IMPLEMENTATION OF A RADAR SYSTEM USING
MATLAB AND THE USRP

A graduate project submitted in partial fulfillment of the requirements
For the degree of Master of Science
In Electrical Engineering

By
Vivian Neil Fernandes
The graduate project of Vivian Neil Fernandes is approved:

__________________________________                          ______________
Professor. James Flynn, Ph.D.                                                        Date

__________________________________                          ______________
Dr. Xiyi Hang, Ph.D.                                                             Date

__________________________________                          ______________
Dr. Sharlene Katz, Ph.D. Chair                                                  Date

California State University, Northridge
# Table of Contents

Signature Page ........................................................................................................... ii
List of Figures ............................................................................................................... iv
List of Tables ................................................................................................................ vi
Abstract ...................................................................................................................... vii
1. Introduction .............................................................................................................. 1
   1.1 Overview .......................................................................................................... 1
2. Background .............................................................................................................. 3
   2.1 Radar and History ........................................................................................... 3
   2.2 Radar Waveforms ............................................................................................ 4
   2.3 USRP (Universal Software Radio Peripheral) ................................................... 10
   2.4 MATLAB/Simulink ......................................................................................... 13
   2.5 GNU Radio Open Source Development ......................................................... 18
3. Implementation of the Chirp Waveform Generator .............................................. 21
   3.1 Chirp waveform types ..................................................................................... 21
   3.2 Chirp Generator Values ................................................................................... 30
   3.3 Chirp Transmission Issues ............................................................................. 36
4. Implementation of Radar Signal Processor ......................................................... 42
   4.1 Interleaving ..................................................................................................... 45
   4.2 Addition of pulses ........................................................................................... 50
   4.3 Matched Filter ................................................................................................ 53
   4.4 Radar Signal processor Simulink model ......................................................... 58
   4.5 Detection Threshold ....................................................................................... 61
5. Summary .................................................................................................................. 69
References .................................................................................................................. 71
Appendix A ................................................................................................................... 72
List of Figures

Figure 2.1: Radar system block diagram (A. Richards, 2010) ......................................................... 4
Figure 2.2: Example of radar equation (O’Donnell). ................................................................. 8
Figure 2.3: Block diagram of USRP ............................................................................................... 11
Figure 2.4: USRP2 and Daughter Cards....................................................................................... 12
Figure 2.5: USRP Transmitter and Receiver Blocks for Simulink ........................................... 14
Figure 2.6: Settings for USRP2 Transmitter and receiver .......................................................... 15
Figure 2.7: Enabled subsystem block ........................................................................................... 17
Figure 2.8: GNU Radio transmitter and receiver blocks for USRP2 ......................................... 19
Figure 3.1: Up Chirp ...................................................................................................................... 21
Figure 3.2: Down Chirp .................................................................................................................. 22
Figure 3.3: Linear and Bidirectional ............................................................................................. 23
Figure 3.4: Bi-Directional Chirp Waveform created using MATLAB (Real) .......................... 24
Figure 3.5: Bi-Directional Chirp Waveform created using MATLAB (Imaginary) ............... 24
Figure 3.6: Spectrum of Chirp waveform ...................................................................................... 25
Figure 3.7: Chirp Generator block ............................................................................................... 27
Figure 3.8: Math function block .................................................................................................. 28
Figure 3.9: Chirp Generator Block Subsystem .............................................................................. 28
Figure 3.10: Transmit and Receive Pulses .................................................................................... 31
Figure 3.11: Complete Chirp waveform ....................................................................................... 32
Figure 3.12: Single Chirp Generator ............................................................................................. 33
Figure 3.13: Single Chirp Waveform (Real) ................................................................................... 34
Figure 3.14: Single Chirp Waveform (imaginary) ....................................................................... 34
Figure 3.15: Total Transmitted Window ....................................................................................... 35
Figure 3.16: Single Chirp Window divided into 5 Parts ................................................................. 37
Figure 3.17: Chirp Waveform Division ......................................................................................... 38
Figure 3.18: Individual Interleaved signals .................................................................................... 39
Figure 3.19: Chirp Generator Division Block ............................................................................... 40
Figure 4.1: Block diagram of radar signal processor ................................................................. 42
Figure 4.2: Transmit/Receive Windows (A.Richards, 2010) ....................................................... 43
Figure 4.3: Simulation model for testing recombination of signal ............................................. 46
Figure 4.4: Recombined Waveform ............................................................................................. 47
Figure 4.5: Simulation of Interleaving and Adder Blocks ............................................................ 48
Figure 4.6: output of receiver buffer4 (each sample is delayed by 30 samples) ....................... 51
Figure 4.7: Adder Block for 20 windows ...................................................................................... 52
Figure 4.8: Output of the adder block .......................................................................................... 52
Figure 4.9: Complex Conjugate ................................................................................................... 54
Figure 4.10: Matched filter output .............................................................................................. 54
Figure 4.11: Simulation Block for Matched filter ........................................................................ 55
Figure 4.12: Matched filter output of received signal and transmitted signal ............. 57
Figure 4.13: Matched filter output of Transmitted and received signal (Zoom) ........ 57
Figure 4.14: Simulation Block for radar processor .................................................. 59
Figure 4.15: Measurement possibilities.(O’Donnell) .................................................. 61
Figure 4.16: Probability of detection. (O’Donnell) .................................................... 61
Figure 4.17: Target Detection (O’Donnell) ............................................................... 62
Figure 4.18: Settings for AWGN Channel ................................................................. 64
Figure 4.19: Radar processor output with Target +Noise as input (SNR -30 db) ....... 66
Figure 4.20: Output of radar processor block with noise only input ....................... 67
Figure 4.21: Radar processor output with Target +Noise as input (SNR -60 db) ........ 68
List of Tables

Table 4.1: Threshold Calculation. ................................................................. 63
Table 4.2: Calculation of Probability of missed detection. .............................. 65
Abstract

Implementation of a Radar System Using MATLAB and the USRP.

By

Vivian Neil Fernandes

Master of Science in Electrical Engineering.

This project report presents an approach to designing a radar system using software defined radio. The project objective was to demonstrate a SDR radar that would transmit and receive chirp waveforms. MATLAB and Simulink were used to implement the logic blocks to process the received data and calculate the range of the target. A methodology for determining an appropriate threshold value for a specified probability of false alarm and probability of missed detection was developed and tested for the system.
1. Introduction

1.1 Overview

The goal of this project is to simulate a radar system using bidirectional Chirp waveforms. The secondary goal is to use the software to develop software defined radio (SDR) based radar to transmit and receive the chirp waveforms, if possible and process the return.

The radar system can be used to detect the height of a target from the ground. It can also be used to detect the range of objects. Since radar systems are used in air transport and other critical civilian and military systems, it is necessary to study the effects of different waveforms on range and velocity resolution of targets.

