Lab 1: Introduction to Xilinx ISE Tutorial

This tutorial will introduce the reader to the Xilinx ISE software. Step-by-step instructions will be given to guide the reader through generating a project, creating a design file, compiling the project, and downloading the design to an FPGA board.

Xilinx software

The Xilinx ISE 4.2i software will be used in this text. All menus structures and screen shots are taken from the ISE 4.2i version. This tutorial will NOT deal with the Foundation family of software.

FPGA development board

The Digilent Digilab IIE board (available from www.digilentinc.com) will be used in this tutorial, however, other boards utilizing a Xilinx FPGA can easily be substituted. The Digilab IIE board contains a Xilinx Spartan-IIE device (XC2S200E) with the equivalent of approximately 200,000 gates. The Digilab IIE board contains a minimal number of prototyping devices, but it contains 6 40-pin I/O headers for attaching daughter boards with additional functionality.

The Spartan-IIE devices are identical to those in Xilinx's Virtex-E family of FPGAs. In fact, the product model stored on the device itself is the Virtex-E model number (for the FPGA on the Digilab board the on-chip device name is XCV200E).

Example Project

This tutorial will step the reader through the VHDL entry, compilation, and downloading of a simple inverter. To test the design, the

pushbutton on the Diglab IIE board will be used as the input to the inverter and the LED on the board will be used as the output.

L1.0.1 VHDL Design Entry

The Xilinx ISE tools allow the design to be entered several ways including graphical schematics, state machine diagrams, VHDL, and Verilog. This tutorial will focus on VHDL entry, but the other methods are similar and can be easily explored once the reader is comfortable with the ISE software.

Starting a project

Start the Xilinx ISE Project Navigator. Choose **File ⇒ New Project**. A popup dialog box will appear. Enter **tutor1** for **Project Name**. For the **Project Location**, select the directory where the project will be stored (i.e., Z:\XProj\tutor1) for your project.

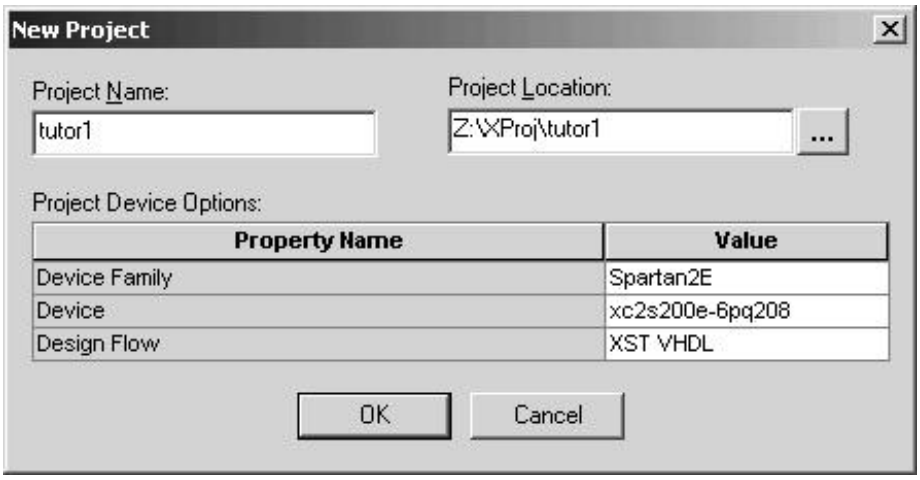

Figure L1.1: The New Project dialog box is used to enter the project and FPGA device information required to create a new project.

Next, the FPGA that will be used with this project needs to be specified. The compilation process is device specific, so the complete device specification (including package type) must be entered when creating a new project. Look on the top of the FPGA you are using. The device model, package type, and speed grade will be printed on it. The Spartan-IIE device used on the Digilab IIE board is shown in Figure L1.2. For the Digilab IIE board, set the **Device Family** to **Spar-**

tan2E. The device model is XC2S200E, the package type is PQ208, and the speed grade is 6, so set the **Device** to **xc2s200e-6pq208**. Finally, set the **Design Flow** to **XST VHDL** and click **OK**.

