ECEN 320 Laboratory Manual

Dr. Michael Wirthlin
Department of Electrical and Computer Engineering
Brigham Young University
Provo, UT, 84602
wirthlin@ee.byu.edu

January 2, 2015

Version 1.3

Contents

Foreword

I have had the opportunity to teach the *Advanced Digital Design* course at BYU for many years. This is a demanding course that requires significant effort by both the faculty instructor, the teaching assistants (TAs), and the students. During this period, students have expressed frustration in the amount of time it takes to properly learn this material and complete the laboratory assignments. One of my goals in teaching this course over the years is to reduce student frustration, time, and unnecessary work involved in these laboratory assignments while simultaneously making sure students properly achieve the student learning objectives. The laboratory assignments described in this manual are my attempt to reduce the frustration students have experienced in this lab over the years. I also try to make the labs enjoyable, interesting, and fun.

This manual is a collection of the laboratory assignments used in the BYU ECEN 320 class. I have solicited feedback on these labs from students in previous semesters in an attempt to improve the labs. Students in my class have been forthcoming and painfully honest in their assessment of these laboratory assignments. I would like to thank my previous students for their feedback on these labs and their patience in completing labs that were at times confusing, difficult, and downright frustrating. I believe these laboratory assignments are much more effective learning exercises based on student feedback. As with any teaching material, these labs can be improved even further. Any feedback you have on these labs will be appreciated and considered.

One problem students often face when completing labs like this is that the FPGA circuit they create does not work properly when downloaded on the FPGA. Even though the student has thoroughly tested the design, the design does not work and the student is forced to debug the circuit using cumbersome and time consuming tools like logic analyzers. Students also create many design iterations during this debug process which each require multiple circuit implement phases (translate, place, and route). In some cases, students never get their design to work.

There are a number of common causes for this problem and I have attempted to highlight these causes within this manual. It is important to follow the instructions in the manual carefully to avoid these common pitfalls. Students that skim through these instructions and miss some of these key suggestions may waste a significant amount of time.

One of the strategies I use to reduce the chance of non-functioning FPGA bitfiles is to provide a carefully designed testbench for each laboratory assignment. These testbenches were created to test your circuit against the requirements of the laboratory assignment and catch common student design errors. It is essential that your circuit pass each testbench test cases before trying to implement your circuit on an FPGA. Students are often tempted to skip these tests and move quickly to the download phase to finish the lab early. Avoid doing this as you will likely spend more time on the lab than someone who carefully simulates his or her design with the testbench.

You are not expected to write complex testbenches as part of this class. While testbench design and other related HDL verification methods are an essential part of HDL design, this class is not a class on HDL verification. Other classes are available to learn these skills. You will be required, however, to perform basic test sequences and write simple simulation scripts. The testbenches and simulation scripts will provide you with the environment you need to identify circuit bugs and fix them.

I hope you enjoy these labs and that the labs will help you learn the fundamental principles of digital circuit design.

Professor Mike Wirthlin
wirthlin@ee.byu.edu
Brigham Young University
Provo, Utah

Introduction

The best way to learn how to create digital circuits with a hardware description language is to practice by creating real circuits in a laboratory setting using commercially available tools. This manual provides a number of laboratory exercises that are designed to help you apply the fundamentals of digital logic design using VHDL. The laboratory exercises were designed with your course lectures in mind and are scheduled to occur during the semester after you have learned the material necessary to complete the lab. These laboratory exercises are progressively more complex and each laboratory exercise requires the understanding of the material covered in the course lectures.

This manual includes several laboratory exercises that you will complete during the course of the semester. Each laboratory exercise is carefully designed to help you learn specific skills and apply the material you learn in the classroom. Each laboratory exercise is organized using a similar format. The format of the laboratory exercises is summarized below.

Learing Objectives

The learning objectives of the lab are summarized to help you recognize what you are expected to learn after completing the laboratory. This will help you understand what tasks, skills, and new design concepts you are expected to learn before you begin the lab.

