2006-277: INTRODUCING FIELD-PROGRAMMABLE GATE ARRAYS INTO SOPHOMORE DIGITAL CIRCUITS COURSE

Sin Ming Loo, Boise State University

Arlen Planting, Boise State University

Matt Murdock, Boise State University

Introducing Field-Programmable Gate Arrays into Sophomore Digital Circuits Course

Abstract

In this paper, we describe our experiences in introducing Field-Programmable Gate Arrays (FPGA) into our sophomore digital circuits course. We describe our findings, the techniques of preparing the laboratory and computer systems, and approaches taken to reduce the number of problems that can appear during laboratory sessions. The students are exposed to digital circuit design using discrete 74xx series components during the first four weeks of the semester. For the rest of the semester, all designs are targeted at FPGA. The use of 74xx series components has been kept in the laboratory for two reasons. First, a 74xx series component might be all that is needed for a simple design. Second, the concept of putting together a design utilizing multiple components (system-level design) can be introduced. Potentially, this encourages the students to optimize their designs so that wiring the digital circuit is easier.

Our findings through this introduction have all been positive. Students are eager to learn the industrial strength tool (Xilinx ISE or Altera Quartus II) while using the FPGA to prototype designs. Due to the complexity of learning the design tools, we developed multiple step-by-step tutorials so that students can learn the basic features on their own. Advanced software features are introduced during lecture or laboratory sessions. Our experiences have shown that during the laboratory sessions while using FPGA, the students do not have to worry about the status of discrete components, prototyping boards, and wiring integrity. Instead, they think about whether they have designed their circuits correctly.

One potential problem for the introduction of this type of computer-aided design tool is the age of the computer systems. We have found that any computer system more than three years old will introduce many problems in the laboratory. This is because the CAD tools are processor and memory intensive. Older computer systems have difficulties satisfying what is required by the CAD tools.

1 Introduction

Teaching a sophomore digital systems course using just basic discrete transistor-transistor logic (TTL) components is no longer a viable and productive option [1-6]. In this case, the pace of technology should dictate what is being taught in the classroom and used in the laboratory, such that students emerging from the course will find the knowledge and skills learned to be useful in the upper division courses and those completing the degree program will be more qualified to obtain advanced careers. One may argue that the use of discrete components and wiring skills learned are highly important and desirable in the real-world engineering environment. However, such skills are usually not the concentration of any four-year degree program. It is strength in design capability and design debugging that makes a good engineer, not prototype wiring. The use of discrete components in real-world engineering problems is very limited. If such use is so limited, should this be the only technology used in the students' learning environment?

One solution to this dilemma is using a combination of mature technology and the latest technology in the learning laboratory. The truth is that basic TTL components are wonderful tools for introducing digital logics principles to those who are new to this field (TTL components are so inexpensive!). The simple connections to power and ground provide insight into getting components to operate. This mature technology should remain as an introduction to the curriculum. The challenge is introducing the new technology into the curriculum.

Our digital systems curriculum starts with a tutorial of design entry and simulation of an industrial strength computer-aided design tool. Xilinx ISE 6.3i [7] and Modelsim Xilinx Edition have been used during the past two semesters. Due to the complexity of learning the design tools, we approach the problem by painstakingly designing multiple step-by-step tutorials so that students can learn the basic features at their own pace. For the first four weeks of the semester, students use this design software suite to design and simulate their laboratory exercises, and the TTL discrete components are used to construct their designs using a proto-board. During the rest of the semester, an FPGA design development kit is used for implementation.

The next section shows how this class fits into the overall electrical and computer engineering curricula. Section three discusses the digital systems laboratory, and tool usage in detail. Section four describes how the upper division courses absorb the changes in the course. Section five discusses the techniques to reduce team and laboratory problems. Section six outlines changes that we will introduce next semester. Our conclusion follows.

2 Digital Systems and Laboratory Overview

The digital systems lecture and laboratory is classified as a 200-level sophomore course. The only pre-requisite to enroll in this course is an introductory programming course (C, C++, or Java). Thus, students enrolled in this laboratory have been introduced to binary number systems. A majority of the students enrolled in this course are electrical engineering and computer science majors. Occasionally, we have physics students in our course. The digital systems course has been set-up with three credit hours of lecture and one credit hour of laboratory, respectively. The lecture section meets for seventy-five minutes per session twice a week. The laboratory meets for one three-hour session per week. During a semester, we have ten hands-on laboratory exercises and one final project. Each lab contains a small portion of pre-laboratory work. For each laboratory exercise, the students are required to work through the problem (pre-lab), enter the design using a schematic capture technique or hardware description language (VHD) (a small subset of VHDL is shown in class along with the schematic symbols), simulate the design, and build the design using discrete components or download to FPGA. Currently, Spartan-3 starter kits are used in the digital system laboratory [7]. The reader can visit the laboratory webpage to find more details about the laboratory coursework [9].

