SOFTWARE DEFINED RADIO FOR STEPPED-FREQUENCY, GROUND-PENETRATING RADAR

A Thesis
Presented to
The Academic Faculty

By

Samuel C. Carey

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2017

SOFTWARE DEFINED RADIO FOR STEPPED-FREQUENCY, GROUND-PENETRATING RADAR

Approved by:

Dr. Waymond R. Scott, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. James H. McClellan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Gregory D. Durgin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 28, 2017

If at first you don't succeed, try, try, try, try... try again.

"Jack O'Neill," Stargate SG-1

To my parents and siblings,

for giving me all the love and opportunities a boy could hope for.

ACKNOWLEDGEMENTS

Thanks to Dr. Scott, my advisor, for providing excellent guidance through the research process, ample facilities and equipment, expert technical insight and advice, and encouragement through the many frustrating stages of this project.

Thanks to the employees of Ettus, especially Jonathan Pendlum, Marcus Müller, and Derek Kozel, for providing excellent technical support and helping me navigate the many nuances of Ettus hardware and software.

This work is supported by the U.S. Army REDCOM CERDEC Night Vision and Electronic Sensors Directorate, Science and Technology Division, Countermine Branch; the U.S. Army Research Office under Grant Number W911NF-11-1-0153; and by Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This research explores the potential of software defined radio (SDR) as a platform for ground-penetrating radar (GPR). SDR is a rapidly developing technology that implements signal processing components partially or completely in software, providing enhanced design flexibility over traditional hardware radio [1]. The stepped-frequency GPR method is used to detect and locate metallic and non-metallic subsurface objects. A multi-tone signal allows the 500-5000 MHz band to be scanned 150 MHz at-a-time. This system is extended to isolate and process the reflections of modulated scatterers. A demonstration prototype is developed, consisting of a transmitting antenna and a receiving antenna, each pointed at the ground and connected via coaxial cable to an off-the-shelf SDR unit. This SDR unit is connected via ethernet cable to a host computer, which controls the system using open-source software libraries. SDR is typically designed for communications applications, so special consideration is necessary for remote sensing. Initial design challenges are weighed against potential advantages of design flexibility and hardware versatility.

CHAPTER 1

INTRODUCTION AND BACKGROUND

1.1 Ground-Penetrating Radar

Ground-penetrating radar (GPR) has been shown to be an effective tool for detecting and locating buried objects such as utilities, archaeological artifacts, and landmines [2]. A GPR transmits an electromagnetic wave that is reflected by underground targets and then received again, as shown in Fig. 1.1. Then, the round-trip time-delay of each reflection is used to determine the depth of each target. The resolution of measurements is highest near the surface and becomes increasingly degraded with increasing depth, soil density, and soil moisture. By systematically sliding a GPR back and forth across a surface and applying advanced signal processing, a user can generate detailed subsurface images.

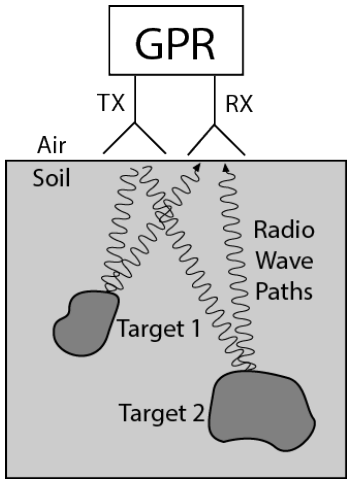


Figure 1.1: A typical GPR scenario. The GPR hardware is held over the ground with its transmit (TX) and receive (RX) antennas directed into soil containing discontinuities in permittivity (targets). Radio waves from one antenna reflect from the discontinuities at different depths and are received by the other antenna.

Reflections can be caused by variations in subsurface permittivity or conductivity. While this sensitivity reveals various materials of interest, it can also produce false alarms when

soil properties naturally fluctuate. A large effective bandwidth can help distinguish targets from false alarms by increasing the resolution of the transfer function. In high performance GPR systems, the ratio of the highest frequency to the lowest frequency of the operation band often exceeds 10.

Many different hardware configurations have been used to measure the transfer function between the transmitting and receiving antennas in a GPR. In pulsed systems, a very short, time-domain pulse is radiated and received. Usually, the pulse is repeated many times, with the receiver sample time shifted slightly for each pulse. The receiver can then build a representation of a single received pulse, with a resolution equal to the time shift.

Some GPRs measure the transfer function by transmitting random or psuedo-random noise signals and correlating them with reflections [3][4]. Frequency-modulated, continuous-wave (FMCW) GPR systems use a transmit signal that has a linearly increasing frequency along with a simple mixer to measure the reflections. [5].

A frequency domain approach is the stepped-frequency method. The GPR transmits a steady tone of a certain frequency and records the magnitude and phase of the reflection. Next, this measurement is repeated at equally spaced frequencies, scanning a certain band. A wider band increases range resolution, and measuring frequencies closer together decreases range ambiguity. The many tuning operations required for each scan can take considerable time, so a reasonable scan rate during a GPR survey can be difficult to achieve. Often, these frequency-domain samples are transformed into the time-domain to obtain the response versus time.

1.1.1 Modulated Scatterers

Modulated scatterers have previously been used to characterize the beam patterns of GPR antennas [6]. A modulated scatterer is a small device that can modulate its own radio backscatter. This can be achieved with a collection of small antennas connected by diodes. With a DC bias, these antennas are electrically joined together to form a larger antenna and

increase the magnitude of reflected signals over a certain band.

In this application, the DC bias is modulated at a frequency much lower than the transmitted frequency. The modulated product of the real bias signal and the complex transmitted signal creates harmonics on either side of the transmit frequency, shifted by multiples of the bias modulation frequency. While the transmitted tone is reflected by everything in the environment, only the modulated scatterer will reflect these shifted tones. By only processing these shifted tones, the receiver can effectively distinguish the device's reflection from that of nearby clutter.

1.2 Software Defined Radio

An SDR is a signal processing chain that converts between physical radio waves and processed digital data. An ideal SDR would implement as much of this chain in software as possible, receiving and transmitting by sampling and synthesizing RF signals, with the ADC and DAC connected directly to antennas. Currently this is not technically feasible. Instead, SDR is implemented as a combination of RF circuits, specialized digital circuits, and software, as shown in Fig. 1.2.

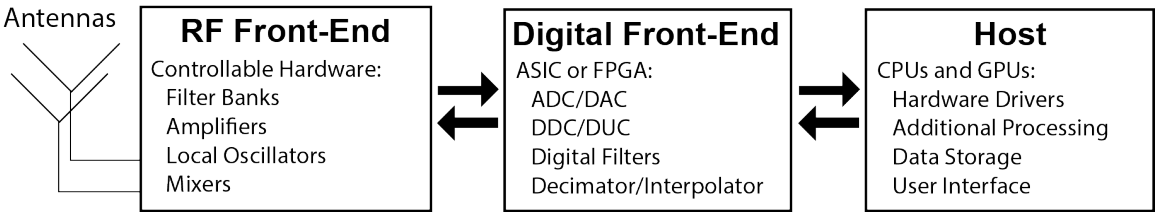


Figure 1.2: A typical SDR. Components are divided into three categories, each representing stages along the processing chain.

The term "software defined radio" refers to all three of these stages working together, although it may also refer to just the front-ends which are typically sold with the expectation that the user will provide their own host computer.

In order to receive a radio signal, an RF front-end is used to amplify, filter, and mix the RF signal down to a lower intermediate frequency (IF). A digital front-end uses an

analog-to-digital converter (ADC) to sample this IF and then passes the samples through a pipeline of digital circuits that may implement digital down-convertion (DDC), digital filtering, and decimation. Finally, the stream of samples is sent to a host computer where it may be processed, stored, and/or displayed to the user.

The process for transmitting a radio wave is essentially the reverse. The host sends samples to the digital front-end, where they may be interpolated, digitally up-converted (DUC), and synthesized with a digital-to-analog converter (DAC). The resulting IF is filtered, mixed, and amplified by the RF front-end.

The U.S. DoD began to develop SDRs in the early 80s in the form of digital baseband receivers and arrays of digital processors [7]. In 1991, Joe Mitola enhanced the design to include a transmitter and higher frequency ADCs and DACs. The next year, he published the concept of "software radio" for the first time and encouraged its development [8]. Over the next decade, the U.S. DARPA and Air Force project SpeakEasy combined several radio systems into a common hardware unit that could be continuously updated to keep up with new communication protocols as they were developed [9]. As the military continued to improve upon their design, falling prices for electronics eventually allowed the front-end hardware to be marketed to non-military researchers and radio enthusiasts. By the late-2000s, improved computing hardware and software allowed sufficiently complex signal processing to be achieved on a civilian budget. Today, users with little analog design experience can purchase a generic, off-the-shelf SDR hardware unit, download compatible software, rapidly prototype almost any radio system in software, and immediately test it in the real world.

This new process increases efficiency at every stage of radio development. For example, SDR allows simulations to be smoothly transitioned to physical tests by simply replacing theoretical models of hardware and signal channels with their physical versions. Hardware fabrication is replaced with software compilation, dramatically reducing the costs and delays of the design iteration cycle. The development process can also benefit from

unprecedented ease of collaboration, where researchers can share new improvements online and ensure that they are working on the most recent versions with their colleagues. Development can even continue after device deployment, if the user has remote access. Parameters may be adjusted at runtime, and completely different designs may be juggled by simply switching programs.

SDR can also simplify hardware manufacturing. Although each device is typically more complicated than necessary for any one application, it can be marketed to a wide range of consumers for a wide range of applications. As the market converges to fewer and fewer devices, economies of scale can reduce costs and increase availability for all consumers.

Several software packages have emerged to support this new hardware, providing drivers to control the front-ends and code libraries to implement digital signal processing. Generally, they are compatible with a range of SDR hardware and allow users to design programs using graphical flowcharts. Major distinctions between these packages arise in licensing and source control. National Instruments provides SDR libraries for its Labview package [10], and Mathworks provides SDR libraries for its Matlab with Simulink package [11], but both are proprietary and closed-source. Consequently, users do not have full control over library details, and any commercial products they develop must negotiate licensing from the providers of the software. On the other hand, there are free, open-source, community developed software projects such as GNU Radio (GR) that provide comparable functionality without those restrictions [12].

All of these software packages communicate with a range of SDR devices by using the appropriate drivers for each device. Many higher-end, consumer devices are designed by Ettus, a National Instruments company, that provides an open-source driver library for all of their devices called UHD [13]. Several Ettus SDRs are equipped with field programmable gate arrays (FPGAs), which are programmed by default to implement the digital, pipelined stages mentioned above.

1.2.1 GNU Radio

A GNU Radio (GR) application is based on a series of connected signal processing blocks called a "flowgraph." A typical flowgraph begins with source blocks that receive data from hardware or files and pass it "downstream" to processing blocks. Eventually, the flowgraph ends with one or more sink blocks that export the data or display it in real-time. This modular architecture allows users to import, connect, and adjust parameters for a large library of existing blocks. Additionally, users may write their own custom blocks and easily incorporate them into any existing flowgraph using standardized interfaces. GR provides scripts to automatically generate "out-of-tree" (OOT) modules and block templates consisting of numerous files of "skeleton-code" for common configurations. This sets up most of the standard interfaces and structure, allowing the developer to focus on implementing their own custom signal processing code.

GR leverages both Python and C++ for tasks to which they are respectively suited. Python is used for operations that are complicated but less computationally intensive, such as establishing connections between blocks, handling the Graphical User Interface (GUI), and launching the application. Python can also describe individual blocks. Alternatively, blocks may be written in C++ to accelerate computation and take advantage of existing C++ libraries. This is possible thanks to the Simplified Wrapper and Interface Generator (SWIG), which allows C++ blocks to be called from Python. By default, each block is assigned its own thread and input/output buffers if appropriate.

If desired, blocks may be registered with the GNU Radio Companion (GRC). GRC displays a graphical representation of the flowgraph, where a user may drag-and-drop blocks and configure connections, variables, and GUI controls.

1.2.2 RFNoC

In recent years, Ettus has been extending UHD and GR to simplify the process of programming the FPGA with extra functionality. The RF Network on Chip (RFNoC) project allows

users to create custom GR blocks with two parts: a synthesized digital circuit on the FPGA and a thread running on the host that configures it at run time [14]. The automatically generated template for a new NoC block is similar to that of a normal host block, with the addition of a Verilog HDL (Hardware Description Language) definition of the block, as well as a System Verilog testbench. The HDL block can be instantiated within the main, HDL source for the FPGA. Then an image is synthesized using Xilinx Vivado (the successor of Xilinx ISE) and loaded onto the FPGA as a bitstream. This synthesis process can take several hours for each design iteration, and once the logic is implemented in the FPGA, it is extremely difficult to debug directly. Therefore, it is essential to simulate and verify the HDL block as completely as possible on the host beforehand, using Vivado and an appropriate testbench.

Currently, Ettus includes a relatively small library of RFNoC blocks within an experimental code distribution. These implement processing operations such as FFTs, FIR filters, addition, and logarithms. There is also a signal generation block for sinusoids and noise that can act as an FPGA-side source block.

1.3 Software Defined Ground-Penetrating Radar

The GPR methods described above are generally implemented using custom hardware that is often difficult to design and build due to the extremely wide bandwidths required for performance. The capabilities of SDR appear to meet the requirements for several GPR methods, suggesting that a GPR could be implemented using SDR and reap the described advantages of SDR. Various radar methods have already been discussed and implemented using SDR [15][16]. Ground-penetrating radar in particular has also been discussed [17], but a search for actual implementations of SDR GPR revealed no examples.

CHAPTER 2

TECHNICAL APPROACH

2.1 System Overview, Goals, and Requirements

This research project seeks to design and implement a stepped-frequency ground-penetrating radar (GPR) using software defined radio (SDR). A simplified system overview is shown and summarized in Fig. 2.1. Later sections will expand this explanation.

