Co-simulation of VHDL and SPICE – Teaching the Whole Picture in Digital Design Labs

Shauna L. Rae
Electronics Workbench
www.electronicsworkbench.com

Abstract – The use of hardware description languages is now an essential part of digital design. As a result, professors require new and innovative ways to deliver course content that teaches both hardware description languages and digital design. Co-simulation of VHDL with other electronic devices allows students to place their VHDL-based components into a circuit with other types of devices such as switches, LEDs, seven-segment displays and other drivers. It provides students with the electronics context that waveform-based simulations fail to provide. We will examine how to incorporate co-simulation of VHDL and SPICE-modeled components into existing digital design curriculum.

Introduction

The methods used to implement and design digital logic into practical circuits have changed dramatically over the last ten years. FPGAs and CPLDs have replaced much of the glue logic used in the past and have allowed more and more complicated designs to be placed on a single chip. These increasingly complicated designs have made gate-based design impractical, causing Hardware Description Languages (HDLs) such as VHDL to replace traditional design methodologies. Over the past decade, instructors have converted introductory and advanced digital design curriculum to include teaching design in conjunction with hardware description languages such as VHDL[1, 4, 7, 8, 10, 13]. Many digital design textbooks have also evolved to include hardware description language examples [2, 6, 9, 12].

The problem arises that VHDL can seem removed from electronics. When students are working in a programming and waveform-only simulation environment, they may lose sight of the context in which the VHDL code is ultimately used. They may also have limited access to labs that would enable them to program their code onto target devices and verify the operation in a real circuit.

Co-simulation of VHDL and SPICE enables instructors to provide students with a simulation environment that resembles a real circuit and helps them to reinforce concepts that go unnoticed in the traditional waveform simulations that usually accompany HDL designs. Students can connect input switches, LEDs and seven-segment displays to their VHDL-based components. This helps them gain context and learn about design elements such as when to use pull-up resistors and the importance of converting hexadecimal numbers to the more readable decimal system. As students progress, they can connect VHDL-based components to more complicated circuitry such as relays, MOSFET drivers, and analog to digital converters. Since these components have SPICE models, the advanced student can simulate an entire design or project.

This paper demonstrates how to integrate co-simulation of VHDL and SPICE into existing courseware. It also highlights the benefits of using co-simulation in conjunction with traditional waveform-based simulations across all levels of digital design.

Introductory Digital Logic Courses

Educators have found that students who are used to working with circuits and components can have difficulties adapting to a hardware description language[7, 13]. With co-simulation, students and instructors can create a component and have it use the VHDL code that they wrote to drive the simulation in a schematic-based environment. This means that they can check the operation of the components in a more familiar CAD type of environment instead of just a code type environment. Co-simulation also means that they can still use all of the existing SPICE/XSPICE modeled components found in the library.

Multisim®, a circuit simulation program produced by Electronics Workbench®[5], allows users to simulate circuit components with both traditional SPICE/XSPICE models and VHDL models simultaneously. A program called MultiVHDL is used for VHDL design entry and waveform testing. After verifying the VHDL code in MultiVHDL, instructors and/or students can create new components in Multisim. To use the VHDL model for the component, the user points to the location where the linked VHDL code resides. Any change to the linked VHDL code is reflected in the behaviour of the component in the Multisim simulation.

By co-simulating a typically used logic gate, for example a TTL AND gate, along side a VHDL modeled AND component, students can gain an understanding of how VHDL relates to traditional gates. Figure 1 illustrates the code a student could write to describe and implement the functionality of an AND gate in VHDL. Figure 2 shows a schematic with both a TTL AND gate and the VHDL modeled AND component.

