Analysis and Design of a Synthetic Aperture Radar System

Final Year Project 2007-08

Group Members

Misbah Ahmad Mussawar

Farrukh Rashid

Furqan Ahmed

Usman Tahir Mir

Supervisor

Mr. Kashif Siddiq

Department of Telecommunication Engineering

National University of Computer and Emerging Sciences

Submission

This report is being submitted to the Department of Telecom Engineering of the National University of Computer and Emerging Sciences in partial fulfillment of the requirements for the degree of BE in Telecom Engineering

Declaration

We hereby declare that the work presented in this report is our own and has not been presented previously to any other institution or organization

____

Supervisor

____

Misbah Ahmad Mussawar (i030713)

____

Farrukh Rashid (i030714)

____

Furqan Ahmed (i040026)

____

Usman Tahir Mir (i040214)

Abstract

A Synthetic Aperture Radar (SAR) is used for all-weather and all-time high resolution aerial and space based imaging of terrain. Being independent of light and weather conditions, SAR imaging has an advantage over optical imaging. Some of the SAR applications include Surveillance, Targeting, 3D Imaging, Navigation & Guidance, Moving Target Indication, and Environmental Monitoring.

This project aimed at the System-Level Design, Modeling and Simulation of a Synthetic Aperture Radar System and the Implementation of the signal processor for SAR using a TI C6416 DSP.

The system parameters have been specified in view of all the constraints and practical limitations. The performance metrics of the system such as range resolution and cross-range resolution, etc have been worked out and the system level specification has been worked out keeping in view the desired performance. Using MATLAB a as major tool, the specified system parameters have been tested for their accuracy and correctness. A simulation of Pulse Doppler radar is completed which includes waveform design, target modeling, LFM pulse compression, side lobe control and threshold detection. A SAR image formation algorithm(Doppler Beam Sharpening) have been implemented in MATLAB.

Table of Contents

List of Figures

Chapter

1

Introduction

1.1 Synthetic Aperture Radar

Synthetic Aperture Radar (SAR) is a type of radar which is used for all-weather and all-time high resolution aerial and space based imaging of terrain. The term all-weather means that an image can be acquired in any weather conditions like clouds, fog or precipitation etc. and the term all-time means that an image can be acquired during day as well as night.

1.1.1 Why to use SAR

The primary reason of using SAR is that the quality of image does not depend on weather or light conditions and images can be taken at any part of the day and in any weather. An optical imaging device (e.g. a camera) can not be used for aerial imaging of earth surface during night or when there is fog or clouds. Hence, synthetic aperture radars have an advantage over optical imaging devices.

A comparison between optical and SAR image of an airport is given in the following set of figures.

During day time, the SAR image with a resolution of 3m (left) and the optical image (right) look like:

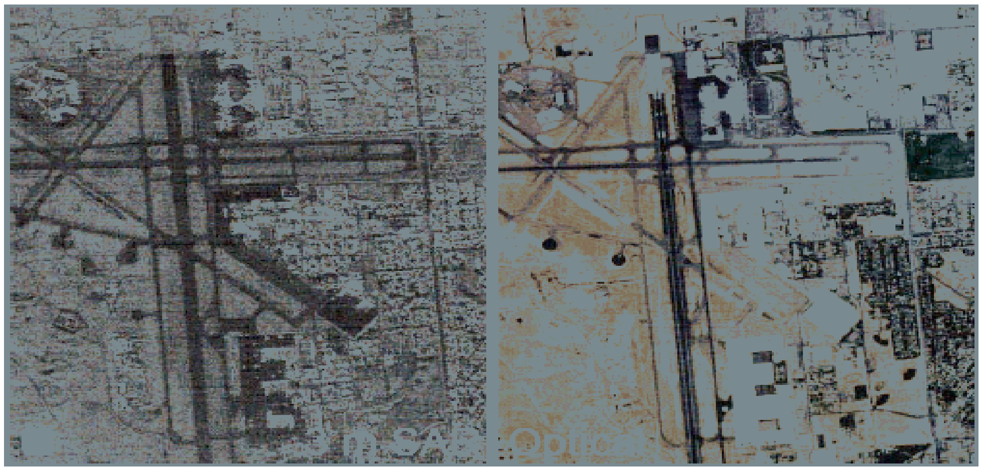


Figure 1.1 Comparison of SAR and Optical Image (Day)
(Source: http://www.sandia.gov/RADAR/sar_sub/images/)

During night, the SAR image with a resolution of 3m (left) and the optical image (right) look like:

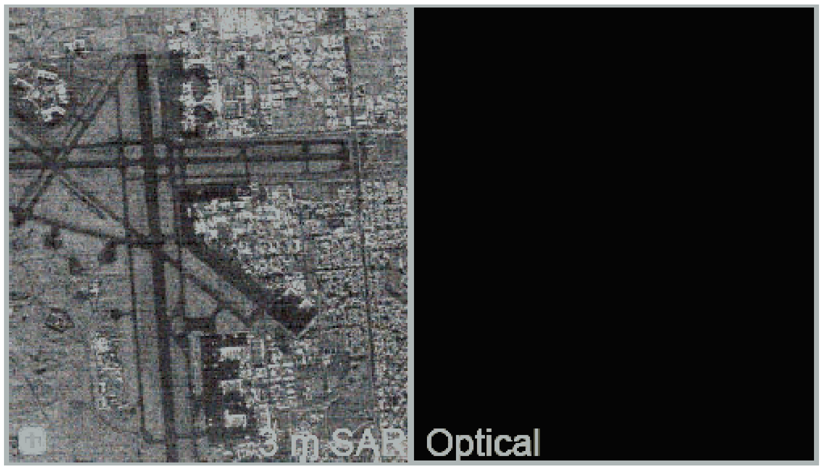


Figure 1.2 Comparison of SAR and Optical Image (Night)
(Source: http://www.sandia.gov/RADAR/sar_sub/images/)

Hence, we can see that the optical image does not provide any information if taken at night but SAR image is still same as that taken in the day time.

1.1.2 Applications of SAR

This section discusses a few of the applications for synthetic aperture radar. The applications increase rapidly as new technologies and innovative ideas are developed. While SAR is often used because of its all-weather, day-or-night imaging capability, it also finds application because it renders a different view of a "target," with synthetic aperture radar being at a much lower electromagnetic frequency than optical sensors.

Targeting , Reconnaissance, and Surveillance
Many applications for synthetic aperture radar are for targeting, reconnaissance, and surveillance. These applications are driven by the military's need for all-weather, day-and-night imaging sensors. SAR can provide sufficiently high resolution to distinguish terrain features and to recognize and identify selected man made targets. More is available at http://www.sandia.gov/RADAR/sarapps.html.

On the Oceans
Most of the man-made illegal or accidental spills are well visible on radar images. Ships can be detected and tracked from their wakes. Also natural seepage from oil deposits can be observed. They provide hints to the oil industries. Scientists are studying the radar backscatter from the ocean surface related to wind and current fronts, to eddies, and to internal waves. In shallow waters SAR imagery allows to infer the bottom topography. The topography of the ocean floor can be mapped using the very precise - ERS Altimeter, because the sea bottom relief is reflected on the surface by small variations of the sea surface height.
The ocean waves and their direction of displacement can be derived from the ERS SAR sensor operated in "Wave Mode". This provides input for wave forecasting and for marine climatology.
At high latitudes, SAR data is very useful for regional ice monitoring. Information such as ice type and ice concentration can be derived and open leads detected. This is essential for navigation in ice-infested waters. More is available at http://earth.esa.int/applications/data_util/SARDOCS/

Treaty Verification and Nonproliferation
The ability to monitor other nations for treaty observance and for the nonproliferation of nuclear, chemical, and biological weapons is increasingly critical. Often, monitoring is possible only at specific times, when overflights are allowed, or it is necessary to maintain a monitoring capability in inclement weather or at night, to ensure an adversary is not using these conditions to hide an activity. SAR provides the all-weather capability and complements information available from other airborne sensors, such as optical or thermal-infrared sensors. More is available at http://www.sandia.gov/RADAR/sarapps.html.

Interferometry (3-D SAR)
Interferometric synthetic aperture radar (IFSAR) data can be acquired using two antennas on one aircraft or by flying two slightly offset passes of an aircraft with a single antenna. Interferometric SAR can be used to generate very accurate surface profile maps of the terrain. IFSAR is among the more recent options for determining digital elevation. It is a radar technology capable of producing products with vertical accuracies of 30 centimeters RMSE. Not only that, but IFSAR provides cloud penetration, day/night operation (both because of the inherent properties of radar), wide-area coverage, and full digital processing. The technology is quickly proving its worth. More about IFSAR is available at http://www.geospatial-solutions.com/geospatialsolutions.

On the Land
The ability of SAR to penetrate cloud cover makes it particularly valuable in frequently cloudy areas such as the tropics. Image data serve to map and monitor the use of the land, and are of gaining importance for forestry and agriculture.
-Geological or geomorphological features are enhanced in radar images thanks to the oblique viewing of the sensor and to its ability to penetrate - to a certain extent - the vegetation cover.
-SAR data can be used to georefer other satellite imagery to high precision, and to update thematic maps more frequently and cost-effective, due to its availability independent from weather conditions.
-In the aftermath of a flood, the ability of SAR to penetrate clouds is extremely useful. Here SAR data can help to optimize response initiatives and to assess damages. More is available at http://earth.esa.int/applications/data_util/SARDOCS/

Navigation, Guidance, and Moving Target Indication
Synthetic aperture radar provides the capability for all-weather, autonomous navigation and guidance. By forming SAR reflectivity images of the terrain and then by correlation of the SAR image with a stored reference (obtained from optical device or a previous SAR image), a navigation update can be obtained. Position accuracies of less than a SAR resolution cell can be obtained. SAR may also be used to guidance applications by pointing or "squinting" the antenna beam in the direction of motion of the airborne platform. In this manner, the SAR may image a target and guide a munition with high precision.

The motion of a ground-based moving target such as a car, truck, or military vehicle, causes the radar signature of the moving target to shift outside of the normal ground return of a radar image. New techniques have been developed to automatically detect ground-based moving targets and to extract other target information such as location, speed, size, and Radar Cross Section (RCS) from these target signatures. More is available at http://www.sandia.gov/RADAR/sarapps.html.

1.2 Project Description

This section describes this project in detail.

1.2.1 Project Title

The project is titled as ‘Analysis and Design of a Synthetic Aperture Radar System’.

1.2.2 Project Objective

This project aimed at the System-Level Design, Analysis, Modeling and Simulation of a Synthetic Aperture Radar System along with the Implementation of the signal processor for SAR using a TI C6416 DSP.

The track followed consisted of basic understanding of radar concepts, various types of radar systems, radar principles, waveform design and analysis, signal processing techniques, SAR system level design considerations, SAR processing and image formation.

1.2.3 Reasons for Choosing This Project

Synthetic aperture radar is an extensive research area in the field of radar. Being used for imaging from air or from space, it has its applications in many important areas like defense, environmental monitoring and earth observations, etc. This importance of SAR was the motivation behind the selection of this project.

Chapter

2

Background

In this chapter, the theoretical background required to understand this project is provided. It includes the basics of radar operation, the terminologies used in radar, performance metrics of radar systems, waveform design for radars, receiver design for radar, radar signal processing, fundamentals of SAR and SAR signal processing.

2.1 Introduction

2.1.1 What is Radar

"Radar" is an acronym for *RA*dio *D*etection *A*nd *R*anging. Radar systems use modulated waveforms and directive antennas to transmit electromagnetic energy into a specific volume in space to search for targets. Targets within a search volume will reflect portions of this energy (returns or echoes) back to the radar. These echoes are then processed by the radar receiver and signal processor to extract target information such as range(distance), velocity, angular position, and other characteristics of the target.

Radars can be classified according to various criteria. These criteria may include the deployment (e.g. ground based, airborne, spaceborne, or ship based radar systems), operational characteristics (e.g. frequency band, antenna type, and waveforms utilized), nature of mission or purpose (e.g. weather, acquisition and search, tracking, track-while-scan, fire control, early warning, over the horizon, terrain following, and terrain avoidance radars).

The mostly used classification is based upon the type of waveform and the operating frequency used. Considering the waveforms first, radars can be Continuous Wave (CW) or Pulsed Radars (PR). CW radars are those that continuously emit electromagnetic energy, and use separate transmit and receive antennas. Pulsed radars use a train of pulsed waveforms (mainly with modulation). In this category, radar systems can be classified on the basis of the Pulse Repetition Frequency (PRF), as low PRF, medium PRF, and high PRF radars.

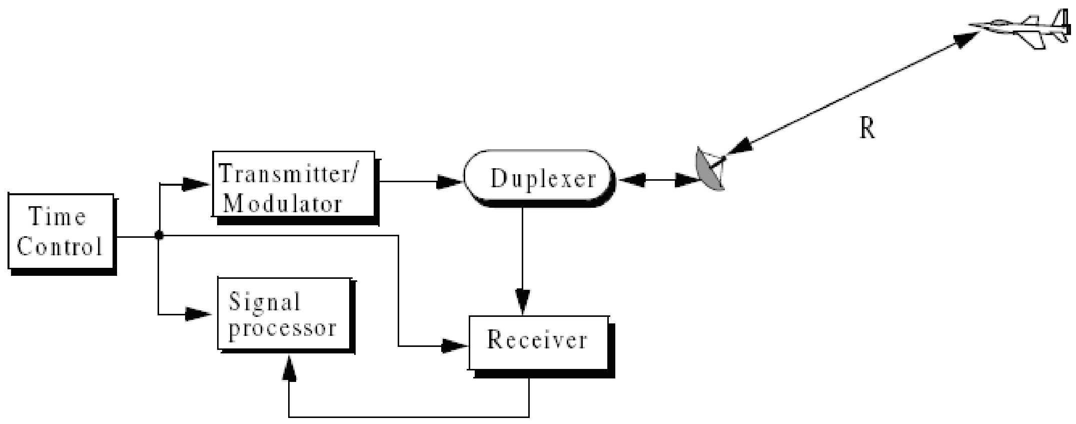


Figure 2.1 A Basic Pulsed Radar System
(Source: Radar Systems Analysis and Design using MATLAB by *Bassem R. Mahafaza*)

2.2 Radar Parameters and Terminologies

The operation of a radar system comprises of some parameters such as range, range resolution, Doppler frequency, and Doppler resolution. These terms are explained in the following text.