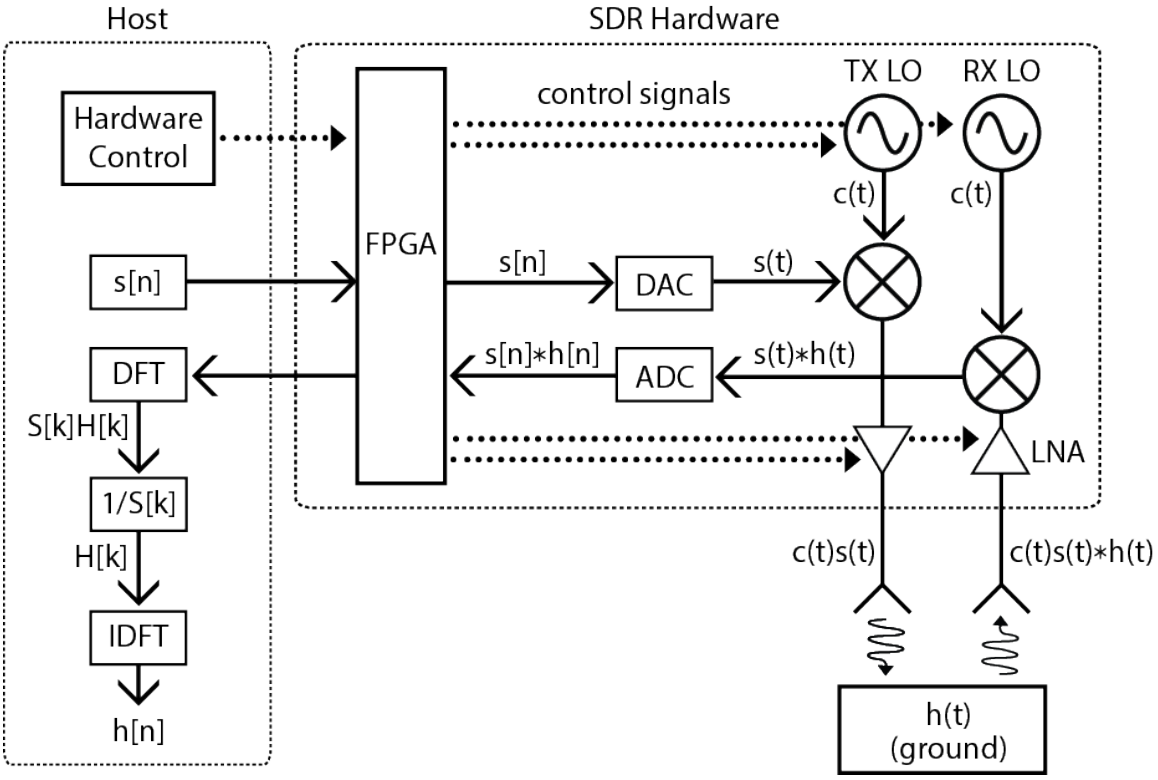


Figure 2.1: SDR GPR system overview. A known digital signal, s[n], is sent from the host to the SDR where it is synthesized, upconverted, amplified, and transmitted into the ground. The ground is modeled as an LTI system, with impulse response h(t). The received signal is modeled as the convolution of the transmitted signal and h(t). By applying a DFT, dividing by the DFT of the known signal, S[k], and applying an IDFT, a digital representation of the ground's impulse response, h[n], can be calculated. The host also sends commands to tune the local oscillators and adjust gain. The FPGA handles communication between the host and the SDR hardware. Note that each mixer shown actually represents two mixers for in-phase and quadrature-phase components. Signals to the left of the mixers are complex.

The stepped-frequency GPR method requires that the transmitter and receiver share a common phase reference. It also requires a large effective bandwidth, preferably covering the 500-5000 MHz band. Measurements from this band should occur as quickly as possible to allow rapid surveys of large areas and redundant measurements for noise cancelation. Off-the-shelf SDR hardware and open-source software are used to maximize the reusability, portability, and flexibility of this project for future contexts.

2.2 Hardware

Ettus UBX-160 RF front-end daughterboards have been chosen for their relatively high bandwidth of 160 MHz [18], as well as their ability to synchronize phase between their multiple local oscillators (LOs) [19]. Two UBX-160 units are housed in an Ettus X310, which serves as the digital front-end of the system. The X310 was chosen for its compatibility with the UBX-160, as well as its relatively large, built-in, Xilinx Kintex-7 XC7K410T field programmable gate array (FPGA) [20]. An Ettus device was also chosen for its support from open-source UHD drivers.

Transmitting and receiving at the same frequency on the same board causes significant coupling between the TX and RX channels. Therefore, separate daughterboards are used for TX and RX for each frequency in order to improve isolation.

2.3 Band Coverage

The X310's maximum sampling rate limits the digital bandwidth of the SDR to 200 MHz. The overall bandwidth of the system is further constrained by the 160 MHz RF bandwidth of the UBX-160 front-ends. Since this is much narrower than the desired 4500 MHz-wide band, this large band is broken into smaller, sub-bands, which are scanned one-at-a-time.

The RF front-ends are sequentially tuned to the center frequency of each sub-band. These centers are chosen in increments of 150 MHz to reduce artifacts from fractional-N tuning and avoid the edges of the RF filters. Therefore, each sub-band covers 150 MHz.

At this spacing of center frequencies, the 500-5000 MHz band takes about 30 retunes to complete a full scan. At each center frequency, the X310 transmits and receives a 200 MS/s signal containing eight tones to cover that sub-band, reducing the number of retunes by a factor of eight. The multi-tone signal is further described in section 2.4. A spacing between tones of

$$\frac{150 \text{ MHz}}{8} = 18.75\text{MHz} \tag{2.1}$$

allows them to be equally spaced in frequency from other tones within that sub-band as well as those in neighboring sub-bands, as shown in Fig. 2.2. This means that the tones immediately to the left and right of the center frequency will now be at relative frequencies of

$$f_t = \pm\frac{18.75}{2} = \pm 9.375 \text{ MHz} \tag{2.2}$$

The magnitudes and phases of the reflected tones are then recovered from the received signal at the respective center frequency using a DFT.

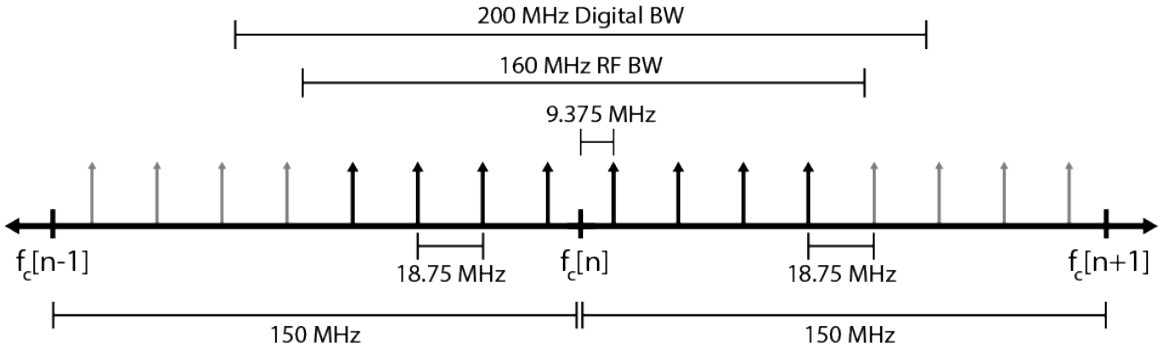


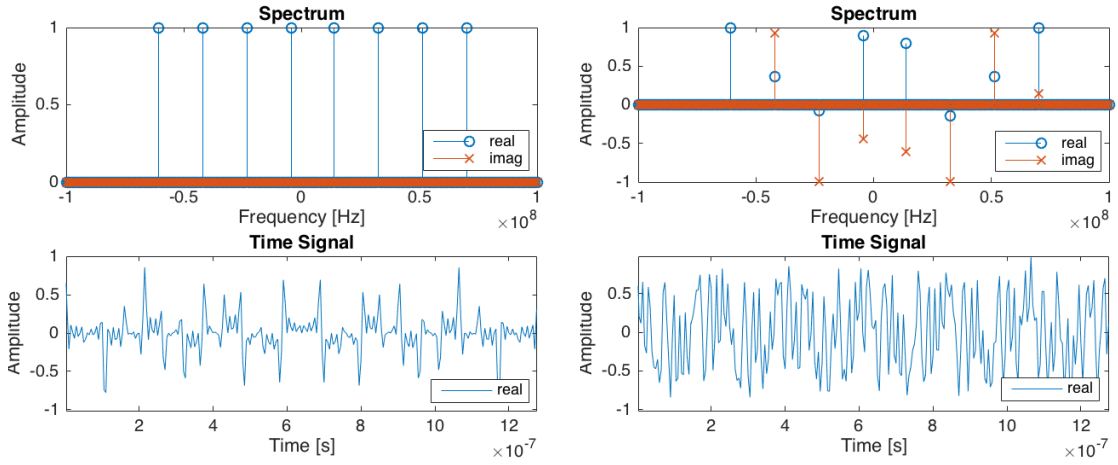
Figure 2.2: Diagram of three neighboring sub-bands. $f_c[n]$, $f_c[n-1]$, and $f_c[n+1]$ are the locations of a current center frequency, the next lower center frequency, and the next higher center frequency respectively. Currently active tones (vertical arrows) are bolded and inactive tones are faded.

Once the equally spaced samples from the entire band are collected, the host performs an IDFT to obtain the impulse response (or range profile). The entire frequency scan is repeated for each new location in the survey.

2.4 Signal Design

To generate the multi-tone signal, a vector representing its spectrum is first initialized to zeroes. Then each desired tone is rounded to the nearest frequency sample, and that sample is set to 1. Then the spectrum is inverted to the time domain. The frequency rounding adds a small amount of quantization error to the measurements, but guarantees an integer number of cycles for all tones in the time domain. In this implementation, a vector length of 256 was chosen to allow shorter bursts as well as longer bursts by repetition.

Setting the initial phase of each tone to zero creates a combined time signal with large spikes and long valleys in amplitude. By shifting the phase of the tones relative to one another, the amplitude of the signal can be more evenly distributed in time [21], as shown in Fig. 2.3. This allows amplifiers to be set at a higher gain without clipping the peaks, thus increasing average power and SNR. Once a DFT samples the magnitude and phase of the reflected tones, these phase offsets can be subtracted out by calibration.



(a) Before compression

(b) After compression

Figure 2.3: Multi-tone spectrum and time signal before and after compression. In (a), all phases are zero, while in (b), they have been shifted. In this case, the peak amplitude was reduced to 52% of its original amplitude.

IQ-imbalances in the quadrature DACs and ADCs can cause tones on either side of the center frequency to be partially mirrored and coupled to the opposite side. While UHD

includes routines to partially calibrate for this, the effect is further diminished by shifting all tones in frequency by a quarter of the tone spacing.

$$\frac{18.75 \text{ MHz}}{4} = 4.6875 \text{ MHz} \tag{2.3}$$

This means that the tones immediately to the left and right of the center frequency will now be at relative frequencies of

$$f_t = \frac{18.75}{4} \pm \frac{18.75}{2} = 4.6875 \text{ MHz and } 14.0625 \text{ MHz}, \tag{2.4}$$

respectively. This leaves tones equally spaced from both intentional tones and their mirrored versions. This additional shift is illustrated in Fig. 2.4.

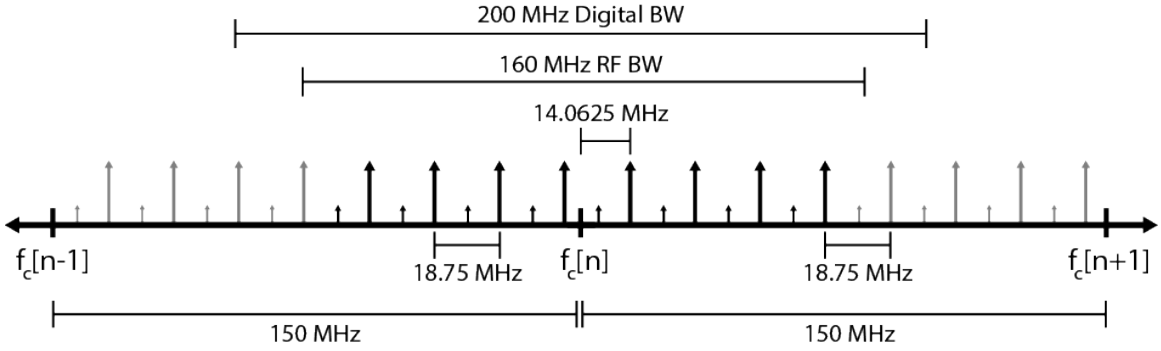


Figure 2.4: Diagram of three neighboring sub-bands with a shift to mitigate the effect of IQ-imbalance. $f_c[n]$, $f_c[n-1]$, and $f_c[n+1]$ are the locations of a current center frequency, the next lower center frequency, and the next higher center frequency, respectively. Currently active intentional tones (larger vertical arrows) are bolded and inactive tones are faded. Mirrored tones are represented by smaller arrows (not to scale).

Note that a zero frequency tone is not used, which avoids the DC-offset of the direct-conversion receiver.

2.5 Synchronization

The TX and RX chains within the two UBX daughterboards must be carefully synchronized to acquire consistent, meaningful data.

First, each chain must be synchronized in frequency. Each UBX daughterboard has two separate LOs for TX and RX. For communication applications, this is useful for transmitting and receiving in multiple bands simultaneously. However, this application requires that the LOs on both UBXs be tuned to the exact same frequency. Fortunately, the X310 provides the option of feeding all LOs with the same clock, achieving frequency synchronization.

Second, the chains must also be synchronized in phase. By default, simple tuning commands from UHD are sent by the host and executed at unpredictable times in the future, depending on numerous factors such as operating system scheduling and buffering. If the two LOs lock at different times, there will be a random phase offset between them. Fortunately, the UBX daughterboards may be locked together in phase using timed commands. Commands are executed by the digital radio controller circuit of each daughterboard instantiated within the X310's FPGA. If the command has a timestamp, the controller waits for its own internal timer to match it before execution. The user must first send a command to both control circuits to reset their timers to zero on the next Pulse Per Second (PPS) rising edge to which they both have access. As long as the command reaches both controllers within the same PPS cycle, their timers will be aligned at the end of that cycle. Next, the user sends a tune command to both circuits with the desired center frequency and an execution timestamp. This causes both LOs to tune on the same clock cycle and lock together in phase [19].

Last, the transmitted and received digital signals must be synchronized in time. As with tuning commands, commands to transmit and receive are sent by the host and executed by the radio controllers at unpredictable times in the future by default. This alignment uncertainty renders any comparison of transmitted and received signals or measurements of range meaningless. Fortunately, UHD also allows users to timestamp transmission and reception commands. The application sends a burst of samples to the TX controller beforehand, where it waits in a buffer for the specified time. The user also sends a request to

receive a burst to the RX controller, where it waits for the specified time. When the time comes, the first samples of the TX and RX bursts should theoretically occur on the same clock cycle.

For an unknown reason, it was observed that the first few hundred samples of the received burst are consistently nonlinearly distorted. Therefore, eight extra repititions of the 256-sample signal are appended to the beginning of the transmit buffer and ignored when received. Also, received bursts consistently begin 48 samples after the begining of transmitted bursts. This offset was measured by sending an impulse signal through a loopback and measuring the difference between its locations in the transmit and receive buffers. This offset may be related to the DUC and DDC on the FPGA that the samples pass through by default. Each has a 24-stage cordic, so the two combined could hold the missing 48 samples. While the exact mechanism for this delay has not been definitively determined, it has been compensated for by ignoring 48 less samples from the beginning. Thus, the first $4 \times 256 - 48 = 976$ samples of the burst are ignored to avoid the effects of both phenomena.

2.6 Scheduling

Breaking the operation into bursts has another advantage. While the X310 is designed to transmit and receive at 200 MS/s (200 million complex samples per second), exchanging this data rate with a host is difficult. Continuous operation requires the designer to choose between lowering the bandwidth to a few megahertz or upgrading host hardware to keep up. With bursts however, the host can use the breaks in between bursts to catch up, if buffers are sufficiently large.

A downside to timed commands is that controllers must wait extra time for a command to occur, instead of executing it immediately upon arrival. If a controller command arrives after its attached timestamp, it cannot be executed at the correct time, making synchronized execution with the other controller impossible. Therefore, timestamps must always be set far enough into the future so as to reduce the probability of this, sacrificing speed for

stability.