3 Digital Systems Laboratory

Table 1 summarizes the flow of laboratory exercises for the digital systems laboratory. This is a very standardize set of laboratory experiments to teach basic concepts of digital circuits. We anticipate that these experiments can be easily ported with little modification for use at other curriculum. The flow of laboratory experiments has been customized to follow the course text-book [8]. During the first and second weeks of the semester, the students work through a self-paced tutorial to learn the basic features of Xilinx ISE schematic capture design entry and waveform testbench creation [9]. Due to the number of features available in the Xilinx ISE and Modelsim tools, we introduce the most basic features to the students to be able to do useful work as soon as possible. In our case, this is the first laboratory exercise. This self-paced tutorial has shown to be very useful and students always refer to this tutorial for several weeks before they are comfortable with all of the necessary steps. With this basic tutorial, the students will be able to test all designs that they will complete for the rest of the semester.

We also stress that they do not have to know everything about the tools initially. We show them where to find the documentation to the software and encourage them to explore. We have found hardworking, enthusiastic students will take this opportunity and also ask whether they can use new features in their laboratory experiments. We also tell them that this is a suite of industrial strength software so it may have software bug problems and may be difficult to learn, and that patience is highly desirable.

Table 1: Topics for the Digital Systems Laboratory

Lab	Laboratory Topic	Details
0	Basics and Tutorial: <i>Introduction to Digital Systems</i>	Introduction to discrete logic prototyping board, logic components, and Xilinx ISE and Modelsim Xilinx Edition. Implement functions in logic gates on discrete logic prototyping board with 7400 series chips.
1	Combinational Circuits: <i>Universal NAND</i>	Build basic logic with just NAND and NOR. Design and simulate with Xilinx and Modelsim. Build circuits on proto-board with 7400 series chips.
2	Advanced Combinational Circuits: <i>"Bad" Logic Circuit Re-Design</i>	Derive Boolean expression from existing circuit. Simplify output equation. Re-design circuit of simplified equation.
3	Introduction to FPGA: <i>Mystery Chip Design</i>	Derive Boolean expression from truth table. Design and simulate in Xilinx. Implement circuits and test their functionality with FPGA.
4	Adders & Subtractors: <i>Arithmetic Unit</i>	Modular design. Design a 4-bit adder by using 1-bit full adder modules. Build an adder/subtract unit with add/sub control bit. Test on FPGA.
	<i>Lab Quiz #1</i>	<i>Derive: truth table, K-map, simplify expressions, design circuit, timing diagram/simulation, Test on FPGA platform.</i>
5	Multiplexer, Encoder, Decoder: <i>Multiplexer and Decoders</i>	Design simple MUXs and decoders with logic gates. Implement certain Boolean Expressions with MUXs and decoders.
6	Adders & Multipliers: <i>Yet Another Arithmetic Unit</i>	Design 4-bit fast adders and multipliers. Multiplex signals from multiple sources into one common data bus.
7	VHDL, Latches & Flip-flops: <i>VHDL & Simple "Memory" Circuit</i>	Design a logic circuit using VHDL instead of schematic capture in Xilinx. Understand/show the differences between latch and flip-flop.
8	Counters: <i>Latch, Flip-Flop, Up-Down Counter</i>	Design a 4-bit up-down counter and download the design to FPGA.
9	Finite State Machines: <i>Sequence Detector</i>	Design a state-machine sequence detector. Show state diagram, state table and assignment table, K-maps, equations, logic circuit, download design to FPGA.
	<i>Lab Quiz #2</i>	<i>Design a state-machine sequence detector. Show state diagram, state table and assignment table, K-maps, equations, logic circuit, download design to FPGA.</i>
	Final Project	See course/laboratory website for details.