2.2.1 Range

Range is defined as the radial distance of the target from the aperture of the radar antenna. In the above figure, range is denoted by 'R'. The target's range 'R' is computed by measuring the time delay Δt ; it takes a pulse to travel the two-way path between the radar and the target. Since electromagnetic waves travel at $c=3x10^8$m/s, the speed of light c, then **$R = c\Delta t / 2$**, where R is in meters and Δt is in seconds. The factor of 0.5 is needed to account for two-way delay.

2.2.2 Pulse Repetition Interval and Frequency (PRI & PRF)

A pulsed radar transmits and receives a train of pulses, as illustrated below:

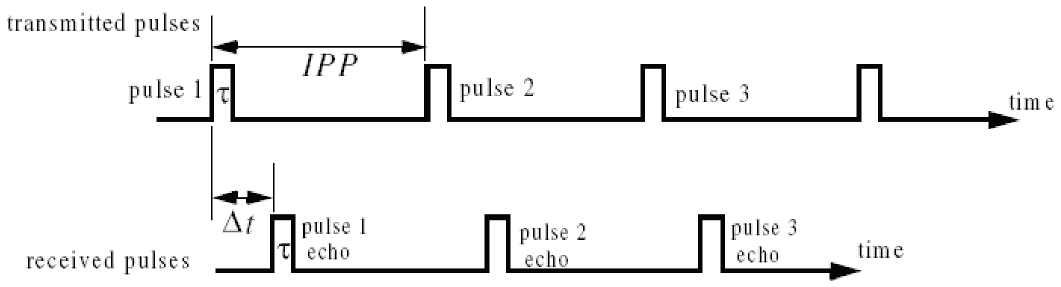


Figure 2.2 PRF & PRI
(Source: Radar Systems Analysis and Design using MATLAB by *Bassem R. Mahafaza*)

In the above figure, each pulse has a width τ and the time between to consecutive pulses is T. The time separation between two consecutive pulses is known as Inter Pulse Period (IPP) or Pulse Repetition Interval (PRI, denoted by 'T') and its inverse is known as Pulse Repetition Frequency (PRF, denoted by f_r).

PRI and the Range Ambiguity

The range corresponding to the two-way time delay 'T' is known as the radar unambiguous range, R_u.
To avoid ambiguity in range, once a pulse is transmitted the radar must wait a sufficient length of time so that the returns from targets at maximum range are back before the next pulse is emitted. It follows that the maximum unambiguous range must correspond to half of the PRI, i.e.

$$R_u = cT/2 = c/2f_r$$

2.2.3 Range Resolution

Range resolution, denoted as ΔR, is a radar metric that describes its ability to detect targets in close proximity to each other as distinct objects. Radar systems are normally designed to operate between a minimum range R_{min}, and maximum range R_{max}. The distance between and is divided into M range bins (gates), each of width ΔR, i.e.

$$M = (R_{max} - R_{min}) / \Delta R$$

Targets separated by at least ΔR will be completely resolved in range. ΔR should be greater or equal to cτ/2. And since the radar bandwidth B is equal to 1/ τ , then

$$\Delta R = c\tau/2 = c/2B$$

Doppler Ambiguity
If the Doppler frequency of the target is high enough to make an adjacent spectral line move inside the Doppler band of interest, the radar can be Doppler ambiguous. Therefore, in order to avoid Doppler ambiguities, radar systems require high PRF rates when detecting high speed targets.

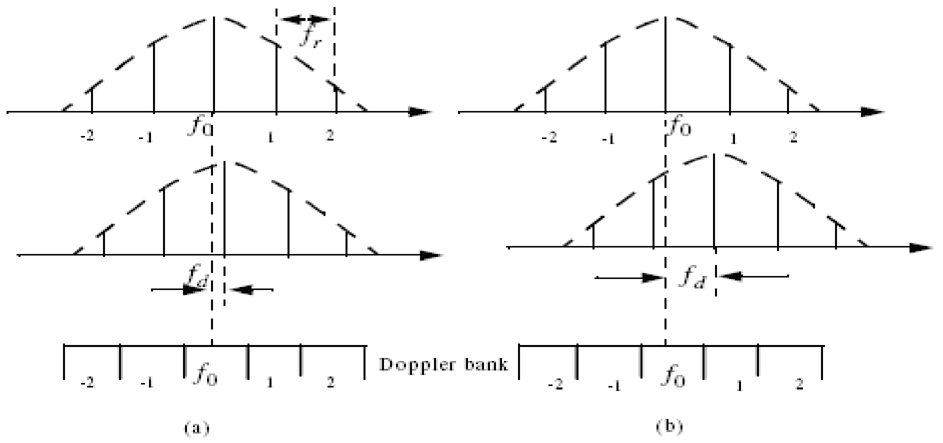


Figure 2.3 Spectra of transmitted and received waveforms, and Doppler bank. (a) Doppler is resolved. (b) Ambiguous Doppler measurement
(Source: Radar Systems Analysis and Design using MATLAB by *Bassem R. Mahafaza*)

2.3 Basic Radar Functions

Most uses of radar can be classified as *detection, tracking*, or *imaging.*

2.3.1 Detection

Detection means to decide whether a specific object is present in the coverage area of radar o not. This is done by comparing the amplitude of received pulses with a threshold. If it crosses the threshold, a target is resent and vice versa.

2.3.2 Tracking

Once an object has been detected, it may be desirable to track its location or velocity. A radar naturally measures position in a spherical coordinate system with its origin at the radar antenna's phase center.

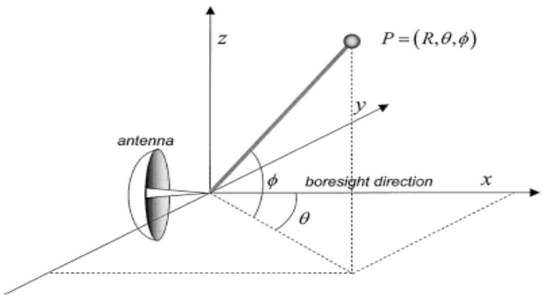


Figure 2.4 Spherical coordinate system for radar measurements
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.3.3 Imaging

Airborne radars are also used for imaging the earth terrain. Radars have there own electromagnetic illumination and hence are independent of light or weather conditions for imaging. Therefore, radar imaging enjoys this benefit over optical imaging devices.

2.4 Radar Data Acquisition - Collecting Pulsed Radar Data

2.4.1 A Single Range Sample

Consider a pulse of length τ whose leading edge is transmitted at time t=0. Then we have to sample the receiver output at time t_0 = $2R_0/c$. Hence, the scatterers over a range of ΔR = $c\tau/2$ meters contribute to the sample. The following diagram explains it:

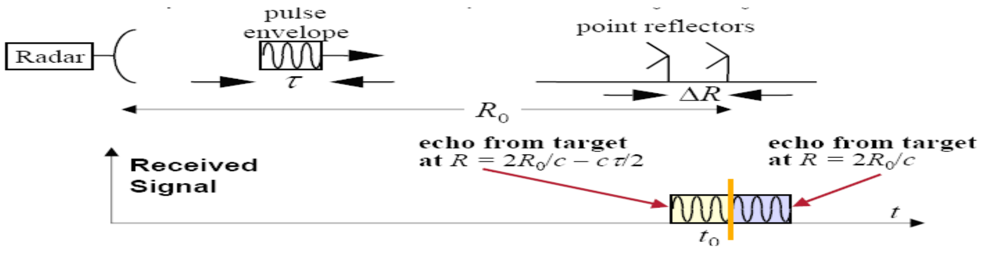


Figure 2.5 A Single Range Sample
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.4.2 A Single Pulse and Multiple Range Samples (Fast-Time)

Now if we want to collect multiple returns caused by a single pulse, we will use a coherent receiver. Each range cell contains 1 complex number and each range cell represents echo from a different range interval. These samples are also called range bins, range gates, or *fast time samples.* Following diagram illustrates fast time samples by one pulse.

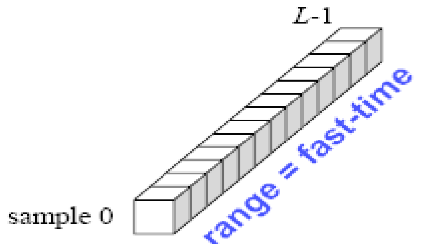
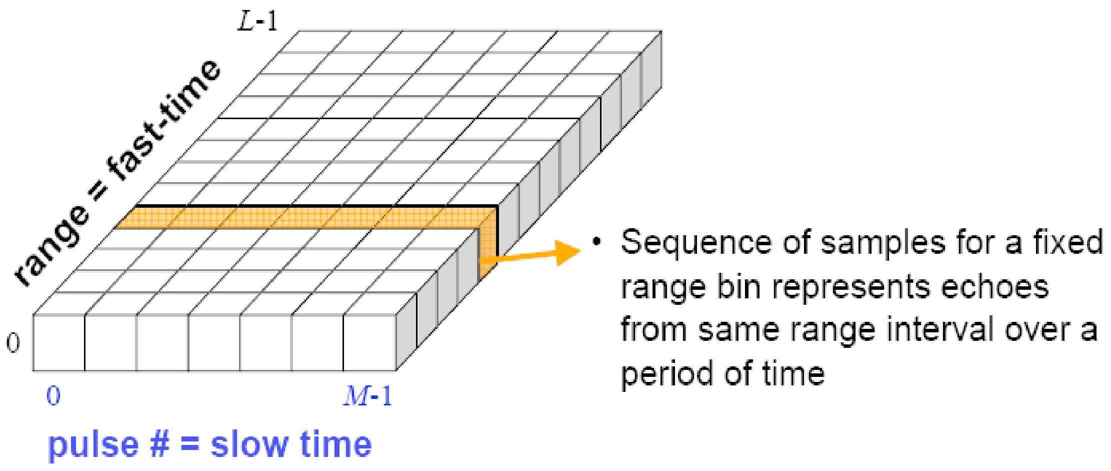


Figure 2.6 A Row Containing Fast Time Samples
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.4.3 Multiple Pulses and Multiple Range Samples (Slow Time)

If radar transmits several pulses, and collects multiple samples of each transmitted pulse, a two dimensional matrix is formed. Pulses are repeated in an interval known as the coherent processing interval (CPI) or dwell. The new axis is called pulse-number or *slow-time.* Below is a diagram showing the 2-D matrix.



L-1
range = fast-time
0
0
M-1
pulse # = slow time
• Sequence of samples for a fixed range bin represents echoes from same range interval over a period of time

Figure 2.7 A Slow / Fast Time Matrix
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.4.4 Multiple Pulses, Multiple Range Samples and Multiple Receiving Channels

If the radar uses more than one receiver (e.g. in the case of multiple phase center antenna, monopulse antenna, *etc)* and collects the above explained 2-D matrix through each channel, a three dimensional matrix or a cube is formed. The cube hence formed is called a radar datacube and is shown below.

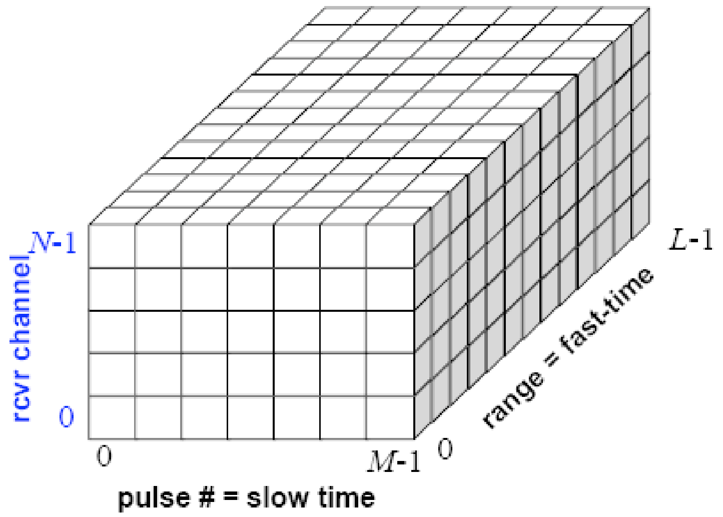


Figure 2.8 A Datacube
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.5 The Radar Datacube

2.5.1 Visualizing the Radar Datacube

Many radars collect a real or conceptual "datacube" of data on which, various operations are performed to achieve different goals.