2.7 Gain Stability

The UBX-160 RX chain uses two different low-noise amplifiers (LNAs), depending on whether the center frequency is set above or below 1500 MHz [22]. The MGA-62563 is used for the lower band with an operating band of 100-3500 MHz [23], while the VMMK-3603 is used for the upper band with an operating band of 1000-6000 MHz [24]. When crossing this barrier, one LNA is turned off and the other is turned on, and onboard-switches reroute the signal path. Only one LNA may be powered on at a time to reduce ringing between them [25]. The commands to switch between LNAs are generated by UHD, depending on which RX frequency the user requests.

Unfortunately, the LNAs take time to turn on and reach a stable gain, but this GPR design requires that the center frequency hop back and forth between the lower and upper bands quickly and often. If a new LNA must be powered up each time the 1500 MHz barrier is crossed, then the application must either wait for the LNA to stabilize or accept measurements of an unpredictably lower gain. The LNA switching also causes a discontinuity in path length, but the extra phase and loss in magnitude could easily be removed by calibration.

To ensure a stable gain across both bands without waiting, the UHD source code has been edited and rebuilt such that only one LNA is used for both bands. Neither LNA is designed for the entire 500-5000 MHz band, but the VMMK-3603 operating band covers a larger portion of it than the MGA-62563. When it is always powered on, the VMMK-3603 outperforms the MGA-62563 in the lower band, due to the warm-up issues with the MGA-62563. Therefore, this UHD modification achieves better stability and SNR for this specific application. Note that this modification would not have been possible if UHD was closed-source.

2.8 Modulated Scatterers

An extension of this GPR allows it to detect modulated scatterers buried underground. Each tone that reflects off a modulated scatterer generates reflected harmonics on either side of it in the frequency domain. When the first DFT is applied to the received burst, instead of extracting the bins containing the transmitted tones, those containing some of the expected harmonic tones are extracted. These bins contain information about the magnitude and phase of the frequency response for the scatterer, as well as the relative phase of the modulation signal. Ideally, the modulation signal would be phase locked with the X310 such that its relative phase is constant. In this system, however, there is no cable between the SDR and scatterer by which to synchronize their phase. Instead a method of estimating the modulation phase based on the received signal is developed.

A periodic pulsed signal such as that in Fig. 2.5 is used to modulate the scatterer. A two level signal is appropriate, given the nonlinearity of the diode.

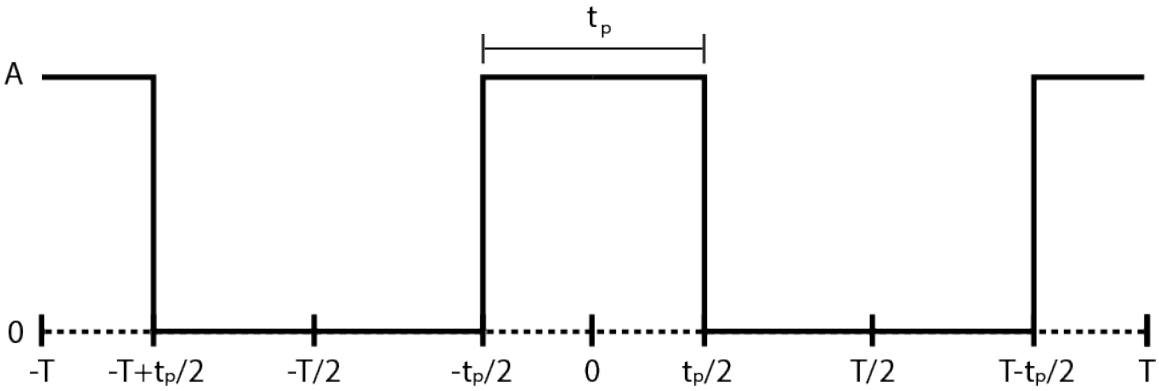


Figure 2.5: A periodic, pulsed time signal with period T, amplitude A, and pulse width of t_p.

The Fourier Series is given by

$$x_m(t) = \sum_{n=0}^{\infty} a_n cos(n(\omega_m t + \theta_m)) \tag{2.5}$$

where $x_m(t)$ is the modulation signal, ω_m is its angular frequency, θ_m is its phase, t is time,

and

$$a_n = \begin{cases} A\frac{t_p}{T} & n = 0 \\ \frac{2A}{n\pi} sin(\frac{n\pi t_p}{T}) & otherwise \end{cases} \tag{2.6}$$

where A is the peak to peak amplitude, t_p is the pulse width, and T is the period. Each received tone from the scatterer is given by

$$x_t(t) = Be^{j(\omega_t t + \theta_t)} \tag{2.7}$$

where ω_t is its angular frequency, θ_t is its phase, and B is its amplitude. Modulating each tone by the square wave produces

$$\begin{aligned} x(t) &= x_m(t)x_t(t) \\ &= \sum_{n=0}^{\infty} a_n cos(n(\omega_m t + \theta_m))Be^{j(\omega_t t + \theta_t)} \\ &= B\sum_{n=0}^{\infty} a_n (e^{jn(\omega_m t + \theta_m)} + e^{-jn(\omega_m t + \theta_m)})e^{j(\omega_t t + \theta_t)} \\ &= B\sum_{n=0}^{\infty} a_n (e^{j(t(\omega_t + n\omega_m) + \theta_t + n\theta_m)} + e^{j(t(\omega_t - n\omega_m) + \theta_t - n\theta_m)}) \end{aligned} \tag{2.8}$$

In the frequency domain, the spectrum can be written as

$$X(\omega) = \begin{cases} Ba_n e^{j(\theta_t + n\theta_m)} & \omega = \omega_t + n\omega_m \text{ , } n\epsilon\mathbb{Z} \\ 0 & otherwise \end{cases} \tag{2.9}$$

Setting $t_p = T/2$ (for a 50% duty cycle) in (2.6) yields

$$a_n = \frac{2A}{n\pi} sin(\frac{n\pi}{2}) \tag{2.10}$$

which equals zero for even harmonics. On the other hand, setting $t_p = T/4$ (for a 25%

duty cycle) yields

$$a_n = \frac{2A}{n\pi} sin(\frac{n\pi}{4}) \tag{2.11}$$

which only equals zero every four harmonics. This allows us to divide the frequency sample of the second harmonic by that of the first to isolate the modulation phase:

$$\begin{aligned} \frac{X(\omega_t + 2\omega_m)}{X(\omega_t + \omega_m)} &= \frac{Ba_2 e^{j(\theta_t + 2\theta_m)}}{Ba_1 e^{j(\theta_t + \theta_m)}} \\ &= \frac{2Asin(\frac{2\pi}{4})}{2\pi} \frac{1\pi}{2Asin(\frac{1\pi}{4})} e^{j(\theta_t + 2\theta_m - \theta_t - \theta_m)} \\ &= \frac{1}{\sqrt{2}} e^{j\theta_m} \end{aligned} \tag{2.12}$$

A similar calculation with the first and second negative harmonics as well as the harmonics of the other 7 tones in the burst will yield the same result. The results of these 16 calculations for $e^{j\theta_m}$ can be weighted by the magnitude of the involved measurements and averaged to increase SNR. Then, the normalized $e^{j\theta_m}$ can be used to cancel the modulation phase in all the harmonics of all the tones in the burst. Finally, the harmonics around each received tone can be averaged to obtain an estimate for that frequency sample of the scatterer.

Fig. 2.6 gives a simplified diagram of a modulated scatterer as it is used in this application.

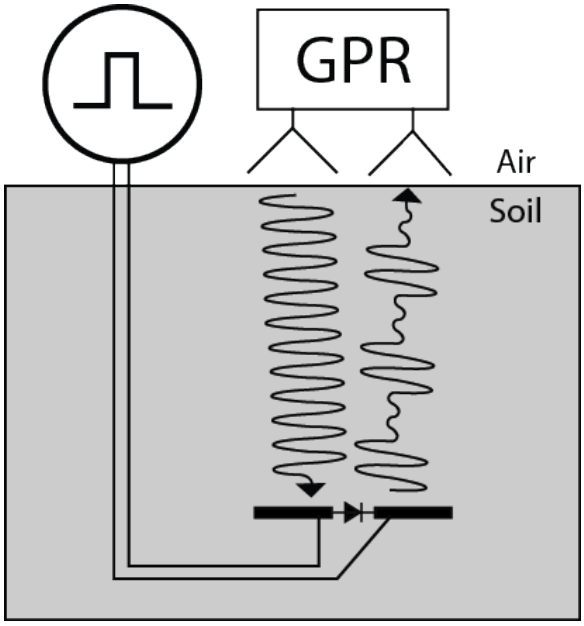


Figure 2.6: Modulated scatterer used with GPR. The function generator produces a periodic pulsed signal and feeds it to the scatter underground. The modulated voltage across the diode connects and disconnects the two halves of the dipole, modulating the magnitude of the backscatter to the GPR. There is no direct wired connection between the GPR and the function generator.

2.9 Software

Using gr-modtool, a new GNU Radio OOT module was created and filled with custom blocks. Fig. 2.7 shows a flowgraph of the blocks connected together to implement the GPR.

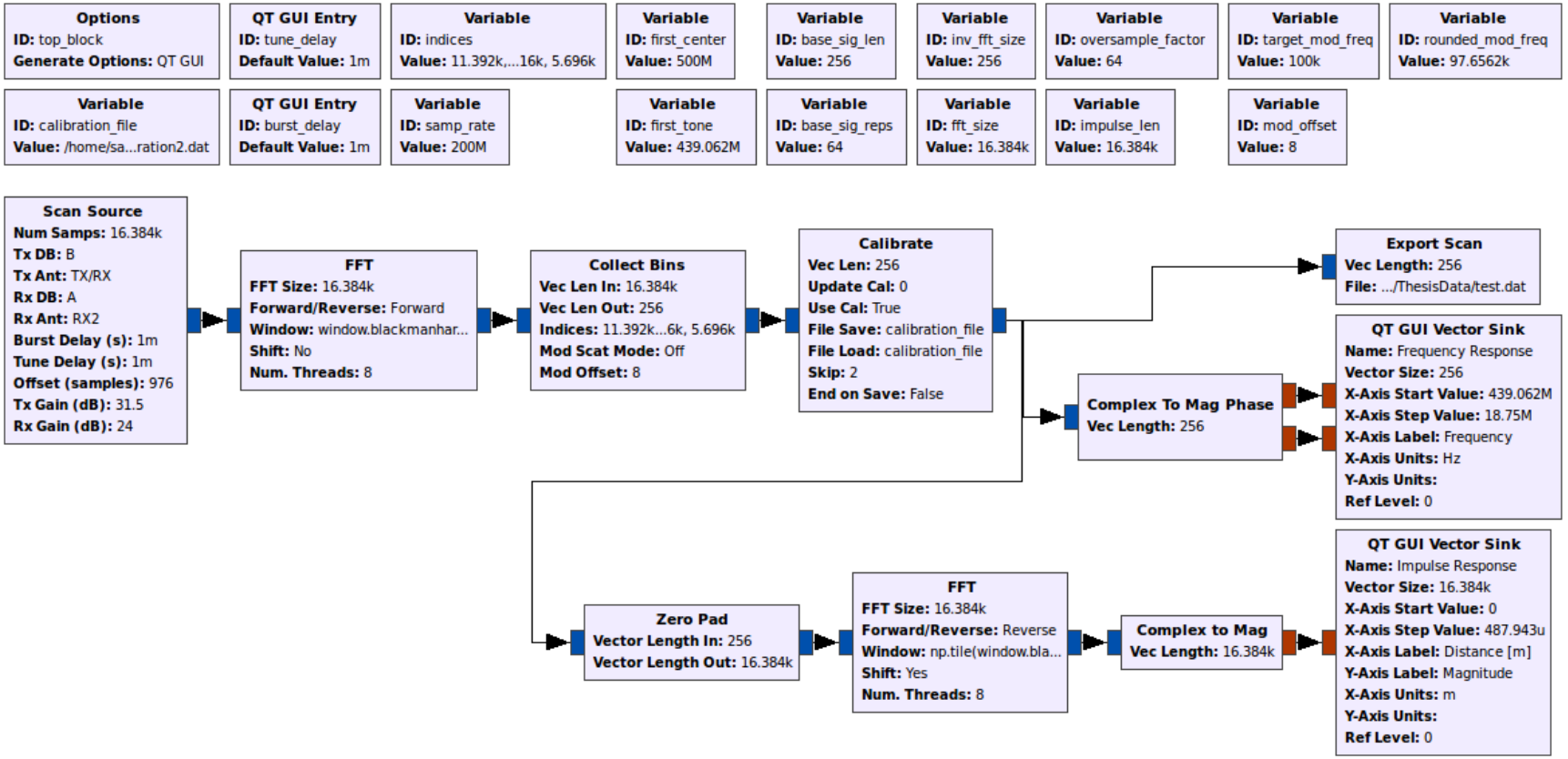


Figure 2.7: A version of the implemented GNU Radio flowgraph on the host, as displayed by GNU Radio Companion. Each block is labeled at its top. The series of blocks connected by arrows illustrate the flow of data through the application, from its acquisition, to its processing, to its storage and display. In this flowgraph, data is passed between blocks in vectors of samples. All passed data is complex, except that to the Vector Sink blocks. The unconnected blocks around the perimeter are input parameters, calculated dependent variables, and configuration menus for GUI widgets such as buttons and text entry boxes.

The leftmost connected block, Scan Source, is a custom block that calls UHD to tune the RF front-ends to each center frequency, transmit bursts, receive bursts, and query the current location of the GPR. Scan Source trims the unstable samples from the received burst, tags it with metadata indicating the current center frequency and location, and passes it to the FFT block.

The FFT block is a default GR block that takes advantage of years of community efforts to establish its efficieny and stability. It computes the DFT of the burst using the user-specified number of threads, and passes the result along with the burst's metadata tag to the Collect Bins block.

The Collect Bins block is a custom block that extracts the 8 frequency samples for the reflected tones from the input DFTs. In this case, 64 repetitions of the base, 256-sample signal are used in the burst, so the spectrum index positions of the 8 tones must be scaled by 64. The block respects the ordering of these positions, so the first four and last four are swapped to place the tones closer to DC in the middle. If the system is being used with modulated scatterers, a virtual switch is flipped and each sample is calculated from the frequency samples of the harmonics on either side, offset by multiples of the modulation frequency. The eight samples are then placed in their correct position in the main 500-5000 MHz frequency response based on their associated center frequency tag. Once samples for the whole frequency response have been collected, the locations for all the bursts are averaged, and the result is considered the location for the entire frequency scan. The full frequency response and this location metadata are passed to the Calibrate block.

The Calibrate block applies a calibration to the frequency response by dividing it element-wise by a frequency response measured and saved previously. The user can save a new response for calibration by pushing a button on the GUI at runtime which saves the current response. This response's file can then be selected and used for future runs.

From this point the frequency response and an oversampled version of the impulse response are displayed in real-time, and the frequency response is saved along with its lo-

cation metadata in a file. All this is handled by default blocks, except for the Zero Padding and Export Scan blocks, which are custom. Afterward, the saved data is imported to MATLAB for further processing and plotting.

2.10 FPGA Acceleration

2.10.1 Motivation

The GPR described above relies on the host application for primary data acquisition and processing operations. For each new location, the host must send the X310 a burst of samples, send a receive command, receive a burst of samples, send a tuning command to both radios, and repeat about 30 times. All of this communication must occur over the ethernet cable, with a time margin between commands to reduce the probability of late commands. Then the host must invest significant computational resources to process burst data in real time, or else waste a significant amount of storage space and process it later.

