DEVELOPING UNDERGRADUATE FPGA CURRICULUM USING ALTIUM SOFTWARE AND HARDWARE

Erik A. Mayer
Electronics Engineering Technology
Pittsburg State University

Abstract

In this paper, the development and assessment of a curriculum covering field programmable gate arrays (FPGAs) will be discussed. An FPGA is a programmable logic device which consists of programmable logic blocks with programmable interconnections. An FPGA may consist of millions of equivalent logic gates.

The motivation for the development of the FPGA curriculum was the need for students to be proficient in FPGAs. This need was assessed by two events. At a meeting of the EET industrial advisory committee, the need for employees proficient in FPGAs was expressed. In addition, the software company Altium Limited expressed an interest in working with the EET program to develop FPGA curriculum material for a university program.

In the fall semester of 2011, the FPGA curriculum was used in an advanced digital logic course. Previously, programmable array logic (PAL) devices and generic array logic (GAL) devices were used. This course serves as a required course in the newly created embedded systems emphasis area in the four-year undergraduate Electronics Engineering Technology (EET) program at Pittsburg State University in Pittsburg, Kansas. It is planned to include the FPGA material in future offerings of the course.

Curriculum

The decision to use Altium software and hardware in the FPGA curriculum was motivated by the fact that Altium expressed an interest in working with the EET program to develop FPGA curriculum material for a university program. Altium Limited[1] has developed an electronic design automation (EDA) software tool called Altium Designer[2]. Altium Designer unifies the design of printed-circuit boards, the design of FPGAs, and embedded system programming. To further facilitate the development of embedded systems, Altium also created NanoBoards which are reconfigurable hardware development platforms[3]. Currently, two versions of the NanoBoard are available: the NanoBoard NB2 and the NanoBoard 3000. For the FPGA curriculum, the NanoBoard 3000 was used. The NanoBoard 3000 has a variety of peripherals including analog-to-digital converters, digital-to-analog converters, audio CODEC, speakers, touchscreen LCD display, pushbuttons, and RGB LEDs[3]. The NanoBoard 3000 has three different variations, each containing an FPGA from a different manufacturer. The variation of the NanoBoard 3000 used in the curriculum contained a Xilinx Spartan-3AN FPGA.

Hardware description languages (HDLs) can be used to configure an FPGA. Two HDLs are mostly used today: VHDL and Verilog. VHDL stands for VHSIC hardware description language while VHSIC stands for very-high-speed integrated circuits. The FPGA curriculum was originally developed using both VHDL and Verilog. However, it was found, at a meeting of the EET industrial advisory committee, that there was a need for employees who knew VHDL. Thus, the decision was made to focus on VHDL. VHDL can be used for two major steps in the design of FPGAs. The first is the *simulation* of a digital circuit to be implemented into an FPGA. The second is the *synthesis* of a digital circuit by configuring an FPGA. The lectures and labs were designed to cover both the simulation and synthesis aspects of VHDL.

Working with Altium, a set of lectures and associated labs for the FPGA curriculum was created that utilized the Altium Designer EDA and the NanoBoard 3000. The majority of the material for the lecture and labs was drawn from documents found on the Altium website[1]. The lectures and labs were designed such that they built on skills acquired in previous lectures and labs. Table 1 shows a listing of the lectures and the labs associated with them. It also shows the hardware and software used in the labs. The lectures and labs covered the following topics: implementing combinatorial and sequential logic circuits using FPGAs, performing simulation and synthesis with VHDL, writing VHDL testbenches, using embedded instruments for testing FPGAs, and configuring and programming soft processors.

Before being used in the undergraduate advanced digital logic course, initial versions of the lectures and labs were used in a graduate course. This helped identify any modifications required in the lectures and labs. It also helped assess possible difficulties that students might encounter. As a result, more background lecture material on FPGAs and programmable logic devices was developed for the undergraduate course. In addition, labs using VHDL for simulation were developed to reinforce fundamental concepts of digital system design.