The CAD representation of the components also helps students to understand what the entity and architecture portions of the VHDL code mean. The entity declaration in VHDL relates directly to the component in the schematic with the inputs A and B shown on the left side of the components and the outputs Y on the right side. The interactive simulator allows students to toggle the position of the switches during simulation and they can see how the inputs affect the outputs of the gates. They quickly see how the architecture section of the code describes how the component behaves and how the two components behave the same (if the code is right).

```
library ieee;
use ieee.std_logic_1164.all;

entity AND_GATE is
     port (
          A: in std_logic;
          B: in std_logic;
          Y: out std_logic);
end AND_GATE;

architecture STRUCTURE of AND_GATE is
begin
     Y <= A and B;
end STRUCTURE;
```

Figure 1: VHDL Description of an AND Gate

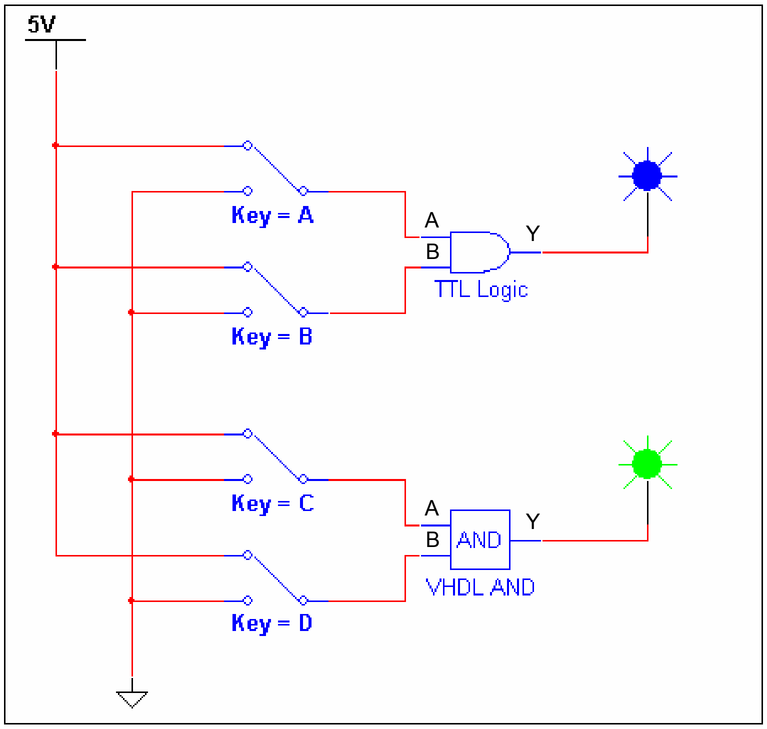
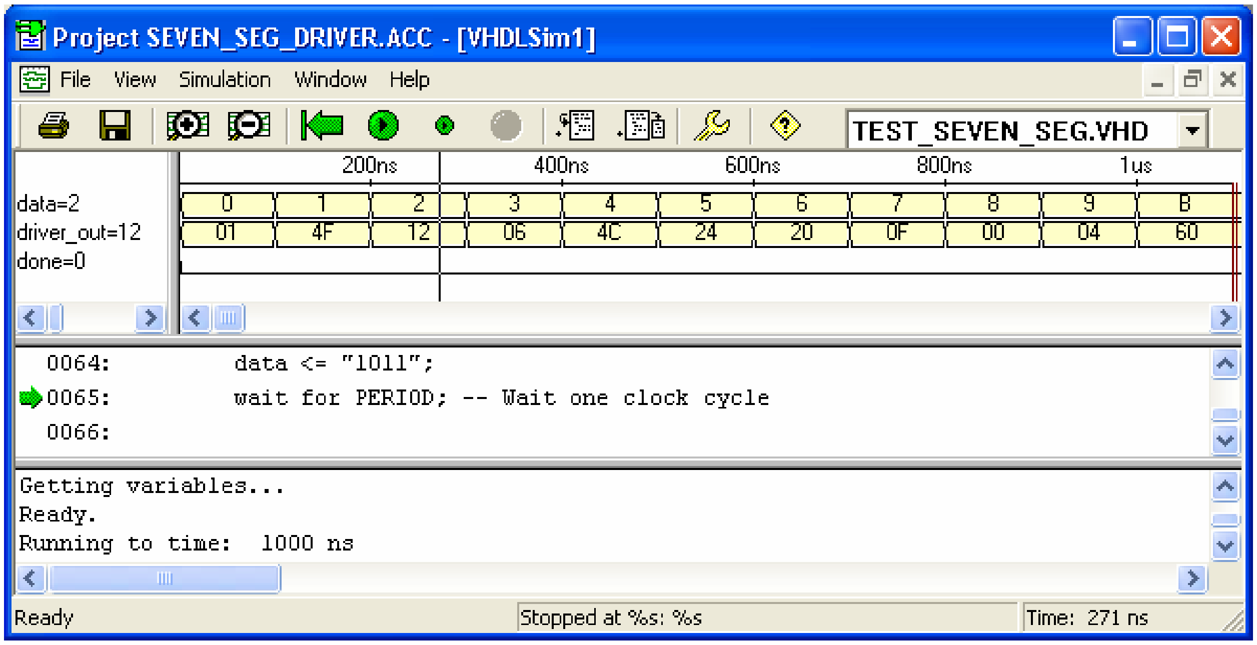