One way to improve the speed and stability of the GPR system is to migrate these operations to the FPGA onboard the X310 using RFNoC. This way, the ethernet cable is used only for programming, occasional commands, and streaming processed data back to the host. Defining operations in deterministic, synthesized circuits can also dramatically increase overall system stability. The probability of dropped packets or late commands can be practically eliminated, and timing margins can be safely minimized. The massively-parallel architecture of an FPGA is highly suited to this processing application, and the default image containing only radio controllers and digital converters leaves most of the FPGA's resources unused. This shift of computation to the FPGA also reduces the strain on the host, allowing it to focus more on other tasks or be replaced with a less powerful device.

2.10.2 RFNoC Design

A theoretical plan to reimplement the previously described GPR system using RFNoC blocks has been devised and partially carried out. An illustration of the planned flowgraph is shown in Fig. 2.8. Unfortunately, due to time constraints, only the first block, RFNoC: Vector Source was successfully implemented. The RFNoC: Scan block was partially implemented. All other RFNoC blocks are theoretical.

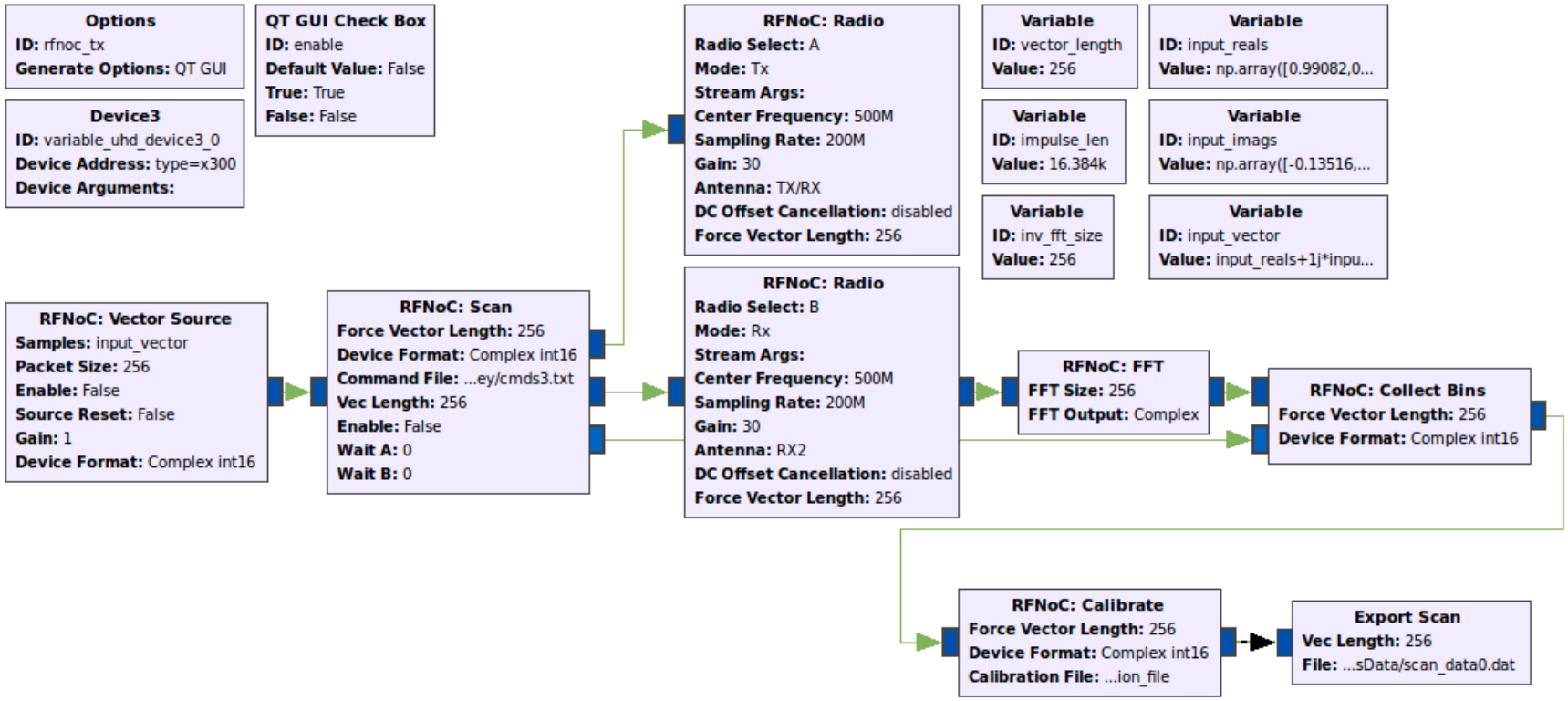


Figure 2.8: The theoretical RFNoC version of the GPR flowgraph, as displayed by GNU Radio Companion. Dummy blocks are used for unimplemented blocks. The last arrow to the Export Scan block represents the flow of processed data to the host over ethernet.

The leftmost connected block, RFNoC: Vector Source, allows the user to define an arbitrary, complex waveform and load it into the FPGA's RAM once when the application is first run. When the user sends the command to enable this block, it will repeatedly stream the samples of this vector until commanded to stop. These samples can be delivered uninterrupted to any other RFNoC block on the FPGA at 200 MS/sec, the maximum digital bandwidth of the X310.

Next, these samples are fed into the RFNoC: Scan block, which replaces the Scan Source block in the host version of the flowgraph. Since FPGA circuits cannot directly query the current location, it must be continuously written to FPGA registers by the host thread. Inside the RFNoC: Scan block is a finite-state-machine (FSM) that is controlled by looping through a list of instructions stored in FPGA RAM at runtime. Instructions are all 64 bits and fall into four categories. Wait instructions set the block in the wait state, in which it idles until the block's internal timer reaches some specified time in the future. Burst instructions set the block into the burst state, in which the samples are allowed to flow from the input to the output. A burst instruction specifies the number of samples to pass before reading the next instruction. It also specifies a time "lead," which sets the timestamp for the passed samples to a time offset into the future, according to the radio's internal timer. Command instructions set the block into the command state, in which a specified number of subsequent instructions are sent as commands to one of the radio controller circuits. Commands retune the RF-front end or reset the radio controller's timer. These commands are also tagged with a timestamp, and an extra bus is added to the default X310 HDL to connect the radio timers to the scan block for reference. Note that this extra bus would not be possible if UHD were closed-source.

A 64-bit command consists of two, 32-bit parts: the address and the data. The address specifies to which bus or register in a block the data should be set. This data eventually controls various hardware components in the X310 and daughterboards through communication protocols such as SPI and I2C. The process to generate the specific commands

necessary to properly operate the front-ends is complicated and would be difficult to recreate in HDL. Therefore, all commands are generated beforehand and embedded within the instruction list, which is loaded into FPGA RAM.

This is achieved by modifying UHD such that it prints out all the commands that it sends to the radio controllers, as it sends them. Then the Scan Source block in the host version of the flowgraph is modified to print out a placeholder macro for the corresponding instruction whenever it requests that UHD send a burst or command to the radios. When the host version flowgraph is run, it prints out the list of instructions neccessary to recreate the same operations in the RFNoC version, with the encoded commands inserted in the correct order, as shown in Fig. 2.9. Placeholder macros are printed for instructions so that the host code can replace them with encoded parameters of the user's choice at runtime. After saving this to a file and trimming it to the appropriate sequence, it can be loaded by the RFNoC: Scan block.

This scan block must be connected to the RFNoC versions of the Radio blocks to avoid passing samples all the way to the host and back. The host thread for the RFNoC RX Radio block has no input, so this must be added. This way, it can be properly connected in the flowgraph. The block's FPGA circuit already has an input and output.

As in the host version of the flowgraph, the received bursts are then passed to an FFT. The default RFNoC: FFT block wraps Xilinx intellectual property (IP) designs, to instantiate an efficient FFT circuit. Unfortunately, RFNoC allows a maximum FFT size of 4096 in order to ensure the result can be properly streamed back to the host. Therefore, a custom FFT block is necessary to wrap the Xilinx IP if larger FFTs are to be used. The size limit is not relevant for this application, because the vector will be reduced in size and repackaged by the RFNoC: Collect Bins block before being sent to the host. This custom block will also trim the received burst, as in the host flowgraph, since this can't be done by the default radio block.

The RFNoC: Collect Bins block operates similarly to the host version, with one nec-

```
{CMD_INFO,0},{CMD_LEAD},
{170},{0x613f4034},
{170},{0x6a00001b},
{170},{0x1d007e42},
{170},{0xa0000},
{192},{0xf07c0000},
{170},{0x1ef0645},
{226},{0xfffff829},
{227},{0x5d6},
{224},{0xffff15ca},
{225},{0xffffef11},
{CMD_INFO,0},{CMD_LEAD},
{170},{0x613f4034},
{170},{0x6a00001b},
{170},{0x1d007e42},
{170},{0xa0000},
{192},{0x7cf000},
{170},{0x1ef04aa},
{WAIT_INFO},{WAIT_TIME_B},
{BURST_INFO},{BURST_LEAD},
{WAIT_INFO},{WAIT_TIME_A},
```

Figure 2.9: Intructions for RFNoC: Scan block. (a) shows an excerpt of actual instructions directly printed by the host version of the application. Following each command instruction are the commands to send to a radio, with the address printed in decimal and the data printed in hexidecimal. This set of instructions commands the transmitter and receiver to tune to the first center frequency, waits for a number of cycles for the LOs to settle, sends a burst to be transmitted and commands that a burst be received, and waits for the burst to complete. The complete list of instructions includes these instructions, followed by those for scanning the rest of the center frequencies using unique command sequences.

essary adjustment. RFNoC does not allow metadata tags to be attached to data as it flows through the FPGA. Therefore, the metadata indicating the center frequency and location of each burst is passed as a separate data stream directly from the RFNoC: Scan block to the RFNoC: Collect Bins block. This metadata arrives early and is stored in a FIFO until the FFT data also arrives, so that they are paired and processed together. Each full frequency response is packed with its average location metadata in the same vector and sent to a custom Calibrate block.

The RFNoC: Calibrate block functions almost identically to the host version, dividing each vector by a fixed vector loaded at runtime, and passing the result and packed metadata to the host, where it can be handled as it was in the host version of the flowgraph.

CHAPTER 3

RESULTS

3.1 Basic Measurements

The following tests characterize the performance of the system under the operating conditions imposed by this GPR design. A minimum of 30 dB of external attenuation was inserted with RF attenuators for all loopback tests to ensure that the receiver could not be damaged [26]. See Appendix A for photos of some of the equipment used.

3.1.1 Calibration

Figure 3.1 shows the uncalibrated discrete frequency response of a loopback with a coaxial cable. In addition to the 30 dB of external attenuation, the TX and RX chains were each set to include 3.5 dB of internal attenuation. The vertical lines separate the samples into the sub-bands of 8 tones that were collected at each center frequency. The bump-like shape in the magnitude of each band is due to the low-pass filters of the transmitting and receiving RF front-ends. Because the tones of each burst are shifted slightly to the right for IQ-imbalance mitigation, the 4th sample of each sub-band is slightly closer to the center of the RF filters than the 5th, causing it to have a greater magnitude. One can see that the phase of each tone increases roughly linearly from one center frequency to the next, as expected for a loopback. This indicates that LO synchronization of phase and frequency was successful.

After the frequency response is saved and divided from subsequent measurements, the magnitude and phase responses become approximately flat lines at zero, as shown in Fig. 3.2. This calibrated response is approximately what one would expect from an ideal loopback of zero-length. In Fig. 3.3, after calibration, a "line stretcher" device has been inserted to precisely increase the signal's path length by 30 cm, resulting in a corresponding shift of

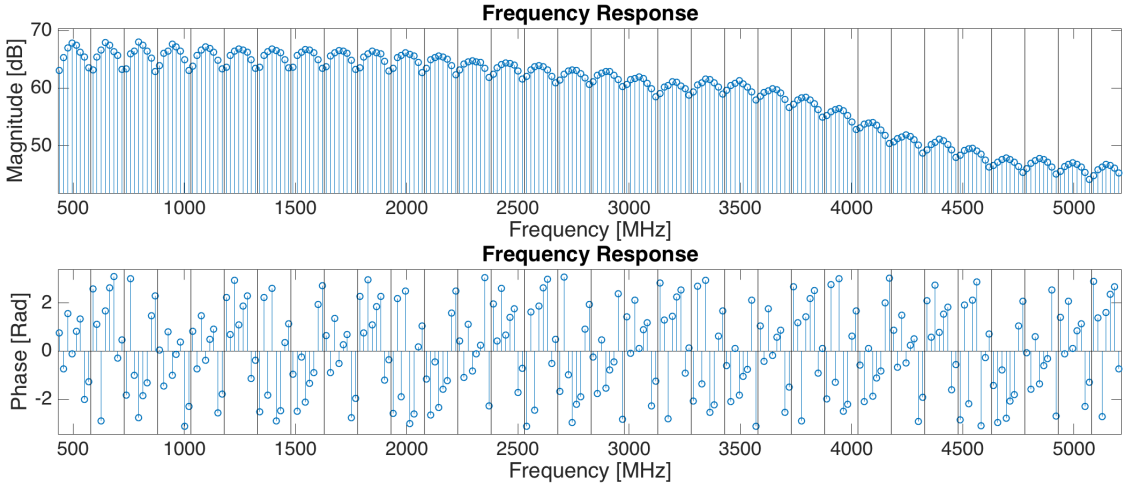


Figure 3.1: Uncalibrated scan of frequency response.

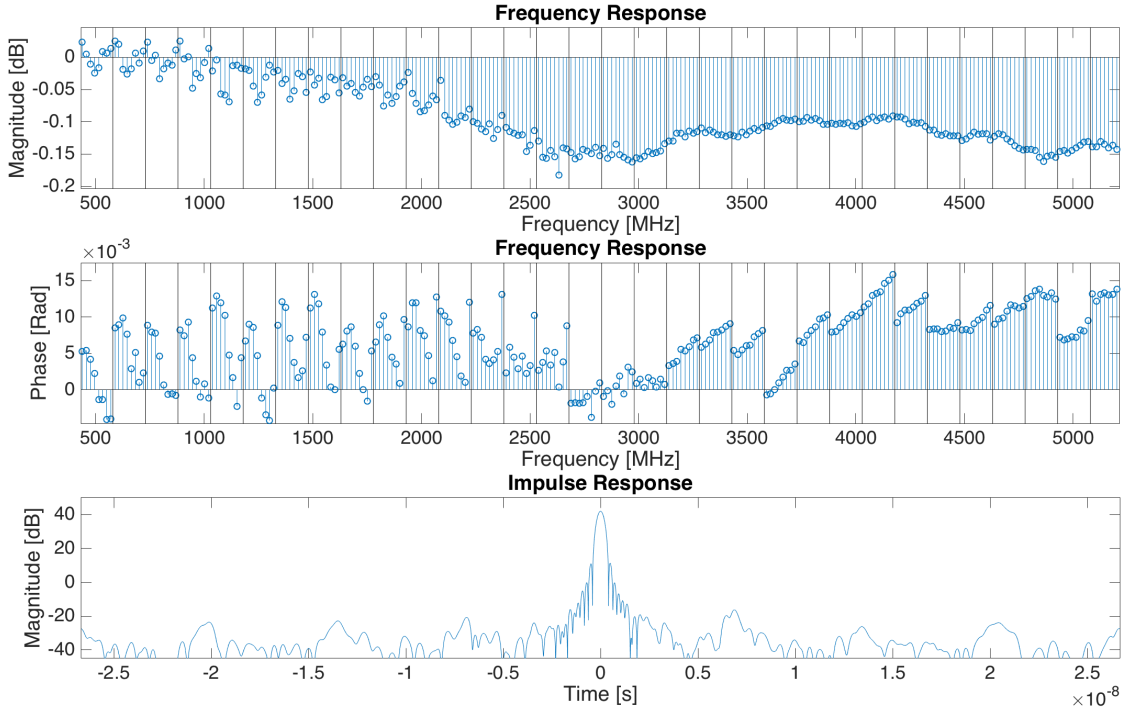


Figure 3.2: Calibrated scan of frequency response and resulting impulse response. The impulse response is oversampled by a factor of 256 and has a peak located at 4.069×10^{-13} seconds.

