6.111 Final Project Abstract: Real-Time RF Fingerprinting for Amateur Radio Repeaters

Oliver Trevor

October 18, 2022

1 Background

Since almost the beginning of radio, interfering transmissions have caused problems for amateur and professional operators alike. This is especially a problem for repeaters, which are (often public) radio devices that retransmit users' signals at higher power from an antenna in a high location, allowing a small handheld transmitter to reach a wider area. Historically, repeater control operators have had no recourse against trolls or other unwanted users other than using crude radio direction-finding techniques to track down the offending user and apply legal or physical threats to their person. However, beginning around 20 years ago, radio operators discovered that, because the PLL in a modern FM transmitter does not immediately start at the target frequency but instead "wanders" for a fraction of a second when the transmitter first keys up, the exact key-up characteristics of a signal can be used to uniquely fingerprint a transmitter. This effect is used by cellular companies to detect cloned SIM cards and by the government to identify and track illegal transmitters. However, most schemes for doing this use a computer for processing, which means that–especially if the database of possible key-up signatures to compare against is large–this can only be done retroactively on recorded data, making it useless for preventing the interference in the first place. To create a system for locking unwanted users out of a repeater without delaying the transmissions of legitimate users would require identifying the key-up signature in the fraction of a second before the user starts talking and making a decision as to whether to allow the repeater to retransmit the signal. This is an ideal project for an FPGA because it requires massively parallel DSP with a hard time limit on coming to a result.

2 High-Level Specification

The FPGA will digitize the output of an FM discriminator (the raw demodulated "audio" signal without any of the usual post-processing that radios apply) and detect when a transmission is incoming. Upon receiving a transmission, it will record it into a buffer.

Then, an array of digital matched filters will convolve the received signal against known RF "fingerprints" to attempt to find a similar one. A large part of the complexity of the project will live in this part, as the number of individually identifiable repeater users will be a function of how many digital matched filters can be fit into the FPGA fabric *and* how fast those filters can run, since the filters will likely run fast enough that each one can run a comparison against multiple stored fingerprints.

Optimizing the combination of parallelization and serialization of the DSP filters to be able to handle an environment with many possible users while still positively identifying transmissions in real time will be the challenging part.

Beyond producing a lockout/allow signal for the repeater controller, I would also like to explore what other useful insights the FPGA could give by storing identifications on an SD card or passing them to a computer over UART/Ethernet/etc. I have written FPGA things that used "raw" SD card access before, so I could use some combination of dd and Python scripts to read the stored files off the card without having to implement WAV and FAT32 on the FPGA. Alternatively, I could stream the recordings of blocked calls over UDP to a computer that saves them to persistent storage. This part of the project would provide a way to include more complex peripherals and protocols.

3 Design Goals

This project will prioritize mass parallelization for speed. The goal is to be able to maximize the number of distinct users the system can recognize in real time using a single FPGA's fabric.

I already have a radio with a "9600 baud" packet radio output, which I believe is essentially a direct FM discriminator tap. Depending on whether I use the XADC of the FPGA, I may need a external ADC board. The bandwidth does not need to be very high, so it should not be particularly expensive. An I2S audio ADC would probably work, as long as it does not perform any filtering/post-processing.

4 Work Distribution

This is an individual project, so the work will not be distributed. I will, however, try to broadly split the project into a few phases:

1. Pre-work: recording a selection of Radio Society's transmitters' key-up characteristics using a digital oscilloscope, feeding the data into Python, and prototyping how the DSP filters will work. This will let us decide things like sampling rate and buffer length that will inform how the Verilog gets implemented. It will also inform whether we need to use the DDR RAM or can store enough patterns in the BRAM blocks.
2. Filter implementation: creating an initial implementation of a single matched filter that compares one recording against one stored recording.
3. Optimization and parallelization: working out how to fit more filters into the FPGA, which will become a speed/area/memory tradeoff problem.
4. Outputs: piping identification data to the repeater controller and to a computer.