Project 9: Full SDR with Ethernet (Zybo-Z720) (ZYNQ 7020) – Using Vivado 2019.1 VHDL

Summary

This project will put together SDR Tuner controller, PS7 (no Microblaze), and using custom AXI peripheral (with internal FIFO) to move the output data from the PL to the PS where it can stream it to a PC over Gigabit Ethernet.

Goals

1. Scatter-gather DMA and the general operation of the Ethernet peripheral.
2. Using UDP at a low level for sending data to connected devices from minimalist systems.
3. Extend AXI peripheral to serve as a data collection interface.

References

Provided:

- Project 5 SDR Tuner controller
- Project 8 custom AXI peripheral
- MATLAB script to display in real-time the streaming signal out via Ethernet/UDP. This program will digest and display packets in the format described in the appendix of this project. "Collect_Data_Complex.m"

System Parameters and Capabilities

Functionality project 5, the complete DDC should be preserved with the ability to tune a fake ADC (DDS), and the Radio Tune frequency and also use the uUdD command keys for incremental adjustments of the Fake ADC. Volume control (+/- command keys) should also be preserved.

A new option shall be added to the main menu (in SDK), which is a command to enable / disable streaming of data via ethernet. When the program starts, ethernet streaming should be disabled. When the user requests the streaming to be enabled, all the data from the filters (the same data going to the headphones) should be sent via UDP to a broadcast address 255.255.255.255, where it is displayable using "Collect_Data_Complex.m"

The system shall print "Full SDR with Ethernet (Zybo-Z720)" and a list of serial commands. The command list shall be displayed any time the space bar is pressed via serial command, or slider switch, the user can enable or disable ethernet streaming of the decimated IQ (DAC left & Right) data.

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 the intention to guarantee this never happens. However, you should generally be able to capture consecutive packets and observe data via ethernet. The counter in the packet format will make it easy to see if packets are occasionally being dropped.

Appendix : UDP Data Format

The provided matlab code 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 missing multiple frames will be known.
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.