Models were created which simulate and verify that the radar equations hold true for bidirectional chirp waveforms. The simulations were performed using Simulink. A USRP2 would be used to transmit and receive waveforms. The project also looked at different transmission and processing techniques to overcome system and resource constraints.

Chapter 2 will discuss the background of Radar and its history. It will also look at the various radar waveforms and the radar equation. The chapter will also introduce the USRP2 and development platforms like MATLAB and GNU radio.

Chapter 3 discusses how the chirp waveforms are generated and transmitted. It also discusses the values used to generate the chirp waveform used for this project and the issues faced during transmission.

Chapter 4 discusses the radar signal processor and its development. This chapter introduces the use of a matched filter and signal integration for the detection of a signal in
a high noise environment. It also introduces probability of detection and detection threshold and their importance in target detection.

Chapter 5 will summarize the results of the tests that were conducted in the lab using MATLAB/Simulink.
2. Background

2.1. Radar and History

Radar is an acronym for Radio Detection And Ranging. A radar system uses electromagnetic waves to determine the range, altitude and speed of a fixed or moving target. Due to the property of radio waves that allows them to be reflected from solid objects, radar can be used to detect any target that has a big enough surface area so that the waves can “bounce” off it and return back.

Radar is also used to map surfaces which are not easily accessible or for making maps of large regions. During World War II radar was used to detect enemy aircraft and was used as an early warning system. After the end of WWII radar found uses in the commercial arena for aircraft traffic management and also for studying and predicting weather patterns.

Radar also sees use in astronomy, nautical systems, ocean-surveillance systems, outer-space surveillance, meteorological precipitation monitoring, altimetry, flight control systems and ground penetrating. With the advent of high speed DSP processors, radar is also used to create high-definition maps of the ground over large areas.
2.2. Radar Waveforms

The operation of radars is based on one of the basic properties of electromagnetic waves, i.e. they are reflected when they strike a target. The amount of energy reflected back and its direction is based on the shape of the target and the material used to create it. Various waveforms are used in radar system and each type of waveform has its own properties. The basic principle of radar is illustrated in figure 2.1.

Figure 2.1: Radar system block diagram (A. Richards, 2010)

A signal with specific properties and wavelength is emitted by a transmitter in the direction of the target. The signal strikes the target and an echo signal is reflected back. The signal strength reflected back depends on the reflective properties and shape of the target. The echo signal strength is very small compared to the original signal due to scattering, absorption and spreading loss and the signal to noise ratio (SNR) of the received signal is generally very low.

Scattering occurs because the entire signal is not reflected in the direction of the antenna. Depending on the shape and size of the target, signals are scattered or absorbed thereby reducing the amount of signal power that is reflected back. Before discussing
radar waveforms and detection techniques, the radar equation and the relationship between transmitted power and received power is presented. The average noise power at the receiver input is given by,

\[ N = k \times B_n \times T_s, \quad \text{(eq. 2.1)} \]

where,

k = Boltzmann constant = 1.38 \times 10^{-23} \text{ joules/°K}

\( T_s = \) System Noise Temperature (°K).

\( B_n = \) Noise Bandwidth of receiver, (Hz).

The power density at a distance R from an isotropic radiator is given by,

\[ PD_{isotropic} = \frac{P_t}{4 \times \pi \times R^2} \]

where,

\( P_t = \) Transmitted power.

The spatial Power Density of the echo signal at the receiver input is given by,

\[ PD = \frac{P_t \times G_t \times \sigma}{(4 \times \pi \times R^2)^2}, \quad \text{(eq. 2.2)} \]

where,

R = Distance between the receiver and target.

\( \sigma = \) Area of Cross Section of Object in meters.

\( G_t = \) Gain of the transmitter

Also the transmitter gain is given by,

\[ G_t = \frac{4 \times \pi \times A_t}{\lambda^2}, \quad \text{(eq. 2.3)} \]
where,

\[ A_t = \text{Effective area of the Transmitter antenna,} \]
\[ \lambda = \text{Wavelength of the transmitted signal.} \]

For a radar system the receive and transmit antennas are the same so that,

\[ G_r = G_t = G \] and both antennas have the same effective area.

The received power \( P_r \) is obtained by multiplying the power density at the radar receiver (eq.2.2) with the effective the area of the receiving antenna (\( A_R = A_T \)).

Hence the above equation can be rewritten as,

\[ P_r = \frac{P_t \times G_t \times \sigma \times A_R}{(4 \times \pi \times R^2)^2} \] \hspace{1cm} (eq. 2.4)

Hence the S/N (Signal to Noise Ratio) is hence given by,

\[ \frac{S}{N} = \frac{P_r}{N} = \frac{P_t \times G^2 \times \lambda^2 \times \sigma}{4 \times \pi^3 \times R^4 \times T_s \times B_n \times k \times L_s} \] \hspace{1cm} (eq. 2.5)

where,

\[ L_s = \text{Total System Losses,} \]
\[ \lambda = \text{Wavelength of the transmitted signal.} \]

The received return signal is applied to a digital signal processing algorithm to determine the time between the original transmission and receipt of the signal. This time difference \( T_r \) is proportional to the distance travelled by the signal, which is twice the distance of the target from the receiver.
The distance to the target or the range R is given by,

\[ R = \frac{c \times T_r}{2}, \quad \text{(eq. 2.6)} \]

\( c = \) Speed of Electromagnetic energy in free space = \( 3 \times 10^8 \) meters/sec.

\( T_r = \) Time taken to reach target and travel back to the receiver.

The most common radar signal is a set of short duration pulses modulated by a high frequency carrier wave, also called a pulse train. Figure 2.2, illustrates the transmission of a pulse train. Each pulse has a pulse width of \( 1\mu \) sec and a power of 1MW. The duty cycle is 0.001, which means a pulse is transmitted every 1m sec. The transmitter transmits a signal for \( 1\mu \) sec and the receiver listens for a response for 0.999m secs. Thus the average power over each cycle is 1KW. For the example in figure 2.2, since the receiver can listen for 0.999 ms, the maximum unambiguous range is given by equation 2.6

\[ R = 3 \times 10^8 \times \frac{0.000999}{2} = \sim 15000 \text{ meters} \]

It can also be seen from the figure that the power of received target echo is small compared to the transmitted signal. This is because of noise and scattering, where the entire signal is not reflected back in the direction of the antenna.
Figure 2.2: Example of radar equation (O’Donnell).

The ability of a radar system to detect a target is based on the return power. Equation 2.5 shows that the received power decreases as the 4th power of the range, which means that the reflected power from the target is very small. Hence it is necessary that the transmitted waveform contains enough power in the direction of the target. One goal of the radar system is to send as much power as possible in the transmission time. A technique known as pulse compression is used. For most radar systems, the most important requirement is a high range resolution. The required range resolution may be obtained by using short pulses (pulses with small pulse widths). However it is difficult to meet the high power requirements in the short period of transmission.

