

ECEN 449: Microprocessor System Design
Department of Electrical and Computer Engineering
Texas A&M University

Prof. Sunil Khatri
TA: Monther Abusultan
(Lab exercises created by A. Targhetta / P. Gratz)

Laboratory Exercise #1
Using the Integrated Software Environment (ISE)

Objective

The aim of this week's lab exercise is to familiarize you with the Xilinx FPGA design flow via ISE by stepping through a simple example. We will use ISE to create hardware, which lights up the LEDs on the XUP board depending on the status of the on-board DIP switches. The hardware will be entirely implemented in FPGA fabric and described in ISE using Verilog. After completing the aforementioned example, you will be expected to implement a simple counter and jackpot game on your own with the knowledge gained from the first part of this lab.

Procedure

1. Launch the ISE Project Navigator and create a new design project.
 - (a) Open a terminal window in the CentOS workstation and run the following commands:

```
>source /softwares/Linux/xilinx/10.1/ISE/settings64.csh
>ise
```

The former sets up the environment in order to run ISE, while the latter actually starts the ISE toolset.

- (b) Once in ISE, select **File** → **New Project**
The New Project Wizard opens (Figure 1).

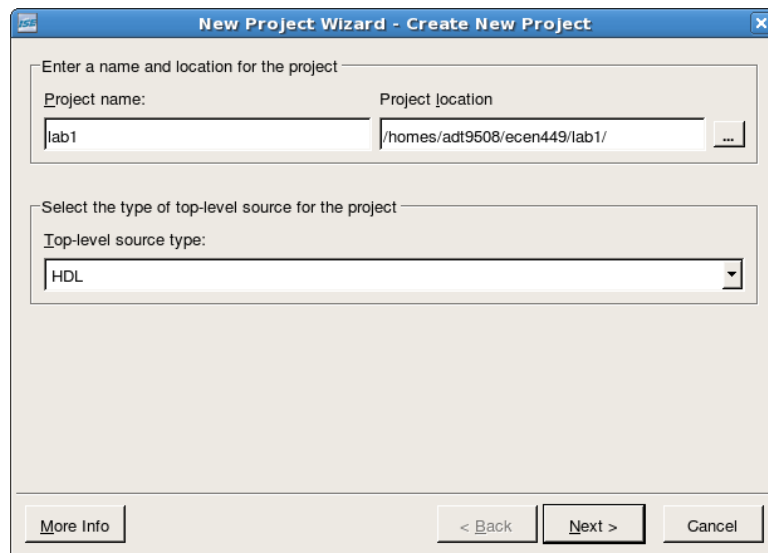


Figure 1: Create New Project

Select a Project Name (ex. lab1) and a Project Location (ex. /ecen449/lab1 in your home directory). Then chose HDL as the Top-Level Source Type and click Next.

- (c) Next, the ‘Device Properties’ window appears (Figure 2).

Set the device properties to the following:

Device Family: Virtex5

Device: XC5VLX110T

Package: ff1136

Speed Grade: -1

Additionally, set the following design flow properties:

Synthesis Tool: XST (VHDL/Verilog)

Simulator: ISE Simulator (VHDL/Verilog)

Preferred Language: Verilog

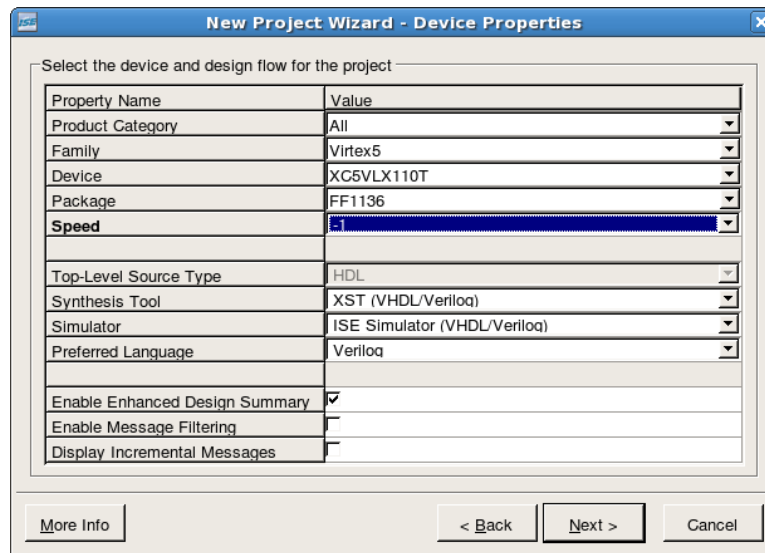


Figure 2: Device Properties

- (d) Next, the 'Create New Source' window appears (Figure 3). Click on 'New Source' to create a new source file.

