EECS 270 Laboratory Overview

Acknowledgments

During the summer of 1999, the EECS 270 laboratory was re-tooled to use advanced field-programmable logic chips and boards along with associated PC-based design software from the Xilinx and XESS corporations. The lab experiments were also completely re-designed to take maximum advantage of the new hardware and to enhance the students' digital design experience. Much of the credit for this effort goes to Ghassan Shahine who had assumed the position of Senior Engineer/ Lab Coordinator. Gordy Charichner was instrumental in helping with the CAEN installation of the Xilinx Foundation tools. Finally, credit is due the Lab GSIs and EECS 270 students who labored through the first offering of this lab in the fall semester of 1999.

EECS 270 introduces you to the exciting world of *digital logic design*. Digital devices have proliferated in the last quarter century and have become essential in just about anything we do or depend on in a modern society. Computers of all varieties are now at the heart of commerce, communications, education, health care, entertainment, defense, etc. Personal computers have become standard fixtures in most homes and even in preschools. But while the "computer" is the most visible digital computing device, it is by no means the only one. Embedded (invisible) digital controllers can be found in such diverse applications as automobiles, airplanes, elevators, telephones, televisions, cameras, and (fancy) kitchen appliances, to name just a tiny few. We are in the midst of a digital revolution that is transforming our way of life in ways far more profound than any other technological invention in the history of mankind. It is breathtaking.

This course provides you with a basic understanding of what digital devices are, how they operate, and how they can be designed to perform useful functions. It forms the foundation necessary for the more advanced hardware and software design courses in our curriculum. You will learn about digital design through a combination of lectures, homework, and a hands-on laboratory. The laboratory is an integral part of the course that shows how the theory of digital design learned in lectures is applied in practice to construct real digital systems. This overview document describes the equipment you will be using in the lab, the lab procedures, and a schedule of the seven experiments you will be performing throughout the semester.

1.0 Getting Started . . .

The EECS 270 laboratory is located in room 2341 in the EECS building. Each student will be assigned a workbench equipped with a Pentium III computer connected to a "logic board" that houses the actual digital system being designed (more on this later.)

Each student will have a “home” directory on drive Z, located on a common server, to store his or her design files. You need to physically come to the lab in order to establish your account and set a password on the EECS 270 server. You can then create subdirectories in your home directory to keep the design files for each lab experiment separate. Thus, if your uniquename is janedoe you should put your design files in:

Z:\janedoe\lab1	for the design files of experiment #1
Z:\janedoe\lab2	for the design files of experiment #2
. . .	
Z:\janedoe\lab9	for the design files of experiment #7

Note that you can create and verify your designs on your own PC or on the PCs in the College of Engineering Computer-Aided Engineering Network (CAEN) labs (except possibly for labs in the Media Union). To conduct the in-lab portion of your experiments, however, you need to copy your design files to the EECS 270 server. The easiest way to do this is to use the archive/restore commands in the Xilinx Foundation project manager: a) *archive* your project to generate a single compressed ZIP file, b) copy the file to the EECS 270 server, and finally c) *restore* the project. You can copy the ZIP archive using a floppy disk (if it fits) or using ftp (file transfer protocol); the ftp address of the EECS 270 server is **ee270-srv.eecs.umich.edu**. Note that the EECS 270 lab is not attached to the CAEN network; you must first come to the lab and set a password on the EECS 270 server before you can log in or use ftp.

2.0 Once Over Lightly

In your role as a digital logic designer you need to go through the following steps to accomplish your job:

1. Understand the required functionality of the design by reading and interpreting a set of English language specifications. These specifications are typically given to you by your customers; in your case, your customers are your EECS 270 instructors who will give you written specifications for each design in this lab.
2. Apply your knowledge of how to design digital systems (which you will hopefully acquire this semester) to translate your understanding of the required functionality into a “paper” design. Until a few years ago, that was literally true: you had to do manual design and draw the resulting system on paper. Today, however, the situation has improved considerably. You will be using special *design automation* software to help you with the mundane aspects of the design process and to let you focus your creative energies on the more challenging aspects. This design paradigm is commonly referred to as *Computer-Aided Design* (CAD for short) and the software programs used to assist you with design are frequently called CAD tools.