Exercises

Each laboratory contains a number of exercises that must be completed to pass of the laboratory assignment. These exercises build upon each other and should be completed in the order in which they are presented. Most of the exercises will require you to answer questions, complete a partial design task, or perform a design simulation. The exercises are organized in a manner that breaks up the laboratory assignment into smaller, more manageable tasks. Each exercise has a specific learning outcome and subsequent exercises rely on the learning outcomes from previous exercises.

Questions

Each laboratory contains a number of questions that must be answered to complete the lab. The answers to these questions must be supplied online (Learning Suite in our case). The questions will be given within the various exercises of each laboratory assignment. It is very important that you answer these questions in the order they are given and before completing subsequent exercises or questions. These questions were carefully chosen to help you learn the skills material you need to complete the lab. Students who skip over the questions and proceed directly to the laboratory exercises will not be prepared to complete the exercise. Questions will be provided in the laboratory material as follows:

Question: <Question to be answered>

If you do not know how to answer a question, consult with a teaching assistant, other students, or the faculty instructor. It will be very difficult to complete the laboratory exercises properly without having a good understanding of the answers to the questions in the lab.

Screencasts

Many of the laboratory exercises contain supplemental material in the form of electronic "Screencasts". Screencasts are video segments that demonstrate important concepts that are difficult to describe in written form. These screencasts will demonstrate important concepts visually along with an audio discussion. These screencasts are helpful and will save you significant time in completing the laboratory if you carefully watch them. You will need a way to listen to the audio portion of the screencast and it is recommended that you bring a set of headphones to the laboratory. These screencasts are large files (10s of megabytes) and were optimized for video quality, not file size. These screencasts are best viewed while on campus and in the digital design lab.

When you are given a screencast for a laboratory exercise, you will see the following annotation in the laboratory write-up:

Screencast: <Name of Screencast>

You can access the electronic file for the given screencast in the online course resources (LearningSuite).

File Upload

Most laboratory assignments require you to upload files that you create during the laboratory exercises. These files will be reviewed and graded by the teaching assistant. When a file upload is required, it will be shown in the laboratory material as follows:

Upload: <File or resource that must be uploaded>

Use the appropriate online learning software (Learning Suite in our case) to upload your laboratory files.

Resources

You will need access to a number of resources outside of this manual to complete each laboratory. These resources may include VHDL files, online manuals, or other files that are necessary for completing the lab. The laboratory exercises will list these resources and provide HTML links were possible. Access to these resources will be provided online in Learning Suite.

Personal Exploration

Most of the laboratory activities you complete are carefully described in the laboratory write-up. While these activities are carefully written to help you achieve the learning objectives of the lab, learning is most effective when it is self-directed and motivated by your personal interests. For each laboratory assignment you will be required to spend some time completing a "personal exploration" activity. Personal exploration activities are learning activities that you choose and interest you. Each laboratory assignment will include several suggestions for a personal exploration activity. You will be required to document your personal exploration activity in your laboratory write-up.

Pass-Off

To complete each laboratory assignment, you must pass off the lab in person with a TA. The requirements for each pass-off will be described in the laboratory manual. Most laboratories will require you to demonstrate the proper function of your digital circuit to the TA.

Chapter 1

Xilinx ISE Design Suite

The purpose of this laboratory is to introduce you to the Xilinx ISE Design Suite and the various tools you will need to compile, simulate, and implement your VHDL designs on the Nexys2 FPGA board. Because you will use these tools throughout the semester it is essential that you learn how to use them well and become familiar with its user interface.

