EE183 LAB TUTORIAL

Introduction

You will be using several CAD tools to implement your designs in EE183. The purpose of this lab tutorial is to introduce you to the tools that you will be using, Xilinx Foundation with Synopsys FPGA Express. You will learn how to use Foundation for verilog design and simulation.

An Overview of Foundation

Foundation is the program that manages your design in 183. It has, among other things, the schematic capture elements that you used in 121. In 183, we only use it to for simulation and to generate the bitfile that is downloaded to the FPGA. The verilog development can be done in any text editor (there are especially nice tools for developing in emacs), but Foundation does come with an hdl editor that is adequate. The actual synthesis is handled by FPGA Express, which is bundled into Foundation. This tutorial walks you through a simple design which exercises all of these tools. In addition, you will create a file to download onto an actual FPGA to demonstrate functionality in hardware.

Honor Code

Please note that EE183 is an individual project class. Discussion and group learning is encouraged, but sharing of schematics is a violation of the honor code.

Starting Foundation

Foundation is located on the local disk of every machine. In order to run Foundation simply open the Project Manager shortcut on the desktop. If there is not a shortcut, from the start menu go to programs->Xilinx Foundation 4->Project Manager

Projects

To get started, you need to create a project. You must have a Project file for each of your labs. Project files have the contain information about the libraries you are using and help you manage your entire project.

- A window should pop up asking what project you wish to open. Select "Create a new project." If this window doesn't pop up, go the menu bar and select File→New Project.
- A dialog will pop up asking you to name your project. Call it "tutorial". It will also give you a choice between the schematic and hdl flows. Choose hdl.

Where do I work? Store files?

Please read the computing FAQ for information on where files should be stored. You should work off the local users directory (e.g. C:\USERS\MATT\) on any particular machine (make one if it isn't there), but do not leave your files on a machine after you logout. Always save your files to your home directory on the server, and delete the local copy of your work.

- After you have selected the directory, name and flow for your project click OK and your project should be ready for use.

Design Entry

This tutorial assumes you use the hdl editor associated with Foundation.

Starting things off

Follow the steps above to create a project and or set the current project to one you created previously. To create a new module, click on the picture labeled HDL within the Design Entry panel (it's in the upper right hand corner of window.

A top level schematic of the tutorial design is shown at the end of the tutorial. You will end up with a design that works like that. Do not attempt to understand its purpose. There isn't any. It's just a tutorial.

Creating a new module

- You can either use the HDL design wizard to set up your module or do it yourself. It is up to you.
- For the first module we will do a 4 bit, 2 input and gate. Enter the following module:

```
module and_4bit(a, b, z);
input [3:0] a, b;
output [3:0] z;

assign z[3:0] = a[3:0] & b[3:0];
endmodule
```

- Save the module in a file named and_4bit.v. It is a good habit to put each module in its own file.
- You can check the syntax of your module by selecting Synthesis ->Check Syntax. This is like a practice run of the compilation process.
- Congratulations! You have designed your first block. Note that something as simple as this usually doesn't warrant its own module. This has just been done by way of demonstration.
- You now need to add this file to your overall project. Go to Project Manager and select Project-> Add source file(s). Find your new verilog file and add it in. It is now visible to any other files in your design.

Adding a parameterized module

- A nice shortcut is the use of parameters. Let's use these to create a module that will synchronize an external input of arbitrary width. Create the following module:

```
module synchronizer (in, out, clk) ;

parameter SIZE = 1;

input [SIZE-1:0] in ;
input clk ;

output [SIZE-1:0] out;

wire [SIZE-1:0] x;

dff #(SIZE) dff_1(.d(in[SIZE-1:0]), .clk(clk), .q(x[SIZE-1:0]));
dff #(SIZE) dff_2(.d(x[SIZE-1:0]), .clk(clk), .q(out[SIZE-1:0]));

endmodule
```

- This module uses one of our 183 library flops so don't forget to add that file to the design as well.
- You should use a synchronizer on each input to your design so this module should come in handy.

Big-picture

You should now be able to place the two new blocks you have created on the top level of the design (which is another verilog module). The last two lines of the synchronizer example demonstrate how to instantiate modules.

One-pulse and control components

You should be able to create the one-pulse and control component from the schematics attached at the end of the tutorial. The rest of the design can be implemented in the top level module. Follow the schematics and code it up.

Done?

You should be about done with the design entry. Did you make any mistakes? Well, we'll have to *verify* that.

Synthesize the design

The next step is to synthesize our design so it can be simulated. Make sure all your source files have been added to your project then click on the synthesis button.