Hence to have the bandwidth of a short pulse and the power of a long pulse, the long pulse is modulated either in frequency or phase. This technique, using frequency or phase modulation, allows radar to simultaneously achieve the energy of a long pulse and the resolution of a short pulse.
This report shall discuss the various waveforms used for a radar system and their benefits. Although there are different types of waveforms, the chirp waveform was chosen because it provides a number of advantages including:

- Higher signal to noise ratio (SNR).
- Consumes less bandwidth than other waveforms.
- More power in every signal period due to pulse compression.
- Higher resolution.
- Ability to process in an analog way to realize low power solutions.
- Chirp Waveform systems do not fail because of Doppler shifting.

Chirp signal waveforms can be processed asynchronously and offer advantages in comparison to other systems. The chirp signal waveform uses pulse compression, which is achieved by modulating the transmitted waveform with a high frequency carrier wave. Chirp waveforms using pulse compression are discussed in detail in section 4.
2.3. USRP (Universal Software Radio Peripheral)

The Ettus Universal Radio Peripheral (USRP) is a hardware platform used in software defined radios. For this project, the USRP2 was used as a transmitter and receiver. The USRP is a comparatively inexpensive hardware platform. (USRP Networked Series, 2012)

The USRP has high speed Gigabit Ethernet link which is used to connect to a host machine. The host machine then uses software to implement the entire signal processing functions in the radio. These functions can also be loaded on to the onboard FPGA for faster execution.

The USRP2 and WBX daughterboard used in the project is capable of working with a maximum frequency of 2.5GHz.

The USRP contains,

- A Xilinx Spartan 3-2000 FPGA
- Gigabit Ethernet interface.
- Dual 100 MHz 14-bit ADCs
- Dual 400 MHz 16-bit DACs
- Two 100 MS/s, 14-bit, analog-to-digital converters.
- Two 400 MS/s, 16-bit, digital-to-analog converters.
- SD card reader.

Figures 2.3 and 2.4 show the block diagram of the USRP and photos of the hardware.

In a SDR, the high sample rate processing is done by the USRP, while more involved
lower sample-rate processing is handled by the host computer. Also the USRP has fully
coherent sampling clocks and local oscillators.

The USRP has a high speed FPGA, which translates, filters and decimates the
outputs of the high speed ADC, from 100 MS/s to a slower speed which can be handled
by the host computer. Similarly the FPGA up-converts the signal from the PC to
400MS/s.

Figure 2.3: Block diagram of USRP
Figure 2.4: USRP2 and Daughter Cards

The two software development platforms that were considered for in this project were:

- MATLAB/Simulink.
- GNU Radio Open Source.
2.4. MATLAB/Simulink

The project was developed and tested using MATLAB code along with Simulink blocks. This includes the Chirp Waveform generator, and the Radar Signal Processor. The blocks that interface MATLAB/Simulink with the USRP are provided by Mathworks.

In spite of its ease of use, due to the stringent requirements of processing power and memory, MATLAB is not efficient enough for radar processing. It does not give the real time performance that is needed. During the development of this project, performance problems and Ethernet packet drops were encountered when communicating with the USRP at sampling rates of higher than 8MHz. However for simulation purposes, MATLAB is sufficient. The Simulink models used to communicate with the USRP are shown in figure 2.5.
Figure 2.5: USRP Transmitter and Receiver Blocks for Simulink
Figure 2.6: Settings for USRP2 Transmitter and receiver

Figure 2.6 shows the parameter settings for the USRP transmit and receive blocks in figure 2.5. The USRP2 is set to transmit and receive at a center frequency of 450MHz. A single chirp window is generated and stored as a MATLAB variable *originalsignal*, which is 141 samples in length. The chirp signal is padded with zeros to make the complete signal 564 samples in length, which is the complete transmission window, also called a chirp window. The transmitter transmits 141 samples and then listens for a response after for the rest of the chirp window period. The process of generating this waveform will be described in detail in section 3.
During the execution of the model (see figure 2.5), the Signal from Workspace block reads the *originalsignal* variable at the specified sampling rate of 10MHz. This is then passed to the *SRDu Transmitter* block which communicates with the USRP2 via Ethernet. The interpolation value needs to be set so that the chirp waveform is up-sampled to a rate of 100 MS/s.

The receiver reads a frame of 564 samples, which is the length of the receive period, before it triggers the *Data Len* signal, which in turn triggers the enabled subsystem to read the data from the *USRP2 Receiver* block (see figure 2.5). The Enabled Subsystem then saves the data in the workspace by using the signal to workspace block (see figure 2.7). Since the USRP2 transmitter and receiver blocks require extensive PC resources to operate at peak performance, the incoming data is stored, to be processed later on by another subsystem.
Figure 2.7: Enabled subsystem block

[Diagram of enabled subsystem block]

[Block Parameters: Enable dialog box]

Enable Port
Place this block in a subsystem to create an enabled subsystem.

Parameters

States when enabling: [Production, Hold, Run]

- [ ] Hold
- [ ] Run

Show output port
Enable zero-crossing detection

[Buttons: OK, Cancel, Help, Apply]
2.5. GNU Radio Open Source Development

An alternative to MATLAB/Simulink is GNU Radio. It is an open source software development toolkit, with signal processing blocks. GNU radio provides the ease of development that MATLAB /Simulink provides, with better processing speed and less PC resource consumption. GNU radio blocks are developed using Python script, which when compiled is converted into C code, which can be developed to be Operating System/Platform independent. Since it is open source, code modifications are easy since the source code is freely available and can be ported to embedded platforms.

It also provides a graphical interface which is similar to the one provided by MATLAB. Although it does not support all the extensive modules as MATLAB, it does support the basic structures like signal sources and sinks.

Figure 2.8, shows a sample model of how to use the USRP2 Source /sink blocks in GNU Radio.
Figure 2.8: GNU Radio transmitter and receiver blocks for USRP2

As illustrated in the figure, the USRP Sink /Source blocks are configured to communicate with the USRP2 using the Ethernet link. The sample rate for this example is maintained at 12.5MHz and the center frequency is maintained at 450MHz.

The benefits of using GNU radio over MATLAB in this project are as follows.

- Simplicity of programming using C++ and Python programming.
- Does not have resource hungry UI interfaces during runtime.
- Runs on Linux and Windows with not much code change.
- Since the code is in C++ and Python it gives us the real time performance that is needed to interface with the USRP.
Despite all these benefits, Simulink was used to implement the simulations, because of the availability of different blocks and toolkits, thereby giving more design flexibility.
3. Implementation of the Chirp Waveform Generator

3.1. Chirp waveform types

Chirp signals are angle modulated sweeping signals, the frequency of which sweeps over a frequency band within a specific time period. If this band of frequencies is swept from the lowest to the highest frequency limit, the resulting waveform is called an Up chirp (see Figure 3.1). In the opposite direction it is a Down chirp (see Figure 3.2).