Learning Objectives

After completing this laboratory, you should be able to:

- Create a new project within the Xilinx ISE design suite using the proper FPGA device
- Simulate a VHDL architecture within the Xilnix ISim simulator and view the simulation in the waveform viewer
- Create simple simulation scripts for controlling a VHDL simulation
- Simulate a VHDL architecture using a testbench
- Synthesize your VHDL architecture and implement the architecture on the FPGA
- Download the bitfile of the VHDL architecture on the FPGA

Exercises

Exercise 1 – Opening Project Navigator and Creating a New Project

The Xilinx ISE is a suite of tools used to create, simulate, and implement designs for Xilinx FPGAs. One of the tools in this suite is Project Navigator. This tool provides a graphical user interface for all of the tools within the ISE suite that you will use in this class. The Project Navigator will help you manage individual design projects for each of the labs described in this manual. Within Project Navigator you will describe your VHDL circuits, compile your VHDL circuits, synthesize your circuits, and implement your circuits on the target FPGA. Because the Project Navigator will be used for each laboratory exercise, it is important that you become familiar with the user interface and the tools that it contains. Do not worry too much about the details—we will discuss these tools in more detail in future labs.

The Xilinx ISE tools have been donated to us through the Xilinx University Program—we appreciate the generosity of Xilinx for providing these tools to the university. All of the laboratory assignments and training materials were created with Xilinx ISE Version 14.1[1]. For this first exercise, you will learn how to open the Project Navigator and create a new project. Before completing this exercise, watch the following screencast to learn the various options you have for accessing the ISE tools.

Screencast: Accessing the Xilinx ISE Tools

Once you have access to the ISE tools you are ready to begin the laboratory by creating a project. To complete this exercise, watch the Project Navigator screencast provided with the lab materials. You are encouraged to follow along with the screencast to create your project for this lab. You will add your newly created VHDL file to your project.

[1]The tools that you will use in the laboratory are version 14.6. While there may be some small differences between the training materials and the tools you are using it should be relatively straightforward to adpat these materials to the newer tools.

Screencast: Creating a New Project

As a part of this exercise, you will need to type in a new VHDL file and add this file to project. Type in the VHDL from Listing 2.1 from the textbook (the listing is copied below for your convenience).

```
--==============================
-- Listing 2.1 Even detector
--==============================
library ieee;
use ieee.std_logic_1164.all;

-- entity declaration
entity even_detector is
   port(
      a: in std_logic_vector(2 downto 0);
      even: out std_logic
   );
end even_detector;


-- architecture body
architecture sop_arch of even_detector is
   signal p1, p2, p3, p4 : std_logic;
begin
   even <= (p1 or p2) or (p3 or p4) after 20 ns;
   p1 <= (not a(2)) and (not a(1)) and (not a(0)) after 15 ns;
   p2 <= (not a(2)) and a(1) and a(0) after 12 ns;
   p3 <= a(2) and (not a(1)) and a(0) after 12 ns;
   p4 <= a(2) and a(1) and (not a(0)) after 12 ns;
end sop_arch ;
```

Figure 1.1: Listing 2.1 from the Textbook.

Upload: Submit your Listing 2.1 VHDL file

Throughout the course you will need to create new projects for each of the laboratory assignments that you complete. Figure 1.2 provides a screen shot of the project settings you need for your future laboratory assignments.