A dialog should pop up. Select your top-level module (probably called tutorial in this case). Also set the target FPGA. The family is Spartan II, the Device is 2S100TQ144, and the speed grade does not matter.

To make the simulation easier to understand we want to preserve the hierarchical design. Click on the Synthesis Settings, SET button. Then make sure there is a check mark next to Preserve hierarchy. Without this only the top-level ports can be referenced by name.

When all this is set click Run. Hopefully it should go off without a hitch. If there are any errors you probably have syntax errors and need to correct them.

Simulating the Design

The design is now done. The next step is to simulate it to see if it works correctly. To do this you need to simulate your design. Simulation involves several steps.

Opening a Logic Simulator window

- Click on the simulation button in project manager. It will open up a new window with your design ready to be tested

Command File/Stimulus

You can stimulate your design by manipulating the waveforms by hand, but it is recommended that you create a text file that has all the commands you want and then run the command file. To use a command script go to Tools ->Script Editor. It will bring up a program that manages command files. Enter the following command file, you don't have to type in the comments, i.e. lines that start with "|". You should perform each function in the order specified, or your circuit may not simulate properly.

```
| MyFirstCommandFile.cmd
|
| You refer to every net in your circuit by the name you gave it.
| To make it easier to refer to the IN signals, define a "vector"
| of the signals and then refer to the vector name.
vector data data3 data2 data1 data0
vector out out3 out2 out1 out0
| Set the base you'd like to see the numbers in, e.g. bin, hex
radix bin data out
| View the following waveforms
watch select button clk data loada loadb out
| Setup the clock period
stepsize 1000ns
| Set the inputs.
| For individual nets use either "l" or "h" to set lo or hi
l select button
| For vectors or buses, use assign <vector> <value>
assign data 0000
| Define which signal is the clock, clock <clock signal>
clock clk 0 1

| Step 1
| Load reg_a
assign data 0110
sim 2000ns
h button
sim 2000ns
l button
sim 2000ns
| Step 2
| Load reg_b
assign data 0011
sim 2000ns
h button
sim 2000ns
l button
sim 2000ns
| Step 3
| Toggle between AND and OR outputs
h select
```

```
sim 2000ns
l select
sim 2000ns
| Step 4
| Load reg_c
h button
sim 2000ns
l button
sim 4000ns
| Repeat steps 1 thru 4 to try other tests
```

Where am I Now?

This tutorial was meant to take you through the basics of Foundation. You have entered a fairly simple verilog design. You synthesized a netlist, which allowed you to simulate your design, using logic simulator. You setup a command file to stimulate the design and view the output waveforms.

Inputs and Outputs

In general, the Xilinx software will choose the pin assignments to make best use of the routing resources (this usually will change with each run of the implementation step). You will probably want the ports to come out on specific pins to make wiring easier.

The pads can be set to specific pins by creating a constraint file. Simply create a file with the same name as your project with the extension .ucf (in this case "tutorial.ucf) in your project's directory. Foundation should automatically use it when picking pins. The syntax for this file is as follows:

NET "button" LOC = P80;
NET "out<0>" LOC = P83;

This will assign the Button port to pin 80 and Out[0] to pin 83.

Next Step

The next steps would be to synthesize your design for a specific FPGA. To do this simply click on the Implementation button in the project manager flow. This will generate a bitfile that can be downloaded to the FPGA. Check out the webpage for details on this.

What Do I Turn-in?

Copies of all your source code, command file, and waveforms. Include waveform plots of the following:

1. 1011 stored in REG_A; 1001 stored in REG_B; and the result of ANDing stored in REG_C.
2. 0101 stored in REG_A; 0010 stored in REG_B; and the result of ORing stored in REG_C.

The waveforms should at least show all primary input and outputs as well as the contents of REG_A, B, and C.

Next

Don't forget to do lab 1.

ADVANCED TOPICS

Projects

As projects are moved from machine to machine, links to libraries and project directories may be changed. Be sure to check the project manager periodically to see what directory it has set as your project directory and which libraries it's pointing to.

Translating a design to FPGA

There are a few steps involved in getting your design ready for an FPGA.

1. The tool will map all of the logic components you have used to the FPGA specific blocks. i.e. it will map your gates to look up tables within the FPGA logic cells.
2. Partition: Foundation tool will partition the design into large functional blocks.
3. Place: All of the logic cells will be placed, i.e. they will be assigned to a specific, physical logic cell.
4. Route: The cells will be routed together.