Figure 3.1: Up Chirp

As illustrated in figure 3.1, the signal sweep starts at a specified low frequency and then increases exponentially till the maximum frequency is reached, within the given time period.
Similarly, the down chirp is a sweep starting at the specified maximum frequency and the frequency decreases to the minimum frequency in the given time period (see figure 3.2). A Bidirectional chirp is obtained by starting at a specified high frequency and then decreasing the frequency until the minimum specified frequency is achieved, within half the given time period. It is then increased (linear increase) until the original maximum frequency is reached in the remaining time period (see figure 3.3). A Bidirectional linear chirp is used in this project to get a better SNR and it also gives a better output at the matched filter.

The chirp waveforms illustrated in figures 3.1, 3.2 and 3.3, are swept cosine, since they are created by sweeping a cosine wave, by varying phase and frequency.
The chirp signal used in this project is given by the expression

\[ S_n = e^{-j\pi \frac{W}{T} \left( nT_s - \frac{T}{2} \right)^2}, \quad \text{(eq. 3.1)} \]

where

- \( T \) = Time Duration of the chirp.
- \( W \) = Bandwidth of the Chirp signal.
- \( T_s \) = Sampling time.
- \( n \) = discrete sample index.

The waveform produced using equation 3.1 is a complex waveform producing the real in phase (I) and imaginary quadrature (Q) components of the baseband signal. (See figures 3.4 and 3.5).
Figure 3.4: Bi-Directional Chirp Waveform created using MATLAB (Real)

Figure 3.5: Bi-Directional Chirp Waveform created using MATLAB (Imaginary)
Figure 3.6: Spectrum of Chirp waveform.

In order to have sufficient power in the return signal, the transmitted signal must have a long enough period; $T$. The power of a signal is given by

$$P_{AV} = \frac{1}{T} \int_0^T s(t)^2 dt,$$  \hspace{1cm} (eq. 3.2)

For this project each pulse of 7μ secs length has 1 watt of power. The determination of the chirp length is discussed in section 3.2. The chirp waveform created using equation 3.1 is shown in figures 3.4 and 3.5. The 3dB bandwidth is shown in figure 3.6, and plays a very important part in the chirp signal generation. The 3dB bandwidth is approximately 10MHz.
A simulation model was developed to implement equation 3.1 and is shown in figure 3.7. The model produces a chirp waveform which has a period of $7\mu$ secs sampled at 10 MHz. The bandwidth $w$ for this model is $10MHz$. The math function block in the model implements an exponential math function (see figure 3.8). The rest of the blocks are basic math blocks and constants.
Figure 3.7: Chirp Generator block
Figure 3.8: Math function block

![Function Block Parameters: Math Function](image)

Math

Mathematical functions including logarithmic, exponential, power, and modulus functions. When the function has more than one argument, the first argument corresponds to the top (or left) input port.

<table>
<thead>
<tr>
<th>Main</th>
<th>Signal Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function:</td>
<td>exp</td>
</tr>
<tr>
<td>Output signal type:</td>
<td>complex</td>
</tr>
<tr>
<td>Sample time (-1 for inherited):</td>
<td>-1</td>
</tr>
</tbody>
</table>

Figure 3.9: Chirp Generator Block Subsystem

![Image of Chirp Generator Block Subsystem]

The subsystem shown in figure 3.9 contains the simulation model from figure 3.7.

The Limited Counter block generates the values for \( n \), which drives the model to generate the Chirp waveform. The counter generates a count from 1 to 141 and then resets itself to 0. \( T_s \), \( W \), and \( T \) are constants where;
The output of the Chirp Generator block is a single chirp with pulse width defined by the $T$. A chirp waveform with different parameters can be created, by just changing the values of the constants $T_s$, $W$ and $T$. The upper limit of the counter Limited block is the product of $W$, $T$ and an oversampling rate if needed.

$$\text{Counter Range} = p \times T \times W$$

$$= 2 \times 10^6 \times 7 \times e^{-6} = 140 \text{ samples}$$

For this project the counter upper limit is 141. So the counter will count till 140 and then reset itself to 0. The oversampling rate is used to increase the number of samples during the chirp pulse period, giving better signal resolution. The oversample rate is selected based on the bandwidth and performance restrictions of the radar system.
3.2. Chirp Generator Values

The project described in this report had the following requirements.

- Measure the distance to targets between 1000 and 3000 meters away.
- Operate at a Carrier frequency of 450 MHz

The pulse width determines the minimum target distance that can be detected accurately. The time taken for a signal to travel 1000 meters and back is given by the radar equation (see equation 2.6)

\[
T_r = \frac{2 \times 1000}{3 \times 10^8} = \sim 7\mu \text{seconds}
\]

Thus, a pulse width of 7\(\mu\) seconds was selected (see figure 3.10). This insures that the first sample of the return signal should reach the receiver only after the last sample of the chirp waveform has been transmitted. The radar system is hence capable of detecting a target at 1000. If the target is at 1100 meters, there is a delay of 0.3\(\mu\) seconds between the transmission of the last sample and receipt of the first sample. If the response is received before the transmission is completed, the result is an error because the receiver will only listen for the response after the transmission is completed (see figure 3.10). The pulse width determination also depends on various other factors, none of which are considered in this project.
Using the same range formula (eq. 2.6), the minimum receive period needed for detecting a target 3000 meters is,

$$ T_r = \frac{2 \times 3000}{3 \times 10^8} = \sim 21 \mu \text{seconds} $$

The radar system transmits a Chirp waveform with a pulse width of 7\( \mu \) seconds and waits during a receive period of 21\( \mu \) seconds (see figure 3.11).
Based on the analysis done above, the following chirp waveform parameters were selected:

\[ T = 7e^{-6}; \quad \% \text{length of chirp pulse in seconds} \]

\[ W = 10e^6; \quad \% \text{bandwidth of chirp pulse (Hz)} \]

\[ p = 2; \quad \% \text{oversampling rate for chirp pulse} \]

\[ f_c = 450; \quad \% \text{Center frequency of radar (MHz)} \]

The simulation block shown in figure 3.9 was modified to produce a chirp waveform with the above parameters. In order to minimize the complexity of the block and increase speed of execution, the chirp generator block was simplified by using the equation 3.1 and implementing it with the \textit{embedded MATLAB function} block. Based on trial runs this method was determined to have faster execution time. Based on the trials it was also determined that a oversample rate of 2 was adequate for this project. The output
of the *embedded MATLAB function* block (see figure 3.12) is the same as the output of the simulation block shown in figure 3.9.