Figure L1.2: The Xilinx Spartan-IIE FPGAs have the model number (XC2S200E), package type (PQ208), and speed grade (6) printed on the top of the chip.

A project can be retargeted (i.e., compiled for another FPGA device) after the project is created by selecting the **xc2s200e-6pq208-XST VHDL** item in the **Module View** window and then choosing **Source** ⇒ **Properties**.

Creating a design file

Choose **Project** ⇒ **New Source...**. A popup dialog box will appear. Select **VHDL Module** from the list on the left and enter **top_level** for the **File Name**. Make sure the **Add to Project** option is checked and click **Next**.

The next screen of options allows you to enter the input and output signals for this module. Leave the default values for **Entity Name** and **Architecture Name**. For this project, only one input and one output are needed. For the input, enter **pb** for the **Port Name** and **in** for the **Direction**. Likewise, for the output, enter **led1** and **out** respectively. Since both of these signals are only one bit, the **MSB** and **LSB** fields can be left blank. If a multi-signal bus was being specified, then the bus width would be specified in these fields. Click **Next**. A summary of the new file to be created will be displayed next. Verify that all of the information is correct. Use the **Back** button to return to any

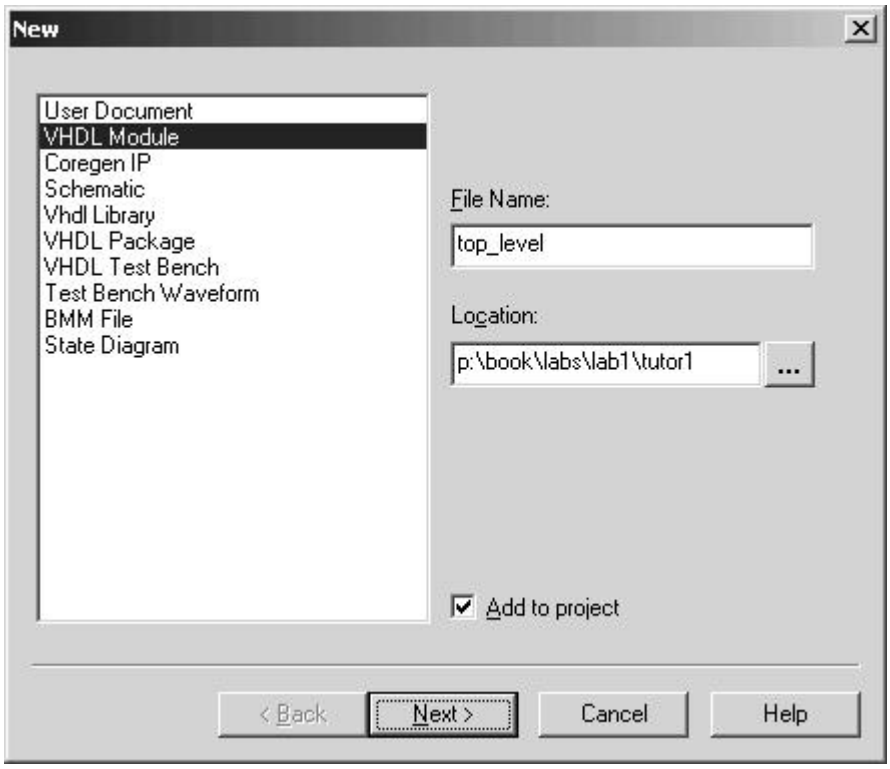

Figure L1.3: Specify the name and type of the new source file in the New Source dialog box.

previous screen and make corrections. When you have verified that everything is correct, click **Finish**. The design file top_level.vhd will be created, added to the current project, and opened for editing as shown in Figure L1.5.

