RPI TEAM:

Number Munchers

CSAW 2008

Andrew Tamoney
Dane Kouttron
Alex Radocea

Contents

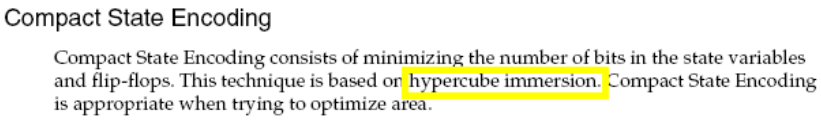
Introduction:

The following paper details the design of a 'Trojan' for use in embedded systems to extract secure information, using methods of low frequency data transmission in conjunction with a low tech RF receiver constructed from a pair of standard, dollar-store, headphones. Our entry utilizes a monitor as a transmitter, and a pair of 'broken' headphones connected to a standard microphone port, as a receiver.

Tactics Implemented:

Attacking the Compiler

With the given time constraints and additional requirements, matching the benign Alpha device's device utilization proved difficult. More specifically, the synthesizer performs fairly powerful optimizations and is difficult to significantly outperform. For proof of this, consider the following example from page 252 of the XST User's Guide.



Compact State Encoding

Compact State Encoding consists of minimizing the number of bits in the state variables and flip-flops. This technique is based on hypercube immersion. Compact State Encoding is appropriate when trying to optimize area.

Additionally, manual optimizations with the FPGA Editor at the gate level are not possible since the shipment of the Trojan-ed alpha *must* be in the form of source code. [1]

Unsatisfied with the few placement constraints offered by ISE, the RPI Electronics Club Team determined an alternate method to hide the code size of our additions.

Within our project directory we have included a hidden dynamic link library, "libGc_ReportViewer.dll". It is a patched version of the original libGc_ReportViewer.dll, normally loaded from the ISE installation directory.

Fortunately for us and K, and unfortunately for the Orange Army, the standard behavior on most versions of Windows is to check the current directory when resolving library paths.
If our entry is opened by double-clicking on the project file in our submission directory, even if another version of ISE is running, the patched binary will be loaded. It should be noted that an attack of this type was not explored on Linux. It is likely that some other attack vector would be required. [2]

The patched library has been modified to always display the benign alpha device's utilization.

[1] Manually editing of logic gates is enjoyable anyway. That said, the fpga editor can be found at the following location in the ISE install path: ISE\\bin\\nt\\fpga_editor.exe
[2] Preliminary investigation did not find simple file format vulnerabilities which could be used to gain code execution

Low power RF transmission

Our team searched for a method of making the data available from outside the FPGA by using low power AM transmissions. We developed a means of producing and recovering data using this method, as documented below.

General Overview

Our team developed a method of injecting two separate carrier frequencies into the blue video carrier on the outbound VGA port on the Basys FPGA board. In determining the frequency of the carrier, we searched for the highest frequency that would result in the least perturbation of the video signal's content; I.E. the one with the least visible fingerprint. We decided to use the VGA cable, because it is the longest trace to function as an antenna we could find. In a CRT, the analog signal is amplified, and used to aim an electron gun, so theoretically the rf signal is slightly amplified.

We chose to use a differential – 3 band method of encoding the data, as it would result in the least perturbation of the video signal. By slowly switching between specifically chosen carrier signals, we are able to encode data, albeit slowly, which can propagate to an appropriate receiver.

Finally, a program running on the receiving side, continuously performs an FFT on the microphone, searching for the 18.5, 19 and 19.5 kHz bands. Upon finding the appropriate start bit sequence, the program converts the incoming RF data into the 128 bit security code string. This process takes approximately 92 seconds, as the baud rate of data transfer has purposely been set very slow. A graphical interface displays the number, as its being decoded, in a format akin to those used in cheesy sci-fi movies.