Warning Messages

Throughout this entire tool-chain, you'll receive a lot of info, warning, and error messages. Please pay attention to them. There are symbol warnings that tell you about floating attributes, schematic warnings about unconnected nets, etc. In general, there should be no red-colored lines in the Console window.

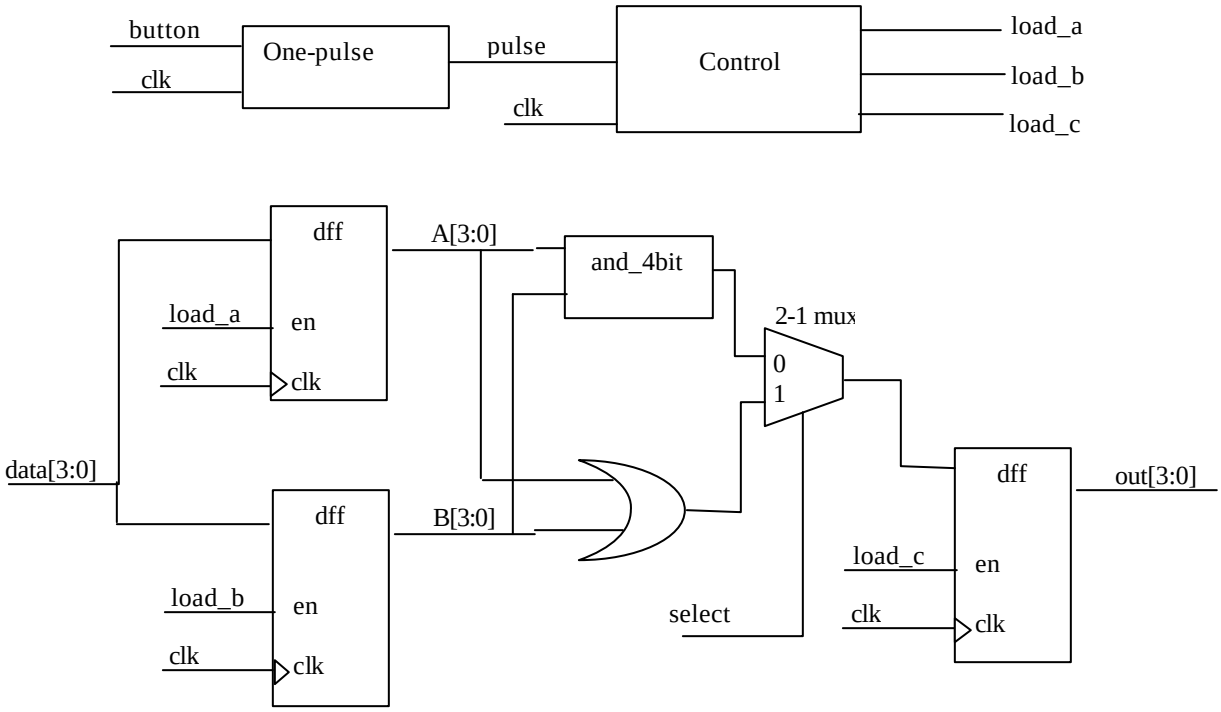
More Help

All of the tools have accompanying help files. Also, there are several copies of the Foundation reference floating around in the lab.

Tutorial schematics and state machines

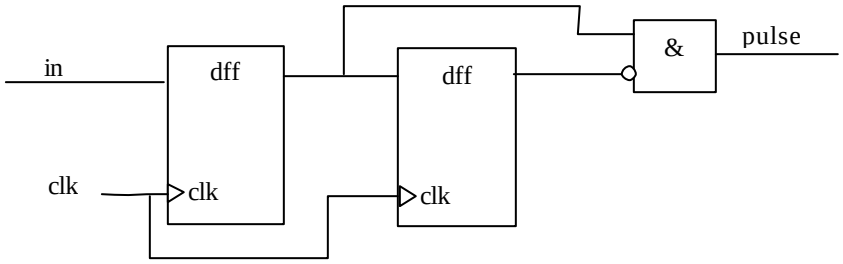
Top level

Inputs: button, select, data[3:0], clk
Outputs: Out[3:0]



One pulse

Inputs: in, clk
Output: pulse



Transition table for Control

Pulse	S1	S0		Next_S1	Next_S0	Load_A	Load_B	Load_C
0	0	0		0	0	1	0	0
1	0	0		0	1	1	0	0
0	0	1		0	1	0	1	0
1	0	1		1	0	0	1	0
0	1	0		1	0	0	0	1
1	1	0		0	0	0	0	1
X	1	1		0	0	0	0	0