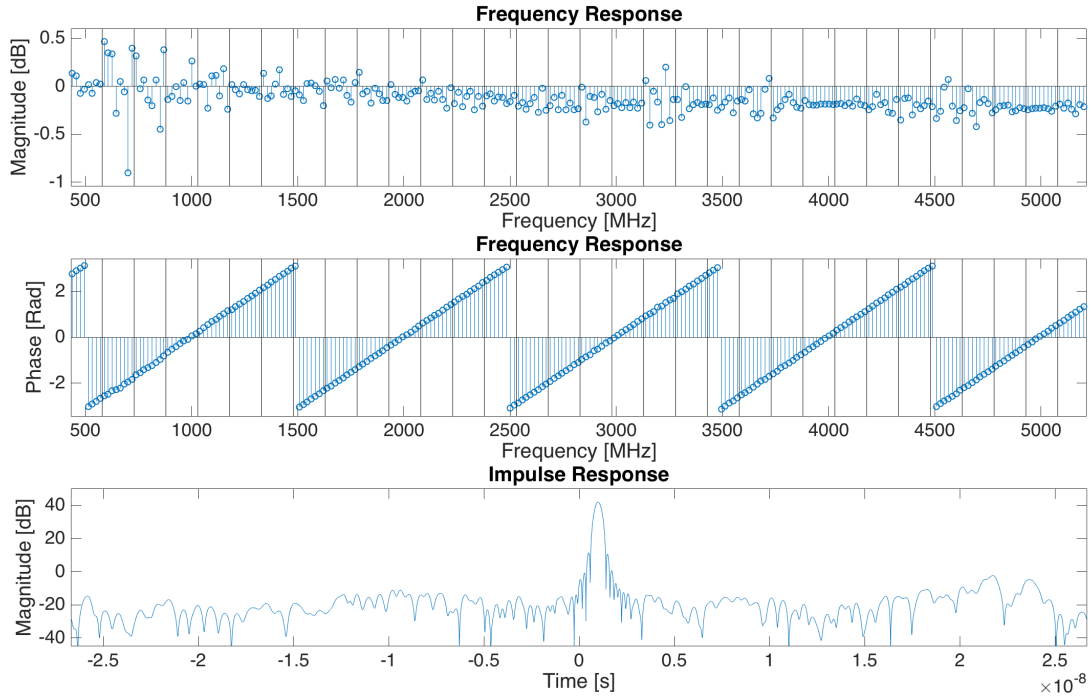


Figure 3.3: Calibrated scan of frequency response and resulting impulse response, after a path length extension of 30cm. The impulse response is oversampled by a factor of 256 and has a peak located at 1.001×10^{-9} seconds.

the impulse in time. The change in path length can be divided by the change in time of the impulse response's peak to calculate the velocity of propagation in the line stretcher:

$$\frac{30 \times 10^{-2}}{1.001 \times 10^{-9} - 4.069 \times 10^{-13}} = 1.0001c \tag{3.1}$$

where $c = 299,792,458$ [m/s] is the speed of light in a vacuum. Since the line stretcher is air filled, this computed velocity is essentially correct.

3.1.2 SNR

The SNR for the GPR was measured as a function of several parameters. SNR is defined here as the ratio of the peak squared magnitude of the impulse response of a point target to the average squared magnitude of the half of the impulse response furthest away from the

peak.

$$SNR = \frac{|h[n_{max}]|^2}{2/N \sum_{n=n_{max}+N/4}^{n_{max}+3N/4} |h[n \mod N]|^2} \tag{3.2}$$

where $h[n]$ is the discrete impulse response of length N and n_{max} is the index of its peak.

The TX and RX chains on the UBX are each equipped with internal, adjustable attenuators, with a maximum attenuation setting of 31.5 dB. In UHD, 31.5 dB of attenuation is considered 0 dB of gain, and the gain is adjustable from 0 to 31.5 dB. Figure 3.4 plots SNR vs. gain for each chain as the other is held at a setting of "0 dB."

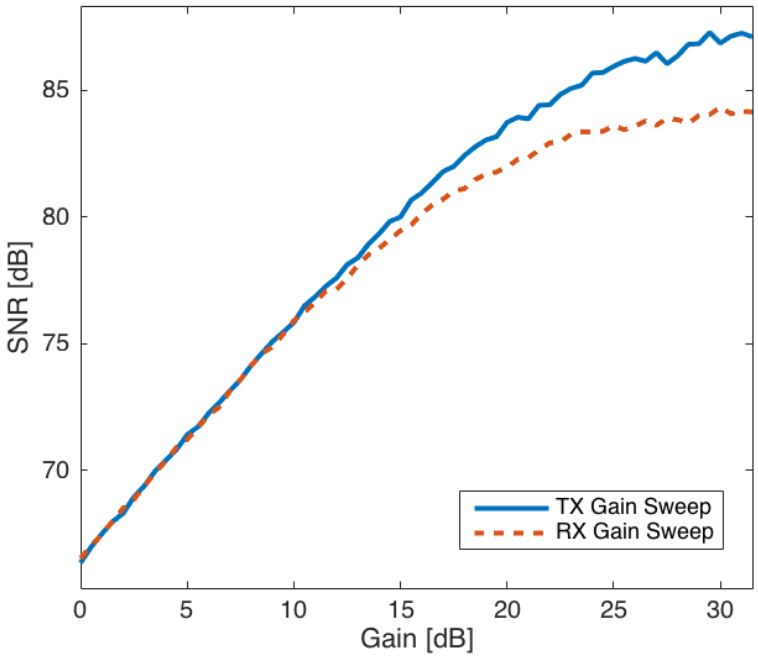


Figure 3.4: SNR vs. gain for the TX and RX. Each sweep holds the other gain at "0 dB" with 30 dB of external attenuation which keeps the receiver from saturating and causing distortion. This causes peak SNR to occur as each approaches maximum gain. Diminishing effects of increasing the gain are seen by the flatting of the SNR curves for gains above approximatively 15 dB.

The total SNR was plotted vs. DFT length in Fig. 3.5, where the DFT length was adjusted from 1 to 64 repetitions of the original, 256-sample signal. The slope of approximately 3 dB per doubling of DFT/burst length confirms the system is behaving as expected.

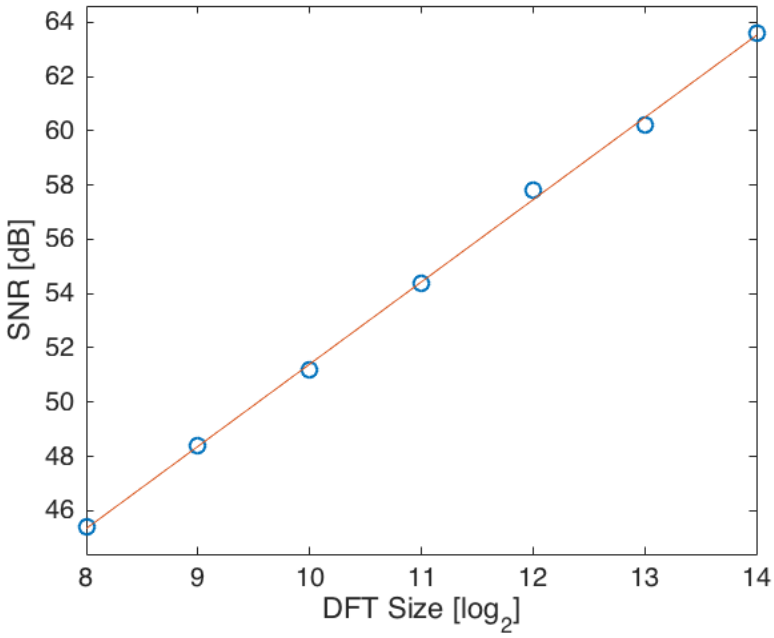


Figure 3.5: SNR vs. DFT length. The fitted line has a slope of 3.0286 dB per doubling of DFT/burst length.

The modification to the LNA selection code in UHD significantly stabilized received signals, which improved the SNR as shown in Fig. 3.6.

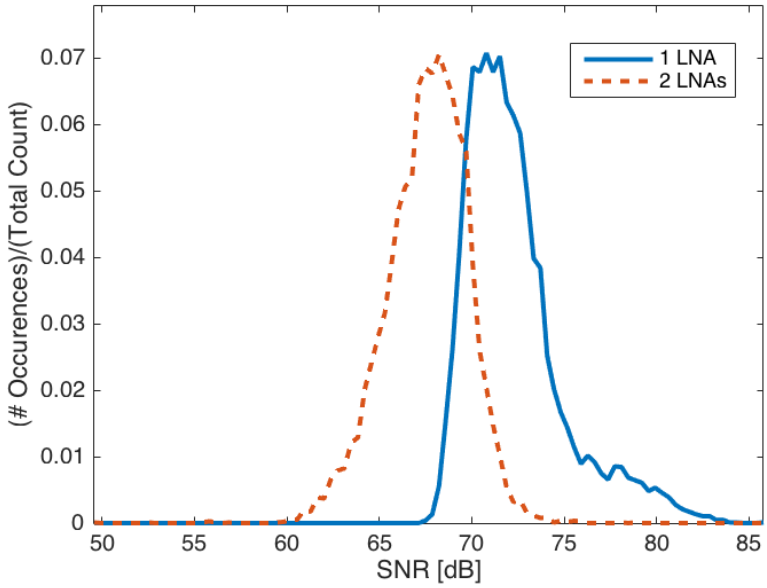


Figure 3.6: Normalized histograms of SNR before and after UHD modification. Each experiment measured SNR about 10,000 times over the course of 20 minutes, was immediately preceded by a calibration, used a 16,384 point DFT, used 30 dB of external attenuation, and used maximum internal gain (no internal attenuation). The only difference was the use of a single LNA instead of two. The average increase in SNR was 4.7 dB.

3.1.3 Coupling

The average received power of all tones at each frequency was used to measure coupling effects between the TX and RX chains. First, power was measured for a coaxial loopback from one daughterboard to the other through 60 dB of external attenuation, for reference. Then a matched load was attached to all four ports of the X310. Intraboard coupling was measured as the received power when receiving on the same board that is transmitting, and interboard coupling was measured as the received power when receiving on a different board within the same X310. Both are plotted as a function of center frequency, relative to the 60 dB loopback in Fig. 3.7. Tones were extracted using a 16,384 point DFT. The TX and RX gains were set to their maximum settings.

The intraboard coupling is about 10 dB stronger than the 60 dB loopback, and the interboard coupling is about 40 dB weaker the the 60 dB loopback. This direct coupling

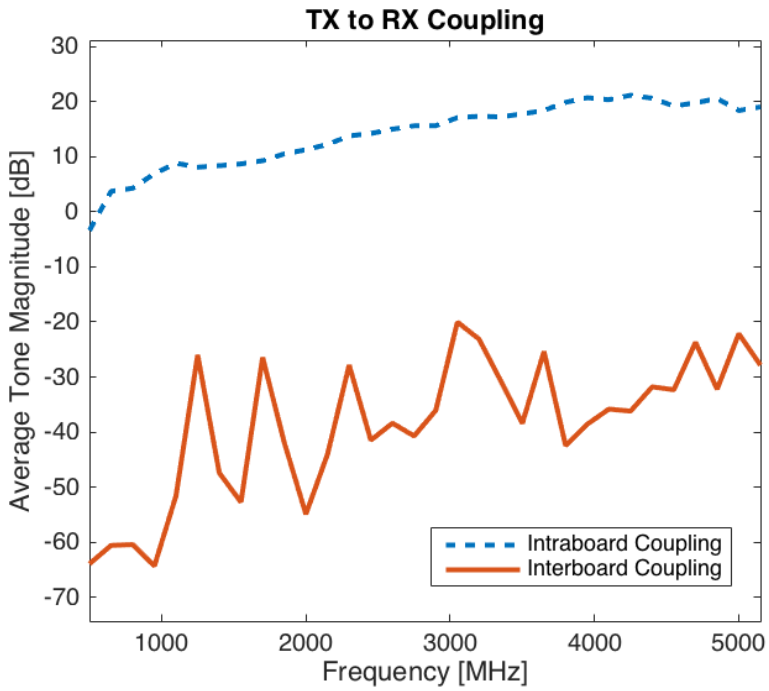


Figure 3.7: Direct coupling from TX to RX on the same daughterboard (intraboard) and separate daughterboards (interboard) within the X310, relative to a loopback with 60 dB of attenuation. A DFT length of 16,384 and maximum internal gain were used.

can compete with signals from targets, so separate boards should be used for TX and RX if possible to reduce the effect by about 50 dB. The coupling could be partially removed by calibration, but the dynamic range of the system would still be degraded.

3.2 GPR Measurements

The X310 was connected to two broadband, linear, co-polar antennas, made by Sustman [27], which are based on an antenna design by Kim and Scott [28]. The antennas were scanned over the surface of a 1.5 m deep pit of dry sand containing various test targets. For single dimensional scans, the path of motion was in the H plane of the antennas. Targets were oriented with their shortest dimension aligned with the depth axis and their longest dimension aligned in the E plane of the antennas and perpendicular to the page. Cylindrical target dimensions are given as diameter and height, respectively. Measurement locations were spaced horizontally 1 cm apart, 10 cm above the sand or highest target. The TX gain

was set to the maximum setting of 31.5 dB, and the RX gain was lowered by 7.5 dB down to 24 dB to avoid overloading the RX chain. A DFT length of 16,384 was used for extracting tones.

Impulse responses from each location were averaged and placed beside those of adjacent locations to form B-scans. The impulse responses on the left and right edges were averaged and subtracted from those in the middle to remove common signals. No SAR focusing was attempted. The magnitude of the real part of each image is plotted in decibels in figs. 3.8-3.11 for a variety of targets. The range axis is oversampled by a factor of 8 for a range resolution of 3.9 mm, assuming a dielectric constant for sand of 1. The top layer of horizontal reflections are from scattering at the surface of the sand that is not removed by subtracting the signals at the edges. Hyperbolic shaped reflections are due to scattering from the targets as the antennas move closer to and then further from each.