In the 'Select Source Type' window (Figure 4), select Verilog Module as the source type. For 'File Name', enter "switch.v". Ensure the location points to your lab1 project directory and hit 'Next'. See figure below:

Next, the 'Define Module' window appears (Figure 5). This allows us to define the ports for our hardware module. Xilinx will then auto generate part of our source file based on the information provided. Specify a port called "SWITCHES". Set its direction to "input", set the Most Significant Bit (MSB) to 3, and set the Least Significant Bit (LSB) to 0. Specify another port called "LEDS" and set direction to "output", MSB to 3, and LSB to 0. This will create a 4-bit input port, which will connect to the on-board DIP switches, and a 4-bit output port, which will connect to the on-board LEDs.

Click 'Finish' on the Summary window and 'Next' on the 'Create New Source' window to create source file. Likewise, hit 'Next' on the 'Add Existing Sources' window as we have no additional source files to add at the moment. Finally, review the information in the 'Project Summary' window and hit 'Next' to create project.

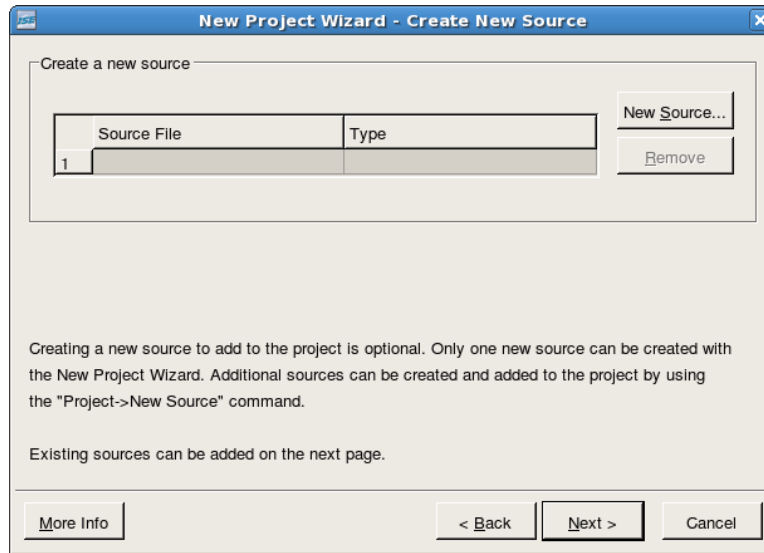


Figure 3: Create New Source

2. At this point, ISE has created a new project and source file for us to modify. We will now create code to provide the desired functionality (i.e. to turn on LED[i] when Switch[i] is high).
 - (a) From the ‘Sources’ window, open the “switch.v” file. It will contain a Verilog module with the port declarations described in part 2(d).
 - (b) Above ‘endmodule’, add the following line of code:
assign LEDS[3:0] = SWITCHES[3:0];

The resulting verilog module should be as follows:

```

module switch (SWITCHES, LEDS);
  input [3:0] SWITCHES;
  output [3:0] LEDS;

  assign LEDS[3:0] = SWITCHES[3:0];

endmodule

```

- (c) Click on **File** → **Save** to save your changes.
3. After making our simple modifications to “switch.v”, we want ISE to perform a syntax check.