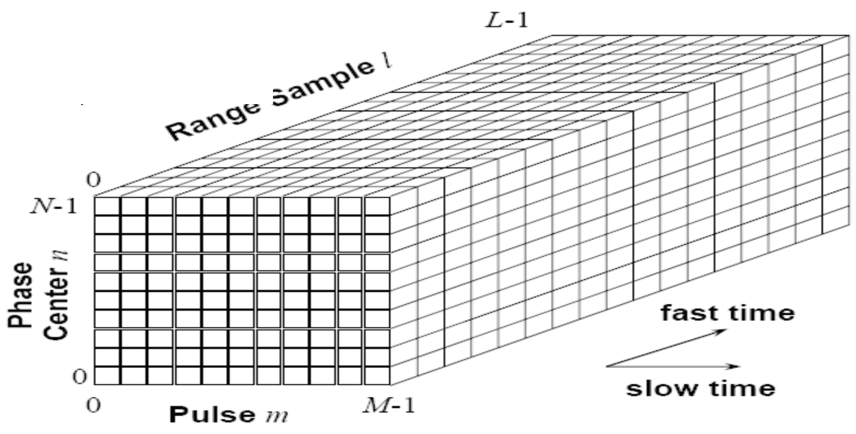


Figure 2.9 The Radar Datacube
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.5.2 Processing the Radar Datacube

Standard radar signal processing algorithms correspond to operating in 1-D or 2-D along various axes of the datacube. The selection of various axes with their use is illustrated in the following diagram.

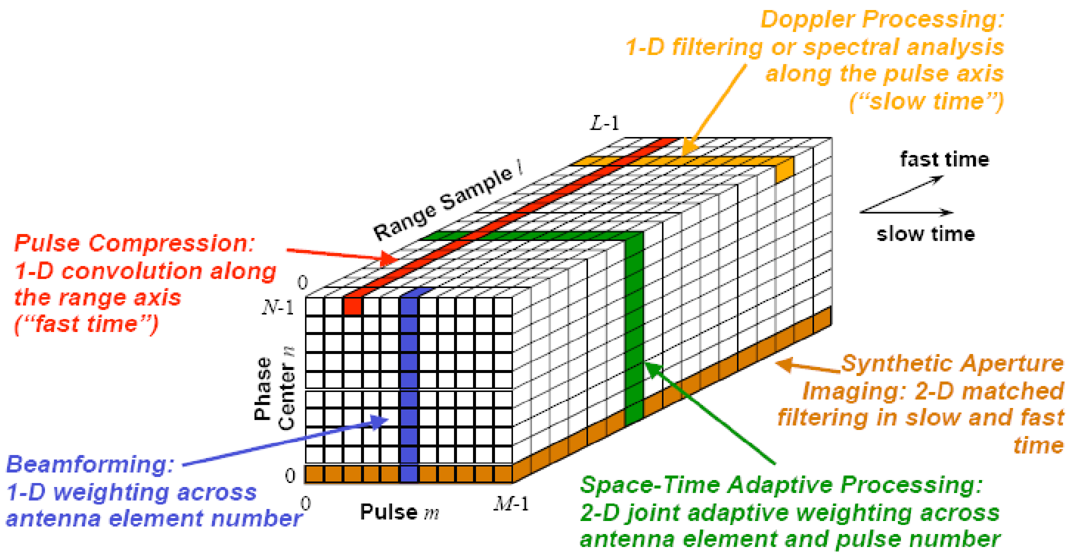

Doppler Processing: 1-D filtering or spectral analysis along the pulse axis ("slow time")
L-1
Range Sample l
fast time
slow time
Pulse Compression: 1-D convolution along the range axis ("fast time")
0
N-1
Phase Center n
Synthetic Aperture Imaging: 2-D matched filtering in slow and fast time
Beamforming: 1-D weighting across antenna element number
0
0
Pulse m
M-1
Space-Time Adaptive Processing: 2-D joint adaptive weighting across antenna element and pulse number

Figure 2.10 Processing of the Datacube
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.5.3 The Data Matrix

In synthetic aperture radar, the output 2-D matrix of a single receiving channel is used for processing. The outputs of each receiver channel are processed independently.

A 2-D data matrix is depicted below and its various parameters are illustrated.

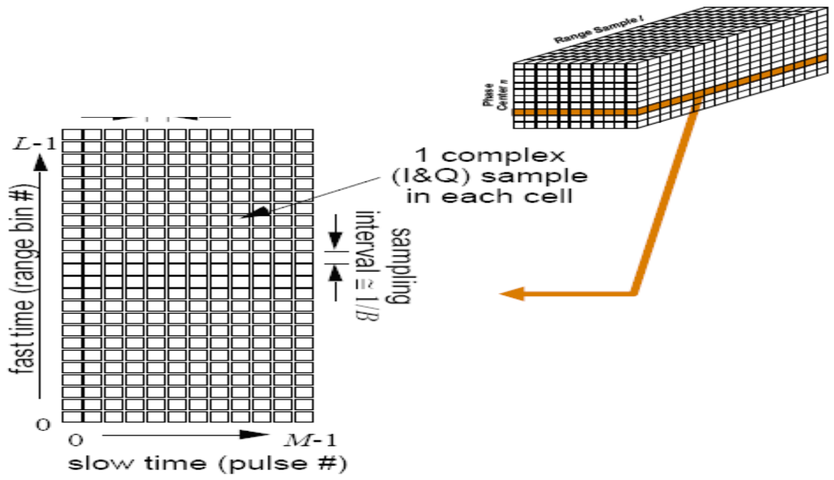


Figure 2.11 A 2D Matrix as a part of Datacube
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.6 Waveform Design for Radar

In a radar system, it is important to choose the waveform design which is appropriate to the purpose and requirements of the radar system. This chapter will enlighten various considerations regarding selection of the waveform. Few types of waveforms and a brief comparison between some types is also given in this chapter.

2.6.1 Waveform Characteristics

Every waveform has some properties that describe it and help selecting a waveform for use. They are listed below.

- Duration
- Power
- Energy
- Bandwidth
- Range Resolution
- Range Sidelobes
- Doppler Resolution
- Doppler Sidelobes
- Doppler Tolerance
- Range Doppler Coupling

2.7 Waveform Selection Criteria

For selecting a waveform the most important characteristics of a waveform i.e. energy, range resolution and Doppler resolution are considered.

- **Energy**
- **Range Resolution**
- **Doppler Resolution**

2.8 Types of Waveforms

This section will introduce a special type of waveform called *linear frequency modulated* waveforms and will provide the motivation behind its use and its benefits. Then a comparison of simple pulse with LFM pulse will be provided.

2.8.1 Linear Frequency Modulated (LFM) Waveforms

It is a modulation technique which is used to decouple waveform energy and range resolution. Frequency or phase modulated waveforms can be used to achieve much wider operating bandwidths. Linear Frequency Modulation (LFM) is commonly used. In this case, the frequency is swept linearly across the pulse width, either upward (up-chirp) or downward (down-chirp). The matched filter bandwidth is proportional to the sweep bandwidth, and is independent of the pulse width. The pulse width is tau , and the bandwidth is Beta. The figure below shows the instantaneous value of an LFM signal.

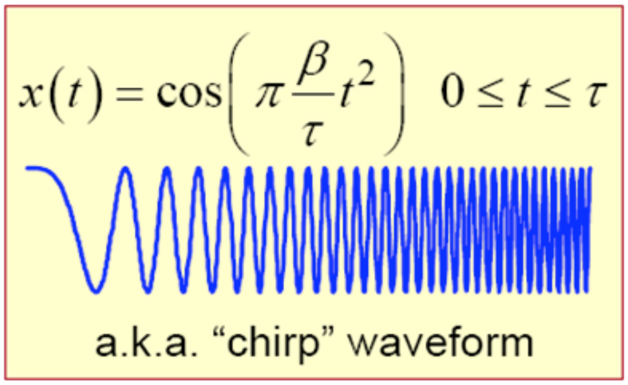


Figure 2.12 An LFM Waveform as a function of time
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

The LFM instantaneous frequency can be expressed by

$$f(t) = \frac{1}{2\pi} \frac{d}{dt}\psi(t) = f_0 + \mu t \qquad -\frac{\tau}{2} \le t \le \frac{\tau}{2}$$

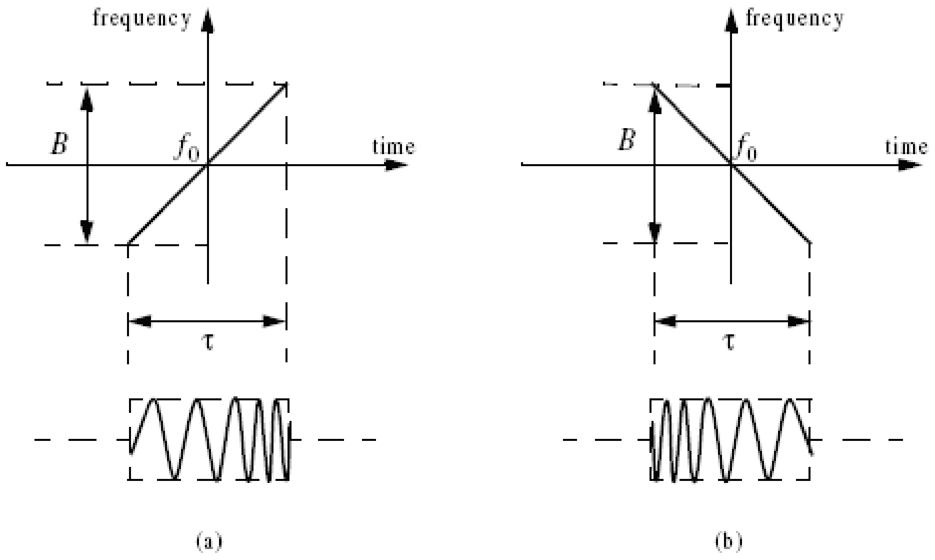


Figure 2.13 Up-chirp and Down-chirp
(Source: Radar Systems Analysis and Design by *Bassem R Mahafaza)*

2.8.2 Comparison of Simple Pulse and LFM Pulse

Simple Pulse	LFM Pulse
Pulse and Energy Coupled	Pulse and Energy Decoupled
Matched Filter Output is Triangle	Matched Filter Output is a Narrow Sinc
Gives Coarse Resolution	Gives Finer Resolution

2.9 Matched Filter for Waveforms

2.9.1 What is a Matched Filter

The matched filter is the optimal linear filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic noise. Matched filters are commonly used in radar, in which a known signal is sent out, and the reflected signal is examined for common elements of the out-going signal.

A matched filter is obtained by correlating a known signal, or template, with an unknown signal to detect the presence of the template in the unknown signal. This is equivalent to convolving the unknown signal with a time-reversed version of the template.

2.10 Pulse Compression

Pulse compression is a signal processing technique mainly used in radar, sonar and echography to augment the distance resolution as well as the signal to noise ratio. This is achieved by modulating the transmitted pulse. Pulse compression allows us to achieve the average transmitted power of a relatively long pulse, while obtaining the range resolution corresponding to a short pulse.

2.11 SAR Fundamentals

2.11.1 Cross-Range Resolution and Aperture Time in Radars

The ability of a radar to distinguish to targets in the azimuth dimension is known as cross-range resolution. In real beam imaging, the cross-range resolution is proportional to range. By using synthetic arrays, we can decouple cross-range and range.

In a real-beam, forward looking radar if Θ_{az} is the azimuth beamwidth and R_o is the range, the cross-range resolution is given by $\Delta CR = R_o\Theta_{az}$.

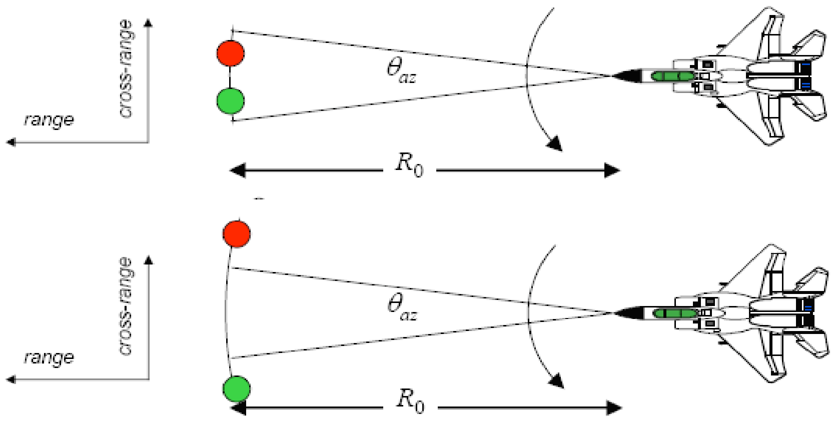
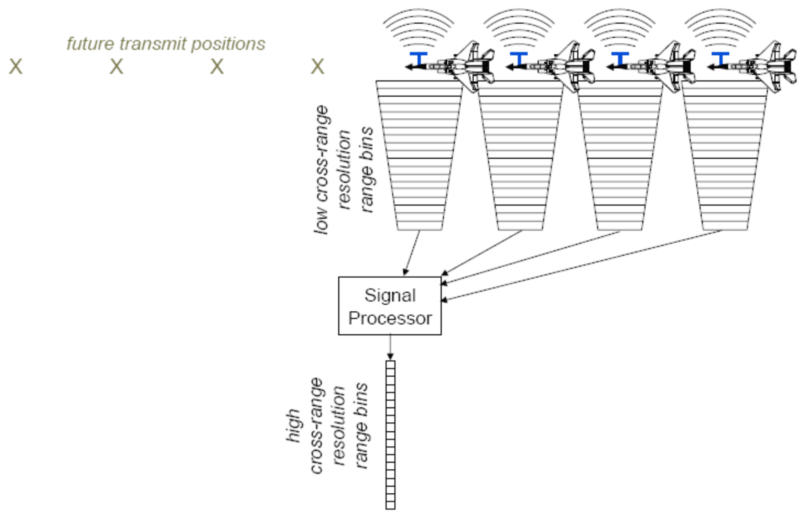


Figure 2.14 Cross-Range Resolution using Real Beam Radar
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

In the above figure, at a fixed range, two scatterers are considered to be just at the point of being resolved in cross-range when they are separated by the width of the (3 dB) main beam and beamwidth of the antenna is $\theta_{az} = \lambda / D_{az}$ where D_{az} is the width of antenna in the azimuth direction. Therefore, cross-range resolution of a real beam radar is $R\lambda / D_{az}$. *Hence, in real beam radar, cross-range resolution is dependent upon operating range.*

The synthetic aperture viewpoint enables us to synthesize a virtual large antenna array. The physical antenna is one "element" of the synthetic array. Data is collected at each position sequentially, and then processed together. The effective aperture size is determined by the distance traveled while collecting a data set. The following figure shows the concept synthetic aperture array.



future transmit positions
low cross-range resolution range bins
Signal Processor
high cross-range resolution range bins

Figure 2.15 Collecting Highr Cross-Range Resolution Samples
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

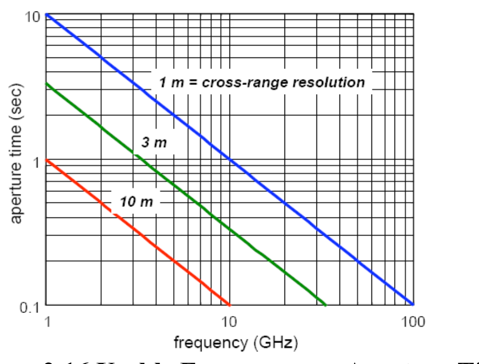
Cross-Range Resolution over an Aperture Time T_a
Suppose we integrate data over an aperture time T_a, let the speed of the radar platform be $\boldsymbol{v}$, then the resolution we will obtain can be found as:

$$D_{SAR} = vT_a \; ; \; \theta_{SAR} = \lambda / 2vT_a$$

$$\Rightarrow \quad \Delta CR = R\,\theta_{SAR} = R\,\lambda / 2vT_a$$

$$\Rightarrow \quad T_a = R\,\lambda / 2v\Delta CR$$

For some given cross-range resolutions, the useable frequency and aperture times are given in the following graphs.



