Embedded System Design Lab 4

Stephen A. Edwards

Due August 10, 2005

Abstract

Design and implement, in synthesizable VHDL, a circuit that sums the contents of an on-chip memory. Validate it using the simulator, run it through logic synthesis, and verify it works on the FPGA.

1 Introduction

For labs 1 and 2, you treated the FPGA board as a target for C programs. If this were our only objective, we would have chosen a board with, say, a small processor and a few peripherals. Instead, the board has a very flexible FPGA and beginning with the lab, you will get a chance to take advantage of this flexibility by designing and implementing your own hardware. For this lab, you will only design hardware; your simple system will not include the Microblaze processor or any related components.

2 Programming the FPGA

The *lab4.zip* file contains a project consisting of a main module with a small ROM whose contents are displayed using the *hexdisplay* module. Launch the lab4.npl file to start Project Navigator and try downloading the design to the board. There is a task under the main.vhd file called "Generate Programming File." If you invoke "Configure Device," it will run the iMPACT program, which can download a bitstream to the board.

iMPACT asks a number of questions when it starts. Say "I want to configure device via *Boundary-Scan Mode*," and "Automatically connect to cable and identify Boundary-Scan Chain." It should find two chips in the chain: the xc3s400 (the main chip) and an xcf02s (used for programming interfaces). Assign the *system.bit* file to the first chip (the main FPGA) and say "Bypass" for the second chip.

Finally, to program the FPGA, select it (it should turn green) and right-click to bring up a menu. Select "Program" and "OK." This should download the design to the chip.

3 Simulating the Design

There is also a testbench VHDL file in the lab4 project. You can run it directly, but to make the output of the display reasonable for humans, there are a number of very large counters that slow down the project, making the simulation very slow.

I have included commented versions of faster clocks (i.e., that do not slow the design down) in both main.vhd and hexdisplay.vhd. I suggest you use these faster clocks when you are testing your design and finally switch them back to the slow versions so you can see the results on the FPGA.

4 The Assignment

In *lab4.zip* you will find a simple project that contains a simple ROM represented using the template discribed in class and displays "00" on the two LED displays using a pair of simple hex-to-seven-segment decoders.

Your job is to add a controller (i.e., some sort of state machine) that sums the contents of this ROM and displays the result on the LEDs. Add an accumulator, an adder, and a counter that generates addresses for the ROM. Draw a block diagram of these components and a timing diagram showing signals such as the ROM address, the data coming from the ROM, and the accumulated sum.

As usual, we have provided a skeleton of the code for this lab in *lab4.zip*, available from the course website. This zip file contains VHDL files (suffix ".vhd") and a few configuration files that together produce a bitstream for the FPGA that counts in hex on the LEDs on the Digilent board. Open the project in Project Navigator, modify the source files, and either simulate the design or synthesize it to the board.

Show your working solution to the TA, have him sign off on it, and turn in a listing of your VHDL source files along with block and timing diagrams.

As usual, short, elegant solutions (this is more difficult in VHDL, but not impossible) will receive better grades than messy ones.

