CSEE W4840 Embedded System Design Lab 2

Stephen A. Edwards

Due February 21, 2013 for last names starting A–N, February 28, 2013 for last names O–Z

Abstract

Learn to use the Nios II IDE (programming environment) to implement an Ethernet chat client on the DE2 board.

1 Introduction

You will develop software in this lab. We supply a hardware design for the DE2 that includes the Nios II processor, memory, an Ethernet controller, a VGA controller, and a controller for the PS/2 keyboard.

You will implement an Ethernet-based chat client on this hardware. The user should be able to type in a line of text using the attached keyboard and see it appear on the video display. When s/he presses *Enter*, the contents of the line should be sent as a UDP broadcast packet.

The board should also display on the screen every UDP broadcast packet it receives, which assumes all packets come from similar projects.

To clarify who is typing what, the first few characters of each packet should contain the user's name. Have the user type this when your system starts, save it, and send it automatically at the beginning of each packet.

We have connected all the boards in the lab to hubs to form a local-area network that is not connected to the Internet to avoid causing problems for others. You will want to use a "packet sniffer" such as *Wireshark* or *tcpdump* to observe details of the packets you send and receive.

Program the board for this lab in two steps. First, open te DE2_NET.qpf project in Quartus and download the DE2_NET.sof file to the board (no need to recompile the project). This configures the hardware but leaves the software unconfigured. To bring the board to life, download and run software on it using the Nios II IDE, described below.

2 The Nios II IDE

Start the Nios II IDE by typing *nios2-ide*. When it starts, it may give you a few icons to choose among. Select "Workbench" to get started.

First, set your workspace. This is the directory in which the IDE will put files. Select File→Switch Workspace and select your directory for lab2.

When you select a new workspace location, it begins empty (Figure 1). Start by importing the provided projects.

We have provided two projects that you should import. Select File→Import, then General→Existing Projects into Workspace (Figure 2). Click Next>.

Select the lab2/software from the lab2 tarball (Figure 3). Make sure you have both the lab2 and lab2_bsp ("board support package") projects selected. Click Finish.

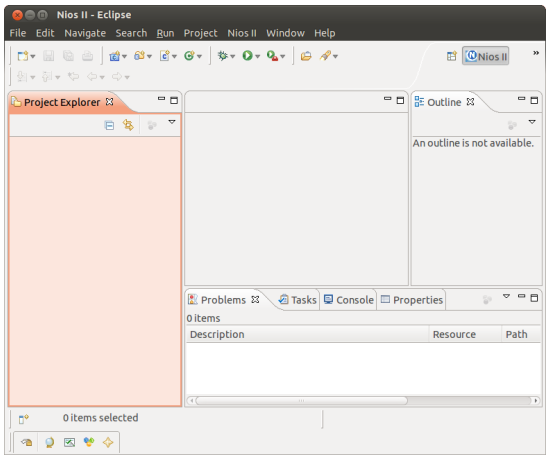

Figure 1: The Nios II IDE workbench (empty)

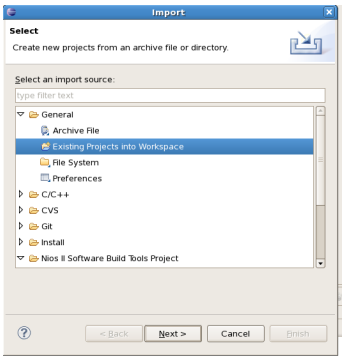

Figure 2: Importing projects into the IDE

Now, select Project→Build All.

After importing both lab2 and lab2_syslib, build them by selecting "Projects" and "Build All." Once built, you can run it on the DE2 by selecting "Run," "Run As...," and "Nios II hardware." This choice does not appear unless the project has been built, and it will not work unless you downloaded the DE2_NET.sof file to the board using Quartus.

It may complain about a mismatched system ID (Figure 4). Scroll down and check "Ignore mismatched system ID" and "Ignore mismatched system timestamp." If it still fails after this, you may have to recompile the system in Quartus before proceeding with the software.

After the Nios II IDE runs your program, it connects a terminal to the DE2 board that allows you to print and type to your running C program through standard *printf* and *getchar* calls. Figure 5 shows a running project.