- In this lab you will be using the *Xilinx Foundation 1.5* CAD software. It is available on your lab computer as well as on the CAEN PCs. You can also have your own personal copy of the software to run on your own PC; it comes packaged, on two CD-ROMs, with the course textbook (John Wakerly, Digital Design: Principles and Practices, 3rd Ed., Prentice-Hall, 2000.)

  The Foundation software allows you to enter your design and to verify it by *simulating* it to make sure it is actually behaving as required by the specifications. If you detect errors in the design, you should modify and re-verify it before proceeding to the next step.

3. Implement the design by downloading it to the logic board attached to your lab computer. The main component on this logic board is a Xilinx field-programmable gate array (FPGA) integrated circuit (IC) chip consisting of thousands of logic building blocks that can be “wired” according to the pattern of your specific design by appropriately opening and closing tiny electronic switches. The miracle of software hides all of this detail; seconds after you issue the command to download your design, the FPGA becomes the hardware embodiment of that design.
4. “Play” with your design. The logic board contains a variety of switches (for input) and indicators (for output) to help you apply stimuli to and observe responses from the FPGA. You can even control the FPGA from the attached PC and display its responses on the PC screen.

FIGURE 1. The XS40 and XStend Board Layouts

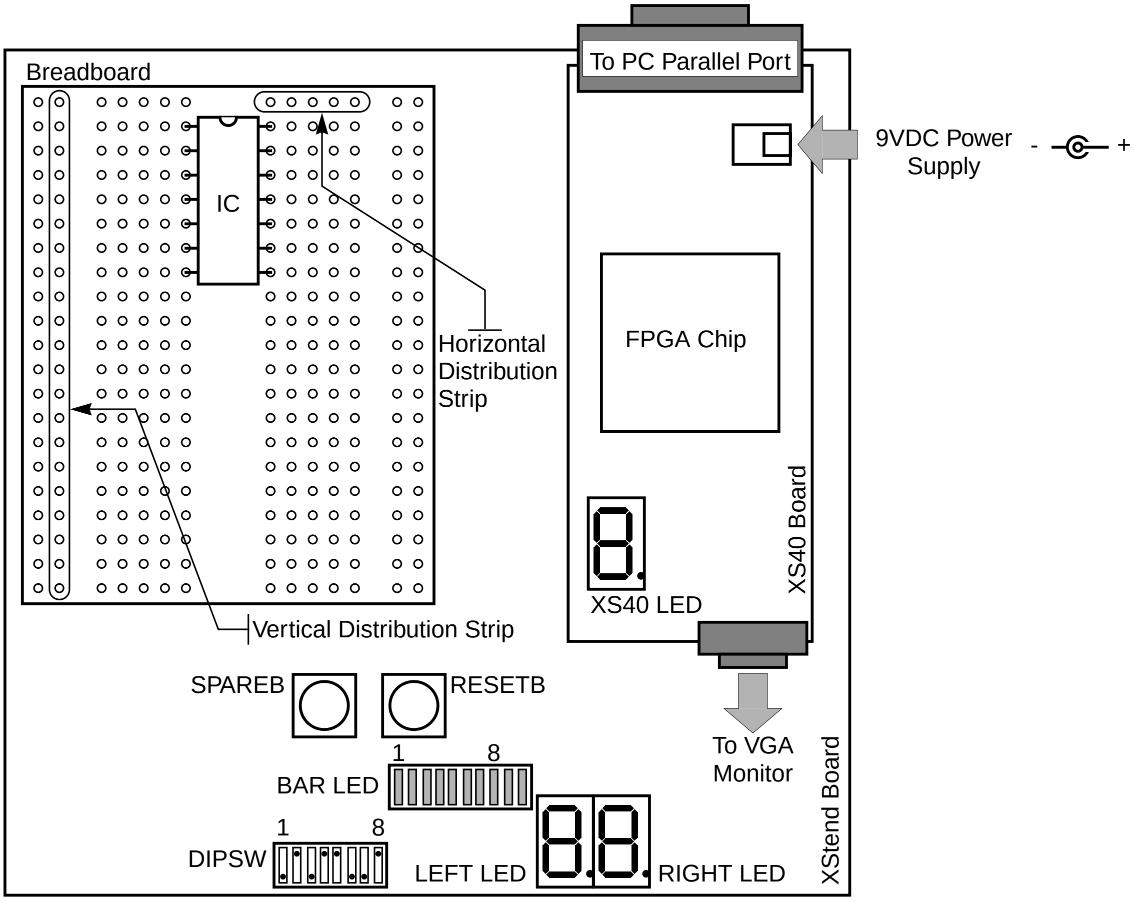
XS40 Board

XStend Board

3.0 The Hardware

As mentioned earlier, each lab workbench has a Pentium III computer and a programmable logic board. The logic board is made by the XESS corporation and consists of two distinct components (see Figure 1): the small XS40 board which holds, among

FIGURE 2. Resources on the logic board relevant to the EECS 270 lab