Figure 3.12: Single Chirp Generator

Figure 3.12 shows a simplified model of the model shown in figures 3.7 and 3.9. This simulation block (see figure 3.12) produces a single chirp waveform, having a pulse width of 7\(\mu\) secs and a bandwidth of 10 MHz. The output is then passed to the *To Workspace* block which saves the output as the *simout* variable in the MATLAB workspace. The *Embedded MATLAB Function* takes 3 values as inputs shown by the *constant* blocks. The Code for the Embedded MATLAB Function is can be found in Appendix A. The chirp wave produced by the code is illustrated in figures 3.13 and 3.14. Note that the output of the above code is the same as the chirp waveform illustrated in figures 3.4 and 3.5.
Figure 3.13: Single Chirp Waveform (Real)

Figure 3.14: Single Chirp Waveform (imaginary)
Figure 3.15: Total Transmitted Window

The chirp waveform shown in figures 3.13 and 3.14 are then padded with zeros to create a complete chirp transmission packet or a chirp window (see figure 3.15). The zeros are added to separate two consecutive chirps from each other. They are separated by the receive period. That is, the two consecutive chirps are separated from each other by the time that the receiver listens for the return signal, which is $21\mu$ seconds. This generated waveform is saved in the MATLAB workspace which is then read by the USRP2 Transmitter Block of MATLAB (see figure 2.5).
3.3. Chirp Transmission Issues

The chirp waveform that was generated is transmitted using the USRP2 block in Simulink which shifts the chirp signal to a 450 MHz carrier frequency. During testing it was found that the bandwidth and speed of the host PC was not adequate for reception of the incoming signal. On receive, packets were lost or delayed. In order to counter this problem a method was devised to reduce the required sampling rate of the received signal. The original chirp signal is divided into parts and then transmitted, in a process called interleaving. Interleaving is a way to arrange data in non-contiguous way to decrease the bandwidth required for transmission. A sampled signal of 36 samples is shown in figure 3.17. The signal is divided (interleaved) into 5 parts which are labeled A to E. The output of the interleaving process is show in figure 3.18. Each of these parts is transmitted separately and in order. First all of the “A” values are sent, followed by the “B” values and so on. The sample time is now five times that of the original signal resulting in a reduced processing demand on the PC. When the interleaving process is applied to the signal illustrated in figure 3.15, the output is shown in figure 3.16.
Figure 3.16: Single Chirp Window divided into 5 Parts
Figure 3.17: Chirp Waveform Division
Figure 3.18: Individual Interleaved signals
The chirp generator block (see figure 3.19) produces a Chirp waveform as described in the section 3.2 (see figure 3.15). The chirp division block code takes a single Chirp Transmission window and divides it into five sections, each of which is an offset of the previous one by five samples (see figure 3.16) with each part separated by an receive period of \(21\mu\text{seconds}\). The Chirp Division Block also outputs the complex conjugate, the original chirp window and the five individual parts of the signal. Each part of the chirp waveform is \(7\mu\text{seconds}\) long and has an receive period of \(21\mu\text{seconds}\). The code for the Chirp Division block can be found in Appendix A. The Signal to Workspace block saves the output of the chirp division block to the MATLAB variable \textit{FinalChirp}.

At the receiver, each part is received in the order that they were transmitted in. Since the order of transmission is known, it is recombined in the same order. The result is a return signal which has similar characteristics to the return signal that would have been received without interleaving. Putting the signal back together in the right order is of paramount importance. This will be discussed in section 4.
4. Implementation of Radar Signal Processor

The final goal of this project was the implementation of a radar signal processor in MATLAB that would take the noisy return signal containing bursts reflected from the target, and accurately determine the distance of the target. This section of the report describes the signal processing system developed to determine the distance of the target from the received radar return signal.

The signal waveform used was a series of bidirectional chirp waveforms. The design is based on the starting time, the length of the pulse, and the inter-pulse period (receive period) and the T/R cycles (see figure 4.2). The system alternates between modes, first transmitting a pulse, and then listening for echoes.

Figure 4.1 shows the block diagram of the radar signal processor. The return signals is received at the USRP2, where it is down converted and decimated. It is then passed to the interleaving block. The interleaving block recombines the five T/R cycles corresponding to a single chirp window. The interleaving block buffers 20 such recombined T/R cycles to be passed to the Adder block. The output of the Adder block is then passed to a matched filter along with the complex conjugate of the original transmitted signal.

Figure 4.1: Block diagram of radar signal processor
The *Compare and Threshold detection* block takes the output of the matched filter and scans it for valid target information. If valid target information is found, the distance of the target from the receiver is calculated by using the range equation (eq. 2.6). Each of these blocks is discussed in-depth in this section.

Figure 4.2: Transmit/Receive Windows (A.Richards, 2010)

![Radar Diagram - Transmit/Receive Windows](image)

Figure 4.2 illustrates the operation of a radar system. The system transmits during the transmit window. It stops transmitting and starts listening for incoming data during the receive period. The receive period is the time between two pulses, and is very important in determining the range of the radar system, as discussed in the previous sections. A T/R cycle consists of a single transmit and receive period (see figure 4.2). This section will discuss how the received data is processed and passed through a detection algorithm.
However before discussing the detection algorithm, it is necessary to understand each block of the radar signal processor, and the role they play in target detection. A Simulink test model was created which implements the blocks of the radar signal processor. Pre-generated signals are used as inputs. The chirp waveforms used in the simulation were created using the *chirp generator* block (see figures 3.10 and 3.15).
4.1. Interleaving

Before the radar signal processor can detect the target, the incoming signal has to be processed to increase the signal to noise ratio. Interleaving is a process where the received data is arranged in a non-contiguous way to deal with the bandwidth limitations as described in section 3.3.

Before developing the interleave block for the radar processor, the concept was tested using the simulation block illustrated by figure 4.3. The chirp generator block outputs the signal divided into 5 parts (5 T/R cycles), which is the signal illustrated in figure 3.16. The signal is then delayed by 15μs seconds using the Integer Delay block, in order to simulate a return signal. The delayed signal is then passed to the Chirp Combine Block. The Chirp Combine block takes 5 T/R cycles and interleaves them to get a complete recombined chirp waveform illustrated in figure 4.4. The code for the Chirp Combine block is shown in Appendix A. The Simulink model was tested for different values of delay to confirm that the recombination logic gives the accurate output.
Figure 4.3: Simulation model for testing recombination of signal
Figure 4.5 illustrates the simulation model created to test the interleaving block for the radar signal processor, based on the simulation model illustrated in figure 4.3. The *signal to workspace, chirp division* block, *delay* block and *AWGN* block are used to simulate a return signal received by the USRP2 (see figure 4.5). A return signal with different parameters can be simulated by changing the properties of these blocks.