Getting to know the Project Navigator

Take a few minutes to familiarize yourself with the Project Navigator layout. As shown in Figure L1.5, there are four window panes in the default layout. All text files (design files, report files, etc.) are displayed in the main window on right-hand side of the screen. In the upper left-hand pane, you may select the **Module View**, **Snapshot View**, or **Library View**. Click on each of these tabs and look at the information provided. Then click on Module View to return to the default. **Module View** displays the design files in the current project and shows the relationship between the files. Double-clicking on any design file in the

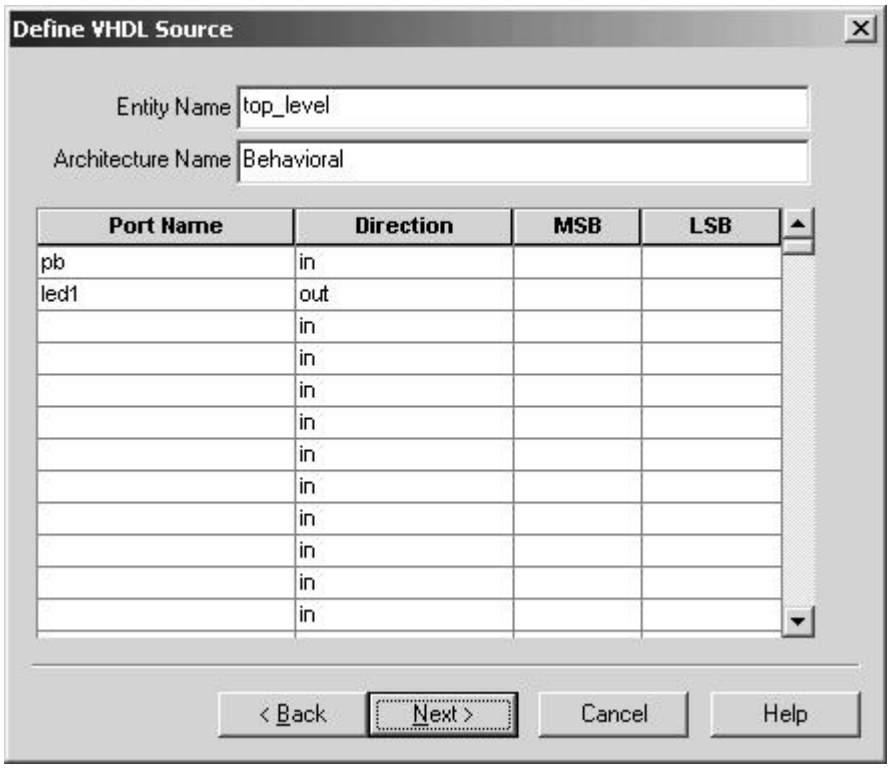

Figure L1.4: Enter the input and output signals for the new VHDL module in the VHDL Source Definition dialog box.

Module View window will open up that file in the main window.

The **Process View** window is displayed in the lower left-hand side. This window provides access to the pin assignment file, simulator, compiler settings, compilation reports, compiler controls, and FPGA programmer. Click on the + sign beside each item in the **Process View** window to view the different options available.

Finally, the bottom pane is the **Console** window. All text output from the Project Navigator, including warning and error messages, are displayed here.

Writing the VHDL model

Look at the top_level.vhd file that you generated earlier. The first four lines of code indicate which VHDL libraries and packages are needed for this VHDL module. IEEE.STD_LOGIC_1164,