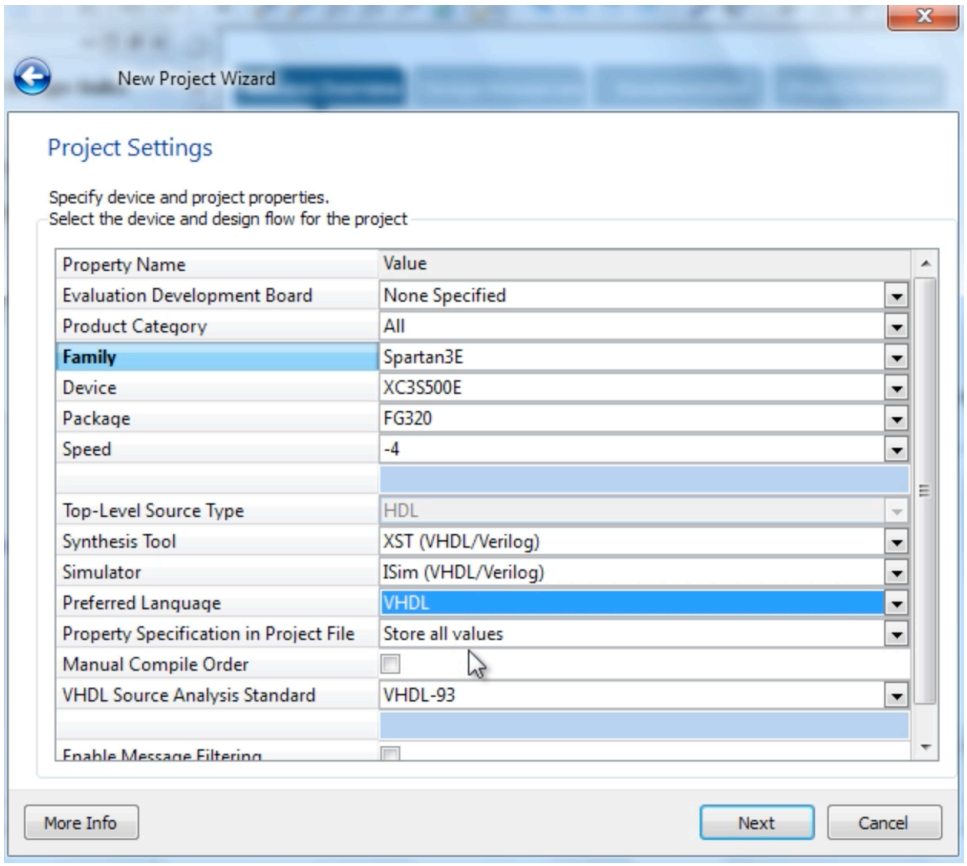

Figure 1.2: Suggested Project Settings for Laboratory Assignments.

Exercise 2 – Simulating a VHDL Architecture

Logic simulators are an essential tool for any digital design project. Simulators provide visibility into all parts of your circuit and are necessary for verifying the correct operation of your circuit before synthesis. If there are problems in your circuit, it is far easier to identify them with a simulator than to find the problem in hardware on the FPGA. We will use the Xilinx ISim simulator throughout semester to simulate the laboratory assignments and perform various experiments. To succeed in this course, it is essential that you learn to use ISim well and become familiar with its operation and function. While you will be given basic instruction on the use of ISim, you are strongly encouraged to learn more about the many features, capabilities, and tools that are a part of the ISim framework.

To learn how to run the ISim simulator with your new project, watch the following screencast and follow along to learn how to setup and run a basic simulation on Listing 2.1.

Screencast: ISim HDL Simulator

Question: Use the waveform viewer to determine the *longest* time it takes for the output 'even' to change after any one of the inputs has changed (note that the time to change may depend on the input sequence - choose the transition time that is the longest).

There are many commands and GUI buttons you can use to aid in your circuit simulation and verification. You are encouraged to review the user interface and various commands in the ISim command users guide. You will need to access this guide to answer several questions in the laboratory write-up. You can access this guide in the Help menu: Help→ISim Users Guide. Chapter 10 of this manual contains the details of the commands used in ISim. Reference this chapter when answering the following questions on the laboratory write-up.

Question: How do you zoom into a region within the waveform viewer? (describe the button(s) you press)

Question: How do you "zoom out" so you are looking at the whole waveform?

Question: How do you determine the value of a signal at a specific time period?

It quickly becomes tedious to manually change the input behavior by typing force commands and run commands. Simulation scripts can be written to simplify this process. We create simulation scripts in ISim in the form of ".tcl" (pronounced "tickle") files. You can type all of the commands needed to perform a simulation in a ".tcl" file script and execute the commands by running the .tcl file script.

Scroll back through the "Console" window and see what the command arguments were that did your force and run commands. You can also see the commands you have run by running the "history" command line function. Create a .tcl file in a text editor and type in all the commands needed to fully simulate this circuit. Make sure you save it in the working directory of the project. You can verify the current working directory by typing "pwd". Here is an example of the opening lines of such a .tcl file:

```
# initialize the simulator with the "restart" command
restart
# initial value for a
put a 000
# run the simulator for 100ns
run 100 ns
# place a different value for a and run
put a 111
run 100ns
# More commands for forcing and running
```

After creating your .tcl simulation script, run the script by using the "source" command:

```
ISim> source listing2_1.tcl
```

Verify that your .tcl script tests all input values for this circuit (there are eight different values of 'a').