The first buffer block in the simulation model (*buffer1* in figure 4.5), buffers 5 T/R cycles. These T/R cycles need to be recombined together to get a complete return signal. The *Chirp Combine* block takes the output from the first buffer (*buffer1*) and interleaves the 5 under sampled T/R cycles to get a complete chirp window.
Figure 4.5: Simulation of Interleaving and Adder Blocks

The output of the *chirp combine* block is then passed to the second buffer (*buffer4 in figure 4.5*). The second buffer will store twenty combined chirp windows before passing it to the *Embedded MATLAB Function* block which implements the adder block. The number of chirp windows that need to be buffered before any further processing is done, is determined by the adder block discussed in section 4.2. The one down side to
using interleaving is that latency is introduced into the system because of the recombination process.
4.2. Addition of pulses

Once the receiver has received the response, some processing is necessary to reduce the effects of receiver noise and improve the detection of the signal. This can be done by adding several consecutive return signals together to get a signal whose power is the total of the power of all the received waveforms. The noise power of the received signals is also added up during the process. However since noise is a random process, noise components will tend to cancel, improving the signal to noise ratio in the combined signal.

For ease in development and processing speed, coherent integration was selected for this project. As discussed in section 4.1, the output of the chirp combine block is a single chirp window (a single recombined T/R cycle) which is recombined from 5 T/R cycles. The output is then passed to the second buffer block which stores twenty such recombined chirp windows (see figure 4.5 buffer4 block). The process of adding these windows should be done as soon as they are received because of performance and memory constraints. The output of the buffer (see figure 4.6) is passed to the Embedded MATLAB Function block shown in figure 4.5. The Embedded MATLAB Function first divides the input from the buffer into twenty individual chirp windows (T/R Cycles). These individual chirp windows are then added together.

Figure 4.7 shows the simulation model created to add twenty chirp windows together. The input to the embedded MATLAB function blocks is the buffered signal that has twenty chirp windows (see figure 4.6). These are broken down into each individual chirp windows and then added by the adder blocks. Each Embedded MATLAB Function block outputs ten individual chirp windows each. The twenty individual chirp windows (see
figure 4.6) are then passed to the Add blocks and are added together. The output of the Add2 block is the aggregate of all the twenty windows which has a better Signal to Noise Ratio. This signal is then passed to the matched filter. The block illustrated in figure 4.7 is embedded into the Embedded MATLAB function blocks illustrated in figure 4.5. The output of the adder block for a input of twenty samples (see figure 4.6) is illustrated in figure 4.8. The code for the embedded function blocks is shown in Appendix A.

Figure 4.6: output of receiver buffer4 (each sample is delayed by 30 samples)
Figure 4.7: Adder Block for 20 windows

Figure 4.8: Output of the adder block
4.3. Matched Filter

A matched filter is used to locate the chirp pulse within the received signal. A matched filter is implemented by integrating a time shifted and reversed copy of the original chirp waveform with $g(t)$, the output of the adder block $f(t)$ (Nadav Levanon, 2004).

\[
(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau
\]

\[
= \int_{-\infty}^{\infty} f(t - \tau) g(\tau) \, d\tau \quad \text{(eq. 4.1)}
\]

The matched filter gives the maximum SNR at the output for a given signal. The chirp waveform is symmetrical around the point at $t = 0$. Thus the time reversed copy of the original chirp waveform is the same as the chirp waveform itself (see figures 3.13 and 3.14). The output of the Adder block is passed to the Embedded MATLAB function that implements the Matched filter (C.S. Burrus, 1994). The matched filter block also takes in as a input, the complex conjugate from the Signal from Workspace block.

If the original chirp waveform (see figures 3.13 and 3.14) and the complex conjugate (see figure 4.9), are given as a input to the Matched filter block, the output is a peak at sample 561 as illustrated in figure 4.10.
Figure 4.9: Complex Conjugate

Figure 4.10: Matched filter output.
Figure 4.11: Simulation Block for Matched filter

Figure 4.10, shows the matched filter output. Note that the peak corresponds to the time that the original and Complex conjugate waveforms are matched when there is no delay. By detecting the peak in the matched filter output, the delay between the transmission and the receipt of the chirp waveform is determined. This value will be used to calculate the target distance.

To illustrate the calculation of target distance, the output of the signal from workspace3 block was delayed by 30 samples using the delay block in Simulink (see...
figure 4.12). This simulates a received signal, which took $3 \mu$ seconds to hit the target and return back after transmission was complete. The matched filter output of the blocks for the transmitted and received signals is illustrated in figures 4.12 and 4.13. Figure 4.13 shows 2 peaks, the first one at sample number 561 and the second one at 591. The peak at sample 561 is the original transmitted signal and the one at 591 is the received signal. The delay between them is 30 samples, which translates to a delay of $3 \mu$ seconds (at a sampling rate of 10MHz). Hence using the range equation the distance of the target is therefore,

$$R = 3 \times 10^8 \times \frac{3 \times 10^{-6}}{2} = 450 \text{ meters}$$

If the two peaks get closer to each other, so that they overlap, the system is not able to distinguish between them and hence will give an incorrect target reading. If the response to the previous chirp is received during the transmission of the present chirp, then the received peak will be before the transmitted peak and will also give a incorrect reading.
Figure 4.12: Matched filter output of received signal and transmitted signal

Figure 4.13: Matched filter output of Transmitted and received signal (Zoom)
4.4. Radar Signal processor Simulink model

In order to simulate and verify the implementation of the radar signal processor, a complete model was created as illustrated in figure 4.14. This model implements the blocks of the radar signal processor as illustrated by figure 4.1. Sections 4.1, 4.2 and 4.3 discuss the implementation of the individual simulation blocks. The model shown in figure 4.14 is the integration of all the above mentioned blocks and implements the detection and range calculation blocks.
Figure 4.14: Simulation Block for radar processor

Signal generator, Interleaving and adder blocks
Illustrated in figure 4.11
The *sort* blocks (see figure 4.14), sort the incoming samples in ascending order. So the first index in the array is the highest value or the peak which is used to calculate the delay between transmission and receipt of the chirp. The *range calculator* block, which is an *Embedded MATLAB function* block, calculates the range of the target by using this delay. The *range calculator* block takes in the output of the two *sort* blocks as inputs. The block then searches for the index whose value is above the *threshold* value and calculates the delay between the two indices. This delay is then used to calculate the range of the target. The threshold value and its determination as discussed in section 4.5.
4.5. Detection Threshold

For every measurement that is performed by the radar processor, there are two hypotheses (see figure 4.15):

- Target absent, noise only, $H_0$.
- Target present, signal plus noise, $H_1$.

Hence for any measurement there are four possible outcomes as shown in figure 4.16.