Figure 2: Comparing VHDL parts to TTL parts

Once students are more comfortable with the concept of VHDL, and how it relates to electrical components, they can learn topics that are more complicated. Introductory courses on digital design typically include lessons on encoders and decoders[6, 10]. Figure 3 illustrates the waveform results of a testbench written to check the operation of a hexadecimal to seven-segment decoder. The VHDL implementation of the decoder is in Appendix A.


Project SEVEN_SEG_DRIVER.ACC - [VHDLSim1]
File View Simulation Window Help
TEST_SEVEN_SEG.VHD
200ns 400ns 600ns 800ns 1us
data=2
driver_out=12
done=0
0 1 2 3 4 5 6 7 8 9 B
01 4F 12 06 4C 24 20 0F 00 04 60

```
0064:         data <= "1011";
0065:         wait for PERIOD; -- Wait one clock cycle
0066:
```

```
Getting variables...
Ready.
Running to time:  1000 ns
```

Ready | Stopped at %s: %s | Time: 271 ns

Figure 3: Waveform results of the Seven-Segment Decoder Testbench

The waveform results produced by the testbench provide students with verification that their VHDL code is working properly but it fails to provide context. It is not clear that the seven-segment decoder is actually outputting the correct output to drive a seven-segment display. It only shows that the inputs produce the outputs that the student thought to be correct. If, for instance, they thought that input “4” should produce the outputs “4D” instead of “4C” their waveform testbench would appear correct.

Co-simulating the decoder, allows them to verify that the given inputs are generating the desired outputs. In other words, when they input 0011 they will see the number 3 displayed on the seven-segment display, see figure 4. The context that co-simulation provides is especially important for students learning digital design at a distance where they may not have access to hardware labs to test their VHDL code. They can use the switches to control the inputs to the driver and watch the output automatically update as it would in a hardware implementation with an FPGA or other programmable logic device.