Encoding method

The triple carrier method we chose to use produced the least interference visible on the monitor. For the frequencies of 18.5, 19 and 19.5 kHz, introduced to the blue VGA line are difficult to notice, visually, but propagate well in the rf spectrum. Our start bit is more than 1 second of high output on the 18.5 band, with 19khz and 19.5 KHz held low. After the start bit has been established, the master key is transmitted over the 19 KHz band with 19.5 KHz always being the inversion of 19 KHz. During the transmission section 18.5 KHz is low. This scheme ensures that exactly one band will be on at any one time. If multiple bands are turned on it becomes very noticeable on the CRT. We also never want to have no bands on since then you will see a darker black which is slightly distinguishable from 1 band on.

Hiding in the noise

It is important to note that data communications are very rare at such low frequencies, namely because of the extremely long antennas necessary. As such, most EMI detection schemes wouldn't target these frequencies.

Incognito receiving of low power RF transmissions

The Receiver

We were very limited in what we could use as a receiver, adding an am radio receiver wouldn't be counted, and external hardware wasn't acceptable. After further research, we concluded that a 'broken' set of headphones would be suitable for an antenna into a standard microphone port, which would function as a low power, low frequency RF receiver. For this project, we consider a standard pair of headphones essentially as a dipole antenna with a load, the headphone, at the end (see figure 4). By removing the load (cutting the headphone speakers from the headphone cable), the electrical potential across the dipole antenna increases dramatically, allowing the perception of the 19 and 19.5 kHz frequency transmission.

A standard microphone card has the capability of recording approximately 44 kHz, or twice the human hearing range. The nyquist theorem details that the reproduction of data on any frequency requires a recording of twice that frequency, therefore by choosing 19 and 19.5khz, the minimum frequency of sampling on the microphone side would be 39khz, well within the spectrum of a microphone input.

The required antenna length for properly receiving and transmitting data on the 19 kHz band would be orders of magnitude longer than any trace on the board, or any pair of headphones, so our group relies on a very low baud rate of data transmission, and a relatively short data transmission distance; as an example, a covert op would have to be within a few meters to receive the data with a pair of headphones, but could better receive the data from much further distances using a more sophisticated receiver.

Software FFT and data collection

FFT Background

An FFT, or Fast Fourier Transform, creates a plot of the frequency domain of a system. As our system transmits data by modulating 18.5, 19, or 19.5 kHz onto the blue analog line of the VGA cable, the software implementation required to demodulate the data and make it useful is split into three main parts (fig 4).

Primarily, noise filtering of the microphone input proved to be vital to getting valuable data through the system. The noise filtering is implemented in a sigma-delta fashion, wherein we compare the FFT output of 18.5, 19, and 19.5 kHz. The delta of magnitude between each frequency is recorded. The delta values are summed over .5 second intervals and finally passed as average deltas. The averaging is necessary due to the imperfections in the microphone hardware, as well as *flickering* in FFT response caused by system resource mismanagement.