Figure 4.15: Measurement possibilities. (O’Donnell)

<table>
<thead>
<tr>
<th>For each measurement</th>
<th>Measurement</th>
<th>Probability Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target absent hypothesis, $H_0$</td>
<td>$x = n$</td>
<td>$p(x</td>
</tr>
<tr>
<td>Noise only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target present hypothesis, $H_1$</td>
<td>$x = a + n$</td>
<td>$p(x</td>
</tr>
<tr>
<td>Signal plus noise</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.16: Probability of detection. (O’Donnell)

<table>
<thead>
<tr>
<th>For each measurement</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth</td>
<td>$H_0$</td>
</tr>
<tr>
<td>$H_0$</td>
<td>Don’t Report</td>
</tr>
<tr>
<td>$H_1$</td>
<td>Missed Detection</td>
</tr>
</tbody>
</table>

The probability of detection (PD) is the probability that $H_1$ is chosen when $H_1$ is true. The probability of false alarm (PFA) is the probability that $H_1$ is chosen when $H_0$ is true.
The detection threshold is the minimum amount of voltage above which a signal will be considered as a valid target signal (O’Donnell). Any sample below this threshold will be considered as noise. A false alarm is when the noise voltage level is above the threshold but there is no target (see figure 4.17). Generally the threshold is calculated using probability density functions for different models of noise. In this project the threshold is determine from the simulation results.

The concept of detection of a target involves deciding for each sample of the incoming waveform, whether a target exists at that time. This is established by setting a threshold signal level, on basis of interference from external noise, clutter, thermal noise etc. The voltage level of every sample is then compared against this threshold value and decision is made whether the target exists. The threshold value depends on the probability of false alarm that is used to design the radar system.

By using the simulation models (see figure 4.14), it was determined that, for a probability of false alarm of 2%, the minimum threshold value is ~93 volts at the output.
of the matched filter (see Table 4.1). In order to determine the threshold value the simulation was run without any target data. The simulation without target data was achieved by disconnecting the signal from workspace blocks from the rest of the blocks (see figure 4.14). In order to get the threshold voltage value, the matched filter output of the received signal was analyzed (see figure 4.20). The simulation was run a number of times to get the average threshold value at a fixed SNR value. The AWGN Channel Block settings are illustrated in figure 4.18. These noise settings are fixed and never changed throughout the simulations.

Table 4.1: Threshold Calculation.

<table>
<thead>
<tr>
<th>Trial Threshold</th>
<th>Upper Limit</th>
<th>Lower Limit</th>
<th>$P_{fa}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>156</td>
<td>150</td>
<td>165</td>
<td>0.03%</td>
</tr>
<tr>
<td>100</td>
<td>125</td>
<td>90</td>
<td>0.12%</td>
</tr>
<tr>
<td>90</td>
<td>100</td>
<td>85</td>
<td>3.54%</td>
</tr>
<tr>
<td>93</td>
<td>98</td>
<td>88</td>
<td>2.25%</td>
</tr>
<tr>
<td>90</td>
<td>95</td>
<td>85</td>
<td>5.56%</td>
</tr>
</tbody>
</table>
Table 4.1 shows the probability of false alarms for different threshold values. At a threshold of 93 volts the probability of false alarms is about 2.25%. Since noise is a random variable, the threshold value cannot be determined on only one pass of the simulation. The simulation was run multiple times so as to confirm the threshold value of 93 volts.

The simulation in figure 4.14 was run using target data and with the AWGN Channel settings shown above. The amplitude of the input signal was decreased, so as to decrease the SNR of the output of the AWGN Channel block. The simulation was run multiple times at different amplitudes to determine at what SNR value, the probability of a missed detection ($P_{md}$) is above 0.5%. Table 4.2 shows the output of the simulations.
with target data. It can be seen from the data in table 4.2 that as the amplitude of the input signal decreases, the SNR decreases and the probability of a missed detection increases.

Table 4.2: Calculation of Probability of missed detection.

<table>
<thead>
<tr>
<th>Input signal Amplitude</th>
<th>SNR(dB)</th>
<th>$P_{md}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
<td>0.0012%</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0.0084%</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.01%</td>
</tr>
<tr>
<td>0.1</td>
<td>-20</td>
<td>0.435%</td>
</tr>
<tr>
<td>0.5</td>
<td>-30</td>
<td>0.467</td>
</tr>
<tr>
<td>0.01</td>
<td>-40</td>
<td>0.635</td>
</tr>
</tbody>
</table>

As long as the voltage at sample 591 (see figures 4.19 and 4.21) is greater than the threshold value, the target is detected properly. At a SNR of $-40$ dB or higher, the probability of missed detection ($P_{md}$) drastically increases to approximately 0.65% (see Table 4.2).
Figure 4.19: Radar processor output with Target + Noise as input (SNR -30 db)
Figure 4.20: Output of radar processor block with noise only input
Figure 4.21: Radar processor output with Target + Noise as input (SNR -60 db)
5. Summary

Current radar systems are largely hardware systems that have to be fully replaced or largely upgraded to allow for new waveforms and modulation types. This is time intensive and costly. Future radar systems, both ground and airborne will use SDR (software defined radio). Just like radios, radars could have multiple applications with modifications to the software.

This project analyzed the chirp waveform and different simulation models used to produce them. The project also analyzed the matched filter and its use to extract target information from incoming signal data. The analysis, which was conducted using MATLAB to simulate the chirp waveforms and the matched filter, provides an environment to test waveforms and filters before an actual system is implemented.

While the original project goal was to implement a working radar system, this proved much more difficult to implement than anticipated. High processor and resource requirements hindered the performance of the radar system. Some of the difficulties were getting real time performance from MATLAB and minimizing the Ethernet packet dropouts during transmission and receiving using the URSP2. Due to these issues the chirp waveform transmitter and receiver could not be implemented successfully. However the simulation models were used to simulate a radar system with a range of 1000 to 3000 meters.

Some areas of additional work would make it possible to implement the radar system designed in the project. The USRP2 has gigabit Ethernet which makes it possible to handle high bandwidth of data. However due to the limited resources of current PCs it is not possible realize the USRP2 transmit and receive blocks, running on the same PC.
simultaneously. One option is to be able to use 2 different PCs, one to transmit and the other to receive. This will reduce the processing load on the PCs since each PC will be responsible for a single task. The simulation models developed in this project were used to study the effect of noise on target detection. It was also used to empirically determine the threshold value required to detect a target from a noisy return signal. The radar processing block can also be modified and developed to calculate the velocity of a moving target by detecting any Doppler shifts.
References


http://ewh.ieee.org/r1/new_hampshire/


*USRP Networked Series*. (2012). Retrieved from Ettus Research:
https://www.ettus.com/product/category/USRP_Networked_Series

http://en.wikipedia.org/wiki/Matched_filter
Appendix A

Matlab Code

Code to generate Single Chirp Waveform

function finalchirp = fcn(T,W,p)

% length of burst in seconds
T=7e-6;

% bandwidth of burst (Hz)
W=10e6;

% chirp parameters
p=2;
N = p*T*W;

% chirp vector
n = 0:N;
alpha = 1/(2*p*p*T*W);
s = exp(2i*pi*alpha*((n-.5*N).^2));

% chirp waveform
finalchirp = complex(zeros(1,length(s)*4));
for idx=1:1:length(s)
    finalchirp(1,idx) = s(1,idx);
end