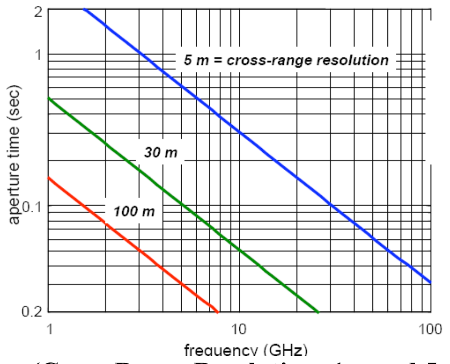


Figure 2.16 Usable Frequency vs. Aperture Time (Cross-Range Resolutions 1m and 5m)
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

The maximum practical value of aperture time is limited by the physical antenna on the platform. Any one scatterer contributes to the SAR data only over a maximum synthetic aperture size equal to the travel distance between two points shown, which equals to the width of the physical antenna beam at the range of interest, namely $R\Theta_{az.}$ Maximum synthetic aperture size is the maximum distance traveled while target is illuminated. The corresponding maximum effective aperture time is $R_o\Theta_{az} / v$.

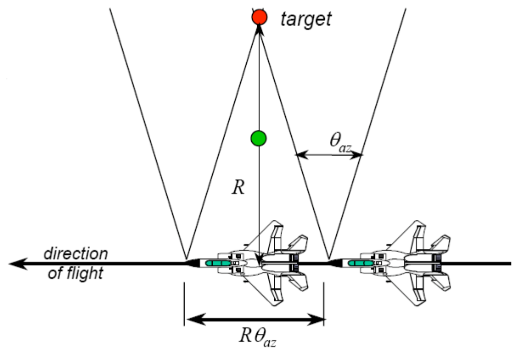


Figure 2.17 Maximum Synthetic Aperture Size
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

From the above figure, we have $D_{SARmax} = R\theta_{az}$

2.11.2 Sidelooking Stripmap SAR

In this mode, a phased array aperture is formed parallel to the velocity vector. Therefore array factor boresight is perpendicular to the velocity vector. Phase steering can change this steering of the physical antenna squints the "element pattern". "Element pattern" is not a broad typical sinc-like antenna pattern. In stripmap SAR, physical antenna does not scan but rather, a physical beam is dragged along with platform motion. Effective aperture size D_{SAR} or aperture time T_a determined by how many pulses contribute to each pixel. This basic SAR operational mode is called stripmap SAR.

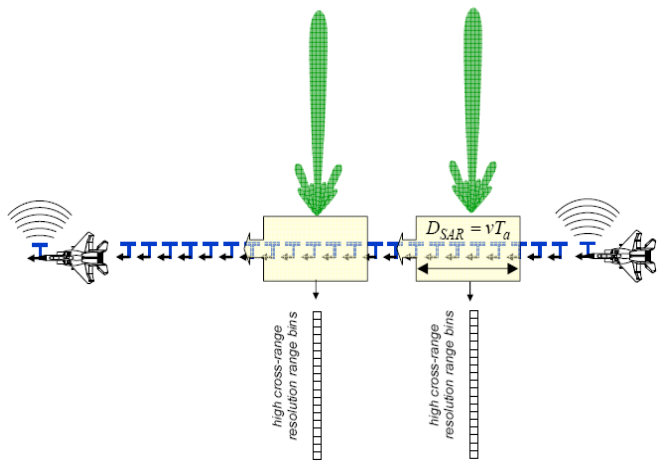

$D_{SAR} = vT_a$
high cross-range resolution range bins
high cross-range resolution range bins

Figure 2.18 Stripmap SAR Data Collection
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

Best Case SAR Stripmap Resolution
Cross-range resolution corresponding to the maximum synthetic aperture size is given by

$$\theta_{SAR} = \lambda / 2D_{SAR} = \lambda / 2R\theta_{az} = D_{az} / 2R$$

$$\Rightarrow \quad \Delta CRmin = R\,\theta_{SAR} = D_{az} / 2$$

Hence, a larger potential aperture gives a finer resolution. Increase in effective aperture size exactly cancels increase in beamwidth with range.

Conclusion
- By using synthetic aperture, cross-range resolution is now independent of range.
- By using synthetic aperture, cross-range resolution is now much smaller and comparable to range resolution.
- By using synthetic aperture, we have achieved huge integration gains.

2.11.3 Stripmap Image Size
The image size of a stripmap SAR which is also known as "Swath length" denoted by L_s is upper bounded by elevation pattern footprint on ground.

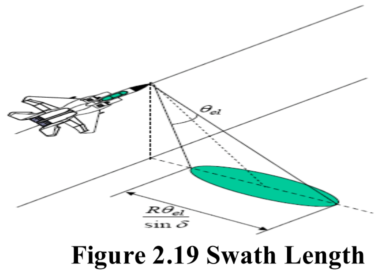


Figure 2.19 Swath Length
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

Doppler Bandwidth

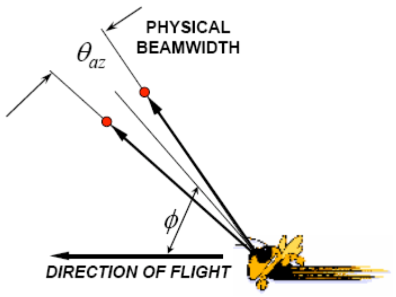


Figure 2.20 Doppler Bandwidth
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

The radial velocity relative to stationary scatterers differs across the physical beamwidth. The difference between the maximum and the minimum Doppler shift across area of the beamwidth is known as Doppler bandwidth.

Upper and Lower Bounds on PRF
The upper and lower bounds on the PRF of a sidelooking stripmap SAR are given by the relation:

$$2v / D_{az} <= PRF <= cD_{el}\tan\delta / 2\lambda R$$

Cross-Range Sampling Interval
The Nyquist sampling interval for sidelooking stripmap SAR is

$$T_s = D_{az} / 2v \text{ sec}$$

2.11.4 Range Migration

Range migration describes the downrange curvature of the data as cross-range position varies. Range migration means that returns from a given scatterer will be spread over multiple range bins even after pulse compression. Shifting and interpolation in range can straighten this out.

Range Walk
It is defined as the difference in range to scatterer at beginning ($u = -D_{SAR}/2$) and end ($u = +D_{SAR}/2$) of the synthetic aperture. It is always zero for side-looking systems.

Range Curvature
It is defined as the difference in range to scatterer at beginning or end ($u = \pm D_{SAR}/2$) and middle ($u = 0$) of the synthetic aperture. For a constant aperture time, range curvature decreases with range and for a constant resolution, it increases with range.

2.11.5 SAR Data Acquisition

Number of Range Samples per Pulse

Number of range samples to be processed per pulse = (swath length)/(range resolution) + (waveform BT product), i.e.

$$L = Ls / \Delta R + (\beta\tau - 1)$$

Number of Pulses

Number of pulses denoted by M contributing to the point target response = (aperture time)(PRF) i.e.

$$M = T_a PRF$$

$$=(\lambda R / 2v\Delta CR)(2v\theta_{az} / \lambda)$$

$$\Rightarrow \; M >= (R\theta_{az} / \Delta CR) = R\lambda / \Delta CRD_{az}$$

2.11.6 Goal of SAR Signal Processing

Echo energy from a single point target is distributed in both range and cross-range (range bins and pulse number, or fast and slow time). This function is called the point spread response (PSR) or point target response (PTR) of the SAR sensor.

Goal of SAR processing is to compress the PSR in both range and cross-range to a single (correctly located) point. This process is the 2D equivalent of pulse compression. The approach is essentially a matched filter but many variations in implementation and accuracy. Shape of point target response depends on range of the scatterer therefore SAR processing is not shift-invariant.

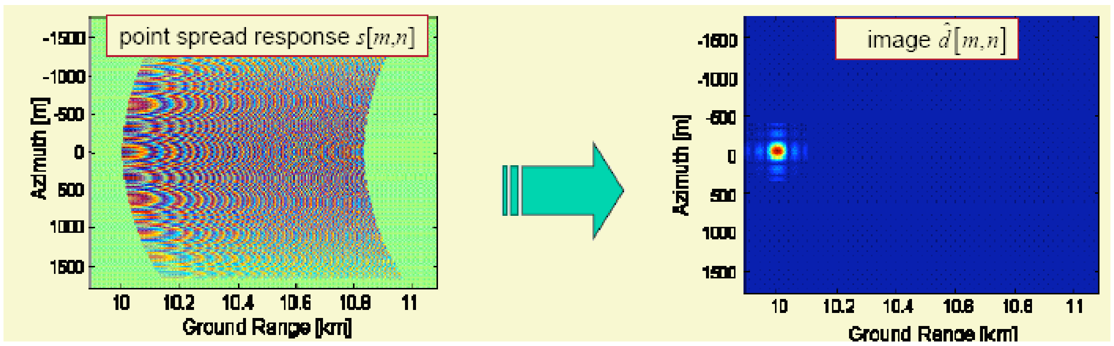


Figure 2.21 From Point Spread Response to the final Image
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.11.7 Doppler Beam Sharpening Algorithm

In this project, Doppler beam sharpening algorithm is used for image formation.. This algorithm is the original form of SAR envisioned. It uses a constant aperture time for all ranges. It provides a substantial improvement over real beam cross-range resolution and has relatively low computational requirements.

DBS Operation Summary

1. Data is collected over an aperture time T_a selected to get desired cross-range resolution.
2. Doppler spectrum is computed in each range bin using DFT. It produces range-Doppler image.
3. Map Doppler axis to cross-range using

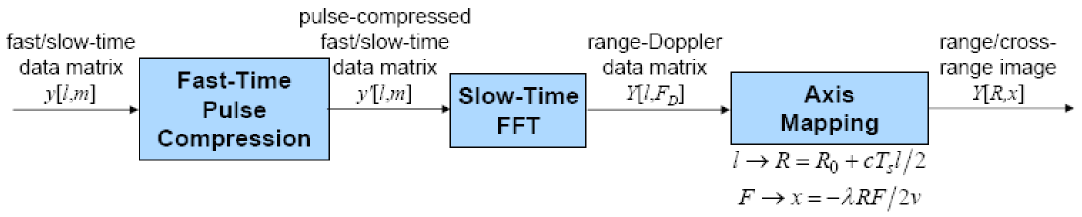

fast/slow-time data matrix $y[l,m]$
Fast-Time Pulse Compression
pulse-compressed fast/slow-time data matrix $y'[l,m]$
Slow-Time FFT
range-Doppler data matrix $Y[l,F_D]$
Axis Mapping
$l \to R = R_0 + cT_s l/2$
$F \to x = -\lambda RF/2v$
range/cross-range image $Y[R,x]$

Figure 2.22 Block diagram of DBS Algorithm
(Source: Fundamentals of Radar Signal Processing by *Mark A. Richards)*

2.12 References

i. Fundamentals of Radar Signal Processing by *Mark A. Richards*
ii. Radar Systems Analysis using MATLAB by Bassem R.Mahafaza
iii. http://www.wikipedia.com/

Chapter

3

Experiments, Simulations and Results

3.1 MATLAB Experiment to study LFM Characteristics

This section contains the results of an experiment performed using MATLAB to demonstrate various characteristics of an LFM Waveform. The output graph and list of observations for each graph is given below.