The same instructor usually teaches the lecture and laboratory sections. When they are not taught by the same instructor, measures are taken to ensure that the lecture and laboratory are synchronized. In the first three laboratory exercises, the students use Xilinx ISE to enter their designs and create testbench waveforms. Modelsim Xilinx Edition is utilized for the actual simulation, although we do not teach the usage of Modelsim in this course. However, a few simulator restarts and signal grouping techniques are shown to students who show interest. When designs are proven using the simulator, the students are required to construct the design using discrete TTL components.

As shown in Table 1, we start the use of FPGA by using a mystery design. The students have to reverse-engineer a design from the input and output they record. During this laboratory, they are given a short tutorial on how to program the FPGA. This tutorial is posted again on the laboratory webpage for reference at a later time [9]. For the rest of semester, all laboratories are simulated using the same techniques similar in fashion as the earlier experiment, but downloaded to the FPGA. Once the design is downloaded to the FPGA, switches and buttons are used to test their design. After the finishing of the experiments, during the sign-off process, we always speak to the students about the use of buttons and switches, more specifically, about the FPGA pin assignments. We are able to determine whether the concept came across by giving a one-hour hands-on quiz. We use the concept of “spoon feeding” a little and tell them what it is later. We have found that this is very effective, instead of just informing them of everything at the beginning of the laboratory.

The old method of design prototyping using discrete components teaches one a lot of design debugging skills. By migrating the prototyping platform to FPGA, most debugging will be accomplished in the Xilinx ISE. This debugging is carried out with the aid of the simulator. From our experience, the simulator can catch almost all of the design problems. Those designs that do not work usually use lots of complicated and unnecessary logic gates, which the synthesis tools have eliminated.

One problem that we found to be particularly troublesome is design change and update. The use of FPGA tends to promote a change, re-compile, re-synthesis, and coffee break type mentality! Because of ease of change, one accustoms to change and simulate instead of figuring out why a particular circuit doesn't work. This type of laboratory behavior is counter-productive as students spend much time changing a design to arrive at a solution instead of determining the right method of debugging. The laboratory instructor or assistant will have to pay attention when students' designs are not working and remind them to re-work the theory instead of blindly changing a design.

Due to the fact that the software being used is always the latest version (or one or two minor revisions from the latest release), the software being used in the laboratory tends to be problematic. There is no good way to resolve this issue, but there are ways to circumvent such issues in the student laboratory. Design software problems are a fact of life in the digital engineering design world. So, we tell the students that if clicking a particular button will result the design software crashing, a different approach needs to be developed. Many such problems usually occur during the first two weeks of the laboratory. Once they have learned the knack of using the design software, they can resolve most issues and engineer a work-around on their own!

4 Changes and Updates in Upper Division Courses

The changes taking place in the sophomore digital systems course have greatly improved some of our junior and senior courses. For those students taking the computer engineering option, we advise them to take the advanced logic design course as soon as possible. This course focuses on hardware description language. Currently, VHDL is taught in this course. Previously this course was taught with just a simulator (Modelsim). With the introduction of FPGA in the earlier course, the instructor in this course has been able to pick up from where the earlier course left off. The students know how to synthesize and test their designs with the same FPGA prototyping platform. This addition has added a lot of value to our advanced logic course. The students have also the addition to be exciting and useful.

The other impact is on our microprocessor course. The interface logic used to be built using discrete components. Due to this usage, we have had many microprocessor prototyping boards destroyed by students not checking with the laboratory instructors before installing their interface boards to the main microprocessor board. With the use of FPGA, the number of boards destroyed has decreased by using a ribbon cable. Since the students have learned the use of FPGA logic entry and programming, the microprocessor lecture and laboratory have been able to concentrate on the interfacing exercise, instead of wiring and debugging! With this approach, if there are interface problems in the FPGA, it can be fixed easily by changing the design in the software, followed by a re-compile, and re-load the FPGA.

The greatest impact is on our senior capstone design sequences. The students are able to propose and start to work on the design project right away. Local high-tech companies and research faculties contribute to these projects. This provides students with actual real-world problems instead of academic problems, which might not have real-world values!

5 Laboratory Setup & Management

Our laboratory is set-up with sixteen stations and can hold at most thirty-two students. Each station has a Pentium 4 2.8 GHz computer (with Windows XP) and test equipment. A few weeks before the start of each semester, we install and test all of the necessary software on one computer. Software test runs are very important and help reduce the number of problems that can occur on all sixteen machines! When that one machine has been installed and configured with the desired software and setup, we clone the hard disk image and download this image to the other fifteen computers. This guarantees consistency across all computers in the lab, and many problems can be fixed by downloading a new image.