other things, the Xilinx FPGA chip, and the XStend extender board that serves as a carrier for the XS40, provides additional resources (switches and indicators), and has additional space for mounting a prototyping breadboard. Together, these two boards provide a rich environment for prototyping digital systems of substantial complexity. A wealth of information on these boards can be found at the XESS corporation web site (http:\\www.xess.com). For purposes of the EECS 270 lab, we highlight here only those features that are directly relevant to the design experiments you'll be performing.

Figure 2 shows a diagram of the logic board (approximately drawn to-scale) with the following resources indicated:

3.1 On the XS40 board

- The Xilinx XC4010XL FPGA chip which has a capacity of up to 20,000 logic gates, and is packaged in an 84-pin PLCC package.
- A jack to connect a DC power source.

- A DB-25 connector that allows you to attach the board to the parallel (printer) port of a PC.
- A seven-segment Light-Emitting Diode (LED) display (the *XS40 LED*)

3.2 On the XStend board

- Two seven-segment LED displays (the *LEFT* and *RIGHT LEDs*)
- An 8-segment bargraph LED display (the *BAR LED*)
- 8 DIP switches (the *DIPSWs*)
- 2 push button switches (*SPAREB* and *RESETB*)

3.3 The Breadboard

In some experiments you may be asked to use some off-the-shelf IC chips to augment the programmable portion of your design. The prototyping breadboard is provided for this purpose. The board provides a set of holes for plugging chips and wires in order to make the required circuit connections. The holes are connected internally within the board as follows:

1. The holes in the two left-most and two right-most columns are vertically connected. Holes in different columns are isolated. These columns should be used for the power and ground connections to the board.
2. The holes in the center of the board are arranged into two columns of horizontally-connected groups of five. Holes in different rows are isolated. The two columns are spaced apart in order to make it possible to plug an IC as shown in Figure 2. Connections to the IC pins can then be made by plugging wires into the appropriate row of holes.

