISE and Modelsim simulators.

References

[1] A. Etxebarria, I. J. Oleagordia, and M. Sanchez, "An educational environment for VHDL hardware description language using the WWW and specific workbench," in *Proceedings - Frontiers in Education Conference*, 2001, vol. 1, pp. 22-27.
[2] A. I. Hussein, D. M. Gruenbacher, and N. M. Ibrahim, "Design and verification techniques used in a graduate level VHDL course," *Proceedings - Frontiers in Education Conference*, vol. 2 pp. 13-14, 1999.
[3] P. L. Jones, "Getting started with VHDL," in *Proceedings of the IEEE International Conference on Microelectronic Systems Education, MSE*, 1997, pp. 135-136.
[4] A. Wu, "Interactive learning toolbox for logic synthesis with VHDL," in *Proceedings of the IEEE International Conference on Microelectronic Systems Education, MSE*, 1997, pp. 77-78.
[5] M. Chang, "Teaching top-down design using VHDL and CPLD," in *Proceedings - Frontiers in Education Conference*, 1996, vol. 2, pp. 514-517.
[6] J. Pedraza-Chavez, D. Baez-Lopez, and J. M. Ramirez, "Use of VHDL as a tool for the teaching of digital systems," in *ASEE Annual Conference Proceedings*, 1995, vol. 2, pp. 2800-2802.
[7] V. Pedroni, "Teaching Design-Oriented VHDL", in *Proceedings of the 2003 IEEE International Conference on Microelectronic Systems Education, MSE'03,* pp.
[8] Z. Navabi, *VHDL Analysis and Modeling of Digital Systems*, McGraw-Hill, 1998.
[9] D. L. Perry, *VHDL,* New York: McGraw-Hill, 1994.
[10] M. Zwolinski, *Digital System Design with VHDL*, Prentice Hall, 2000.