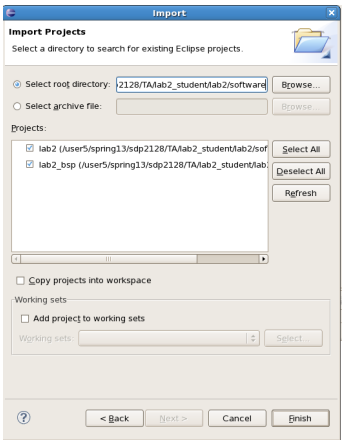

Figure 3: Selecting which existing projects to import. Make sure both lab2 and lab2_bsp are selected.

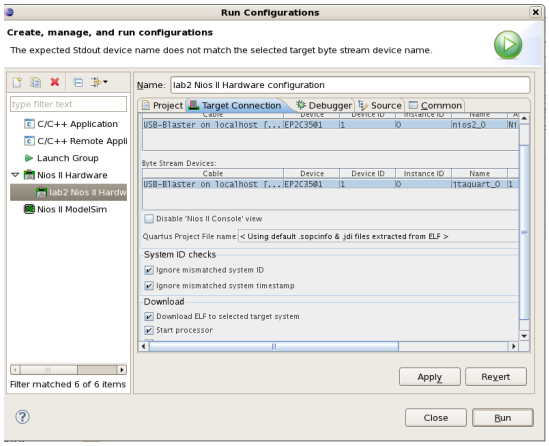

Figure 4: Disabling system ID and timestamp checking

3 The Chat Application

The lab2.tar.gz file has a partially-working skeleton for the application. The code we supplied is awful; modify and discard most of it. Here is what you need to do:

- Make the VGA display work properly. This is a one-bit-per-pixel, 640×480 framebuffer with the ability to set the foreground ("on") and background ("off") colors across the whole screen. There is also a "cursor" mode that draws a cross across the screen; you may ignore it. We have supplied a rudimentary text-mode character generator in *chargen.c* that displays 8×16 characters.
  - Clear the screen when the program starts.
  - Separate the screen into two parts with a horizontal line between. Use the bottom two rows as the user's text input area, and the rest of the screen to record what s/he and other users type.
  - When a packet arrives, print its contents in the "receive" region. Don't forget to wrap long messages across multiple lines.
  - When printing reaches the bottom of the area, you may either start again at the top, or scroll the entry region of the screen.

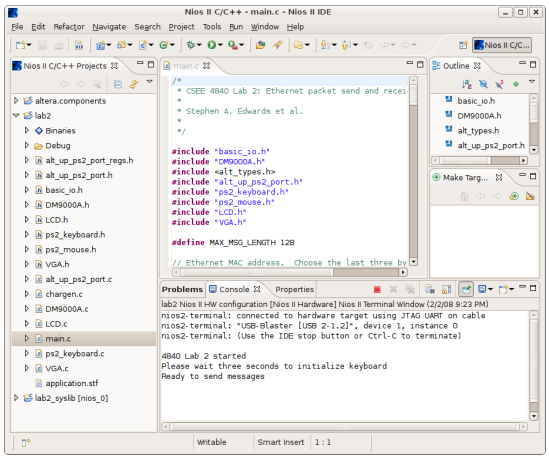

Figure 5: Running a program and observing output

  - Implement a reasonable text-editing system for the bottom of the screen. Have input from the keyboard display characters there and allow users to erase unwanted characters and send the message with return. Clear the bottom area when a message is sent.
  - Display a cursor where the user is typing. This could be a vertical line, an underline, or a white box.
- Make the keyboard input work properly. Specifically,
  - Make both shift keys work (i.e., do upper and lowercase characters)
  - Make the space bar work properly (display a space)
  - Turn off the debugging information for the unrecognized keycodes
  - Make the left and right arrow keys work
  - Make the backspace key work
  - Ignore the other keys (e.g., tab, escape, print screen, the keypad, etc.).
- When the system starts, have the user enter his/her name before going into "chat" mode. Start the string sent by each packet with this name.
- Ensure the UDP packets are well-formed:
  - Make the header checksum correct
  - Make sure the packets are always at least minimum length (64 bytes)
  - Make sure the string sent in the UDP packet is always zero-terminated
  - Make sure the UDP packet length field is correct
  - Make sure the IP packet ID numbers increase
  - Choose a different Ethernet MAC address and make sure you're putting your address in the packet appropriately.

4 What to turn in

Find an overworked TA or instructor, show him/er your working chat application, demonstrate that it is sending well-formed packets, and submit your C source code only (not everything in your lab directory!) via Courseworks.