3.1.1 Effects of BT Product on Chirp Spectrum

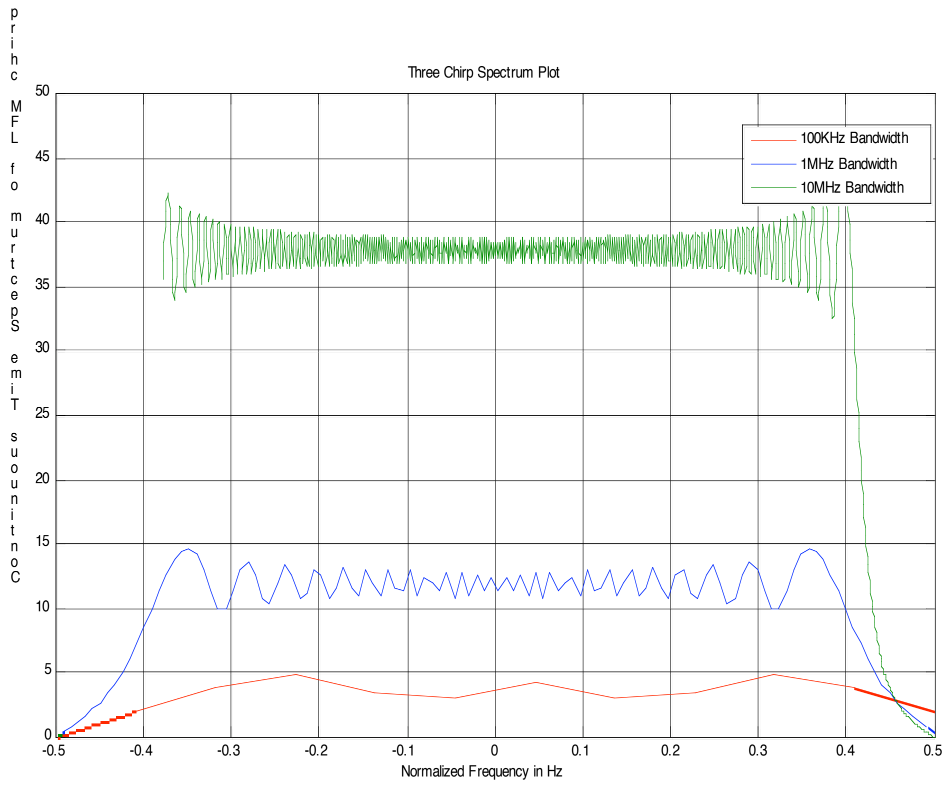


Figure 3.1 Three LFM Pulses with different BT Products

Observations:-

- Three chirp pulses of same pulse duration of 100e-6 seconds but with swept bandwidths of 100 KHz, 1 MHz and 10 MHz were generated at an over sampling factor of k=1.2. It is proved that as bandwidth increases sampling rate also increases.
- All the three spectra have the same bandwidth in the plot as the frequency is normalized.

- The amplitudes of the spectra differed significantly. The spectrum amplitude is proportional to bandwidth. As bandwidth increased spectrum amplitude also increased.

Conclusion:-
It can be said that by increasing bandwidth time product we can approximate magnitude of Fourier transform of a chirp with a rectangle function.

3.1.2 Comparison of LFM pulse and Simple pulse Matched Filtering & Resolution in Time Domain

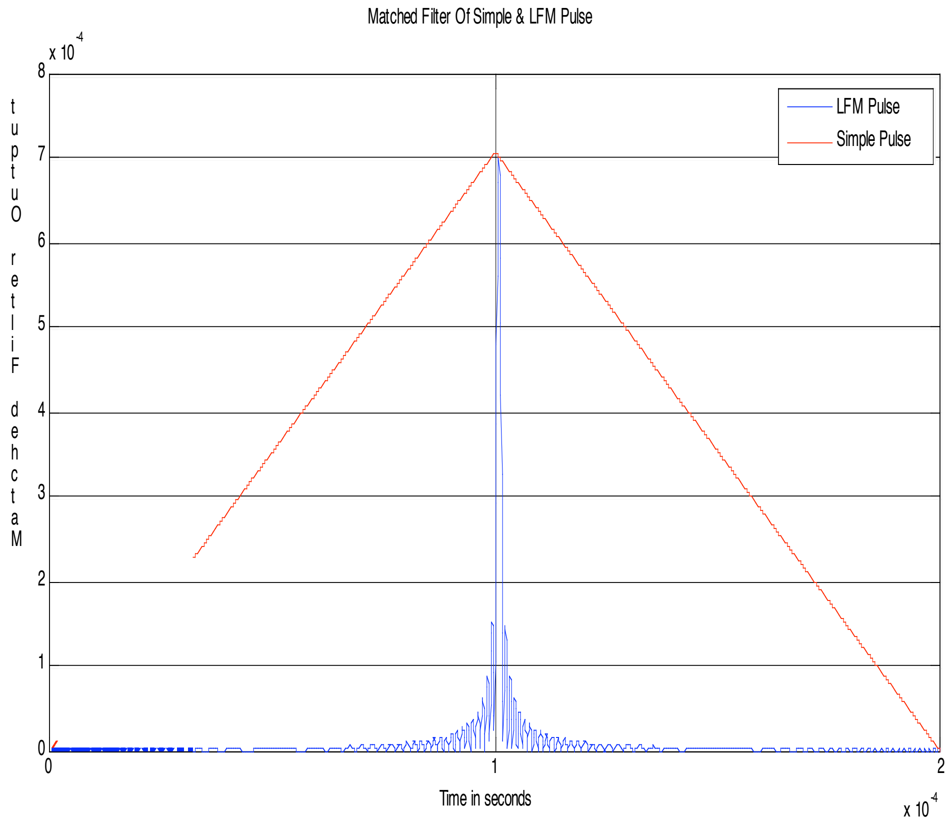


Figure 3.2 Matched filter outputs of simple pulse and LFM pulse

Observations:-
- The approximate bandwidth of simple pulse is 5 KHz.
- A simple pulse & LFM pulse of $k*\tau_p*b$ =10*100e-6*1e6 samples were generated.
- Same peak magnitudes were observed at matched filter output of both simple and LFM pulse.
- The peak-to-first-null width of mainlobe response of simple pulse is 1e-4 seconds. The peak-to-first-null width of mainlobe response of LFM pulse is 0.12e-4 seconds.

Conclusions:-
As same peaks were observed for both the simple and LFM pulse therefore the two waveforms have same energy.

3.1.3 LFM Sidelobe Supression Using Frequency Domain Techniques

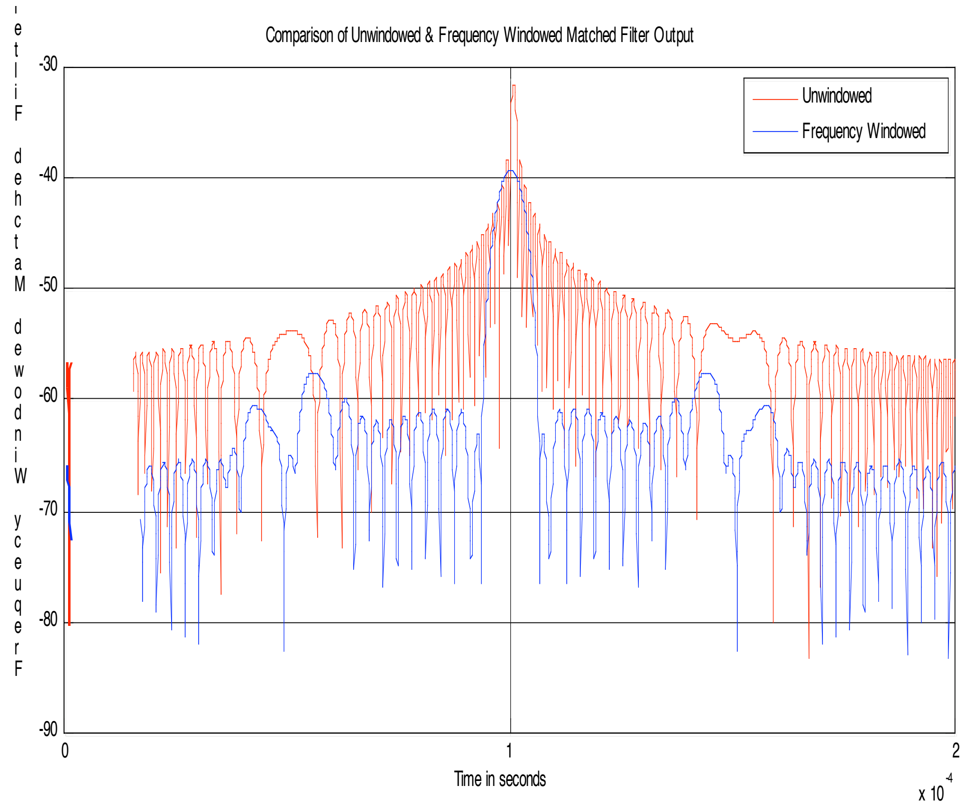


Figure 3.3 Frequency Domain Windowing Result

Observations:-

- The loss in processing gain (LPG) is observed to be 7.74 dB.
- The PSLR without windowing is 13 dB & with frequency windowing it is 36.30 dB.
- The main lobe width before windowing is 0.02e-4 and after frequency windowing it is 0.1321e-4.

3.1.4 LFM Sidelobe Supression Using Time Domain Techniques

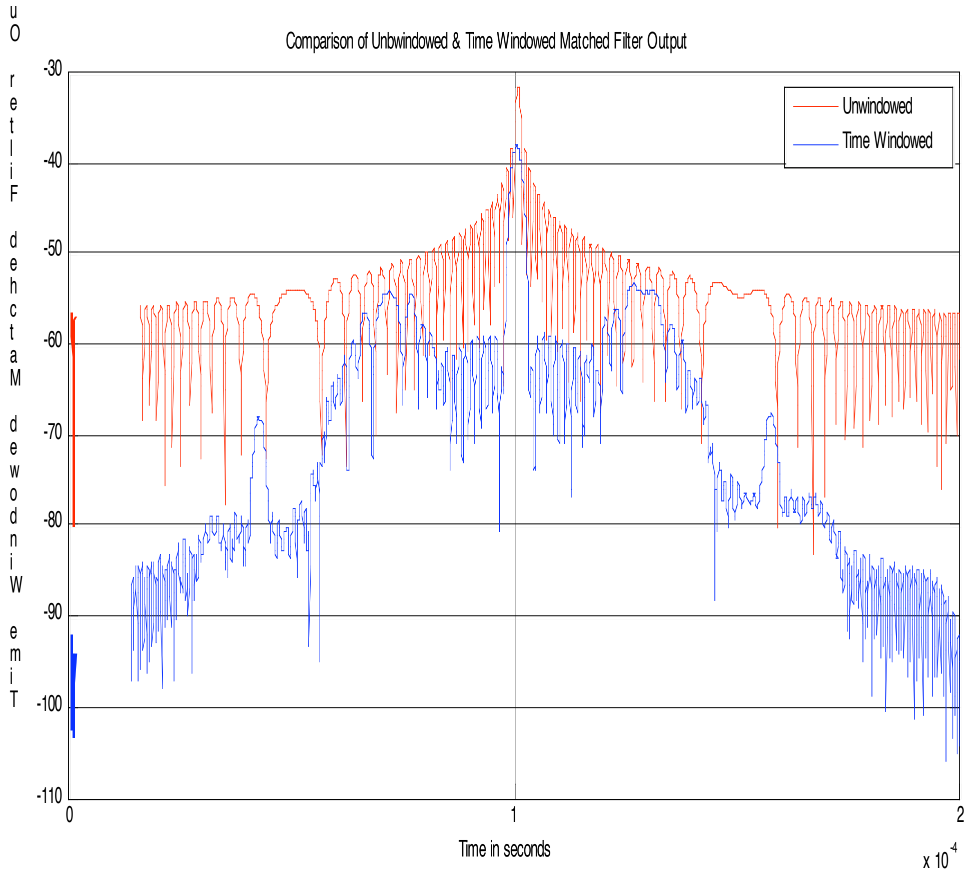


Figure 3.4 Time Domain Windowing Result

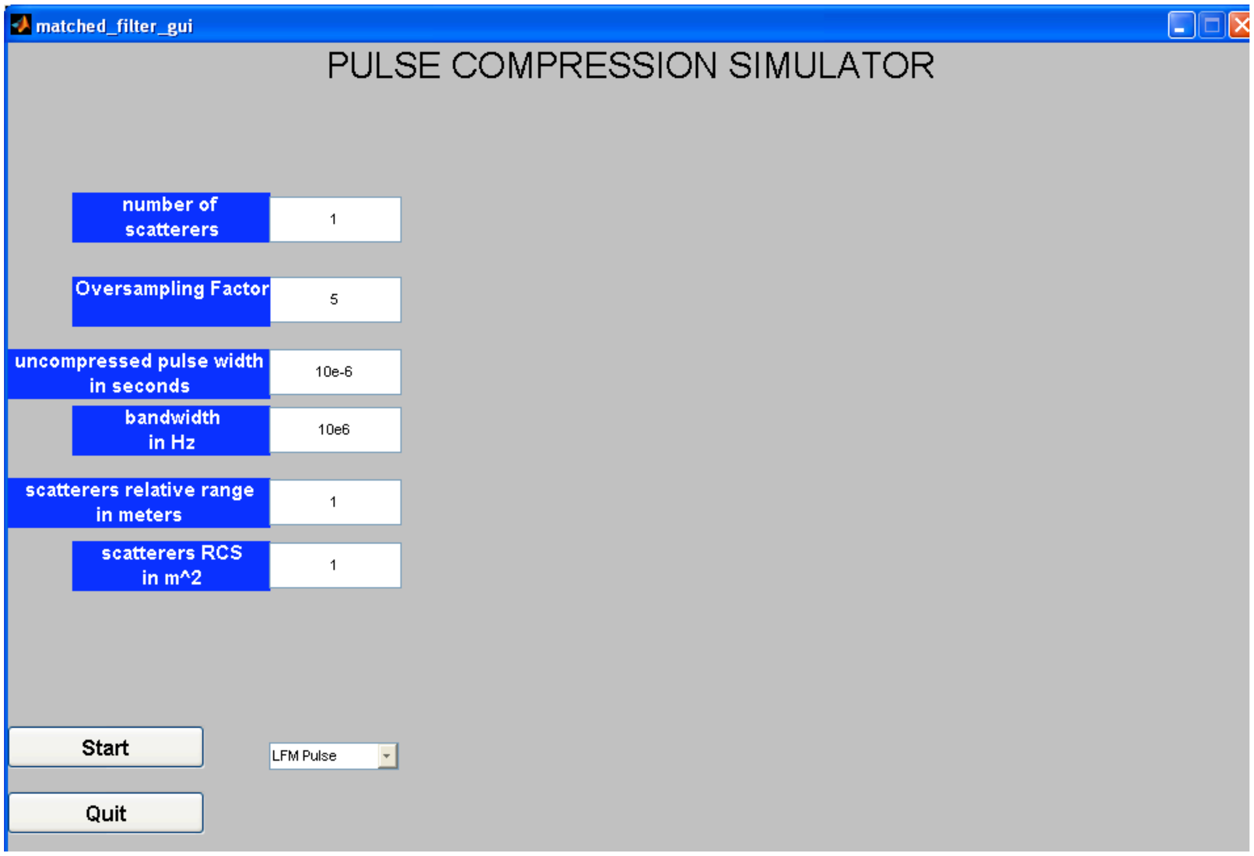
Observations:-

- The loss in processing gain (LPG) is observed to be 6.5 dB.
- The PSLR without windowing is 14 dB & with time windowing it is 32 dB.
- The main lobe width before windowing is 0.02e-4 and after time windowing it is 0.05919e-4.