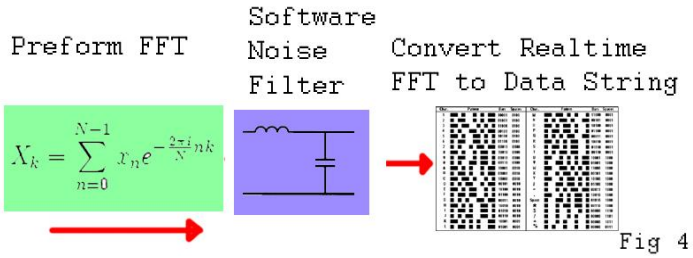
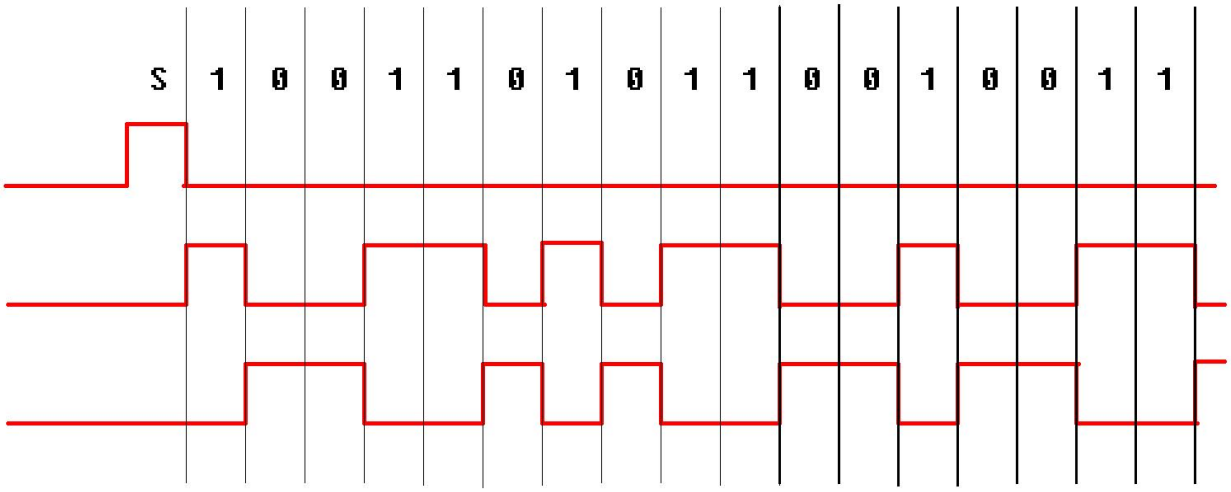


Fig 4

Once the deltas are measured, our team uses a differential scheme to decode the data from the FPGA board.



Future Improvements:

Some of the competition rules forced us to make implementations that were less than ideal. If we had more freedom the range of the device as well of the accuracy could be increased by increasing the frequency of the transmission. As stated before we could not go much higher than 19 KHz due to limitations of a computer sound card. With additional hardware it would be possible to increase this frequency significantly in order to better suit that transmission properties of a standard VGA cable. We would also increase the accuracy by replacing the headphones with a real antenna.

Team Members / Backgrounds

Andrew Tamoney

Andrew Tamoney is a freshman at RPI and is currently dual majoring in Electrical Engineering and Computer Science. Andrew was President of the Robotics Team and Science Team in High School. He was also a competitive rower throughout all four years of high school. He is now a member of both the RPI electronics club and the RPI Formula SAE Hybrid Team.

Dane Kouttron

Dane Kouttron is currently pursuing a dual Degree in Electrical and Electrical Power engineering at Rensselaer Polytechnic Institute. Dane is president of the RPI Electronics club, managing demonstrations and the student run electronics laboratory. He has also worked in hardware and software development for numerous campus programs, ranging from bio-chemistry to the Arts department. Dane is also an active member in the school FSAE team, a competitive racecar design club which competes internationally.