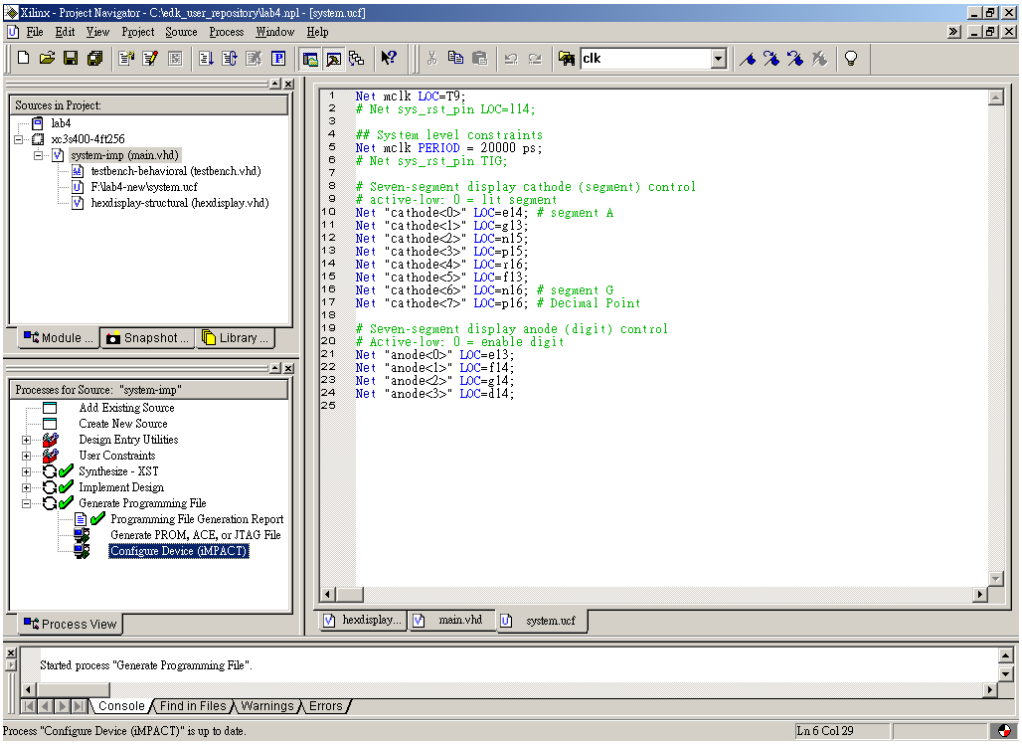

Figure 1: Invoking “Configure Device” in Project Navigator.

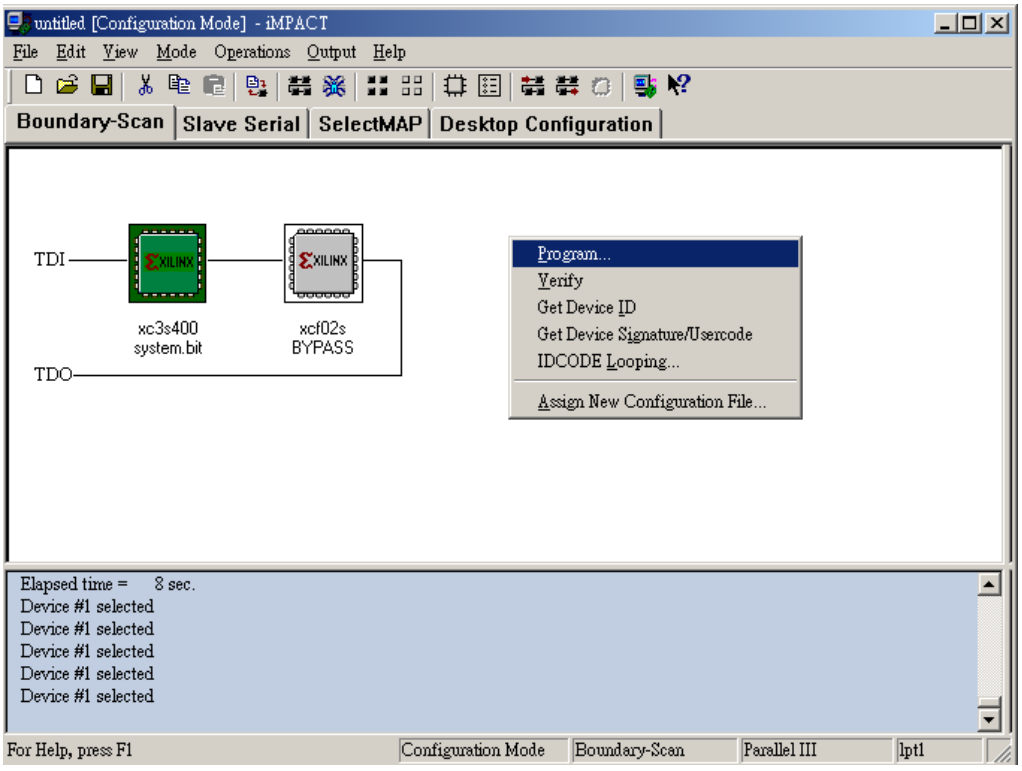

Figure 2: About to program the device in iMPACT.