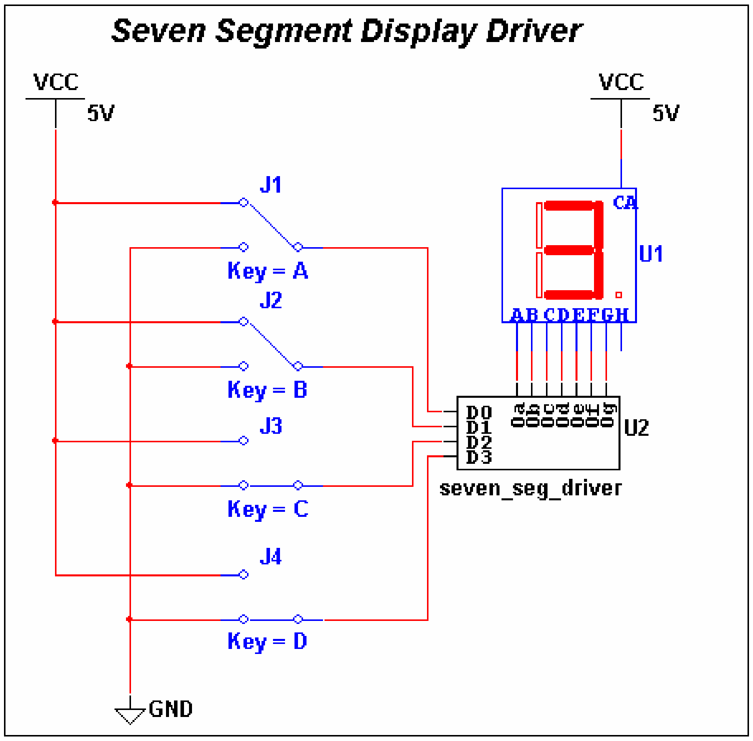
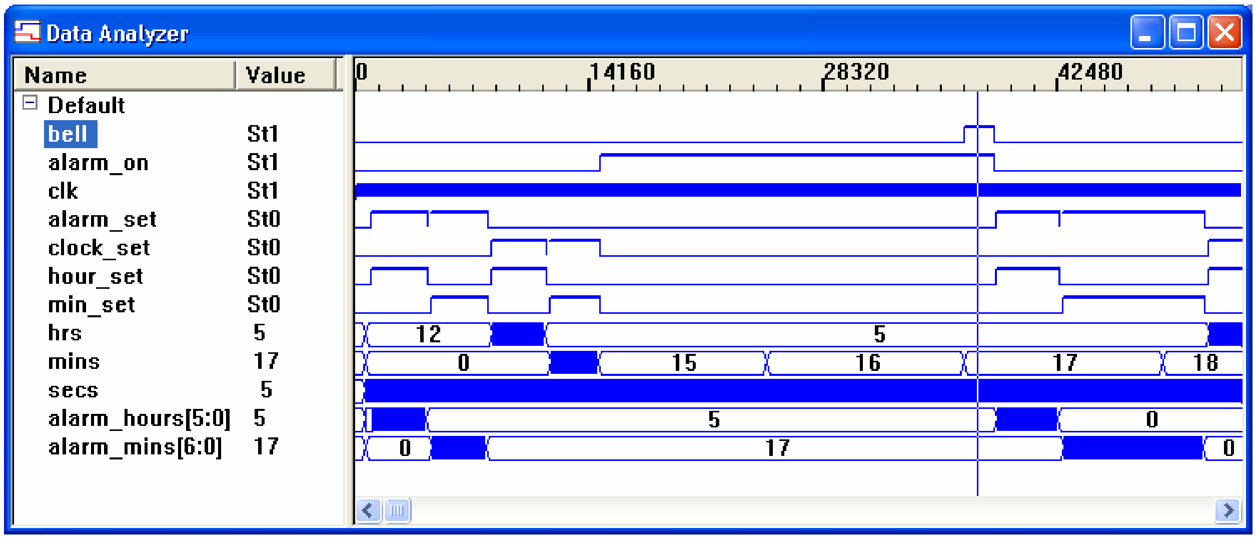


Figure 4: Testing the Seven-segment Display Driver using Co-simulation

Advanced Digital Logic

Once students move on to more advanced designs, co-simulation continues to provide benefits. Beyond simply providing context that would otherwise require programming a target device, co-simulation helps students track down errors. At Chippewa Valley Technical College, Tim Tewalt assigned his students the design of a digital alarm clock in HDL. The results of the waveform testbench, see figure 5, showed that the alarm clock worked as expected. Once the design was implemented in co-simulation it revealed a couple of oversights.