In the graduate course, a textbook was not used. However, the students expressed a need for a book to reference, so a textbook was chosen for the FPGA curriculum. The textbook used was "VHDL: A Starter's Guide" authored by Sudhakar Yalamanchili[4]. This textbook included references to the Active-HDL software which allowed for the simulation of VHDL code[5]. Active-HDL allowed students to perform simple simulations of VHDL code quickly without requiring the writing of a complicated VHDL testbench. Thus, the Active-HDL Student Edition was added to the FPGA curriculum. Unfortunately, the textbook covered only simulation, but not synthesis. A search is being done for a textbook that covers both.

The first Lecture in Table 1, "Introduction to Programmable Logic Devices," covers the history of programmable logic devices, the various types, and the phases of design creation. The "Introduction to FPGA Programming with Schematic Editor" lab serves to introduce the students to Altium Designer and the NanoBoard. This lab introduces schematic capture with the *Schematic Editor* and shows how to input a half-adder logic circuit and program it into the NanoBoard. The students are then assigned to create a full-adder circuit.

The next lecture, "Introduction to VHDL," covers a history of VHDL, its different versions, and its different levels of abstractions. It also covers how to write VHDL for basic combinatorial logic circuits. The "Active-HDL Tutorial" lab has students work through the tutorial in the textbook. The "Binary to Hex Digit display using VHDL" lab has students write the VHDL code for a combinatorial circuit and use Active-HDL to simulate it. The final lab, "Introduction to FPGA Programming with VHDL," demonstrates how to use Altium Designer to write a full-adder program in VHDL and program it into the Nanoboard. The students are then assigned to create a 2-bit adder.

The "Programming Sequential Logic Circuits with VHDL" lecture discusses how to use VHDL for sequential circuits. The "Simulating Sequential Circuits with VHDL" has the students input VHDL code in Active-HDL for a four-bit counter and simulate it. The students are then asked to write the VHDL code for an 8-bit counter. The "Programming Sequential Circuits with VHDL" lab shows how to use VHDL with Altium Designer to program sequential circuits. A BCD counter is used as an example and programmed into the Nanoboard. The students are then asked to create a 4-bit counter.

The lecture "Encoding State Machines with VHDL" covers state machines and how to create one using VHDL. In the lab "Implementing State Machines Using VHDL,"

Table 1 FPGA Curriculum.

Lecture	**Lab**	**Software/Hardware**
Introduction to Programmable Logic Devices	Introduction to FPGA Programming with Schematic Editor	Altium Designer NanoBoard 3000
Introduction to VHDL	Active-HDL Tutorial	Active-HDL
	Binary to Hex Digit display using VHDL	Active-HDL
	Introduction to FPGA Programming with VHDL	Altium Designer NanoBoard 3000
Programming Sequential Logic Circuits with VHDL	Simulating Sequential Circuits with VHDL	Active-HDL
	Programming Sequential Circuits with VHDL	Altium Designer NanoBoard 3000
Encoding State Machines with VHDL	Implementing State Machines Using VHDL	Altium Designer NanoBoard 3000
Test 1		Altium Designer NanoBoard 3000
VHDL Testbench and Simulation	VHDL Testbench and Simulation	Altium Designer
Testing FPGAs with Virtual Instruments	Testing FPGAs with Virtual Instruments	Altium Designer NanoBoard 3000
Using Soft Microprocessor Cores on FPGAs	Embedded Programming with C Language	Altium Designer NanoBoard 3000
Final Project		Altium Designer NanoBoard 3000

students are asked to draw the state graph and write the VHDL code for a state machine that will act as a 3-bit Gray code counter. The students are then asked to program the counter in Altium Designer, run the state machine on the NanoBoard, and verify its correct operation.

Test 1 assesses the students' knowledge of FPGAs that they have gained thus far. For the fall 2011 semester, students were given a state graph and asked to write the VHDL code for the state machine. The students are then asked to program the state machine in Altium Designer and configure it on the NanoBoard.

