ECE532S Digital Systems Design

Warmup Demo Project

Last Updated: Jan 2020

The purpose of this demo is to understand and integrate some of the components that you will require in your projects. You will learn to connect your Nexys board to an Ethernet network, construct and integrate an IP as well as perform on-chip hardware debugging. In advance of your demonstration in the lab, you should have all of the functionality described below complete and ready to demonstrate. This project is explicitly structured as a group project, you should work on it together.

The completed demo is due at the beginning of your demonstration period in the lab. If you do not have the demo prepared in advance of your demonstration period, you will be assigned a grade of 0%. Please ensure that you have signed up for a demo slot or that a TA has informed you of when you are expected to present.

1 Background and Resources

The lightweight TCP/IP project (LwIP) is a small implementation of the TCP/IP stack designed to use minimal memory resources. This stack is well suited for the MicroBlaze processor and Xilinx has incorporated in SDK a version of LwIP that supports the Xilinx Ethernet IP blocks. To construct an example system that implements an echo server on the Nexys 4 DDR board, follow the steps in the Digilent Tutorial. The constructed system will have several major components (DDR memory interface, UART and Ethernet) that can be used as a good starting point for building the design for your course project. Note that the tutorial implements a server, while you will have to implement a TCP client for this project. For more information on LwIP, consult the Wiki, especially the section on the RAW API used in the tutorial.

A second new feature that you will make use of in this assignment is the hardware debugger. Chipscope ILA. A tutorial of an example can be found in Debugging Tutorial. Lab 1, 2 and 9 refer to three different ways to add chipscope to your design (append the netlist, add to verilog or add the ILA ip in your design). Note, the ILA was also covered in the course tutorials themselves. Finally, for more general information on the board, refer to the Nexys 4 DDR reference manual.

2 What to Do

As a group, you are to construct an FPGA hardware system that acts as a network TCP/IP client to a computer server. Within your hardware system you must include a custom AXI Lite IP component that reads the board switches and buttons, and performs some basic manipulation on these input values. The system as a whole is reactive to two of the buttons on the Nexys 4 DDR board, and performs two major operations.

- Operation 1
  - A *transmission* operation is started based on presses of the `BTNL` button on the board.
  - This operation is facilitated by a custom AXI-Lite IP you are to design.
  - The first time the button is pressed, the 16 switch values are read and stored as the lower two bytes of a 4-byte word register. (Hint, a button press corresponds to a pressing and releasing of the button).
  - When the `BTNL` button is pressed a second time, the 16 switch values are read again and stored as the upper two bytes of the 4-byte word register.
  - At this stage, your custom IP should signal to the MicroBlaze that a new valid word is ready. There are two ways you could implement such a signal; using an interrupt line from your custom IP; or using an AXI-Lite register that the MicroBlaze polls until it sees the ready signal. Note, in either case don't forget to implement functionality to reset the ready signal.
  - The software running on this soft processor will read the full word, establish a TCP/IP connection to the computer server and send the word over the Ethernet network as a *POST* message.
- Operation 2
  - A *receive* operation is started when `BTNR` is pressed.
  - This operation is facilitated by a Xilinx provided AXI GPIO IP Core. Note, the Microblaze software could poll this core for a press of the `BTNR` button.
  - Now the MicroBlaze will connect to the computer server and read the current 4-byte word using a *GET* message.
  - The retrieved word will be then shown on the 7-segment displays, in hexadecimal.

A rough block diagram of the main system components is shown in Figure 1. Supporting IPs have been left out of the diagram for clarity.

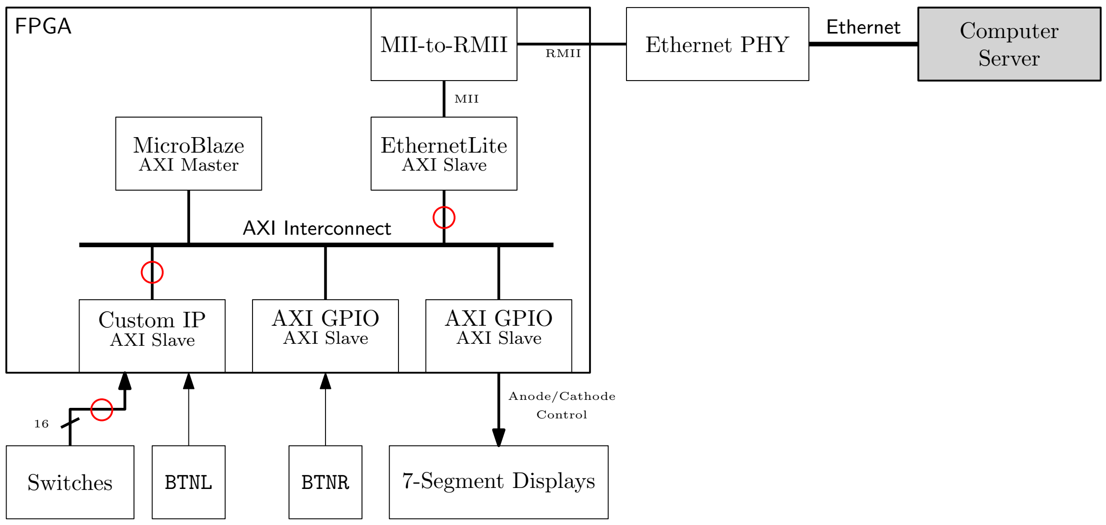

Figure 1: Block diagram of the project. Red circles indicate probe points explained in Sec. 2.2

2.1 Detailed Specifications

When a connection is made to the server, one of two commands is expected. A `GET` command will cause the server to send a 32-bit value corresponding to the value stored in the server. On the other hand, a `POST` command will update the 32-bit value. The `POST` message must be accompanied immediately with the new value to set.

For example the byte sequence 0x50, 0x4f, 0x53, 0x54, 0xba, 0xad, 0xf0, 0x0d is a valid `POST` sequence. The first four bytes correspond to the letters `P`, `O`, `S` and `T` when converted to ASCII and the remaining four bytes make up the new value to set 0xBAADF00D.

The Python3 files `echoserver.py` and `echoclient.py` demonstrate simple versions of the server and client. You may use the server implementation but the client will have to be implemented in the Nexys 4 DDR board.

2.2 Hardware Debugging

Once the system is complete, you should use the Xilinx hardware debugger to ensure all of the hardware is working as expected. Specifically, you are required to set up hardware probes at the three interfaces highlighted with red circles in Fig. 1. You will be asked during the demonstration of your project to trigger transactions at these interfaces and you should be able to explain what is happening.

3 Tips

To get this system to work, you will need to connect the Nexys 4 DDR board to a computer with an Ethernet port. The Nexys 4 DDR Ethernet PHY supports automatic Medium-Dependent Interface Crossover (MDIX), thus a straight or crossover cable can be used to connect the board to a switch or directly your computer.

Python3 must be installed on your personal computer to run the example scripts. Executing the `echoserver.py` script should start a process listening for incoming TCP/IP connections on port 9090 by default. The `echoclient.py` connects to the server via the "localhost" address `127.0.0.1`. You may modify the server and client scripts to better suit your environment.