In Fig. 3.8, plots for an 11 cm diameter metal sphere are presented. The bottom of the sphere is 12 cm above the surface in Fig. 3.8a and sunk about 1 cm into the surface in Fig. 3.8b. The top of the sphere is 2 cm below the surface of the sand in Fig. 3.8c and 28.5 cm in Fig. 3.8d. The reflected signal from the sphere is much weaker when it is 28.5 cm deep, so the background reflections are relatively stronger in Fig. 3.8d. Because the antenna height was always adjusted to be 10 cm from the highest object (sand or sphere), the surface of the sand is seen at ranges of about 68, 55, 45, and 45 cm in plots a, b, c, and d of Fig. 3.8 respectively. When the sphere is submerged in plots c and d, its reflection can be be seen as a stack of hyperbolas at apparent ranges of 45.2-52.5 and 87.7-95 cm, respectively. These ranges assume the relative permitttivity of sand is 1. The actual relative permittivity can be approximated by comparing their apparent ranges (measured from the tops of their reflections) to their relative depths measured with a meter stick:

$$\epsilon_r = (\frac{r_2 - r_1}{d_2 - d_1})^2 = (\frac{87.7 - 45.2}{28.5 - 2})^2 = 2.57 \tag{3.3}$$

where r_1 and r_2 are the apparent ranges and d_1 and d_2 are the measured depths. This number is approximately correct for typical dry sand [29].

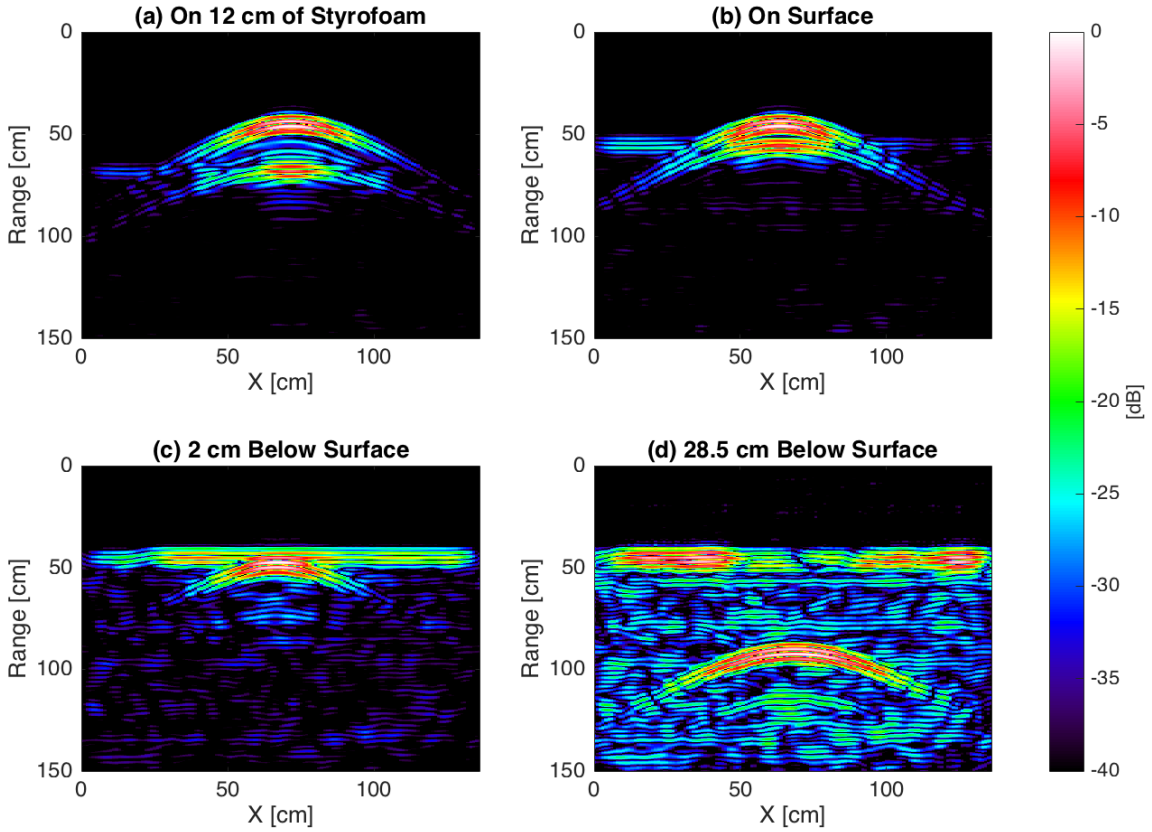


Figure 3.8: Pseudo-color plots of the radar response for a metal sphere with diameter of 11 cm at various heights and depths.

In figs. 3.9 and 3.10, plots for a variety of buried targets are presented. The reflections from the surface of the sand are at about 45 cm, and the reflections from the targets can be seen at deeper depths near $X = 65$ cm.

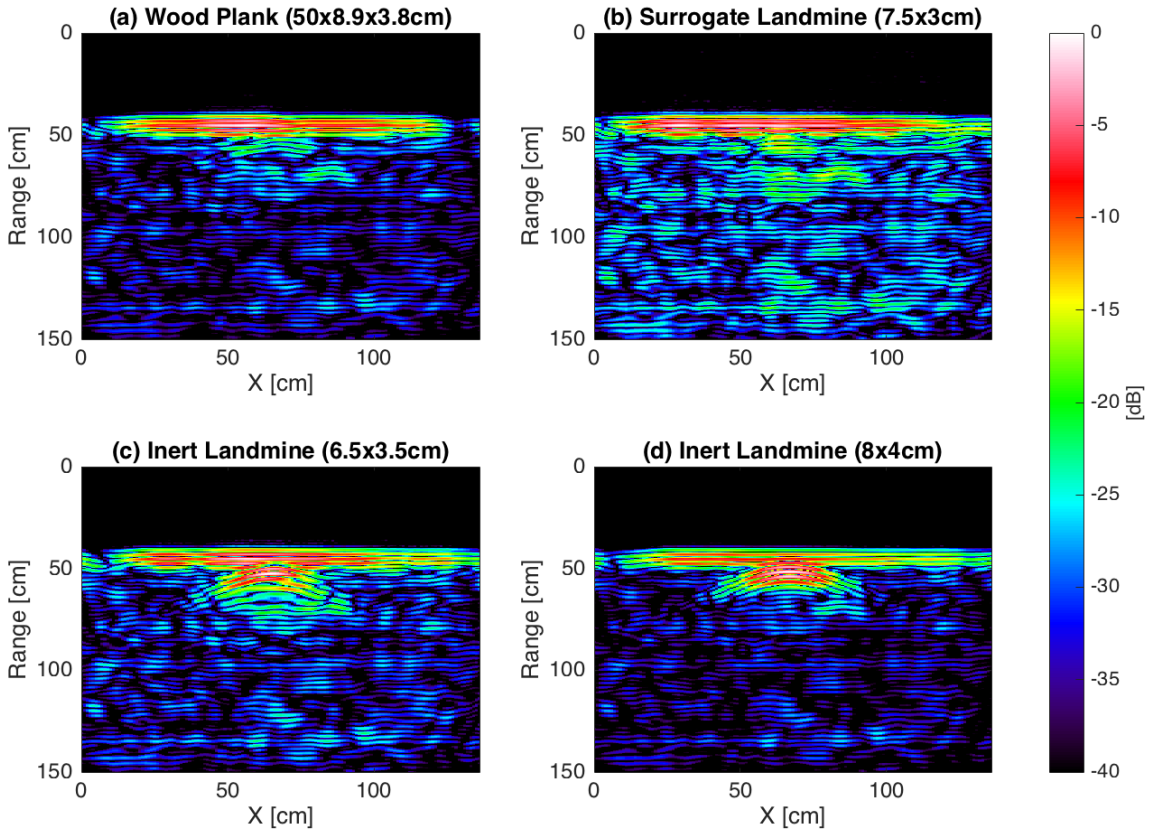


Figure 3.9: Pseudo-color plots of the radar response for various objects buried 2 cm beneath the surface.

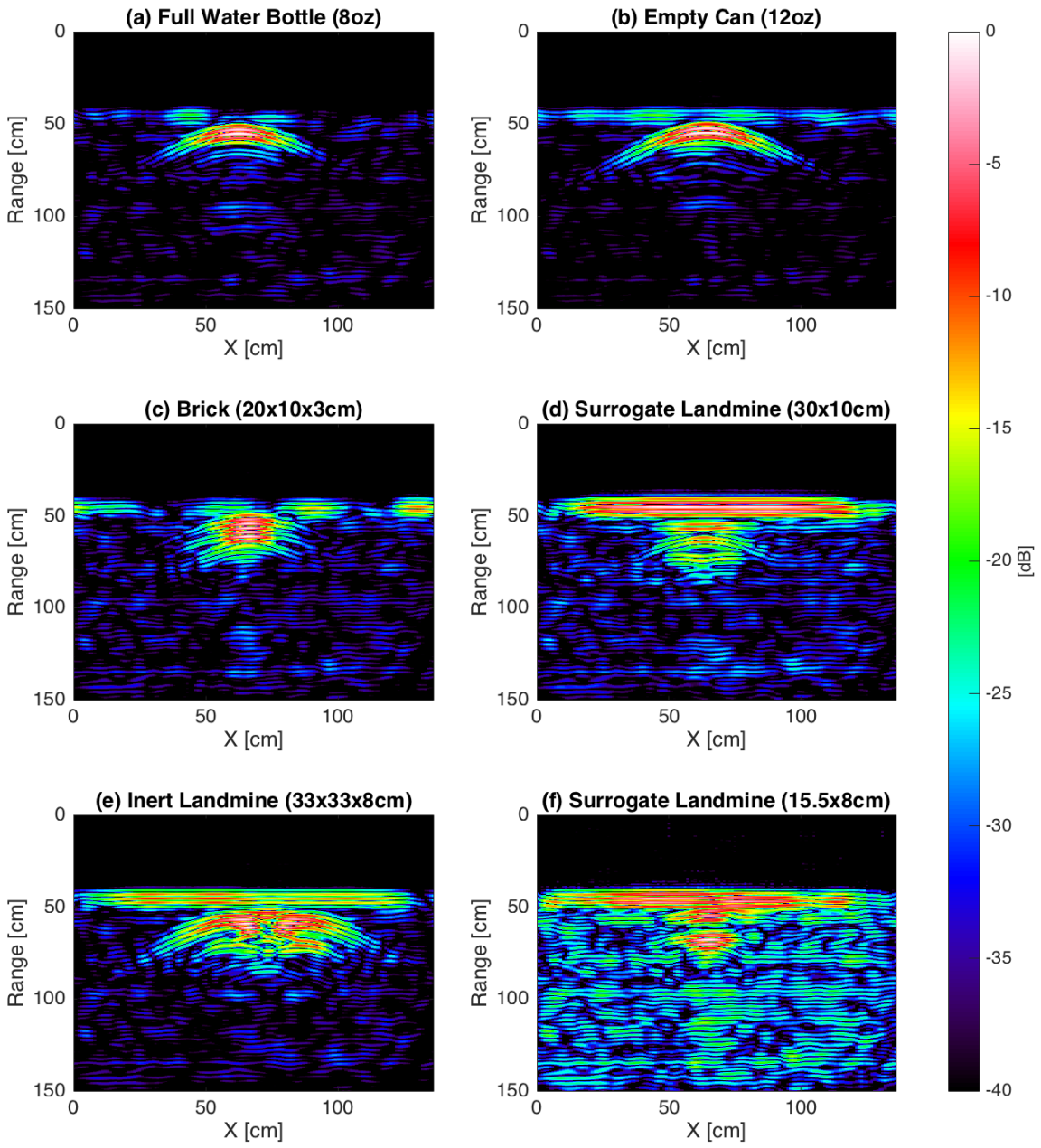
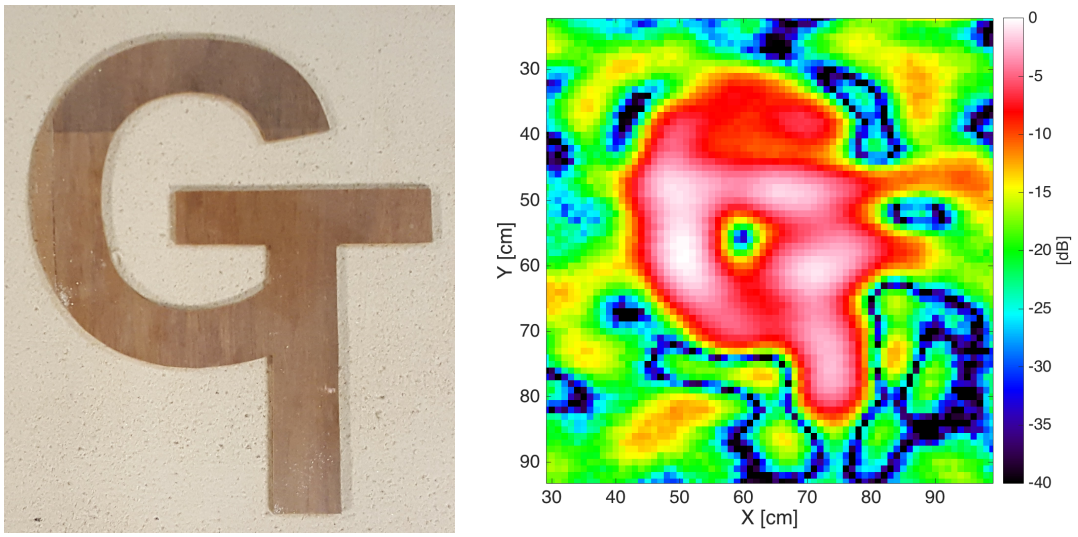


Figure 3.10: Pseudo-color plots of the radar response for various objects buried 5 cm beneath the surface.

A two-dimensional scan over the GT logo target shown in Fig. 3.11a was taken with the target buried 2 cm below the sand. A pseudo-color graph is presented in fig 3.11b for a slice of the response at the depth of the target. The general shape of the target is somewhat apparent even in this raw GPR image. A much more accurate image could be obtained by further processing of the data [30].



(a) Photograph (b) Horizontal slice of scan of logo 2 cm below surface

Figure 3.11: Wood cutout of Georgia Tech logo (46x38.5x0.5cm)

3.2.1 Modulated Scatterers

The following GPR measurements were performed under the same conditions as those above, with the following differences. The host flowgraph was set to modulated scatterer mode with a modulation frequency of about 97 kHz, which shifts the 1st and 2nd harmonics of each modulated tone in the DFT by exactly 8 and 16 frequency samples, respectively. An existing modulated scatterer borrowed from a previous project by Ricardo and Scott [6], is shown in Fig. 3.12.

