525.742

SOC FPGA Design Lab

Lab 9: Linux Full SDR with Ethernet

150 Pts + milestones

**Summary**

This lab will put together all you have built so far (except for the ADC interface since we don’t have a way for you to use the real ADC hardware remotely). You will take your PS7 project from Lab 7, and further adapt it to pull in all of the output data from the radio, and stream that data via. Gbit Ethernet to a connected computer. Along the way, we will gain some experience with how these devices are typically used (running an operating system vs. the “bare-metal” way that we have been developing up until this point)

**Goals**

1. For the student to gain some experience booting and running Linux on the PS7. The benefit achieved by using an OS should be apparent
2. Familiarity with ethernet, UDP, TCP, and how they can be used in minimal systems.
3. Program the PL from the PS without reboot using FPGA Manager
4. Write simple network software in C (or Python) and compile and run that code directly on the PS7

**Requirements**

This lab is expected to be the functional equivalent of the SDR lab (ability to tune the radio, the fake adc…etc) while also having a couple of extra features, the major one being the ability to stream the output data via ethernet. You as the designer have wide latitude over exactly what the interface and commands are (you may certainly keep things the same as before : ‘u’,’U’,’T’,’F’ commands…etc, but you may also adapt for a different style of interface which makes life easier in this infrastructure)

1. **Student must submit:**
   1. **A .bit.bin Xilinx bitfile which can be programmed to the PL via the fpga-manager. The instructor will scp this file to his board and use the fpga-manager to program the fpga before running your software**
   2. **Executable software which runs the SDR lab from the linux shell prompt. Student should provide instructions on how the command should be executed.**
2. **The software must have some means for the user to interactively:**
   1. **Set the tune frequency of the radio**
   2. **Set the fake-adc frequency piped into the radio**
   3. **Enable / disable streaming of radio output data to ethernet**
   4. **Control the destination IP address of the UDP packets (port can be fixed at 25344)**

**References**

Provided for you is a MATLAB script to display in real-time the signal that you are streaming out via Ethernet/UDP. This program will digest and display packets in the format described in the appendix of this Lab. [Collect\_Data\_Complex.m](https://livejohnshopkins-my.sharepoint.com/:u:/g/personal/dwenstr3_jh_edu/EV2dY0zhjldGggghQbOWMjcBhfeQGVwyI8zWvApIB_DkEw?e=8rIE1Q)

Note - While streaming, all samples should be sent via UDP such that there are no discontinuities between packets. Note: it is possible that UDP packets can be dropped by the network or destination computer. It is not our intention to guarantee this never happens. However, you should generally be able to capture consecutive packets and observe your data via. ethernet. The counter in the packet format will make it easy to see if packets are occasionally being dropped by the network.

**Appendix : UDP Data Format**

This laboratory comes with an instructor provided matlab script for viewing the data in real time. If you don’t like the UDP format as presented here, you are welcome to design your own, or simply edit the provided matlab receiver. However, make sure that you submit the matlab with your project so that it can be used for the demo. The instructor provided matlab will listen for UDP packets on the ethernet interface, and display an FFT and time domain plot of the data to the screen.

UDP packets must be formatted in the following manner to be properly displayed:

UDP Destination Port : 25344

1026 total UDP data bytes per UDP frame (256 complex 16 bit samples per packet)

bytes 0-1 : 16 bit unsigned counter, which increments by one each UDP frame. This will let the application know if it has missed a frame, so it can tell the user. This way you know if you are missing lots of frames.

bytes 2-1025 : 16 bit signed I, 16 bit signed Q, 16 bit signed I, ...etc.

all 16 bit values are little endian, and sample rate is assumed to be 125MHz/2560 = approx 48kHz

On receiving these packets, the collect data application will plot the time and frequency domain values of the received data.