Data Analyzer

Name	Value
Default	
bell	St1
alarm_on	St1
clk	St1
alarm_set	St0
clock_set	St0
hour_set	St0
min_set	St0
hrs	5
mins	17
secs	5
alarm_hours[5:0]	5
alarm_mins[6:0]	17

Figure 5: Waveform Results of a Testbench to Check Operation of Digital Alarm Clock

Observe in this testbench, when the alarm_set and hour_set inputs go high the alarm hour time changes to 5. When the alarm_set and minute_set inputs are high the alarm minutes change to 17. The clock is set similarly.

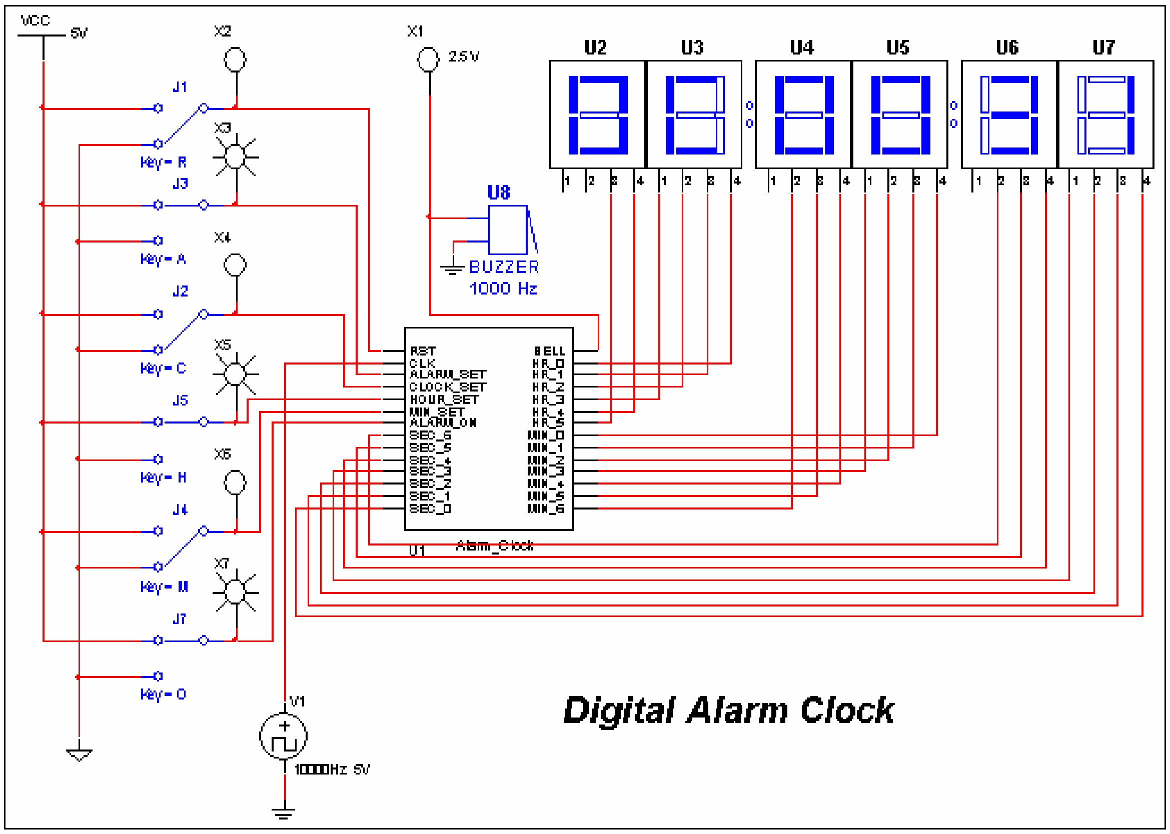


Figure 6: Co-simulation Implementation of Digital Alarm Clock Reveals Issue Hidden in Waveform Test

What students observed once co-simulating the alarm clock in Multisim®, see figure 6, is that when setting the alarm, the alarm time was not sent to the output buffer. The waveform test displays internal registers that a component cannot. Another oversight that the students missed was that the time was displayed in hexadecimal numbers, making this clock un-usable for the general public.