4.0 The Software

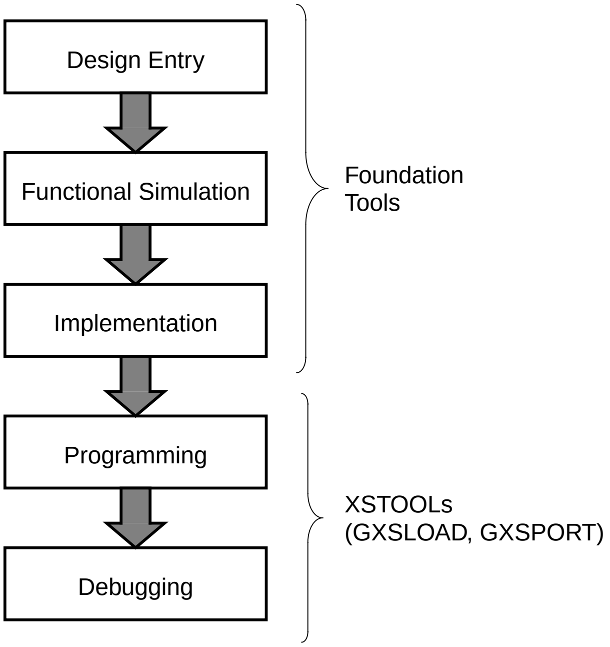
As mentioned earlier, you will be using the Xilinx Foundation CAD tools to enter, simulate, and implement your designs. We do not expect you to become an instant expert in using these tools. Rather, your skills are bound to improve over the course of the semester from repeated use and increased familiarity. The Foundation tools have extensive online documentation accessible from the Help menu. In addition, the class web site has links to frequently-asked questions (FAQs), and further help and tutorials prepared by Jan Van der Spiegel from the University of Pennsylvania. Keeping in mind the availability of such documentation, it is still helpful to elaborate the design steps outlined earlier (on page 2) to make the whole process more concrete and less intimidating.

The design flow consists of five main stages (see Figure 3 on page 7):

4.1 Design Entry

There are basically two methods for capturing a design: a) by drawing a schematic diagram showing the design’s components and how they are interconnected; and b) by writing a text document that describes the structure and behavior of the design in some appropriate notation. We will use both methods of design entry in this lab.

FIGURE 3. The five stage of digital design in the EECS 270 lab

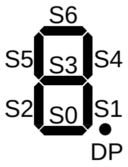


The Foundation *Schematic Editor* allows you to create your design by instantiating components from one or more libraries and by wiring them together. The types of components we will be using ranges from simple logic gates (in the first few labs) to more complex building blocks (such as arithmetic units and data steering logic.)

You can accomplish the same task by using the Foundation *HDL Editor.* HDL stands for Hardware Description Language. These languages are analogous to the programming languages used to write general-purpose software, but are usually augmented with features geared specifically for describing hardware. The HDL Editor supports three such languages: Verilog, VHDL, and ABEL. The first two have powerful constructs that are essential when you're designing complex hardware; they are the primary HDLs used by logic designers in the semiconductor and computer industries. On the other hand, ABEL, which stands for *Advanced Boolean Equation Language,* is a fairly simple language suitable for small designs; it will be more than adequate for our purposes.

Whichever method of design entry you choose (schematic or HDL), you will need to specify your design's inputs and outputs and how you would like them to be mapped to the switches and indicators on the logic board. The easiest way to accomplish this is to create a *User Constraint File* (UCF for short) that maps each of your named inputs and outputs to the pin number of the corresponding resource. This file can be conveniently prepared using the HDL Editor. For example, if you have an input named X that you would like to control with the third DIP switch, and an output Y that would like to observe as the middle horizontal segment of the RIGHT LED, you would create a UCF containing the following lines:

TABLE 1. Programmable pins on the XS40 and XStend boards

Inputs						Outputs							
PC Parallel Port		**DIP Switches***		**Push Buttons***		**XS40 LED**		**LEFT LED***		**RIGHT LED***		**BAR LED***	
Name	**Pin**	**Name**	**Pin**	**Name**	**Pin**	**Name**	**Pin**	**Name**	**Pin**	**Name**	**Pin**	**Name**	**Pin**
D0	44	DIPSW1	7	SPAREB	67	S0	25	LS0	3	RS0	59	DB1	41
D1	45	DIPSW2	8	RESETB	37	S1	26	LS1	4	RS1	57	DB2	40
D2	46	DIPSW3	9	 LED segment names		S2	24	LS2	5	RS2	51	DB3	39
D3	47	DIPSW4	6			S3	20	LS3	78	RS3	56	DB4	38
D4	48	DIPSW5	77			S4	23	LS4	79	RS4	50	DB5	35
D5	49	DIPSW6	70			S5	18	LS5	82	RS5	58	DB6	81
D6	32 (MD0)	DIPSW7	66			S6	19	LS6	83	RS6	60	DB7	80
D7	34 (MD2)	DIPSW8	69					LDP	84	RDP	28	DB8	10