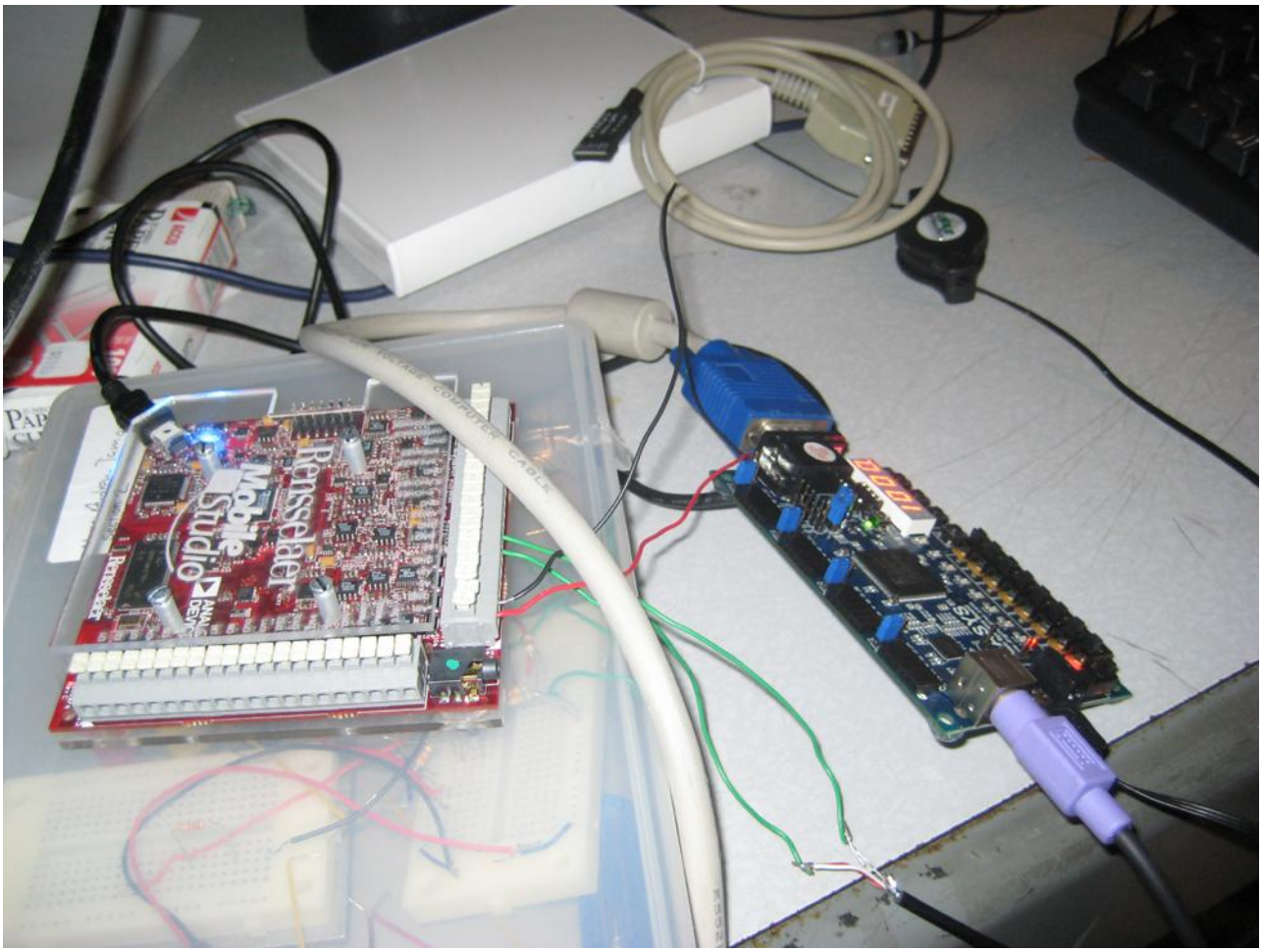
Alex Radocea

Alex Radocea has played war games since the 9th grade when he stumbled upon wargames.unix.se. Alex is currently a sophomore studying Computer Systems Engineering and Computer Science at Rensselaer Polytechnic Institute. He is the current President and Co-Founder of RPISEC, a computer security club.

This project was a joint venture of the RPI electronics club and RPISEC, both from Rensselaer Polytechnic Institute

Appendix

1- Using the Mobile Studio Board to inject carrier frequencies into the Basys board