In the lecture "VHDL Testbench and Simulation," it is discussed how to write a VHDL *testbench* to test and debug a design before it is implemented into an FPGA. A VHDL testbench is a VHDL program used to test a digital design. In it, one can specify the stimulus to the design and record the generated outputs. The lab "VHDL Testbench and Simulation" shows how to use Altium Designer to write a VHDL testbench that tests a BCD counter. The students are then asked to write a VHDL testbench to test the VHDL code they created previously for a 4-bit counter.

The lecture "Testing FPGAs with Virtual Instruments" covers *embedded* or *virtual instruments*. It is impractical to monitor the state of a logic component inside an FPGA with conventional instrumentation unless it is connected to an output. In addition, it is impractical to inject a signal to a logic component inside an FPGA with conventional instrumentation unless it is connected to an input. Embedded or virtual instruments are instruments such as logic analyzers and frequency generators that are integrated into FPGA designs to facilitate on-chip testing and debugging. In the lab "Testing FPGAs with Virtual Instruments," it is shown how to use Altium Designer with the NanoBoard to test a Johnson counter with virtual instruments. The students are asked to use virtual instruments to test a BCD counter they created previously.

The lecture "Using Soft Microprocessor Cores on FPGAs," covers soft processors. *Soft processors*, *soft microprocessors*, or *softcores* are microprocessors implemented using FPGA logic. The lecture also discusses the *OpenBus* system which is used to abstractly represent the processor's connections to peripherals. This reduces the complexity of a FPGA design. The lab "Embedded Programming with C Language" shows how to use Altium Designer to create a soft microprocessor core in the FPGA and program it using C. The OpenBus system is used as well as the *Software Platform Builder* which provides an Application Programming Interface (API) layer between the user code and the hardware. An LED chaser which lights up LEDs sequentially is used as an example. Students are then asked to modify the LED chaser.

For the final project, students are asked to apply the knowledge they have gained about FPGAs from the lectures and labs to a real-world problem. For the fall 2011 semester, the students were asked to apply FPGAs to a bat counter. The bat counter was developed by Rick Laas and Dennis Hewitt who worked with the Department of Biology at Pittsburg State University. The bat counter was designed to count the number of bats that fly through an aperture. The bat counter had a sensor which consisted of a light beams arranged such that a bat flying through the aperture would break two of the beams simultaneously. The students were asked to design a circuit that would use the sensor to record the number of bats.

Assessment

To assess the effectiveness of the FPGA curriculum, the students' capabilities with FPGAs were assessed during the final project. For the final project, the students were given a real-world problem and no guidance on how to solve it. The rubric shown in Table 2 was used to assess the students' ability to select and apply the appropriate knowledge, techniques, skills, and tools to design, test, and improve their final project.

Using the rubric in Table 2, the students were assessed on their choice of how they created their projects. They were assessed on how much of their project was created using pre-defined components in the Schematic Editor, how much of their project was created using VHDL code, and how much of their project was created using a soft processor and the OpenBus system.

The students were also assessed on their choice of how they tested and verified their projects. They were assessed on how much of their testing used simulations performed with a VHDL testbench or Active-HDL. In addition, they were assessed on how much of their testing used virtual instruments.

It would have been valuable to solicit feedback from students on their experience of using FPGAs. However, none was collected for the initial course offering. This is planned for future offerings of the advanced digital logic course.

Table 2.

Category	0 points	1 point	2 points	3 points
Use of VHDL to create project	Not used	Used in less than 33% of project	Used in 33% to 66% of project	Used in more than 66% of project
Use of Schematic Editor to create project	Not used	Used in less than 33% of project	Used in 33% to 66% of project	Used in more than 66% of project
Use of soft processors/OpenBus to create project	Not used	Used in less than 33% of project	Used in 33% to 66% of project	Used in more than 66% of project
Use of Virtual Instruments to test project	Not used	Used in less than 33% of testing	Used in 33% to 66% of testing	Used in more than 66% of testing
Use of VHDL Testbenches to test project	Not used	Used in less than 33% of testing	Used in 33% to 66% of testing	Used in more than 66% of testing
Use of Active-HDL to test project	Not used	Used in less than 33% of testing	Used in 33% to 66% of testing	Used in more than 66% of testing

Table 3.