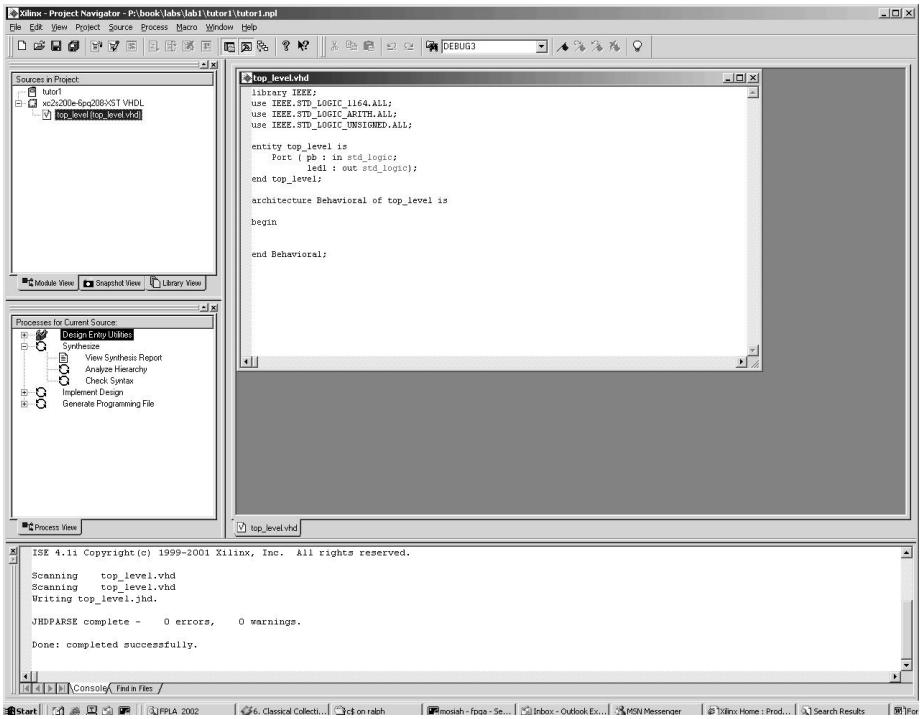

Figure L1.5: The Xilinx Project Navigator is the heart of the Xilinx ISE software. Editing, compiling, and programming can all be accomplished through the Project Navigator. This screen shot shows the default layout which includes the Module View, Process View, Console, and Editing/Viewing windows.

IEEE.STD_LOGIC_ARITH, and IEEE.STD_LOGIC_UNSIGNED are the basic VHDL packages from the IEEE library that define the keywords and operators for the language. These should be included at the top of all VHDL files that you create, unless you have a specific need to use a different library and/or package.

The IEEE library also defines a IEEE.STD_LOGIC_SIGNED package. It should be used instead of the UNSIGNED package if the VHDL module is handling signed values. The only difference between these two libraries will be in the functionality of the comparison operators (less-than, greater-than, less-than-or-equal, greater-than-or-equal).

Following the library and package declaration statements you will find the **Entity** statement. This section defines the name of the module and the inputs and outputs of the module. In this case, **pb** is defined as the single input (**in**) and **led1** is defined as the only output (**out**).

Next, you will find the **Architecture** statement. This is the section where the behavior or functionality of this VHDL module is defined. Enter **led1 <= NOT pb;** on a line between **begin** and **end Behavioral;**. This line of VHDL code indicates that an inverter is to be placed between the input **pb** and the output **led1**.

Choose **File** ⇒ **Save** to save these changes.

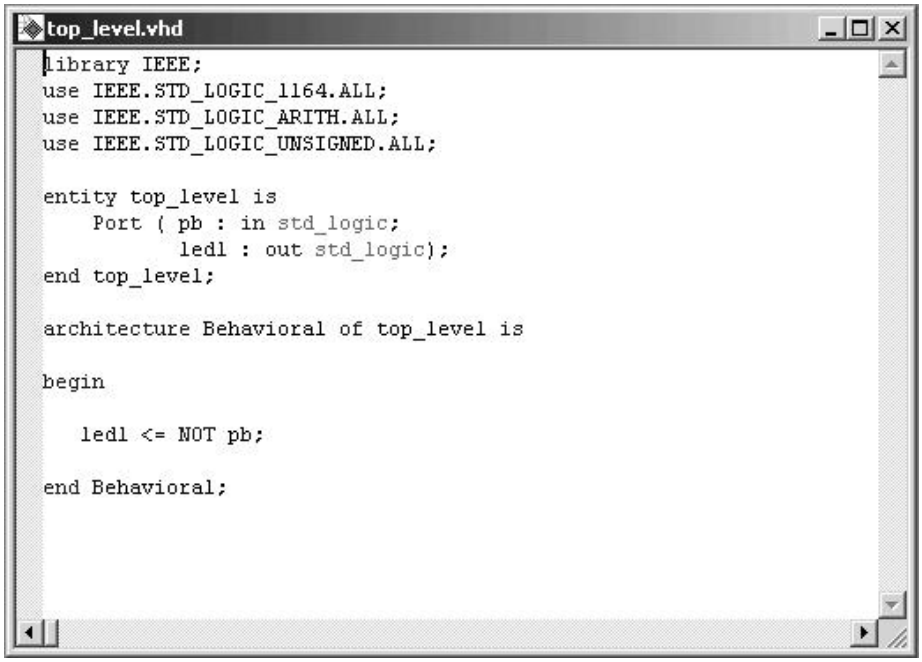

Figure L1.6: The completed VHDL code for the example project, tutor1.