3.2 MATLAB Simulation of Pulse Compression

This section represents a MATLAB Simulation to study Pulse Compression using LFM Waveforms and Time and Frequency Domain Windowing.

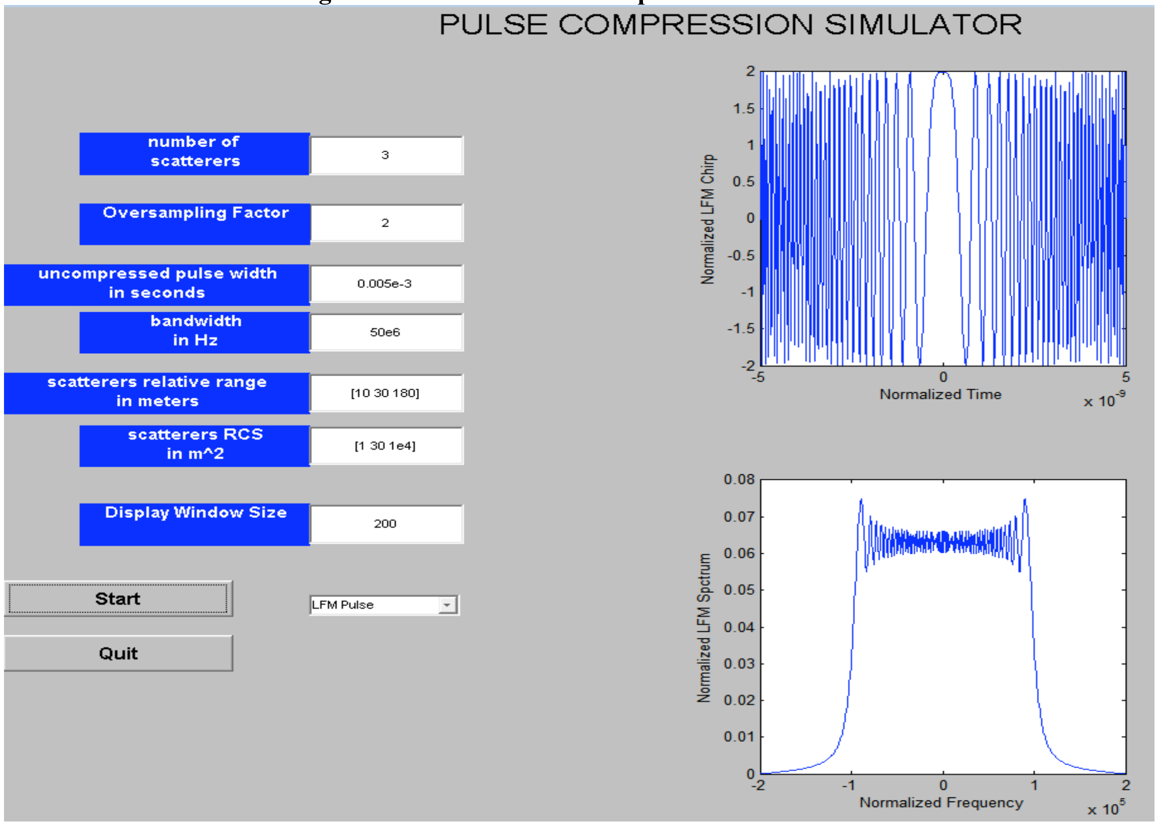
In the project software DVD, this simulation is placed in the folder named 'Section 3.2 Pulse Compression GUI'. To run it, please run 'matched_filter_gui.m' found in the folder.


matched_filter_gui

PULSE COMPRESSION SIMULATOR

number of scatterers	1
Oversampling Factor	5
uncompressed pulse width in seconds	10e-6
bandwidth in Hz	10e6
scatterers relative range in meters	1
scatterers RCS in m^2	1

Figure 3.5 GUI for Pulse Compression Simulation


PULSE COMPRESSION SIMULATOR

number of scatterers	3
Oversampling Factor	2
uncompressed pulse width in seconds	0.005e-3
bandwidth in Hz	50e6
scatterers relative range in meters	[10 30 180]
scatterers RCS in m^2	[1 30 1e4]
Display Window Size	200

Figure 3.6 Transmitted LFM Pulse (Time and Frequency Domain)

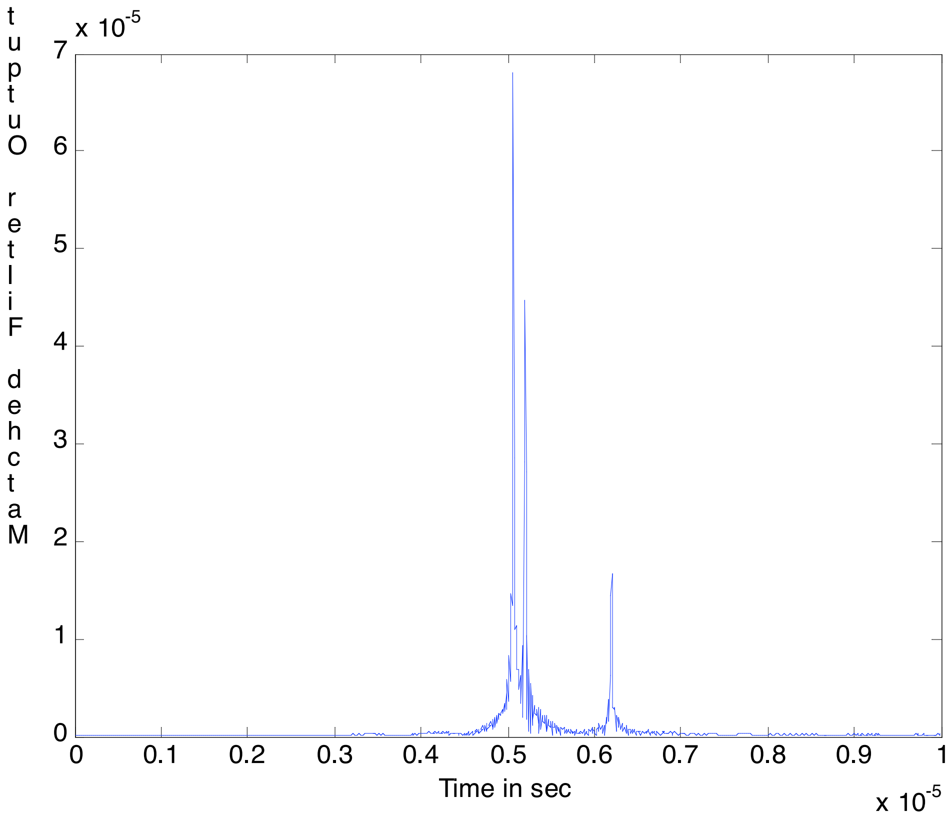


Figure 3.7 Matched Filter Output for 3 Scatterers (Time-Mapped)

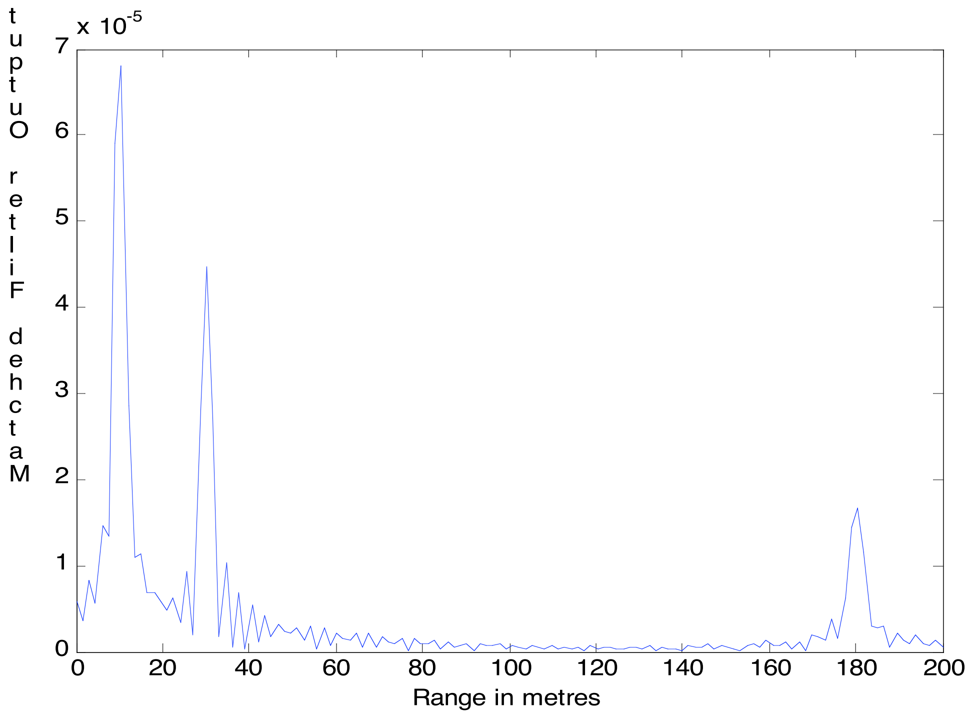


Figure 3.8 Matched Filter Output for 3 Scatterers (Range-Mapped)

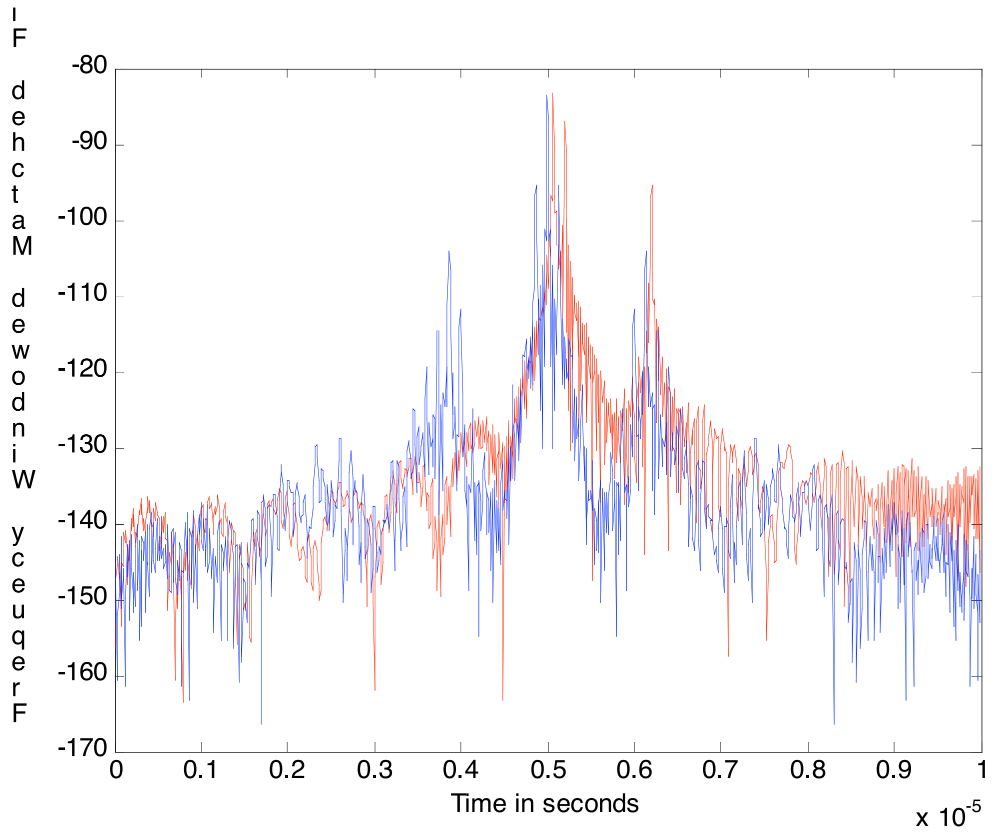


Figure 3.9 Matched Filter Output for 3 Scatterers After Frequency Domain Windowing