Category	0 points	1 point	2 points	3 points	Average Score
Use of VHDL to create project			2	11	2.846
Use of Schematic Entry to create project		11	2		1.154
Use of soft processors/OpenBus to create project	13				0
Use of Virtual Instruments to test project			1	12	2.923
Use of VHDL Testbenches to test project	12	1			0.0769
Use of Active-HDL to test project	12	1			0.0769

Results

For the Fall 2011 semester, 13 students were assessed and the results are shown in Table 3. All 13 students successfully completed the final project. The numbers in Table 3 represent the number of students who were assigned the point values indicated at the top of the column for each category. The average score for each category is recorded in the rightmost column.

It can be seen from Table 3 that students used mostly VHDL in creating their final project as opposed to using pre-defined components in the Schematic Editor. This shows that the FPGA curriculum has done a good job at making the student proficient and comfortable with using VHDL for synthesizing their circuits.

However, the results show that no students attempted to use a soft processor or the OpenBus system in the final project. This shows that more work needs to be done on the curriculum to get students comfortable working with soft processors and the OpenBus system. For the final project, all of the students used the RGB LEDs to display the count in binary representation. The use of a soft processor and the OpenBus system would facilitate the students' use of the NanoBoard's touchscreen LCD display. The LCD display could show the count in decimal representation. In addition, all of the students used a pushbutton to reset the counter. If the students used the LCD display, they could use its touchscreen feature instead of a separate pushbutton. In addition, the use of a softprocessor and the LCD display would allow for the ease of implementation of more features such as storing and displaying counts taken nightly over a month.

Table 3 shows that most students preferred to test the FPGA using virtual instruments instead of using a VHDL testbench or Active-HDL. This shows that the FPGA curriculum needs to be modified to demonstrate to the students the advantage of using VHDL testbenches for testing. A VHDL testbench could perform a more comprehensive test of the final project as it could automatically present stimulus to the inputs of the final project and generate a listing of the generated outputs. When using virtual instruments, the students had to set each input manually and note the generated output.

TAC/ABET Criteria

In the future, it is planned to use the FPGA curriculum to satisfy the accreditation criteria of the Technology Accreditation Commission (TAC) of ABET[6]. Criterion 3, Student Outcomes, from ABET's "Criteria for Accrediting Engineering Technology Programs" (2011-2012) states that a program that is accredited must have student outcomes that are documented and that prepare graduates to meet the program educational objectives[6]. These student outcomes can be used for the assessment and the evaluation of the student outcomes and for the continuous improvement of the program[6]. Table 4 shows the program educational objectives and their associated student outcomes for the advanced digital logic course which contains the FPGA curriculum. To meet Criterion 3, a list of learned capabilities must be included in the student outcomes. The student outcomes for the FPGA curriculum include the following learned capabilities for a baccalaureate degree program[6]:

a. an ability to select and apply the knowledge, techniques, skills, and modern tools of the discipline to broadly-defined engineering technology activities;

b. an ability to select and apply a knowledge of mathematics, science, engineering, and technology to engineering technology problems that require the application of principles and applied procedures or methodologies;

c. an ability to conduct standard tests and measurements; to conduct, analyze, and interpret experiments; and to apply experimental results to improve processes;

Table 4.