We also find that storing design files on file servers can create problems. Xilinx ISE depends on file time-stamp to decide whether a certain operation requires re-run. If the time on the server and computer is out of synch, many undesirable issues can disturb the operation in the laboratory, and these problems are difficult to solve.

In our digital system laboratory, we group the students into team of two. Care is taken so that computer science students are matched up with electrical engineering students. This introduces synergy between students with different majors. We have found that this match-up works in most cases. During discrete component wiring, the EE students help the CS students understand the process. When we introduce a subset of VHDL in lecture and the use in laboratory, the CS students can help the EE students understand the design more quickly.

For each laboratory, we also limit what library, components the students can and cannot use. There are multiple arithmetic units in the library if they used those components, it would defeat the purpose of the laboratory, which might task them to design an arithmetic unit.

We utilize Dr. Felder's accountability form for team members to evaluate each other when the laboratory report is submitted [10]. This has helped us detect team problems early in the process and get the teams together to figure out what works and what does not. If they still fails to work after giving them another try, it is better to split the team. We have found keeping a team together in this situation will create more problems in the laboratory and more work for the instructor. We either let them work independently or create two three-person teams. Note that no matter what the lab instructor does, there is no way to solve all team problems! Remember, you are herding cats! Problems are going to show up here and there. If you work closely with them, problems can be determined earlier and more quickly.

6 What we will do differently

One of the changes to occur next semester is eliminating the pre-lab work. In the past, we allocated a portion of the laboratory as pre-lab work. We enforced the pre-lab by not letting those students finish the lab if they didn't even try to complete any of pre-lab. We have found that enforcing and judging the quality of pre-lab is very difficult. Instead, we will introduce the short concept quiz (five to ten minutes) at the beginning of each laboratory session. Since we synchronize lecture and laboratory, the students will have been provided with the background knowledge to a particular laboratory. The laboratory is to show the practicality of the theory. The short quiz will allow us to find out the strength of the background theory and students' preparation for the laboratory. We think that this will allow us to judge the quality of work and knowledge better and more fairly.

We found that there is only so much you can do with buttons and switches. One of our future projects is to create a computer base GUI with a hardware core that students can use to debug their design.

7 Conclusion

It is time to update the curriculum to what's being practiced in the engineering world. The implementation of FPGA in our student laboratory has benefited both the students and instructors. We

found that students get excited when they find out their laboratory exercises will involve FPGA. More extensive laboratory experiments will be added to our curriculum. For details regarding this transformation, please visit http://coen.boisestate.edu/smluo/smluo_courses.htm.

Acknowledgement

Our experiences wouldn't be possible without hardware and software donations from Altera and Xilinx. These donations are greatly appreciated.

References

- [1] David Hall, "Teaching Design Methodology and "Industrial Strength" EDA Tools in a First-Term Freshman Digital Logic Course," IEEE Transactions on Education, Vol. 41, No. 1, FEBRUARY 1998, pp. 45-49.
- [2] Kevin Nickels, Pros and Cons of replacing discrete logic with programmable logic in introductory digital logic courses, ASEE Conference, Session 2532, 2000.
- [3] Otto Fucik, Bodgan Wilamowski, Michael McKenna, "Laboratory for the Introductory Digital Course," ASEE Conference, Session 20383, 2000.
- [4] Iskandar A. hack, James Haberly, "Low cost FPGA Development Systems for Teaching Advanced Digital Circuits," ASEE Conference, Session 0047, 2001.
- [5] Rafic Backnak, Dulal Kar, Hesham Shaalan, "Digital Systems Laboratory for Teaching and Research," ASEE Conference, Session 1520, 2004.
- [6] Pong P. Chu, "A Small, Effective VHDL Subset for the Digital Systems Course," ASEE Conference, Session 1432, 2004.
- [7] Xilinx ISE, <http://www.xilinx.com>
- [8] Stephen Brown, Zvonko Vranesic, Fundamentals of Digital Logic with VHDL Design, 2/e, McGraw Hill, 2004.
- [9] Boise State University Digital Systems Laboratory Webpage, http://coen.boisestate.edu/smluo/smluo_courses.htm
- [10] Dr. Felder Active and Cooperative Learning, http://www.ncsu.edu/felder-public/Cooperative_Learning.html.