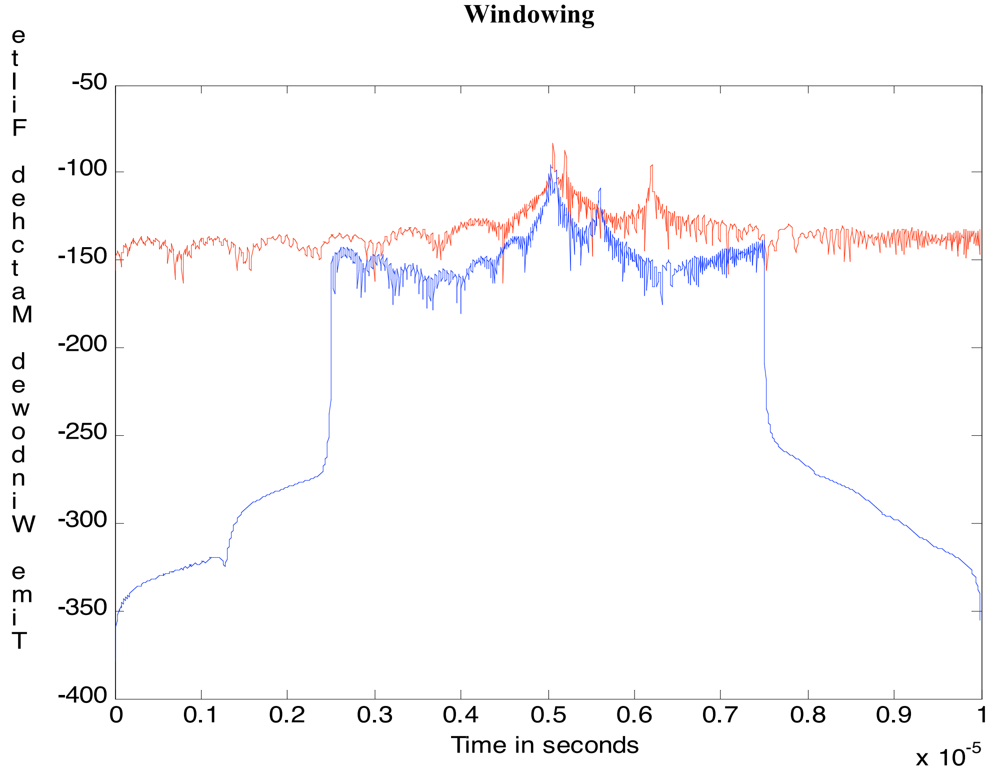


Figure 3.10 Matched Filter Output for 3 Scatteres After Time Domain Windowing

3.2.1 MATLAB Functions Involved

LFM.m:	Generates an LFM Waveform according to input parameters.
Matched_filter_GUI.m:	Driver function for the GUI.
Matchedfilter.m:	Applies correlation processing to the target returns.
Windowedoutput.m:	Applies window on the receiver filter's outputs.

Please see Appendix A for MATLAB Codes.

3.3 Simulation of a Pulse Doppler Radar

A pulse Doppler radar is the one which is capable of detecting a moving target and measure its speed and direction. This chapter presents a simulation of pulse Doppler radar designed using MATLAB and Simulink.

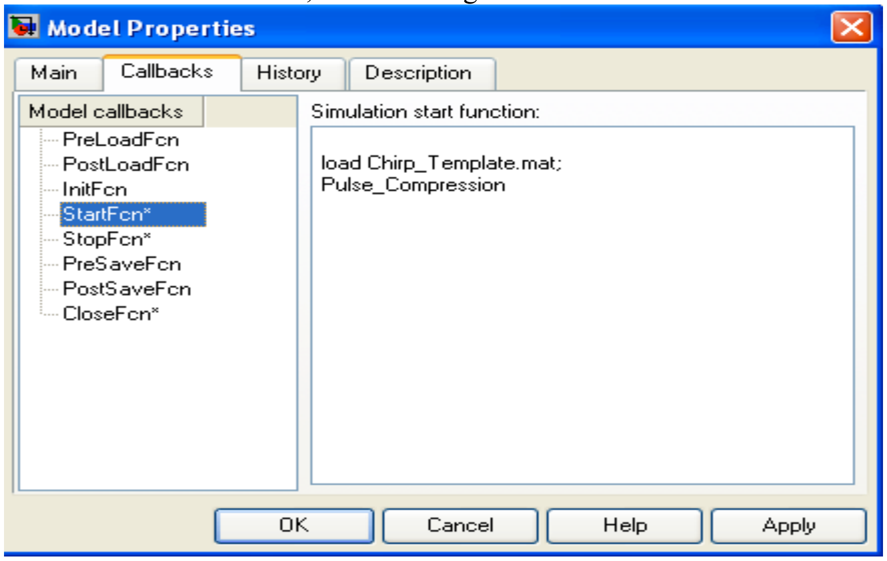
In the project software DVD, this simulation is placed in the folder named 'Section 3.3 Pulse Doppler Radar Simulator'. To run it, please open 'LFM_Wnd_FFT_Complex.mdl' found in the folder. Load the file 'Chirp_Template.mat' into the workspace, run 'Pulse_Compression.m' and then start the simulation from the Simulink model.

3.3.1 Description of the System

This system contains:

i. Model of the transmitter
ii. Model of the target
iii. Model of signal processing end

- Before the simulation is started, the following function is executed:



Simulation start function:

```
load Chirp_Template.mat;
Pulse_Compression
```

Figure 3.11 Function to be executed before simulation

- The file Chirp_Template.mat stores a replica of the transmitted pulse.

- The function Pulse_Compression.m applies window on the Chirp_Template and calculates the receiver filter response which is matched to the transmitted waveform.
- The code listing for Pulse_Compression.m is given in Appendix A.

3.3.2 Overall System

- The 'Upchirp System' block contains the target model, the transmitted waveform and the digital matched filter for the receiver.
- The 'Find Local Maxima' block finds the number of peaks (targets) in the output.
- The displays 'Range', 'Frequency', and 'Velocity' display the radial distance, Doppler shift and the speed of the target respectively.

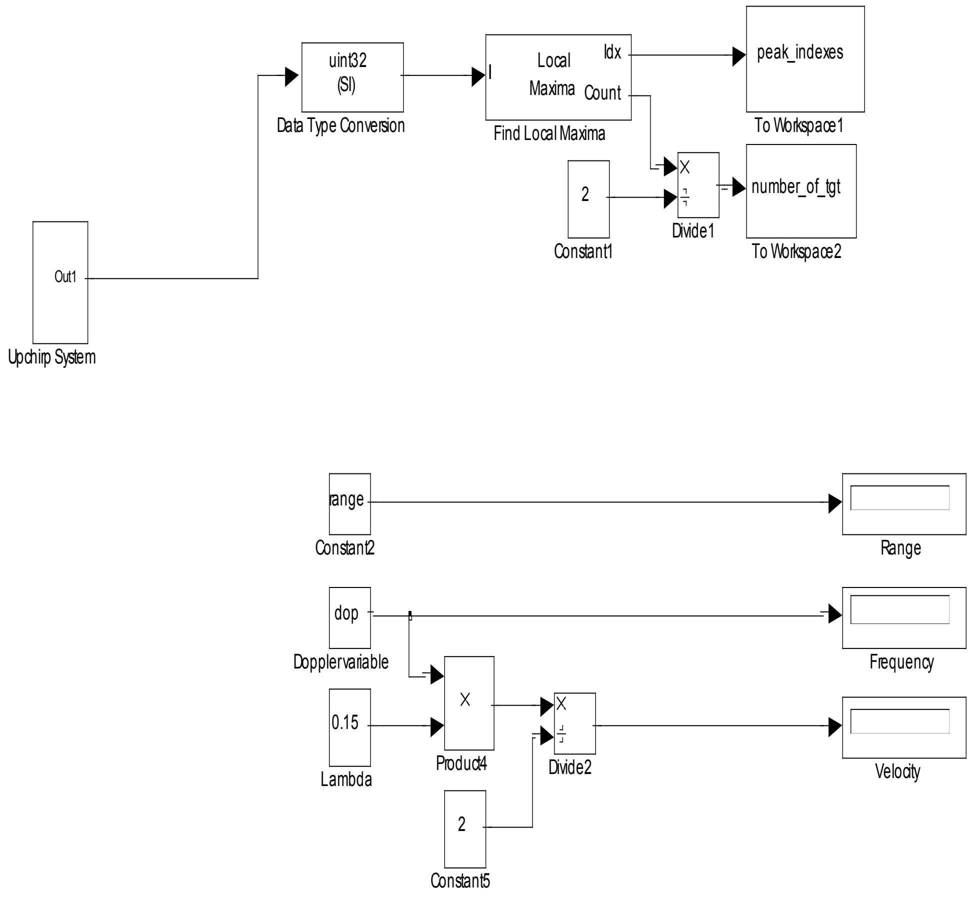


Figure 3.12 The overall system

3.3.3 Pulse Transmitter Model

- The 'Pulse Generator' block generates pulses of a finite duration and amplitude. It provides a pulse train at its output.
- The 'Upchirp' block gives a linearly swept sine wave at its output with a certain bandwidth, target time and sweep time.

- The variable ‘Chirp_Template’ saves a copy of the transmitted signal in the workspace to be later used in the receiver filter.

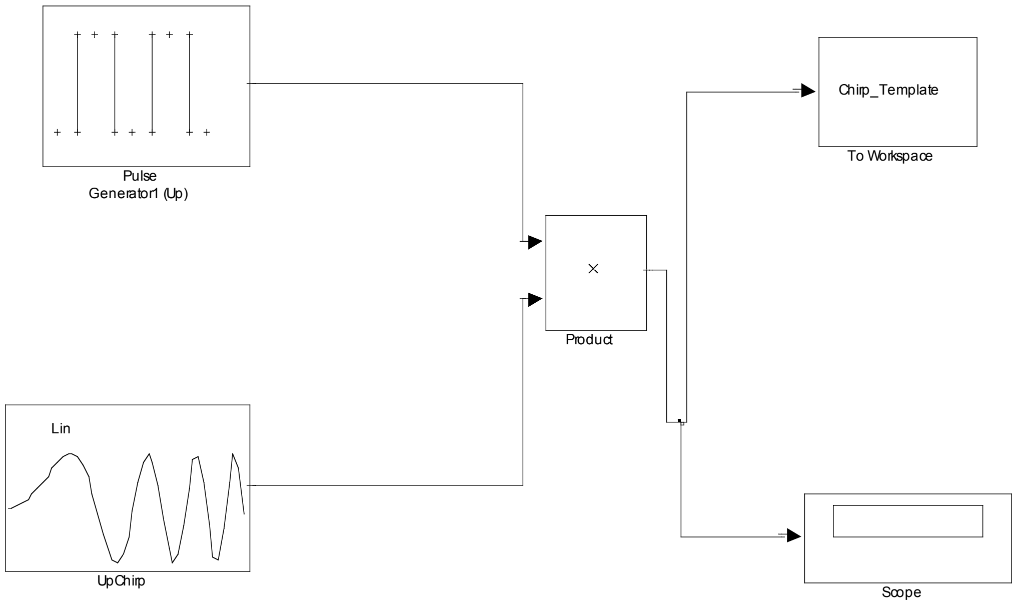


Figure 3.13 Pulse Transmitter of the radar system

- Figure below shows the time domain representation of the transmitted LFM pulse train.

