WIRELESS OPEN-ACCESS
RESEARCH PLATFORM

Lab 2: Introduction to Xilinx Platform Studio

Siddharth Gupta & Patrick Murphy

Rice University

WARP Project

Document Revision 1

March 27, 2010

1 Introduction

In this lab you will use Xilinx Platform Studio to construct a simple hardware & software project that will control various user input/output devices on the WARP board. We will start with a pre-constructed hardware design that is based on the template projects available online.

2 Examining the Base System

Navigate to `C:\workshop\Lab2_XPS\xps`. Double-click on **system.xmp** to open the project in Xilinx Platform Studio (XPS). This is a template project that is available in the online repository as well. The template projects are XPS projects that are built with a certain subset of IP cores. If you are undertaking a new design, the template projects are a good starting point. The project we are using here is the basic version as it enables just the user I/O cores with no support for radios. A quick tutorial on XPS use is in the workshop slides.

3 Compiling the Software Design

Each XPS project is composed of two parts. The first defines the hardware configuration which describes the layout of the various IP cores in logic and the bus connections that enable data transfer between them. The software half describes the code that will run in the embedded PowerPC. The instructions for the PowerPC are stored in block RAMs (BRAMs). Once the bitstream for the hardware part has been generated it does not need to be updated during C-code iteration.

First, we shall download the default software that is part of the template project.

1. From the menu **Device Configuration** click on **Update Bitstream**.

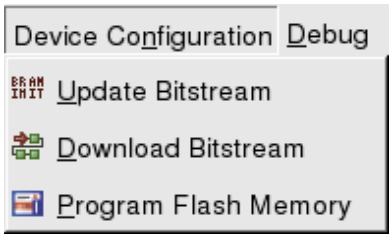

Figure 1: Device Configuration Menu

2. Once the software has been compiled, click **Download Bitstream** from the same menu. This will take the compiled software and hardware bitstream and configure the FPGA.
3. Open Tera Term Pro, connect to COM1 and set the baud rate to 57600. The terminal should display the status of all the User I/O on the board (see Figure 2). At the same time the hex displays and 4-bit LEDs should be counting up.

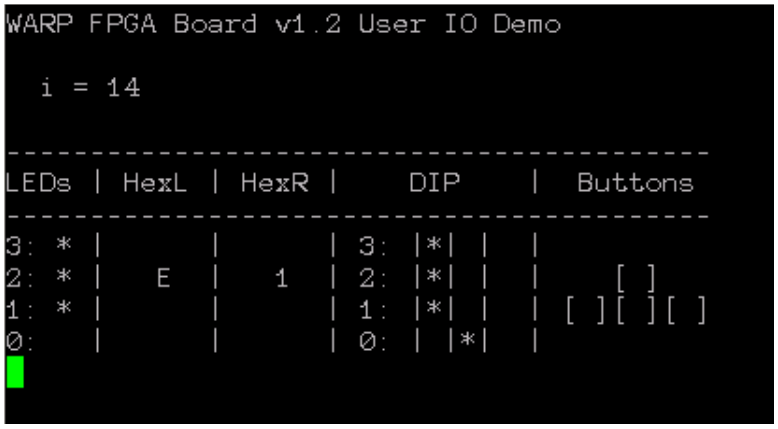

Figure 2: Terminal Output

4 Compiling a New Software Project: LED Counter

One hardware platform can have multiple software projects associated with it. The hardware bitstream is not re-compiled even if a new software project is chosen. Here we will see the steps for using a new software application project to implement the LED Counter software project. The project will implement just the hex display counter part of the default project.

Figure 3: Hex Display Counter

1. Right-click on **Project: LED_Counter** from the **Applications** tab and choose **Mark to Initialize BRAMs**. This will set the C-code associated with this project to be compiled into the hardware bitstream.

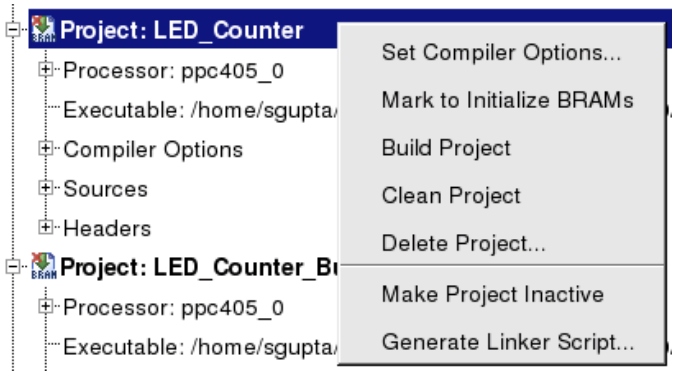

Figure 4: Project Menu

2. Open the source file associated with the project by expanding the **Project:...** and **Sources**.
3. Modify the files as explained in the source code.
4. From the menu **Device Configuration** click on **Update Bitstream**.

5. Right-click on **Project: LED_Counter** and choose **Generate Linker Script...**. Hit **OK** in the resulting dialog box. This will generate the script on how to link the various C-files together if the project has more than one source file associated with it.
6. Update bitstream once again and download it to the board as above.

5 Additional Exercises

Use the same XPS project to complete the following examples.

- **LED_Counter_Button**: holding down a pushbutton should pause the counter. The counter should be paused only as long as the button is depressed
- **LED_Counter_Keyboard**: hitting **p** in the terminal should pause the counter and **u** should un-pause it.
- **Hex_Ticker**: an input from the keyboard (0-9, a-f) is displayed on the hex displays. The old characters should scroll to the left, like a ticker.