Checking the VHDL syntax

Make sure the **top_level.vhd** item in the **Module View** window is highlighted. If it is not, then single click on it to make it active. In the **Process View** window, choose **Synthesize** ⇒ **Check Syntax** (expand the **Synthesize** item and then double-click the **Check Syntax** option). If the VHDL code in the current file is syntactically correct, then a green check mark will appear next to the **Check Syntax** option in the **Process View** window. If any syntax warnings or errors are found, a yellow exclamation point or red "X" respectively will appear. In the case of warnings or errors, a description of the problem will be displayed in the **Console** window. If multiple problems exist,

you may have to scroll the text in the **Console** window to see all of the comments.

L1.0.2 Compiling the Design

If the project were compiled now, the input and output signals would be assigned to random I/O pins on the FPGA by the compiler. To assign the signals to the specific pins for the pushbutton (pin 64) and LED (pin 69) on the Digilab IIE board, these user constraints must be entered.

Entering the pin assignments

Make sure the **top_level.vhd** item in the **Module View** window is still highlighted. If it is not, then single click on it to make it active. Then, in the **Process View** window, choose **Design Entry Utilities** ⇒ **User Constraints** ⇒ **Edit Implementation Constraints File**. The user constraints text file will open in a separate Notepad window. The default file contains a large number of comments documenting the syntax for entering the various types of user constraints. These comments may be deleted. Add the line **NET pb LOC = P64;** to force the signal **pb** to be assigned to pin 64, and add the line **NET led1 LOC = P69;** to force the signal **led1** to be assigned to pin 69. Save the user constraints file (**File** ⇒ **Save**) and close the Notepad window.

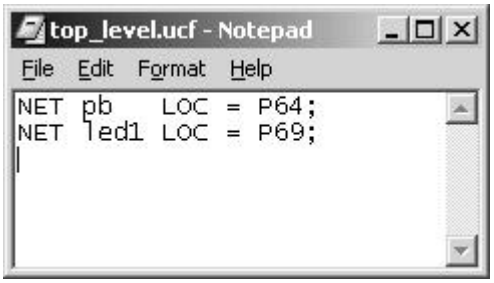

Figure L1.7: The User Constraints File can be used to enter settings such as pin assignments, time requirements, and routing constraints.

A **Notice** message will popup asking if the Implement Design process should be reset. Always select **Reset** in response to this question; otherwise, the pin assignments may not be implemented during compilation.

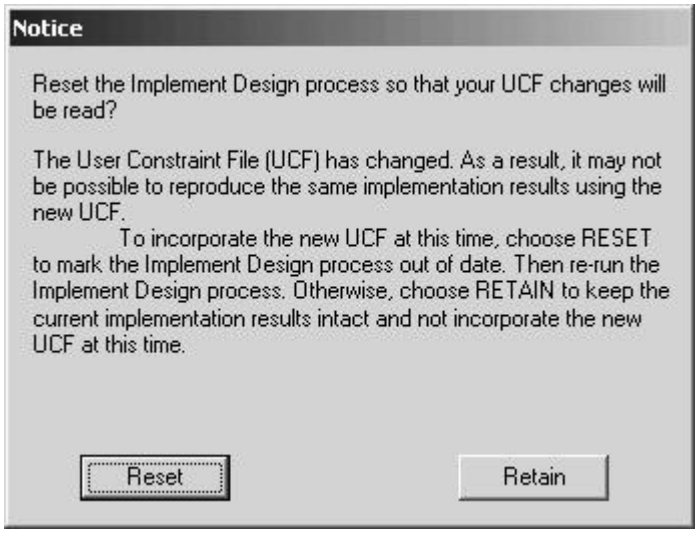

Figure L1.8: This Notice dialog box appears when the UCF file is closed. Click **Reset** anytime it appears.