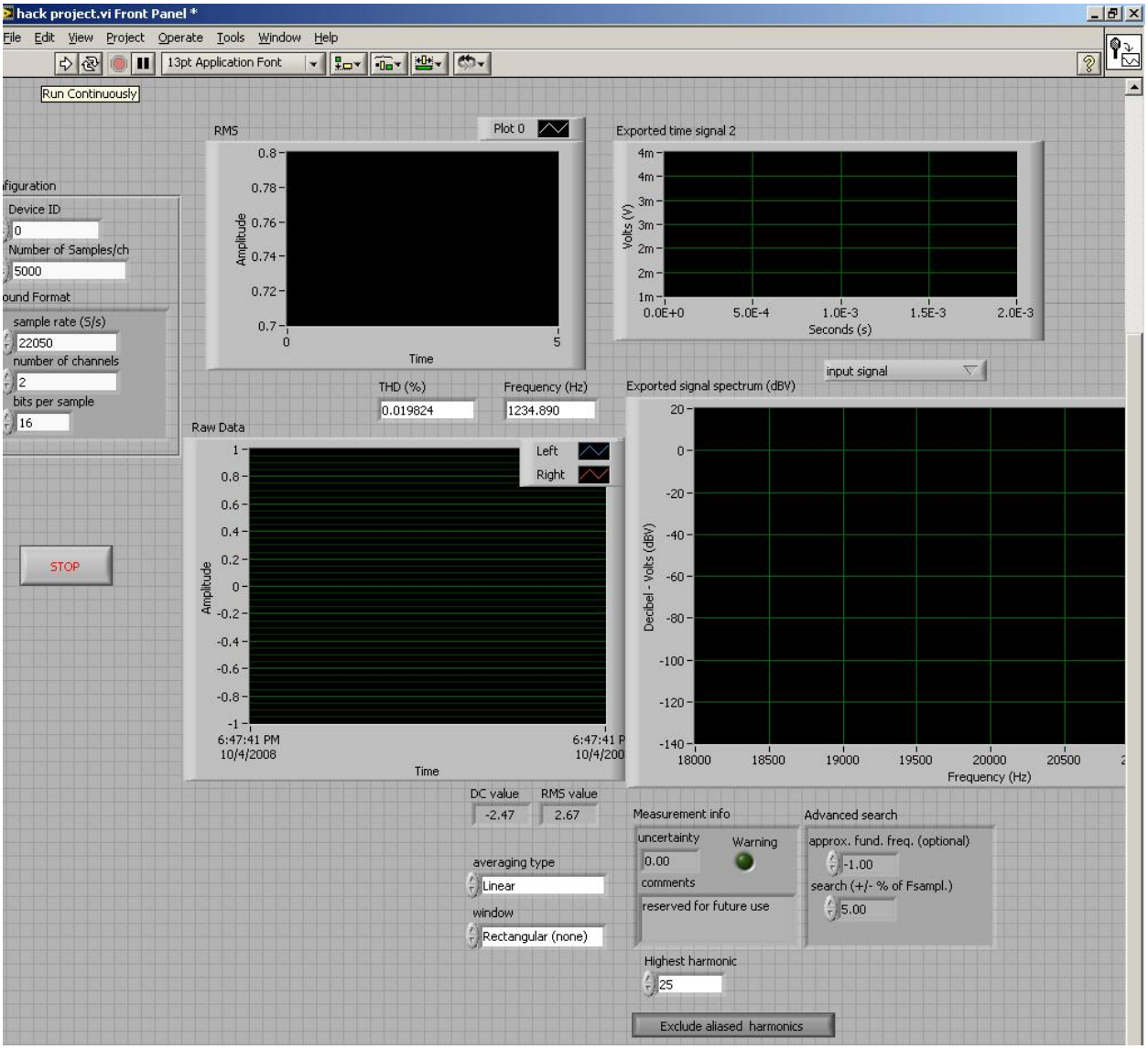


Our team used the mobile studio platform, an integrated function generator + oscilloscope, to begin testing if our ideas on low frequency RF propagation would work. We began by injecting different frequencies into the blue line of the VGA cable, and determining which frequencies were least visible. We did numerous frequency sweeps, at 3.3v pp, to simulate what the FPGA output would be perceived as. After much testing, we concluded that 18.5, 19 and 19.5 kHz were optimal.