Upload: Include a copy of your .tcl file in your laboratory write-up

Exercise 3 – Simulating with a VHDL Testbench

A testbench is often used to simulate a circuit more fully than is possible with commands on the command line and a .tcl script. A testbench is a behavioral model that simulates the environment of the circuit and checks the circuit outputs for proper behavior. Listing 2.7 from the textbook is a testbench written in VHDL to test the operation of the even detector architecture of Listing 2.1.

Download: list_0207_even_tb.vhd

After downloading this file, add it to your current project.

Instructions for adding a file to a project were given in the screencast associated with the previous exercise. When adding a testbench to the project, make sure the file is tagged as "Simulation" under the association column of the "Adding Source Files" dialog box. Setting the association to Simulation instructs the project navigator to use this testbench file for simulation only and not for implementation on the FPGA circuit.

Compile this file and simulate this VHDL architecture within ISim. Determine how long this simulation needs to run in order to complete all tests within the testbench. Keep your waveform of this simulation available, as you will need it for your lab pass-off.

Question: How long does it take for the testbench to test all of the cases of the even detector? Note that the testbench will repeat the test sequence. The question is asking how long it takes to test all of the input cases for the first time.

Now that you have seen the testbench run and test your code, take a look at the code for the testbench itself. Open the testbench file and inspect its overall structure. Even though the commands and syntax may appear foreign, hopefully you can get a feel for what the testbench looks for as it tests your code. In addition, it is important to note that code used to create a testbench uses techniques and structures that cannot always be used to create a circuit. As you examine testbench code throughout this course, realize that this style of code should not be imitated when creating your own hardware design.

Question: What do lines 27 through 42 do?

Question: What is the function of the if statement starting at line 50?

At the bottom of the testbench code (line 65) we see that there is a report made if some failure is detected. To see this report execute, remove the first "not" from line starting with "p1" in the code you copied from listing 2.1. The line should now look like this:

```
p1 <= ( a(2)) and (not a(1)) and (not a(0)) after 15 ns;
```

With this logic removed from the circuit, the test bench should now fail. Re-run the simulation and look at the console window at the bottom. Notice that there were two failures during the test. Look at the time reported for these failures and find these occurrences within the simulation.

Question: At what times did the simulation fail and what were the "test_in" inputs at these times?

Question: How do these test cases correspond to the changes made to the code?

Question: What lines within the if statement starting at line 50 in the testbench test for these cases?

Exercise 4 – Synthesizing a VHDL File

Now that the VHDL file has been verified with the logic simulation tools, you are ready to synthesize the HDL into a circuit. You will use the XST HDL synthesis tool to map the VHDL into a circuit netlist composed of gates and look-up tables. The HDL synthesis tool will analyze your VHDL code and "synthesize" a working digital circuit on the target technology. The synthesis tools perform many important functions such as logic synthesis, logic minimization, and technology mapping.

Watch the following screencast and follow-along to synthesize the even detector into a logic netlist. After completing the screencast, answer the following questions.

Screencast: VHDL Synthesis

Question: Review the synthesis report and determine the "Maximum combinational path delay" (search for this term in the "Synthesis Report").

Question: Copy the equation found within the look-up table (LUT) of your circuit (see "View Technology Schematic").

Exercise 5 – Technology Mapping and Download

The final exercise is to map the synthesized logic netlist onto the FPGA we are using in the lab. This is a complex process that involves logic mapping, placement, and routing. When this process is complete, we will generate a configuration bitfile that is used to program the FPGA with the circuit. We will then download the configuration bitfile associated onto the FPGA.

Watch the following screencast and follow along to generate a bitfile and download the bitfile to your board. Once you have downloaded the circuit onto the board, you are ready to pass-off your lab.