* Active low

```
NET X        LOC=p9;
NET Y        LOC=p56;
```

In English, the above statements mean “map input X to pin number 9, and map output Y to pin number 56.” But how did we get these pin numbers? They are part of the specification of the logic board and can be found by browsing the documentation available on the XESS Corp. web site. To make your life simple, though, the pin numbers of all the programmable switches and indicators we will be using are summarized in Table 1. Note that the pin names in this table are arbitrary and are meant only as convenient labels to identify them in design specifications.

There is one final subtlety that should be mentioned to save you hours of trying to figure out why your design does not work. It turns out that not all pins are created equal. In particular, pins 32 and 34 are special-purpose FPGA *mode* signals and cannot be assigned in the manner shown above. Instead, to use them as inputs in your design you need to instantiate the special library symbols **MD0** (for pin 32) and **MD2** (for pin 34). This is easily done in the Schematic Editor.

4.2 Functional Simulation

One of the advantages of computer-aided design is that you can check out the correctness of your design before implementing it in hardware. Basically, you “exercise” your software model of the design (the schematic or HDL description) by applying various inputs to it and observing the resulting outputs. This process is referred to as *simulating* your design. You can use simulators to check many aspects of your design. For digital design, the two most common classes of simulators are a) *functional* simulators for checking the logical correctness of the design, and b) *timing* simulators for verifying that the design works at the required speed. We will focus almost exclusively on verifying functionality, and only briefly explore timing simulation to get some appreciation of the temporal behavior of logic circuits.

The Foundation tools provide a functional logic simulator that can be invoked after a design has been entered. The simulator allows you to define *stimulators* that you can apply to the inputs of your design, and to observe the resulting logic waveforms on the outputs. Any anomalies detected in this process should be diagnosed to determine their cause, and the design should be modified to eliminate them (by switching back to either the Schematic or HDL Editors). When you are satisfied that your design behaves according to its specification, you can move to the implementation stage.

4.3 Implementation

Implementing a design in hardware takes different routes depending on the nature of that hardware. The choices range from off-the-shelf IC chips that must be physically wired together, to custom-designed ICs that are manufactured in a semiconductor fabrication facility, to a whole range of programmable ICs with varying degrees of flexibility. In this lab, the primary implementation medium is the Xilinx XC4010XL FPGA IC chip; for some experiments we may also use off-the-shelf ICs and wire them up on the prototyping board.

An FPGA can be programmed to implement a particular design by downloading to it a “configuration” file that tells it how to set the state (on or off) of tiny electronic switches (transistors) so that the structure and function of the chip match those of the given design. The process of generating this configuration file for a given design is quite involved, but that need not concern us here; the Foundation tools conveniently take care of this. What we do need to know is that the resulting configuration file is a binary file with a “.bit” extension; it is also referred to as the *bitstream* file.

4.4 Programming

The actual programming of the FPGA involves downloading the bitstream file from the PC to the logic board. This is accomplished by invoking a utility program from the XESS Corp. called GXSLOAD. This should take only a few seconds.

4.5 Debugging

At this point you have a “live” design on the logic board and you are ready to check it by applying inputs using the DIP and push button switches and observing outputs on the various LEDs. You can also apply inputs to your design from the PC using the 8 bits of the parallel port as “software” switches. The GXSPORT utility (also from the XESS Corp.) is used for this purpose.

5.0 What We Need from You

Each experiment goes through three phases which are graded separately: pre-lab, in-lab, and post-lab. Your total grade for an experiment is computed as a weighted sum of these three grades, the weights generally being different for different experiments. The following subsections provide general descriptions of what you're expected to do in each phase. Experiment-specific pre-, in-, and post-lab requirements and their corresponding contributions towards the total grade will be clearly spelled out in each experiment write-up.