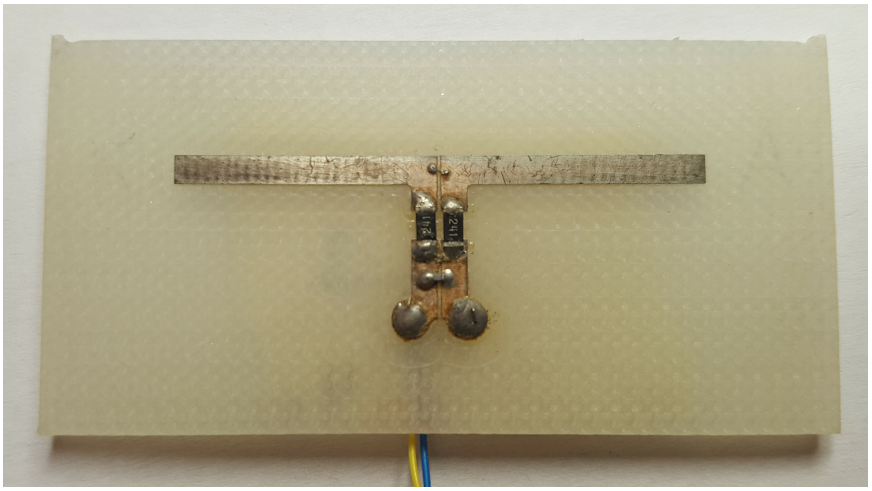


Figure 3.12: The modulated scatterer [6]. The dipole is 2.25x40.3 mm. The gap is bridged by a tiny diode at the top and a capacitor near the bottom. Two resistors are in series on the sides. The feed wires at the bottom are connected on the underside.

An Agilent 33522A function generator produced the modulation signal described in section 2.8 with peaks of 0V and 3V and fed it to the buried scatterer. The result is shown in Fig. 3.13. In these figures, the hyperbolic reflections from the modulated scatter are clearly apparent, but no other reflections are seen. The reflections from the surface of the sand are not modulated so they do not appear either. Notice that the SNR is very high, even when the scatterer is buried 37 cm deep. If this scatterer were not modulated and the GPR were operated in its normal mode mode, the SNR would be very low and the scatterer's reflections would be practically indistinguishable from the background reflections.

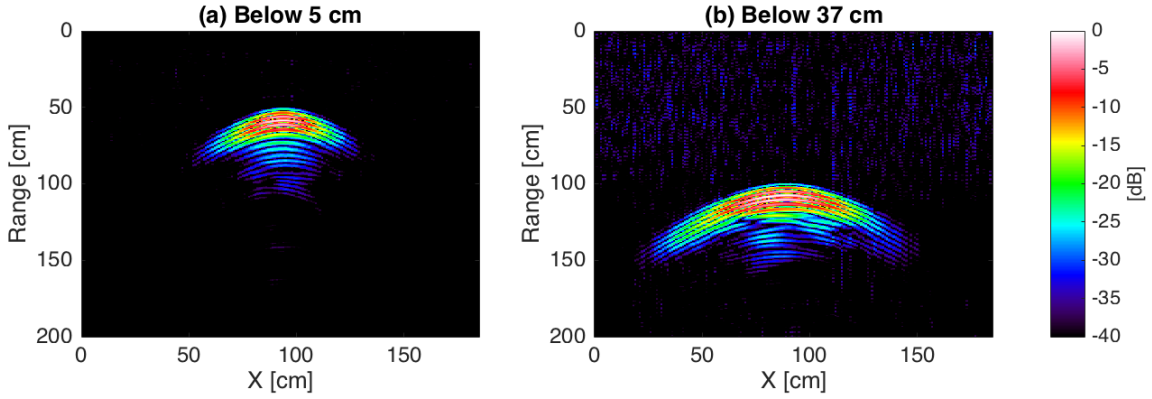


Figure 3.13: Pseudo-color plots of the radar response for the modulated scatterer buried at different depths below the surface.

3.3 Timing

To quantify the timing and stability of the system, 38,000 scan times were recorded over the course of a single run lasting about 11 hours. These times were sorted into a histogram, shown in Fig. 3.14.

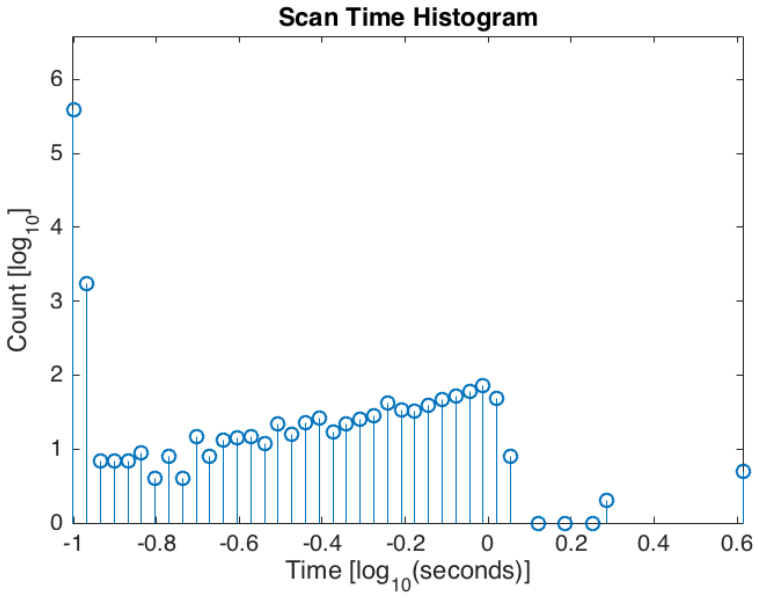


Figure 3.14: Histogram of time to produce each impulse response, on log-log scale.

A burst of 16,384 samples at 200 MS/s should only take about 82 ms. According to an Ettus Engineer, the local oscillators should be capable of retuning and locking in about 100 us [31]. This means that the theoretical minimum scan time for all 32 center frequencies is approximately

$$(\frac{16,384}{200 \times 10^6} + 100 \times 10^{-6}) \times 32 = 5.8 \text{ ms} \tag{3.4}$$

Considering the current average scan time of 100 ms, about 94% of the current scan process can be considered overhead.

By disabling one component of the application at a time, its cost in time can be roughly estimated. Disabling all blocks in the flowgraph besides the Scan Source block had no noticeable effect, suggesting that their computational requirements are not currently a limiting

factor on this host machine.

Eliminating timestamps from tuning commands reduces scan time by up to 25 ms, suggesting that approximately this much time is spent as the radio controllers wait for their timers to match command timestamps. Note that this modification makes the application unacceptably unstable. Completely eliminating tuning commands reduces scan time by an additional 16 ms, suggesting this time is spent negotiating the commands with the radio controllers and actually tuning the LOs

Reducing the DFT size by half, from 16,384 to 8,192, decreased the scan time by about 7 ms while still allowing the system to function normally. This suggests that the transfer of 32 full bursts to and from the X310 over ethernet takes about 14 ms. Completely eliminating bursts reduces scan time by an additional 40 ms, suggesting this time could be for negotiating the bursts, waiting for their timestamps to pass, and actually transmitting and receiving.

The sum of all these time estimates can account for about 95 ms, which is reasonably close to the total of 100 ms. Note that almost all delays are due to communication over ethernet and waiting for timestamps to occur. Theoretically, these delays could be virtually eliminated if the application were implemented on the FPGA, allowing the scan time to approach the theoretical minimum of 5.8 ms.

CHAPTER 4
THEORETICAL PROJECT EXTENSIONS

The progress established in this project invites enhancement and extension in several areas.

4.1 Additional Runtime Configurability

The current system allows several parameters to be adjusted as the application is launched, such as RF gain, burst length, calibration, and whether or not the system is in modulated scatter mode. A future version could allow users to adjust even more of the parameters and processing with a few clicks. Tone spacing, center frequencies, processing options could be adjusted experimentally with ease. Other GPR techniques could even be incorporated and compared side by side in real time.

4.2 Multiple Polarizations

Additional insight can often be derived from looking at GPR surveys of multiple polarizations. For the sake of this discussion, perpendicularly polarized sets of antennas are described as horizontal and vertical, even though they are both polarized parallel to the ground. Transmission and reception with two possible polarizations allows four possible polarization combinations, and certain subsurface features may reflect differently for each.

If the two ports of one daughterboard were connected to two horizontal antennas, and the other connected to two vertical antennas, all four combinations could be measured. Only one transmitter would be used at a time to avoid interference. Even with this precaution, transmitting and receiving on the same daughterboard at the same time would suffer from the coupling measured in Section 3.1.3. Theoretically, this coupling effect could be measured and subtracted out of measurements. However, the only way to preserve dynamic

range would be to use separate daughterboards for transmitting and receiving.

All four combinations could be measured by simply reconnecting antennas each time, but this is tedious. The user could also use an external RF switch, which would need to be controlled by either an extra controller device connected to the host or the array of GPIO pins on the X310.

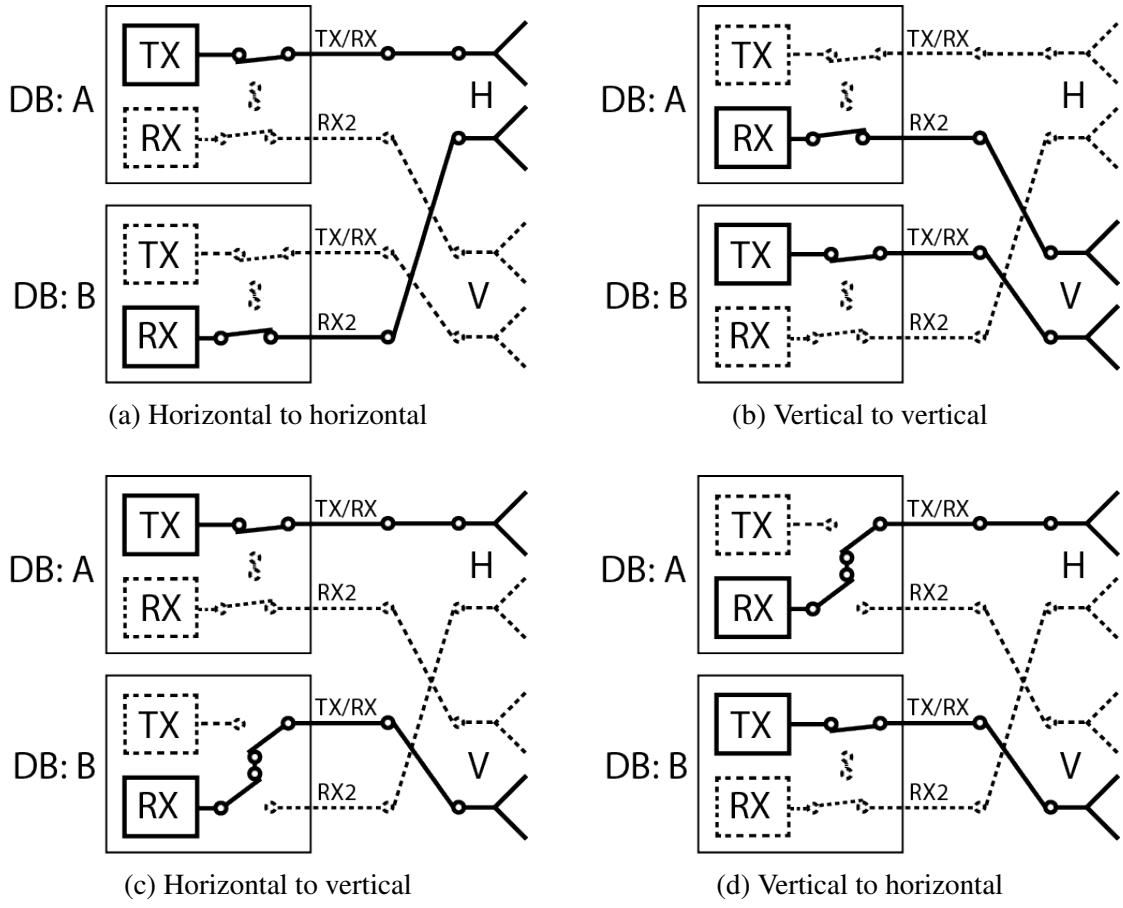
Alternatively, the antennas could be reconnected using the internal switches on the UBX daughterboards. These switches allow the RX chain of each UBX to be connected to either of the two ports on that board, called "TX/RX" and "RX2." The TX chain can only be connected to the TX/RX port, so while that port is connected to the RX chain, the TX chain is unusable. Despite this restriction, all four polarization combinations can still be measured without intraboard coupling, using only these switches and four static connections, as shown in Fig. 4.1.

Measuring the horizontal to horizontal combination and the vertical to vertical combination could be done back to back in software, without changing the switches. They could even be measured at the same time if they were always measuring different center frequencies at any given time to avoid coupling. The cross-polarization combinations would need to switch over the RX chains of the daughterboards to the TX/RX port one-at-a time. The polarization combination used for each burst could be attached to each burst as a metadata tag in the flowgraph.

4.3 GPS

The X310 may be installed with a GPS Disciplined Oscillator (GPSDO) that can receive GPS signals through an extra antenna port on the X310. These signals can be used to help align the clocks of multiple, remote devices in a multistastic configuration.

These signals could be also be parsed as NMEA sentences to acquire location data with which to tag range profiles. This could be useful in environments where precise GPR positioning for a survey is difficult.



(a) Horizontal to horizontal

(b) Vertical to vertical

(c) Horizontal to vertical

(d) Vertical to horizontal

Figure 4.1: Diagram of polarization combinations possible by flipping the internal switches of the daughterboards (DBs). Each DB switch is capable of two states, to connect the receive chain (RX) to either port TX/RX or RX2. The ports are statically connected to two horizontally polarized antennas (H) and two vertically polarized antennas (V). Active paths are bolded and inactive paths are dashed.

4.4 Advanced Modulated Scatterer Applications

The success of the modulated scatterer experiment opens up many possibilities. One obvious application for this system is to embed these scatterers in devices that need to be tracked precisely. If this system were connected to an array of antennas, it could measure the relative distance of the device from each antenna's location and triangulate the device's position. A stationary system in this configuration could even track another system performing a GPR survey. The two systems need not interfere with each other if they are programmed to scan with different arrangements of tones or synchronized to always be scanning at different center frequencies.

When using modulated scatterers, the GPR will only process signals that have, at some point, passed through the location of the scatterer. The signal may reach the scatterer by a direct or reflected path. The signal may return to the receiver by a direct or reflected path. The signal may even reflect multiple times between the scatterer and its surroundings. For any combination of these signal paths, transversed material will affect the received signal. Therefore, it should be possible to invert data acquired from multiple GPR locations to reconstruct a model for the subsurface structure and material properties. Multiple scatterers, operating at multiple locations and modulation frequencies, could provide multiple, independent sets of data to ease the inversion process. See Bolomey [32] for further exposition on such techniques.

Another useful feature of modulated scatterers is that they can reduce concerns about coupling between TX and RX chains. A single daughterboard could theoretically transmit a tone and only process its reflected harmonics, although the dynamic range would still be reduced because the RF front end would have to handle the coupled signal. This would allow all four polarization combinations to be scanned in only two transmitted bursts, without using any switches.

4.5 Motion Detection

4.5.1 Doppler Isolation

A frequency isolation effect similar to that achieved for modulated scatterers could theoretically be achieved for moving objects. Objects moving toward or away from the GPR will shift reflections in frequency due to the doppler effect. The frequency shift for each tone would be unique according to

$$\Delta f = \frac{2\Delta v_t}{v_p} f_0 \tag{4.1}$$

where Δf is the doppler frequency, Δv_t is the relative velocity between the GPR and a target, v_p is the velocity of propagation of the radio wave in the medium, and f_0 is the frequency of the transmitted tone. By collecting these shifted reflections and processing them coherently, it may be possible to generate a range profile for moving targets of a certain relative velocity.