Matlab code for the chirp division block

The code for the chirp division block is show in Appendix A.

function sendC = fcn(oC)

% chirp division
for i=1:length(oC)
    t1=[1:5:length(oC)];
    t2=[2:5:length(oC)];
    t3=[3:5:length(oC)];
    t4=[4:5:length(oC)];
    t5=[5:5:length(oC)];
end
length(oC);
realindex=1;
fC1 = complex(zeros(1,length(t1)*3));
for idx = [1:5:length(oC)]
    fC1(1,realindex) = oC(1,idx);
    realindex=realindex+1;
end
realindex=1;
fC2 = complex(zeros(1,length(t2)*3));
for idx = [2:5:length(oC)]
    fC2(1,realindex) = oC(1,idx);
    realindex=realindex+1;
end
fC3 = complex(zeros(1,length(t3)*3));
realindex=1;
for idx = [3:5:length(oC)]
    fC3(1,realindex) = oC(1,idx);
    realindex=realindex+1;
end
fC4 = complex(zeros(1,length(t4)*3));
realindex=1;
for idx = [4:5:length(oC)]
    fC4(1,realindex) = oC(1,idx);
    realindex=realindex+1;
end
fC5 = complex(zeros(1,length(t5)*3));
realindex=1;
for idx =5:5:length(oC)]
    fC5(1,realindex) = oC(1,idx);
    realindex=realindex+1;
end
length(fC1);
length(fC2);
length(fC3);
length(fC4);
length(fC5);
sendC =
    complex(zeros(1,length(fC1)+length(fC2)+length(fC3)+length(fC4)+length(fC5)));
sendC(1,1:length(fC1)) = fC1(1,:);
sidx = length(fC1)+1;
sendC(1,sidx:(sidx-1)+length(fC2)) = fC2(1,:);
sidx = sidx+length(fC2);
sendC(sidx:(sidx-1)+length(fC3)) = fC3(1,:);
sidx = sidx+length(fC3);
sendC(1,sidx:(sidx-1)+length(fC4)) = fC4(1,:);
sidx = sidx+length(fC4);
sendC(1,sidx:(sidx-1)+length(fC5)) = fC5(1,:);

Matlab Code to recreate chirp waveform from received samples
function [sendC,fCR1,fCR2,fCR3,fCR4,fCR5] = fcn(oC)
%
if length(oC) > 1682
    t1= length(oC);
    t2= length(oC);
t3 = length(oC);
t4 = length(oC);
t5 = length(oC);
fCR1 = complex(zeros(1,(t1))); for idx = [1:1:t1] fCR1(1,idx) = oC(1,idx); end
fCR2 = complex(zeros(1,(t2))); for idx = [1:1:t2] fCR2(1,idx) = oC(1,t1+idx); end
fCR3 = complex(zeros(1,(t3))); for idx = [1:1:t3] fCR3(1,idx) = oC(1,t1+t2+idx); end
fCR4 = complex(zeros(1,(t4))); for idx = [1:1:t4] fCR4(1,idx) = oC(1,t1+t2+t3+idx); end
fCR5 = complex(zeros(1,(t5))); for idx = [1:1:t5] fCR5(1,idx) = oC(1,t1+t2+t3+t4+idx); end
sendC = complex(zeros(1,length(fCR1)+length(fCR2)+length(fCR3)+length(fCR4)+length(fCR5))); realindex = 1; for idx = [1:5:length(sendC)
if realindex<length(fCR1)
    sendC(1,idx)=fCR1(1,realindex);
end;
realindex=realindex+1;
end
realindex = 1;
for idx =2:5:length(sendC)
    if realindex<length(fCR2)
        sendC(1,idx)=fCR2(1,realindex);
    end;
    realindex=realindex+1;
end
realindex = 1;
for idx =3:5:length(sendC)
    if realindex<length(fCR3)
        sendC(1,idx)=fCR3(1,realindex);
    end;
    realindex=realindex+1;
end
realindex = 1;
for idx =4:5:length(sendC)
    if realindex<length(fCR4)
        sendC(1,idx)=fCR4(1,realindex);
    end;
    realindex=realindex+1;
end
realindex = 1;
for idx = 5:5:length(sendC)
    if realindex < length(fCR5)
        sendC(1,idx) = fCR5(1,realindex);
    end;
    realindex = realindex + 1;
end

% sendC(1,1:length(fCR1)) = fCR1(1,:);
% sidx = length(fCR1)+1;
% sendC(1,sidx:(sidx-1)+length(fCR2)) = fCR2(1,:);
% sidx = sidx+length(fCR2);
% sendC(sidx:(sidx-1)+length(fCR3)) = fCR3(1,:);
% sidx = sidx+length(fCR3);
% sendC(1,sidx:(sidx-1)+length(fCR4)) = fCR4(1,:);
% sidx = sidx+length(fCR4);
% sendC(1,sidx:(sidx-1)+length(fCR5)) = fCR5(1,:);
else
    sendC = 0;
    fCR1 = 0;
    fCR2 = 0;
    fCR3 = 0;
    fCR4 = 0;
    fCR5 = 0;
end

Matlab Code to add 10 samples together.

Function [a1,a2,a3,a4,a5,a6,a7,a8,a9,a10] = fcn(u)
function y = fcn(u)
slab1 = complex(zeros(1,2805));
slab2 = complex(zeros(1,2805));
slab3 = complex(zeros(1,2805));
slab4 = complex(zeros(1,2805));
slab5 = complex(zeros(1,2805));
slab6 = complex(zeros(1,2805));
slab7 = complex(zeros(1,2805));
slab8 = complex(zeros(1,2805));
slab9 = complex(zeros(1,2805));
slab10 = complex(zeros(1,2805));
idx = 1;
for idx=1:1:2805-1
    slab1(1,idx)= u(idx,1);
end
idx =1;
for idx=1:1:2805-1
    slab2(1,idx)= u(idx+2805-1,1);
end
idx =1;
for idx=1:2805-1
    slab3(1,idx)= u(idx+(2805*2),1);
end
idx =1;
for idx=1:2805-1
    slab4(1,idx)= u(idx+(2805*3),1);
end
idx =1;
for idx=1:2805-1
    slab5(1,idx)= u(idx+(2805*4),1);
end
idx =1;
for idx=1:2805-1
    slab6(1,idx)= u(idx+(2805*5),1);
end
idx =1;
for idx=1:2805-1
    slab7(1,idx)= u(idx+(2805*6),1);
end
idx =1;
for idx=1:2805-1
    slab8(1,idx)= u(idx+(2805*7),1);
end
idx =1;
for idx=1:2805-1
    slab9(1,idx)= u(idx+(2805*8),1);
end
idx =1;
for idx=1:2805-1
    slab10(1,idx) = u(idx+(2805*9),1);
end