Screencast: Bitstream Generation

One of the steps described in the screencast is to set the "FPGA Start-Up Clock" options to "JTAG Clock" found under the "Generate Programming File" Process step. This step must be applied for *every* project you create throughout the semester. A copy of the screen used to change this setting is shown in Figure 1.3 for your reference. You can refer to this figure for this setting rather than watching the screencast again.

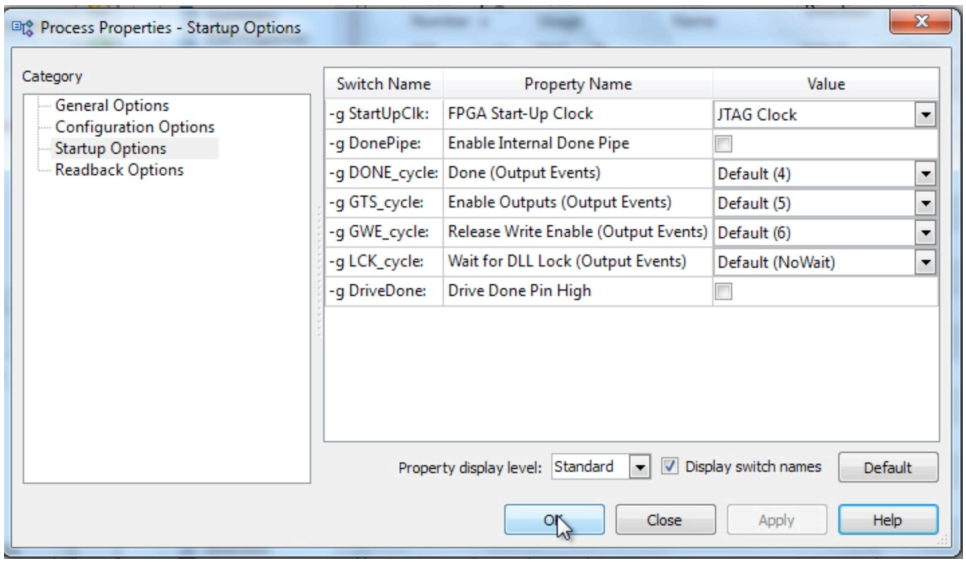

Figure 1.3: Bitstream Generation Settings (FPGA Start-Up Clock).

To complete this exercise, you will need to create a "User Constraints File" (.ucf file). The following text can be copied into your .ucf file. The details of this file will be described in a future laboratory.

```
# User constraints file for Listing 2.1
#
#   A user constraints file is used to specify the pin locations of the
#   top-level signals. The format for a pin location mapping is as
#   follows:
#
#      NET "<port name>" LOC=<PIN LOCATION>
#
#   Note that the keywords NET and LOC are case sensitive. Make sure
#   you capitalize these keywords when creating .ucf files.
#
#   This constraints file will map the first three switches (SW0,SW1,
#   and SW2) to the a input and the first LED (LD0) to the output,
#   even.
#
# Attach a(0) to the switch 0 (SW0)
NET "a<0>" LOC=G18;
# Attach a(1) to the switch 1 (SW1)
NET "a<1>" LOC=H18;
# Attach a(2) to the switch 2 (SW2)
NET "a<2>" LOC=K18;
# Attach even output to the LED 0 (LD0)
NET "even" LOC=J14;
```

Personal Exploration

Spend a few minutes exploring other features of the Xilinx Project Navigator. Summarize your exploration activities in blackboard. Some suggestions for exploration include:

- Identify a few new ISim commands that you can use in future VHDL simulation activities
- Experiment with the ISim GUI and learn the function of the icons
- Explore some of tabs and menus of ISim and ISE to familiarize yourself with the interface (you will spend a lot of time with this tool this semester)
- Explore some of the report files that are generated
- Modify your VHDL design to do something different and display a different logic function on the LEDs

Pass-Off

Demonstrate your testbench simulation and your working circuit on the board to a TA. Have your waveform and board ready when the TA comes to pass you off.