Major Topic Areas	**Significant Class Objectives**	**Significant Laboratory Objectives**	**Student Outcomes**
Students completing course acquire level appropriate content competencies in:	Students completing course will show they can:	Students completing course laboratory will show they can:	Validation of Student Outcomes will be measured in a process that:
Using modern tools utilized in configuring programmable logic devices	Configure FPGAs using Altium Designer EDA	Configure FPGAs using Altium Designer EDA	Configures an FPGA using Altium Designer EDA in implementing a final project
Using hardware description languages in configuring programmable logic devices	Configure FPGAs using VHDL	Configure FPGAs using VHDL	Uses VHDL in implementing a final project
Use of soft processors	Configure and program soft processors in an FPGA	Configure and program soft processors in an FPGA	Configures soft processors in implementing a final project
Testing and verifying programmable logic devices	Use VHDL testbenches and virtual instruments to test and verify FPGA	Use VHDL testbenches and virtual instruments to test and verify FPGA	Uses VHDL testbenches and virtual instruments in testing a final project

Table 5.

EET Assessment Rubric Scale			
0	1	2	3
No meaningful answer, complete misinterpretation of problem	Missing key concepts; significant procedural or methodology errors	Key concepts are essentially correct, minor procedural or methodology errors	Answers are correct

d. an ability to design systems, components, or processes for broadly-defined engineering technology problems appropriate to program educational objectives;

e. an ability to identify, analyze, and solve broadly-defined engineering technology problems.

In addition, the student outcomes for the FPGA curriculum will also meet the following outcomes required by the Program Criteria for an EET baccalaureate degree program[6]:

> the application of circuit analysis and design, computer programming, associated software, analog and digital electronics, and microcomputers to the building, testing,

operation, and maintenance of electrical/electronic(s) systems;

a. the ability to analyze, design, and implement control systems, instrumentation systems, communications systems, computer systems, or power systems.

It can be seen from Table 4 that the student outcomes involve the final project. Thus, the assessment of the FPGA curriculum will occur during the final project. The assessment will be done for each student outcome using the rubric in Table 5.

Conclusion

An FPGA curriculum was created for an advanced digital logic course. This curriculum contained lectures and the labs which covered: implementing combinatorial and sequential logic circuits using FPGAs, performing simulation and synthesis with VHDL, writing VHDL testbenches, using embedded instruments for testing FPGAs, and configuring and programming soft processors.

The efficiency of the FPGA curriculum was assessed by the students' ability to use FPGAs in a final project. It was found that the curriculum taught students VHDL well enough as it was the predominant approach used in creating the final project. However, none of the students used a soft processor and the OpenBus system. This limited the students' use of peripherals in their final project: The students used LEDs to show the output of the final project in binary format instead of using an LCD display. This shows that more work needs to be done to get the students comfortable with soft processors and the OpenBus system.

The assessment also showed that students favored testing their designs with embedded instruments rather than using a more comprehensive test with a VHDL testbench. This shows that the curriculum needs to be developed to get the students more comfortable in using VHDL testbenches.

Bibliography

1. "Altium – Next generation electronic design,"http://www.altium.com/, Retrieved January 12, 2012.

2. "Altium | AD 10," http://products.live.altium.com/, Retrieved January 12, 2012.

3. "Altium NanoBoard 3000," http://nb3000.altium.com/intro.html , Retrieved January 12, 2012.

4. Yalamanchili, Sudhakar, VHDL: A Starter's Guide, Second Edition, Pearson Prentice Hall, 2005.

5. "Aldec – Products – FPGA Simulation – Active-HDL", http://www.aldec.com/en/products/fpga_simulation/active-hdl, Retrieved January 12, 2012.

6. "ABET Criteria for Accrediting Engineering Technology Programs, 2011-2012," http://www.abet.org/tac-current-criteria/, Retrieved January 12, 2012.

Biographical Information

Erik Mayer received his Ph.D. in Engineering Science at the University of Toledo. His areas of focus are power electronics and embedded systems. He was an instructor at Bowling Green State University, where he worked with the Electric Vehicle Institute and taught courses in digital circuit design, microcontrollers, and renewable energy. In addition, he has worked at Visteon designing components for hybrid vehicles. He is currently a Professor at Pittsburg State University where he teaches courses supporting the embedded systems emphasis in the Electronics Engineering Technology program.