Configuring the Compiler

The default settings in the Xilinx ISE software do not work reliably with the Digilab IIE board. You must change two settings in the **Process Properties** section. To make these changes, right-click on **Generate Programming Files** in the **Process View** window and select **Properties** from the shortcut menu. Select the **Startup options** tab. Change the **Start-Up Clock** option to **JTAG Clock**. Click **OK** to save the changes and exit.

Starting the Compiler

Compilation is a multi-step process. In Xilinx, a VHDL design must go through the following stages: Synthesize, Translate, Map, Place & Route, and Program File Generation. (Before running any of the compilation stages, make sure the **top_level.vhd** item in the **Module View** window is still highlighted. If it is not, then single click on it to make it active.) Each of these stages can be executed individually by double-clicking the respective option in the **Process View** window. Alternatively, all compilation steps can be run at once by right-clicking on **Generate Programming File** in the **Process View** window and selecting **Rerun All** from the shortcut menu. As each stage of com-

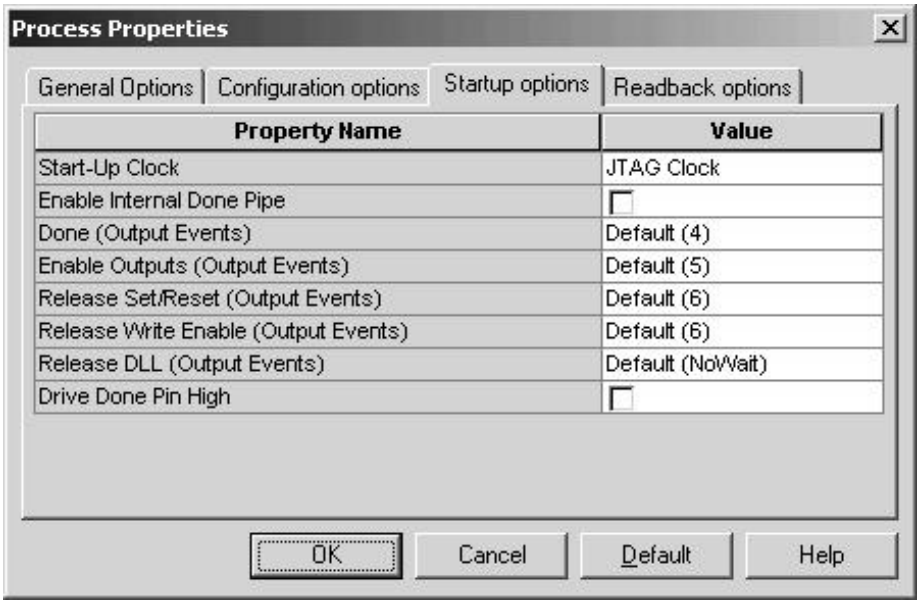

Figure L1.9: Compilation settings can be changed in the Process Properties dialog box.

pilation is completed, a green, yellow, or red mark will appear in the **Process View** window indicating whether it completed without warnings, completed with warnings, or failed with errors respectively. If any stage fails, compilation will stop and the errors can be viewed in the **Console** window. In the case of an error, you may have to scroll the text in the **Console** window to view the full error comments.

L1.0.3 Downloading the Design

Preparing the board

The Digilab IIE board can be programmed using the PC's parallel port. Connect a DB25 cable between the PC and the FPGA board. Make sure that the switch next to the DB25 connector on the board is in the **JTAG** position. Plug in the provided 3.3V power transformer and connect it to the board. There should not be any jumpers installed on the **Mode** pins. The board is now ready to be programmed.

Programming the board

In the Xilinx Project Navigator's **Process View** window, double-click on the **Configure Device (iMPACT)** option under **Generate Programming File**. This will open up the programming file in Xilinx's iMPACT program (see Figure L1.10). Right-click on the green representation of a Xilinx chip and select **Program...** from the shortcut menu. When the **Program Options** popup dialog box appears, click **OK** to program the device. When programming is finished, **Programming Succeeded** should appear in the main window.