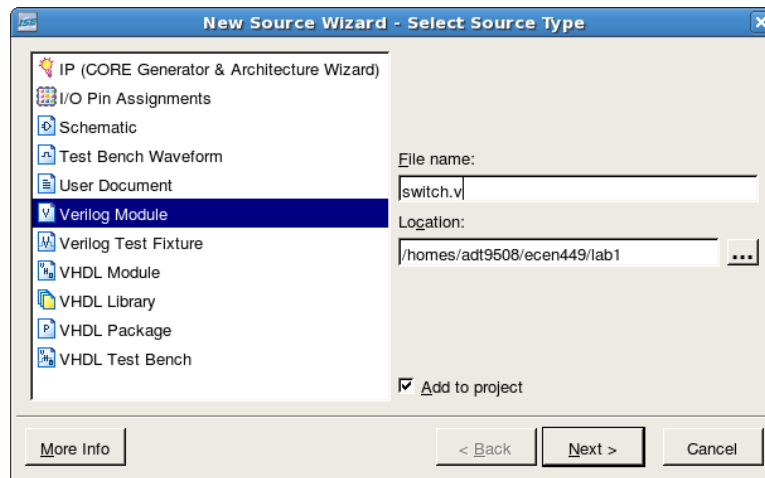



Figure 4: Select Source Type

- (a) Click on the “switch.v” file in the ‘Sources’ window. The ‘Process’ window lists a sequence of steps that need to be completed before the hardware described in the “switch.v” file can be run on the target FPGA.
 - (b) Double click on ‘Check Syntax’. A  should appear next to ‘Check Syntax’ once the check is complete. If ISE finds any syntax errors, please correct them before continuing.
4. We now need to create the User Constraint File (.ucf) containing the location of the DIP switches and LEDs on the XUP Board. The ucf file will be used to connect signals described in the Verilog file (LEDS[3:0] and SWITCHES[3:0] in our case) to pins on the FPGA, which are hardwired to the LEDS and DIP switches on the XUP board.
 - (a) Use your favorite text editor to create a new file called “switch.ucf” in your lab1 project directory and copy the following text into the new file:

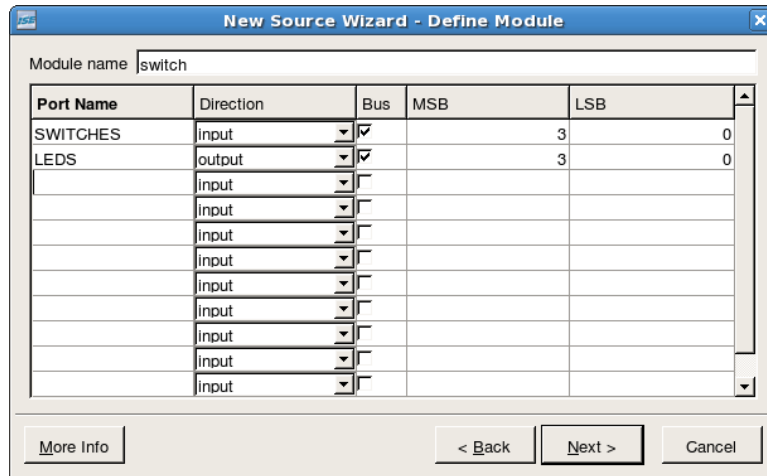


Figure 5: Define Module


```



NET "SWITCHES[0]" LOC = "U25" ;
NET "SWITCHES[1]" LOC = "AG27" ;
NET "SWITCHES[2]" LOC = "AF25" ;
NET "SWITCHES[3]" LOC = "AF26" ;

NET "LEADS[0]" LOC = "H18" ;
NET "LEADS[1]" LOC = "L18" ;
NET "LEADS[2]" LOC = "G15" ;
NET "LEADS[3]" LOC = "AD26" ;

```

Note that the above pin assignments were taken from the XUPV5 user manual accessible from the course website.

- (b) After saving "switch.ucf", return to ISE and right click on the Xilinx FPGA part number within the 'Sources' window. Select 'Add Source' and navigate to "switch.ucf" in your lab1 directory. In the next window, you should see a  next to "switch.ucf". If so, click 'Ok' to add it to the project.
5. At this point, both "switch.v" and "switch.ucf" should show up in the 'Sources' window. It is now time to let the tools create the hardware configuration for our specific FPGA and download the generated configuration onto the XUP board.
 - (a) Select "switch.v" in the 'Sources' window. In the 'Process' window, click on 'Generate Programming File'. This will run all the processes necessary to create a bitstream, which can be downloaded into the FPGA. Running these processes may take several minutes; progress is in-

indicated by the spinning  icon and output to the console. When a process completes, a  appears next to the appropriate process name. The steps that are run before the Programming File is created are: synthesis of the Verilog, mapping of the result to the FPGA hardware, placement of the mapped hardware, and routing of the placed hardware.

- (b) We now need to download the bitstream to the FPGA on the XUP board. Turn on the power to XUP board. In 'Processes' window, double click on 'Configure Target Device'. This will open iMPACT (the tool which performs the bitstream download, also referred to as device configuration). A message warning us of the fact that no iMPACT project file exists will appear. Click 'Ok' to continue. On the next screen, ensure 'Configure devices using Boundary-Scan (JTAG)' is selected and hit 'Finish'.
- (c) After scanning the board, iMPACT should find five devices (Figure 6).