2- CRT displaying standard image and encoded blue carrier data

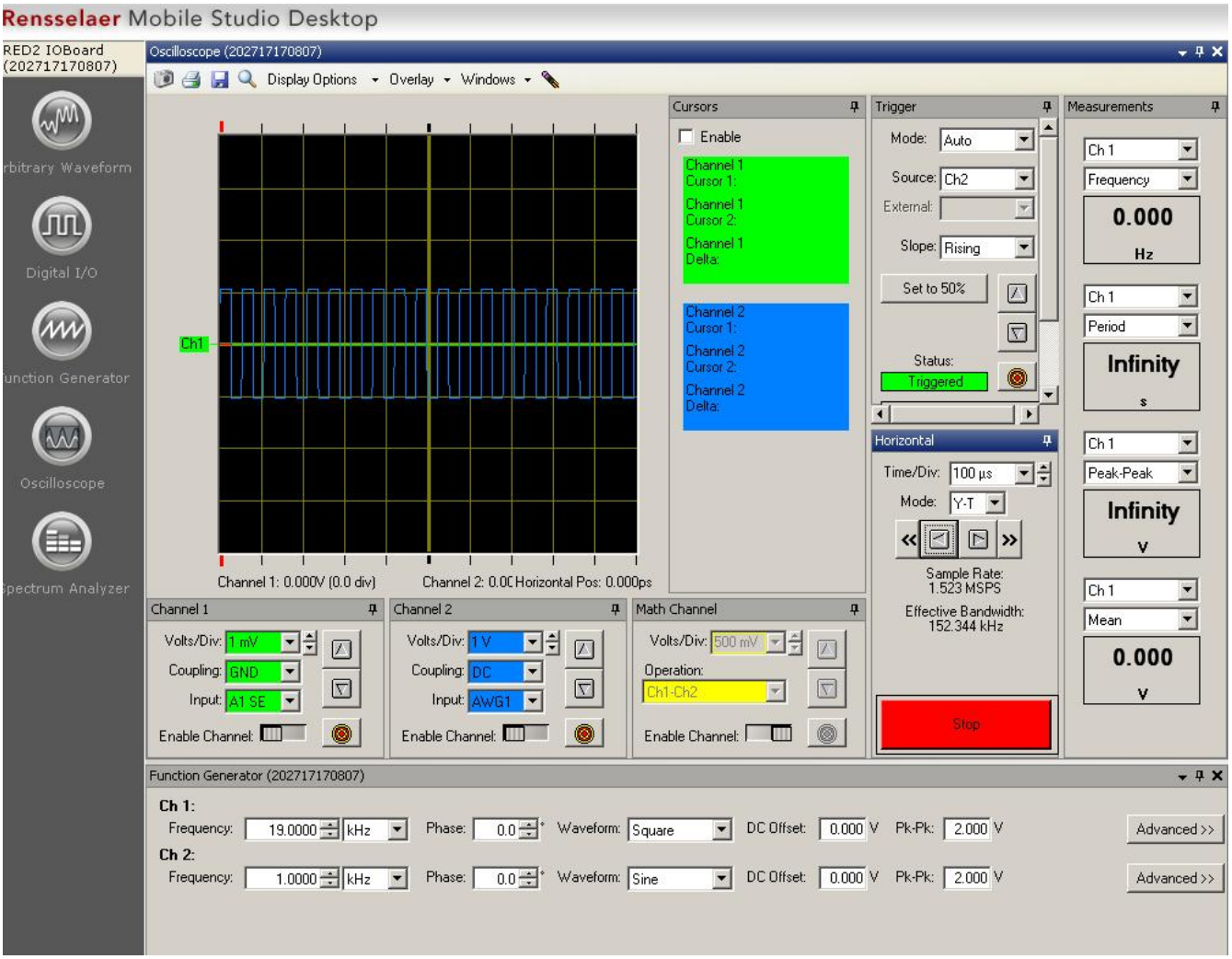
The blue carrier data is hidden by running it at frequencies close to the horizontal sync, the only visible portion is a slight bluish tint of the background color. On the monitor in test, it was very difficult to perceive the color difference.

3- Labview interface developed to analyze microphone data stream, and display an FFT.



When operating, this program would take samples from the microphone port, at 44khz, and output an FFT of the desired bands we were looking for.

4- RPI Mobile Studio Board generating 19 kHz



This is the software interface for the RPI Mobile Studio Board; we used it for testing and performing frequency sweeps on the blue VGA line.