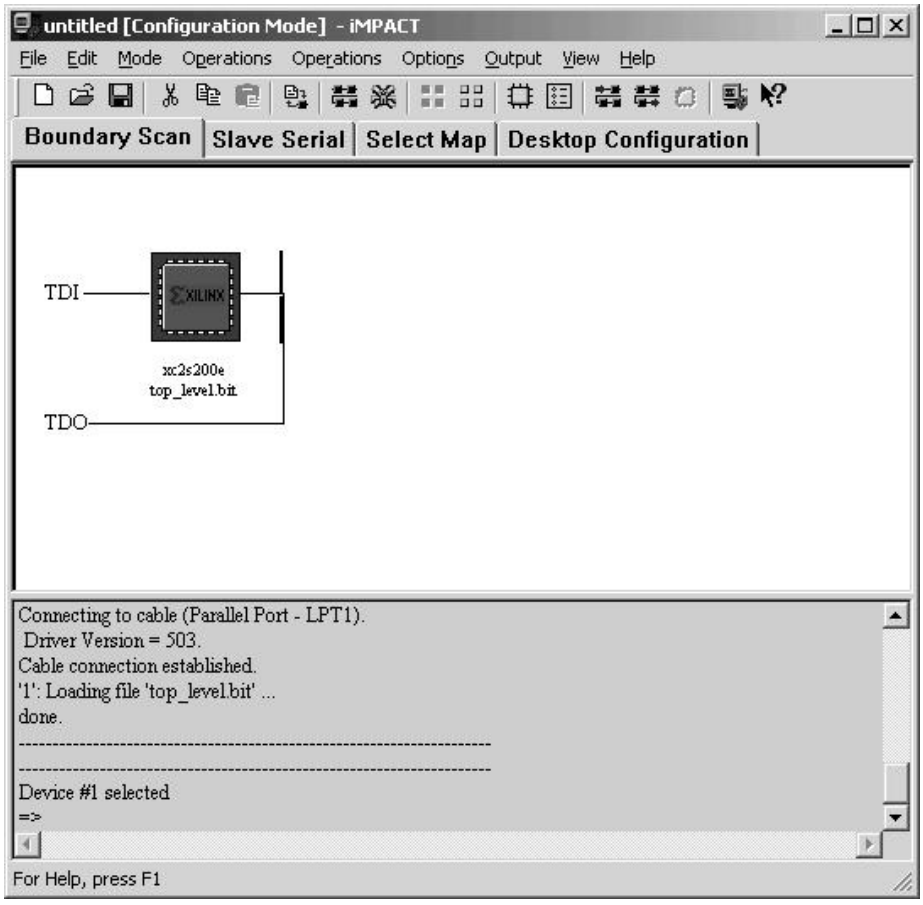

Figure L1.10: Screen shot of Xilinx's iMPACT program, which is used to program the FPGA in the Xilinx ISE software suite.

Remember to get checked-off by the teaching assistant before continuing!

Exercise #1

Create a new project that implements a toggle flip-flop. Refer back to the tutorial if you need help. The VHDL code for a toggle flip-flop is shown in Figure L1.11. In the User Constraints File, assign the the input signal **pb** to pin **P77** and the output signal **led1** to pin **P69**.

top_level.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity top_level is
    Port ( pb   : in  std_logic;
           led1 : out std_logic  );
end top_level;

architecture Behavioral of top_level is
   signal q : std_logic;
begin

   process( pb )
   begin

      if ( RISING_EDGE(pb) ) then
         q <= NOT q;
      end if;

   end process;

   led <= q;

end Behavioral;
```

Figure L1.11: The completed VHDL code for the toggle flip-flop example.

Report for Laboratory 1

The report for Laboratory 1 is due at the *beginning* of the next lab period (Sept. 9). For this report, turn in the following items:

1. Completed Check-off Sheet.
2. Properly documented VHDL code for the tutorial (follow the Format and Style sheet).
3. Properly documented VHDL code for Example #1 (follow the Format and Style sheet).
4. (Optional) List of errors found in the VHDL Tutorial.
5. (Optional) List of errors found in the lab handouts.