As students' skills increase further, they can take advantage of co-simulation to simulate complete design projects. The VHDL modeled parts will co-simulate with all of the components found in the Multisim® library.

Conclusion

Students can benefit from using co-simulation in their digital design labs. In introductory courses, it can help them understand how VHDL code relates to other electronic components. It will also provide them with additional context of how VHDL code is ultimately used in hardware and how it can interact with other devices. Once they move on to more advanced designs, they can take advantage of co-simulation to reinforce their waveform testbenches. They will be able to catch errors easily missed in a waveform test environment. Co-simulation will also help in design projects since students can simulate and document[11] entire projects including signal conditioning and output drivers.

Bibliography

1. Areibi, S. "A First Course in Digital Design using VHDL and Programmable Logic." *Proceedings of Frontiers in Education Conference*, 2001.

2. Brown, S. and Vranesic, Z., *Fundamentals of Digital Logic with VHDL Design*. McGraw-Hill. 2000.

3. Chang, M. "Teaching top-down design using VHDL and CPLD." Proceedings of Frontiers in Education Conference. 1996.

4. Chu, P.P. "A Small, Effective VHDL Subset for the Digital Systems Course." *Proceedings of ASEE Annual Conference and Exposition.* 2004.

5. Electronics Workbench. www.electronicsworkbench.com.

6. Floyd, T.L. *Digital Fundamentals with VHDL.* Pearson Education. Upper Saddle River, New Jersey. 2003.

7. Fucik, O., Wilamowski, B. M. and McKenna, M. "Laboratory for the Introductory Digital Course," Proceedings *of ASEE Annual Conference and Exposition.* 2000.

8. Greco, J. "Designing a Computer to Play Nim: A Mini-Capstone Project in Digital Design I." *Proceedings of ASEE Annual Conference and Exposition.* 2004.

9. Mano, M.M. *Digital Design*, Prentice-Hall, NJ. 3rd Edition, 2001.

10. Parten, M.E. "Teaching Digital Design with HDL," *Proceedings of ASEE Annual Conference and Exposition*, 1997.

11. Tapper, J. “Are our engineering students learning the communications skills they will need as professionals in industry?” *Proceedings of ASEE Annual Conference and Exposition*, 2004.

12. Wakerly, J. *Digital Design: Principles & Practices*. Prentice Hall. 2000.

13. Zemva, A. and Trost, A. and Zajc, B “A Rapid Prototyping Environment for Teaching Digital Logic Design.” *IEEE Transactions on Education.* Nov. 1998.

Biography

Shauna L. Rae received a B. Sc. (Co-op) in Electrical Engineering from the University of Alberta in 1998. After working at Telus Communications and as a hardware design engineer for a consulting company, she returned to the U of A to study. She received her M. Sc. from the Department of Electrical and Computer Engineering in 2003. Shauna now works at Electronics Workbench as an Applications Engineer where she is responsible for ensuring educators, students, and professionals can more effectively use their EDA tools to meet their needs.

Appendix A

```
library ieee;
use ieee.std_logic_1164.all;

entity seven_seg is
     port (
          data: in std_logic_vector(3 downto 0);
          driver_out: out std_logic_vector(6 downto 0));
end seven_seg;

architecture BEHAVIOR of seven_seg is
begin

with data select
     driver_out <=    "0000001" when "0000",
                      "1001111" when "0001",
                      "0010010" when "0010",
                      "0000110" when "0011",
                      "1001100" when "0100",
                      "0100100" when "0101",
                      "0100000" when "0110",
                      "0001111" when "0111",
                      "0000000" when "1000",
                      "0000100" when "1001",
                      "0001000" when "1010",
                      "1100000" when "1011",
                      "0110001" when "1100",
                      "1000010" when "1101",
                      "0110000" when "1110",
                      "0111000" when "1111",
                      "XXXXXXX" when others;
end BEHAVIOR;
```

Figure 7: VHDL Description of a Seven-segment Decoder