A shift of n frequency samples in a DFT would correspond to a relative velocity of

$$\Delta v = \frac{ncr_s}{2f_0N} \tag{4.2}$$

where c is the velocity of propagation, r_s is the sample rate, f_0 is the frequency of the tone, and N is the size of the DFT. Assuming r_s is 200 MHz, f_0 is 500 MHz for the lower end of the operating band, and N is 16,384 as it is in the current system, a shift of a single frequency sample would require a relative velocity of 3662 m/s.

This velocity resolution could be improved by using a longer burst, but this would use up a considerable amount of time, memory, computation. If the target velocity is assumed to be small, the doppler shifts for lower velocities could be processed more efficiently. A single tone could be transmitted and received at a time and mixed down to a frequency of zero. Then a much lower RF bandwidth and sample rate would ignore all but the frequency samples closest to DC, capturing the velocities of interest while throwing away information

about higher velocities.

A remaining challenge would be to combine the doppler frequency samples near multiple tones, considering each frequency would each have a different doppler shift for each velocity. The additional phase from doppler shifting would also need to be canceled out. Although an object moving one direction would cause either a positive or negative doppler shift, a vibrating object, such as a motor, would cause both. Combining the phases from positively and negatively shifted frequency samples may allow cancelation of this doppler phase.

4.5.2 Impulse Response Differentiation

Motion could also be detected by tracking changes in the output of the current system. The point-wise difference between an impulse response and its preceding impulse response could reveal the changing range of a moving target on the other side of a barrier.

Alternatively, a set of sequential impulse responses could be stacked as rows in a matrix. Then a DFT could be taken along each column across the individual responses. This would produce a plot of range vs. movement frequency for the time period over which the set of impulse responses were measured. Targets moving at certain frequencies would be visible in the various frequency rows, while stationary clutter would be grouped together in the 0 Hz row. This could help to isolate movement of a certain frequency, such as the breathing or heartbeat of a buried disaster survivor, or the oscillation of machinery.

While the system currently produces impulse responses at an average rate of 10 Hz, the timing uncertainty of the system, as described in section 3.3, would introduce a fluctuating degree of noise to these measurements. These methods would become much more attractive if the measurement rate was increased and stabilized by transfering functions to FPGA, as discussed in Section 2.10.

CHAPTER 5
CONCLUSION

This research has demonstrated a viable approach to implementing a ground penetrating radar using off-the-shelf SDR hardware and open-source software. The challenges of phase and time synchronization have been solved by carefully selecting hardware and using the proper driver commands. Use of the full digital and RF bandwidths of the X310 has been enabled by dividing operation into bursts. The gain instability from alternating LNAs has been eliminated by a simple change to the driver source code. Shifts of phase and frequency in the multitone waveform have helped to optimize SNR. A system for detecting and locating modulated scatterers without a synchronization cable has been developed and successfully tested. A plan to improve all aspects of the project by reimplementing the project in HDL has been developed and partially carried out.

This GPR system has undergone a variety of tests, verifying its viability as a subsurface survey tool. Metallic as well as non-metallic objects have been imaged at various depths with reasonable levels of SNR. Realistic inert landmines have been shown to be clearly visible in GPR images. Modulated scatterers were located with impressive clutter rejection.

This system demonstrates many of the advantages of software defined GPR over a traditional hardware GPR, such as the ability to scan several frequencies at once, avoid the DC offset of the direct-conversion receiver, and easily collect and process the harmonic tones of modulated scatterers. This project also serves as an example of how SDR can allow an entirely new radio system to be developed in a relatively short amount of time without any actual RF design. Because this project consists entirely of software, it can be conveniently studied, replicated, and extended by new researchers, on new hardware, as it is released. This flexibility will prove invaluable for this project in the future as new features and applications continue to be explored.

Appendices

APPENDIX A

EXPERIMENTAL EQUIPMENT

Experimentation was conducted in the Van Leer building at the Georgia Institute of Technology in Atlanta, GA. Figure A.1 shows the X310 connected to the line stretcher for a loopback test. The line stretcher is a telescoping, air-filled waveguide geared to a hand-crank and distance counter. It is used to precisely adjust the path length of the signal.

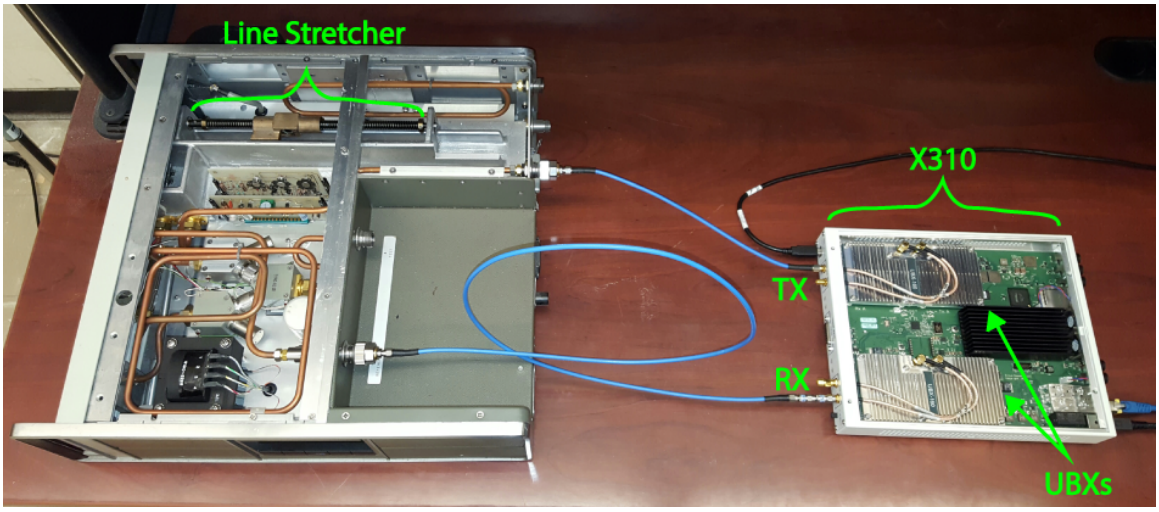


Figure A.1: Two UBX daughterboards inside of an X310 (top panel removed) transmit and receive signals through coaxial cables, attenuation, and a line stretcher. Attenuators are connected in series at the RX port. The line stretcher is housed with other electronics which are not used.

GPR scans were formed by mounting the X310 and antennas on a 3-axis positioning system. The positioning system is suspended over a sand pit in room C 140-I. Positioning was controlled by a Velmex VP9000 stepper motor controller and driver configured to drive and hold the positions of Superior Electric M091-FD09E stepper motors connected to belts to actuate each axis. The VP9000 was programmed by commands generated by a Python script and sent via RS232 from the host computer. Motors were geared to US Digital HB6M encoders, connected to a US Digital USB4 (encoder to USB interface) device. These encoder positions were queried by the host immediately after sending each pair of

transmit and receive commands to the X310 from the host computer.

Fig. A.2 shows the GPR antennas [28] suspended over the metal sphere used to create the plots in Fig. 3.8. Fig. A.3 shows the surrogate landmines used as subsurface targets.

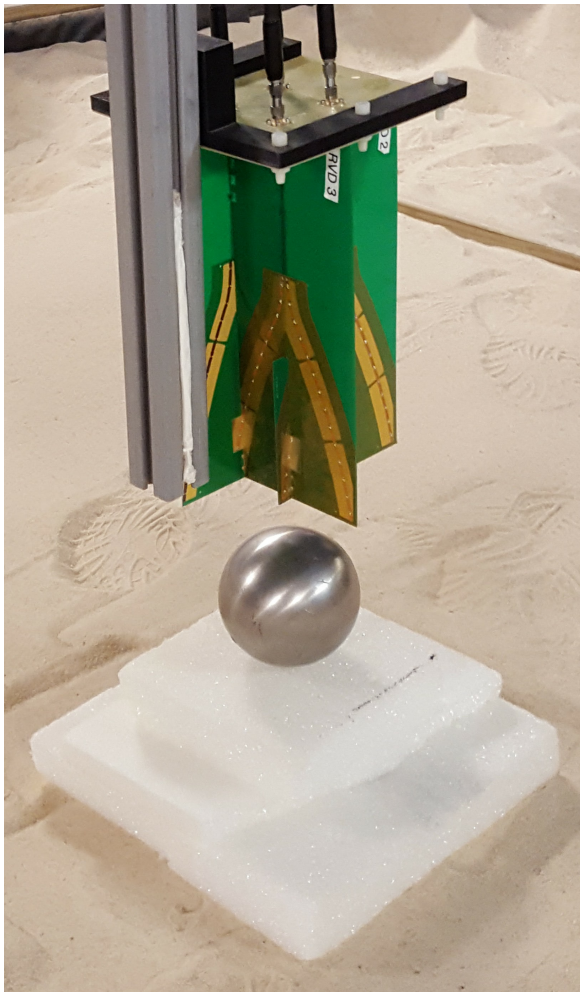
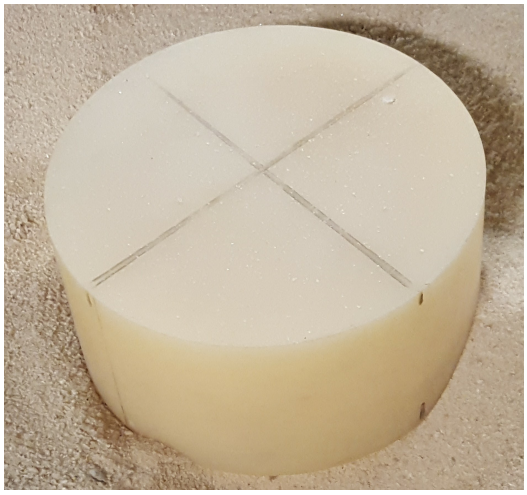


Figure A.2: Photograph of antennas with 11 cm sphere on 12 cm of styrofoam

(a) 7.5 x 3 cm



(b) 15.5 x 8 cm

(c) 30 x 10 cm

Figure A.3: Photographs of surrogate landmines used as targets. Dimensions are given as diameter and height respectively.

REFERENCES

[1] N. A. Markus Dillinger Kambiz Madani, "Software Defined Radio: Architectures, Systems and Functions," in. Wiley & Sons, 2003, ch. Introduction.

[2] D. J. Daniels, Ed., *Ground penetrating radar*, 2nd ed. Michael Faraday House, Six Hills Way, Stevenage, Herts., SGl 2AY, United Kingdom: The Institution of Electrical Engineers, 2004.

[3] G. F. Qunying Zhang Shengbo Ye, "Design and testing of a pseudo random coded gpr for deep investigation," in *IEEE International Geoscience and Remote Sensing Symposium*, 2016.

[4] R. M. Narayanan, Y. Xu, P. D. Hoffmeyer, and J. O. Curtis, "Design, performance, and applications of a coherent ultra-wideband random noise radar," *Optical Engineering*, vol. 37, no. 6, pp. 1855–1869, 1998.

[5] M. Zych, "Measuring experiment of FMCW ground penetrating radar," in *Radar Symposium (IRS), 2011 Proceedings International*, 2011.

[6] R. A. Lopez and W. R. Scott Jr., "Measurement of ground-penetrating radar antenna patterns using modulated scatterers," in *Proc. SPIE*, vol. 5794, 2005, pp. 459–469.

[7] P. Johnson, "New research lab leads to unique radio receiver," *E-Systems Team*, vol. 5, pp. 6–7, 4 1985.

[8] J. Mitola, "Software radios-survey, critical evaluation and future directions," in *National Telesystems Conference*, 1992, pp. 13/15–13/23.

[9] R. Lackey and D. Upmal, "Speakeasy: The military software radio," *IEEE Communications Magazine*, vol. 33, pp. 56–61, 5 1995.

[10] *Software Defined Radio*, `http://www.ni.com/sdr/`, Accessed: 2016-11-29.

[11] *Software-Defined Radio (SDR)*, `https://www.mathworks.com/discovery/sdr.html`, Accessed: 2016-11-29.

[12] *About GNU Radio*, `http://gnuradio.org/about/`, Accessed: 2016-11-29.

[13] *UHD (USRP hardware driver)*, `https://www.ettus.com/sdr-software/detail/usrp-hardware-driver`, Accessed: 2016-11-29.

[14] *RFNoC (RF Network on Chip)*, https://www.ettus.com/sdr-software/detail/usrp-hardware-driver, Accessed: 2016-11-29.

[15] S. Costanzo, F. Spadafora, G. D. Massa, A. Borgia, A. Costanzo, G. Aloi, P. Pace, V. Loscrì, and H. O. Moreno, "Potentialities of usrp-based software defined radar systems," *Progress In Electromagnetics Research B*, vol. 53, 417435, 2013.

[16] *GNU Radio Radar Toolbox*, https://grradar.wordpress.com/about/, Accessed: 2017-4-6.

[17] J. Ralston and C. Hargrave, "Software defined radio: An open source platform for prototype GPR development," in *14th International Conference on Ground Penetrating Radar (GPR)*, 2012.

[18] *UBX 10-6000 MHz Rx/Tx (160 MHz, X Series only)*, https://www.ettus.com/product/details/UBX160, Accessed: 2016-11-29.

[19] *Device Synchronization*, https://files.ettus.com/manual/page_sync.html, Accessed: 2016-11-29.

[20] *USRP X310*, https://www.ettus.com/product/details/X310-KIT, Accessed: 2016-11-29.

[21] Waymond R. Scott, Jr., "Efficient drive signals for broadband CW electromagnetic induction sensors," in *IEEE International Geoscience and Remote Sensing Symposium*, 2013.

[22] *UBX USRP Daughterboard, 10MHz-6GHz*, Rev. C, Ettus Research, Sep. 2015.

[23] *Current-Adjustable, Low Noise Amplifier*, MGA-62563, AV01-0642EN, Avago Technologies, Nov. 2006.

[24] *1-6 GHz Positive Gain Slope Low Noise Amplifier in SMT Package*, VMMK-3603, AV02-2919EN, Avago Technologies, Dec. 2012.

[25] M. West, *Re: [USRP-users] UBX LNA Ramp Up*, https://www.mail-archive.com/usrp-users@lists.ettus.com/msg01962.html, Mailing List, Accessed: 2017-3-16, 2016.

[26] D. Kozel, *[USRP-users] x310 loopback test*, http://lists.ettus.com/pipermail/usrp-users_lists.ettus.com/2016-June/020491.html, Mailing List, Accessed: 2017-3-20, 2016.

[27] J. W. Sustman, "Analysis of resistive-vee dipole antennas for producing polarization diversity," PhD thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 2014.

[28] K. Kim and W. R. Scott Jr., "Design of a resistively loaded vee dipole for ultrawideband ground-penetrating radar applications," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 8, pp. 2525–2532, 2005.

[29] C. Matzler, "Microwave permittivity of dry sand," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, no. 1, pp. 317 –319, 2002.

[30] T. Counts, A. Gurbuz, W. Scott, J. McClellan, and K. Kim, "Multistatic ground-penetrating radar experiments," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, 1247 1257, 2007.

[31] M. Müller, *Re: [Discuss-gnuradio] USRP tuning time*, `http://lists.gnu.org/archive/html/discuss-gnuradio/2015-08/msg00207.html`, Mailing List, Accessed: 2016-11-29, 2016.

[32] J. C. Bolomey and F. E. Gardiol, *Engineering Applications of the Modulated Scatterer Technique*. Norwood, MA: Artech House, 2001.