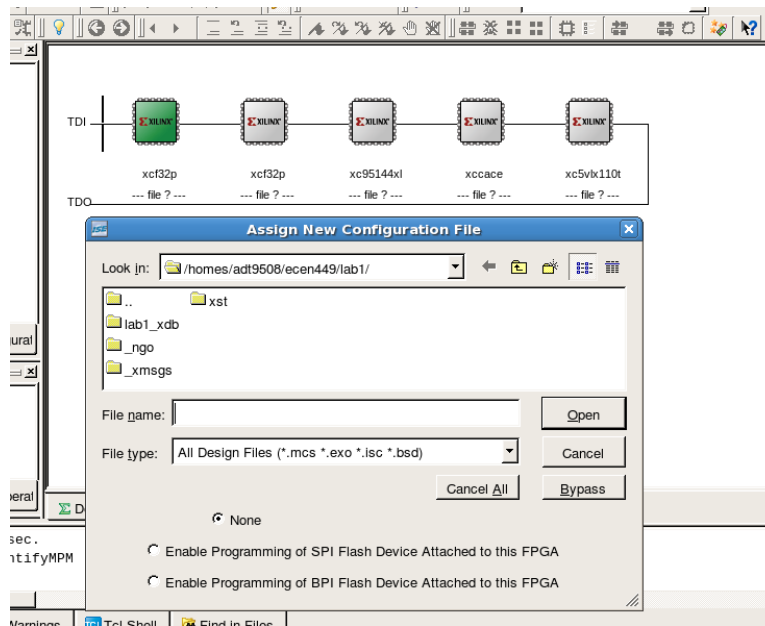


Figure 6: JTAG Chain

The first two, both labeled “xcf32p”, are the board’s non-volatile PROMs. The third, “xc95144xl”, is a CPLD on the XUP board. The fourth, “xccace”, is the System ACE controller (for compact flash). The fifth, “xc5vlx110t”, is the actual FPGA. We only need to program the FPGA. You should now see the ‘Assign New Configuration File’ dialog box. Do not assign configuration files to the first four devices; skip over them by clicking ‘Bypass’.

- (d) The fifth device, the FPGA, is programmed with a .bit file (the bitstream to be downloaded).

iMPACT should automatically show your project directory and a file, “switch.bit” (which was produced after ‘Generate Programming File’). Select this file and click ‘Open’, then click ‘OK’ on the next dialog box to assign the configuration file to the FPGA. A ‘Device Programming Properties’ window will appear. Click ‘Ok’ to continue with the default settings.

- (e) In the iMPACT main window, right-click on the FPGA (xc5v1x110t) and select ‘Program’. The Executing Command box should appear, followed by a blue “Program Succeeded” message.
6. At this point, the FPGA should be programmed to function as described in “switch.v”. Verify this fact by toggling the DIP switches 0 through 3 and observing LEDs 0 through 3. Demonstrate your progress to the TA.
7. Implement a 4-bit counter using the LEDs (You do not need the switches for this exercise). The count value should update approximately every 1 second. Use the up and down push buttons on the XUP board to control the direction of the count. For example, when the up button is pressed, the counter should count up. Likewise when the down button is pressed, the counter should count down. When neither button is pressed, the count should remain the same. Demonstrate this operation to the TA upon completion.

Hints:

- Do not forget to add clock and reset as input pins to your verilog module.
- Skim through the user manual for the XUPV5 board to determine the pin assignments for additional signals. The user manual may be found on the course website.
- After modifying the UCF to include the new ports (and removing unused ports) append the following text to the UCF:

```
NET "CLOCK" TNM_NET = "CLOCK";  
TIMESPEC "TS_CLOCK" = PERIOD "CLOCK" 10 ns HIGH 50% ;
```

Note: The above UCF lines provide ISE with timing constraints necessary to ensure proper design operation and assume your signal for clock is labeled ‘CLOCK’.

- The clock frequency is 100MHz. Updates to the LEDs at this rate will not be visible, and thus, the incoming clock must be divided. Think back to your introductory digital logic class to determine how to divide a clock!
8. Design a “Jackpot game which works as follows: The LEDs glow in a one-hot fashion, which means that the LEDs are turned on one at a time in a sequential manner. Get the transition to happen as fast as you can, while you can still make out which LED is on at a given of time. Assign a DIP switch to each of the LEDs. At any point in time, if you turn on the switch corresponding to the glowing LED, you win a Jackpot and all the LEDs start glowing!

Deliverables

1. [6 points.] Demonstration of working portions of the lab.

Submit a lab report with the following items:

2. [6 points.] Correct format including an Introduction, Procedure, Results, and Conclusion.
3. [6 points.] Commented Verilog files.
4. [2 points.] Answers to the following questions:
 - (a) How are the user push-buttons wired on the XUP board (i.e. what pins on the FPGA do each of them correspond to and are the signals pulled up or down)? You will have to consult the board schematics for this information.
 - (b) What is the purpose of an edge detection circuit and how should it have been used in this lab?