5.1 Pre-Lab Requirements

Pre-lab requirements are due at the beginning of the lab period in which you are scheduled to do the experiment (pre-lab requirements for two- and three-week experiments may be broken down into several installments.) It is essential that you read the pre-lab section of each experiment and do the necessary preparatory work before coming to the lab. In most of the experiments, the pre-lab consists of designing a logic circuit to meet a given specification and of verifying the correctness of the design. You are expected to carry out such design and verification using the Xilinx Foundation tools and to turn in a copy of your design (schematic or HDL) and its simulation results to your lab GSI. You also need to bring in your Foundation design files so that you can transfer them to your assigned lab PC to complete the in-lab portion of the experiment. You can bring your design files on a floppy or zip disk. Alternatively, you may use ftp to copy them over from your CAEN account (see page 2).

You should note that coming to the lab without having done all of this preparatory work puts you at a serious disadvantage; there just is not enough time during the lab period for you to start your design from scratch.

5.2 In-Lab Requirements

During each three-hour lab period, you are expected to complete the hardware portion of the experiment. For most experiments this means merely downloading your design bitstream file to the logic board; for some experiments, you may also need to wire some additional chips on the breadboard. In either case, you must demonstrate the operation of the circuit to your lab GSI. If the circuit is not behaving according to the given specifications, you are expected to diagnose the causes of the erroneous behavior and to make appropriate design modifications. Your in-lab work is considered to be incomplete until your lab GSI certifies the correct operation of your circuit.

5.3 Post-Lab Requirements

Besides being a whiz logic designer, it is essential that you develop your technical communication skills in order to clearly explain and document your designs. We will help you develop these skills by requiring you to submit a written lab report describing what you did in each experiment and providing answers to specific questions related to the experiment.

The report is due one week after completion of the experiment. A late penalty of 10% per day will be assessed for overdue reports; reports more than one week late will be assessed a 90% penalty.

Your report should be prepared with a word processor using 10-point double-spaced text, and should contain the following sections:

1. **Cover Sheet**: this sheet is available as a Portable Document Format (PDF) file that can be printed from the course home page.
2. **Design Narrative**: This is the main part of the report. You should describe here how you interpreted the specifications to arrive at your particular design. You should note any degrees of freedom that you exploited in the specifications to simplify your implementation and/or enhance the design's functionality. You should also note any anomalies you encountered and how you dealt with them. The purpose of this portion of the lab report is to document your thinking process; to explain why you chose to do things a certain way. Your ability to express yourself this way will prove quite invaluable in your future engineering career.
3. **Design Documentation**: Here you should attach the final (corrected) design files including, as appropriate, schematics sheets, HDL programs, and simulation traces.
4. **Answers to specific post-lab questions**: In some experiments, you will be asked to provide answers to a list of lab-related questions as part of your post-lab assignment.

6.0 Lab Schedule

Week	Dates	Experiment		
		Number	Title	Points
1	1/5 – 1/7	No Lab		
2	1/10 – 1/14	1	Introduction to the Xilinx Foundation Design Environment	50
3	1/17 – 1/21			
4	1/24 – 1/28	2	Timing and Delay	100
5	1/31 – 2/4	3	Combinational Logic Design I: A Sprinkler System	100
6	2/7 – 2/11	4	Combinational Logic Design II: A Simple Calculator	150
7	2/14 – 2/18	No Lab		
8	2/21 – 2/25	5	Latches and Flip-Flops	100
9	3/6 – 3/10	6	Finite State Machine Design: A Vending Machine	200
10	3/13 – 3/17			
11	3/20 – 3/24	7	Design and Programming of the M270 Computer	300
12	3/27 – 3/31			
13	4/3 – 4/7			
14	4/10 – 4/14	No Lab		