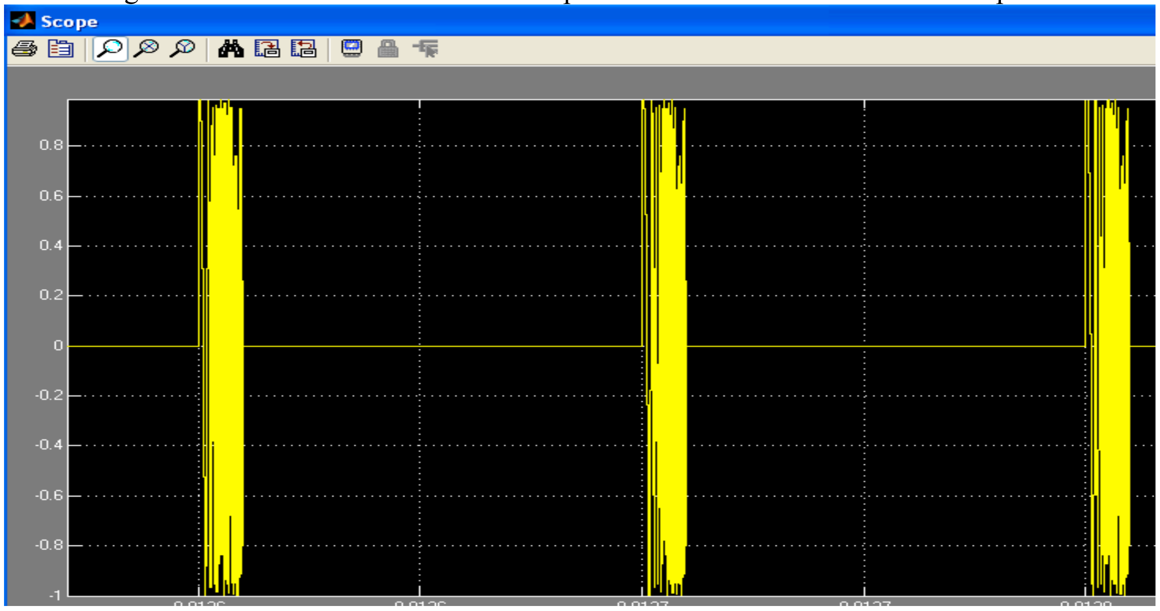


Figure 3.14 The transmitted signal

3.3.4 Target Model

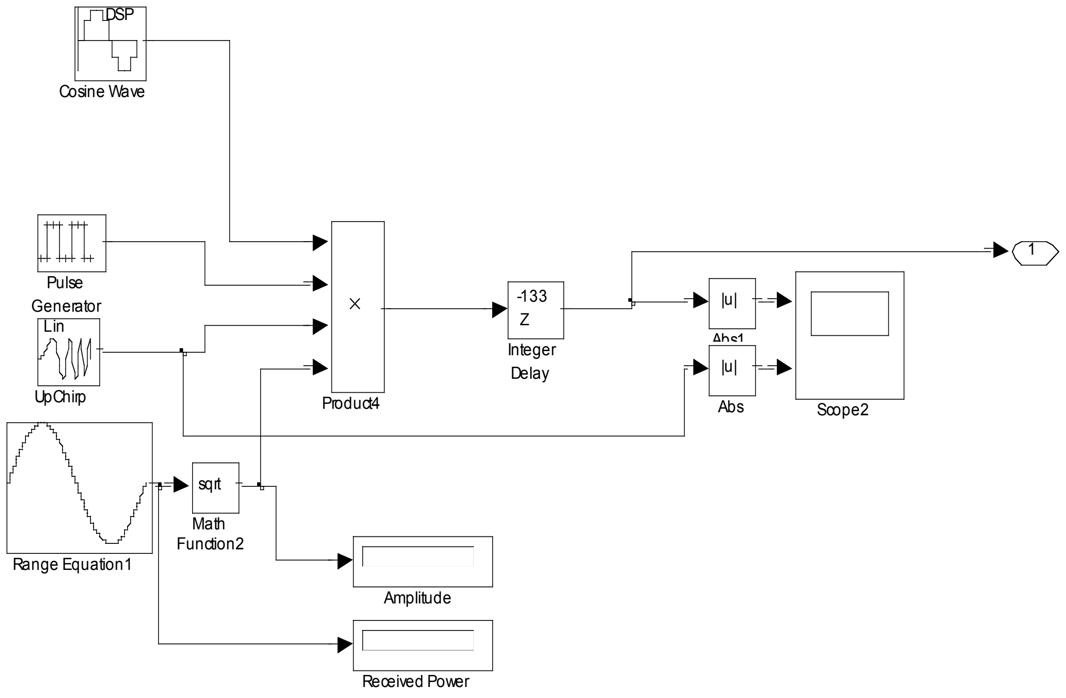
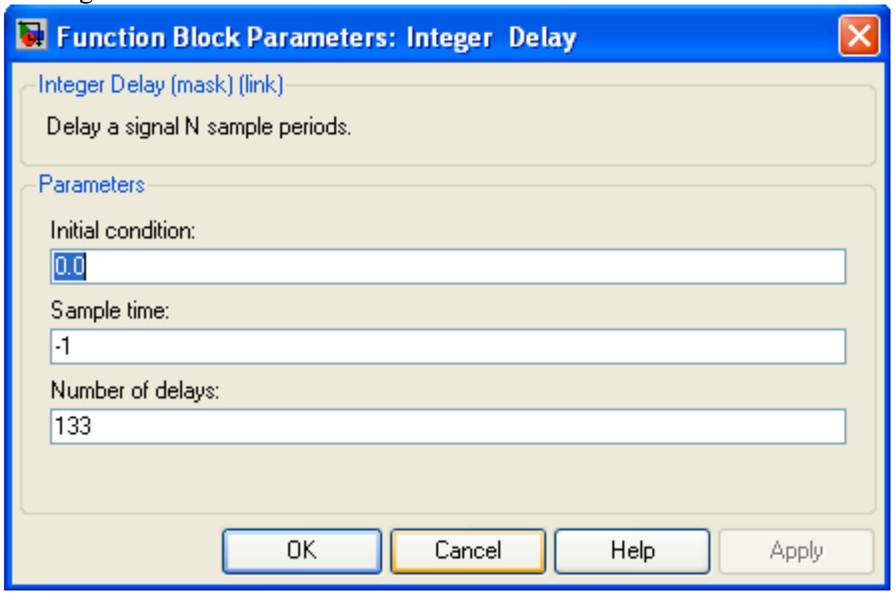


Figure 3.15 Target model

- The 'Cosine Wave block' provides a complex cosine wave which is used to model the Doppler shift caused by a moving target.
- The 'Integer Delay' block is used to model the delay in receiving the return from a specific range.



Function Block Parameters: Integer Delay

Integer Delay (mask) (link)

Delay a signal N sample periods.

Parameters

Initial condition: 0.0

Sample time: -1

Number of delays: 133

Figure 3.16 Setting the delay by the target

- The ‘Radar Equation’ block is used to calculate the amplitude of the received signal.
- The figure below shows the returns from the target (changed in amplitude and frequency)

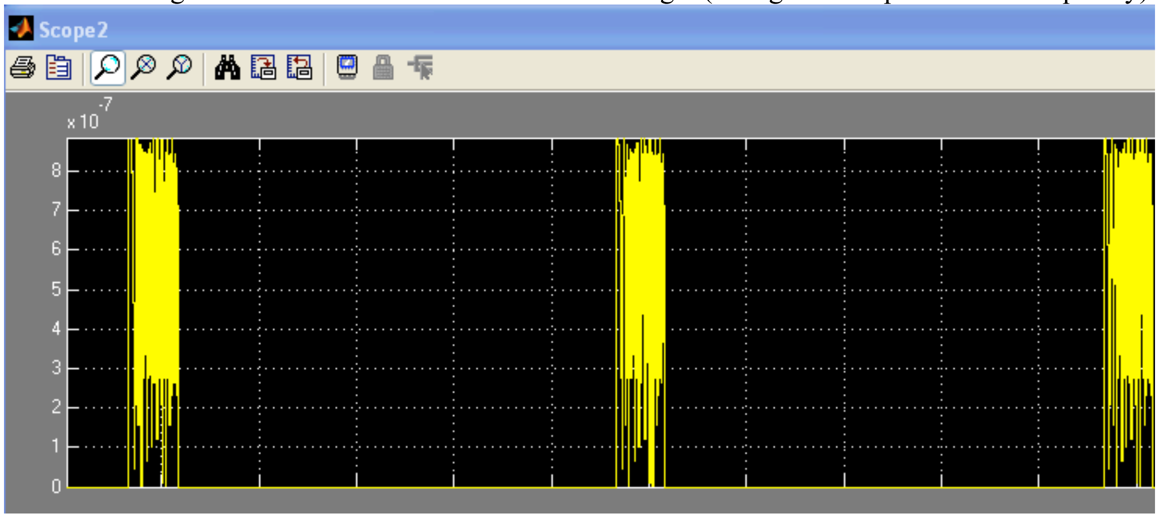
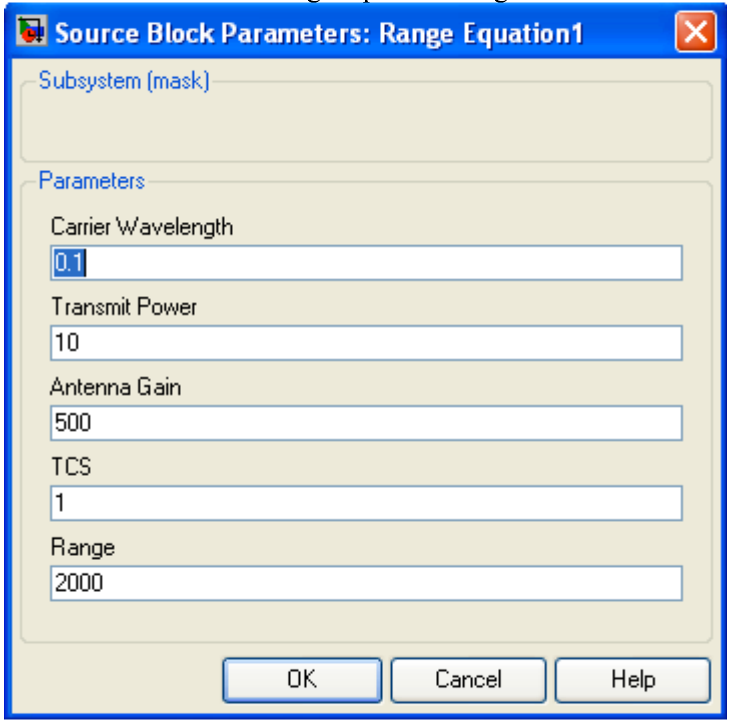


Figure 3.17 The returned waveform from the target

3.3.5 Range Equation in the Target Model

- All the variables involved in radar range equation are given to this block as input.



Source Block Parameters: Range Equation1

Subsystem (mask)

Parameters

Carrier Wavelength: 0.1

Transmit Power: 10

Antenna Gain: 500

TCS: 1

Range: 2000

Figure 3.18 Setting the variables in range equation

- This block returns the amplitude of the received signal.

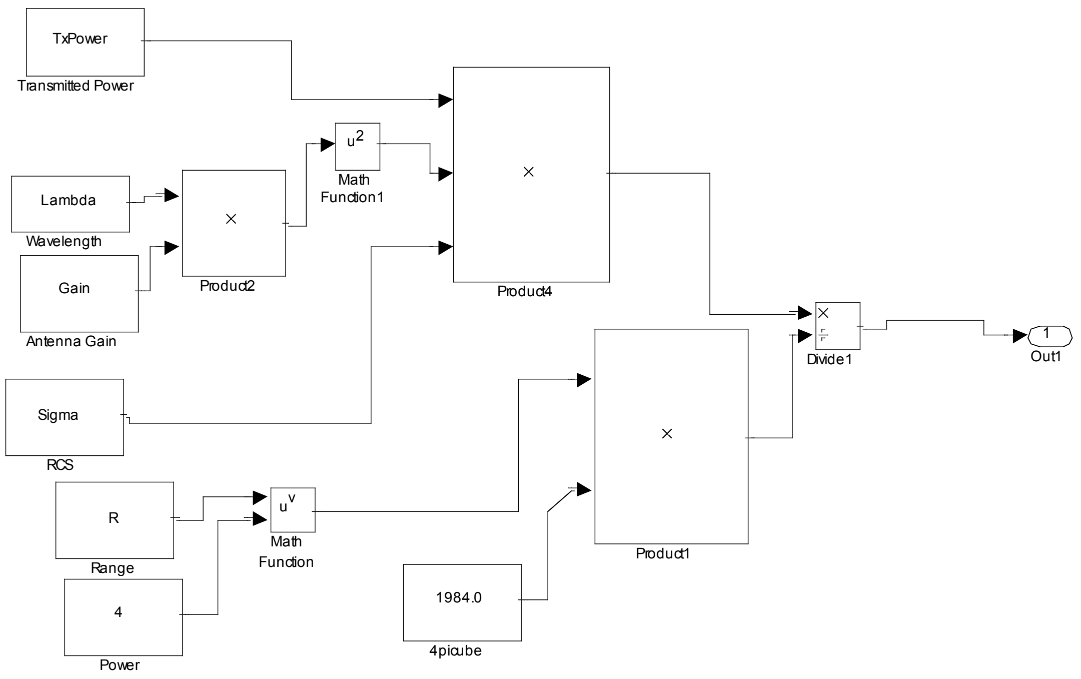


Figure 3.19 Computing the received power from range equation

3.3.6 Signal Processor Model

- This model receives the return from the target.
- The received signal is then passed from the matched filter (for pulse compression)
- Then using the buffers, slow/fast time matrix is formed.
- FFT is applied on each column of the matrix.
- Threshold is applied on the amplitude of the pulses to decide about presence of the target.
- Results are stored in the workspace.

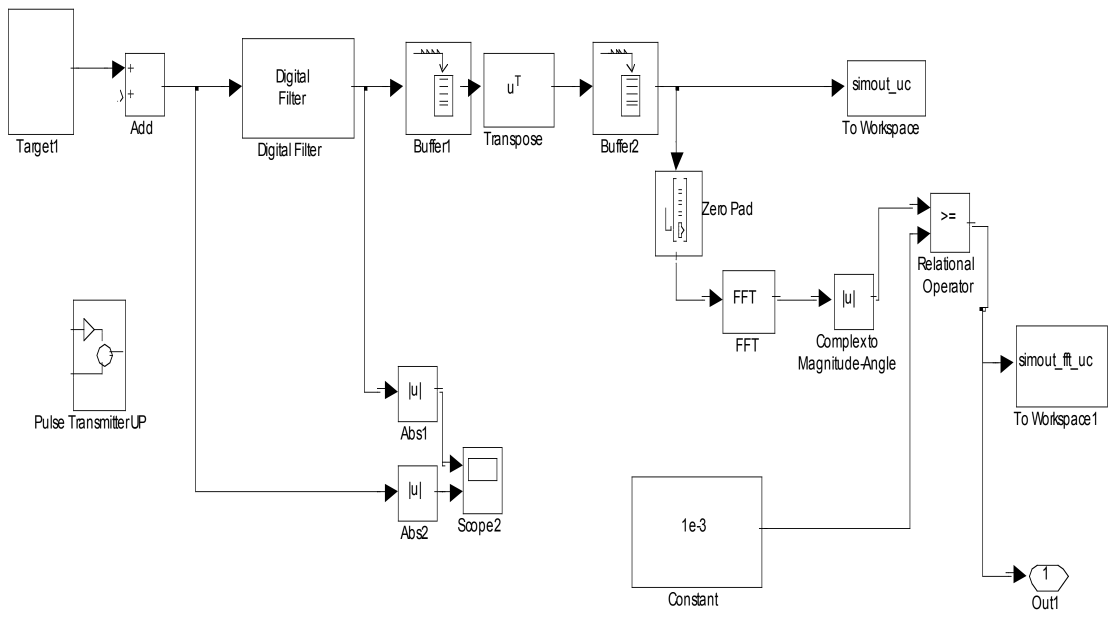


Figure 3.20 The signal processor model

- The figure below shows the input to the matched filter (lower half) and the compressed output of the matched filter (upper half)

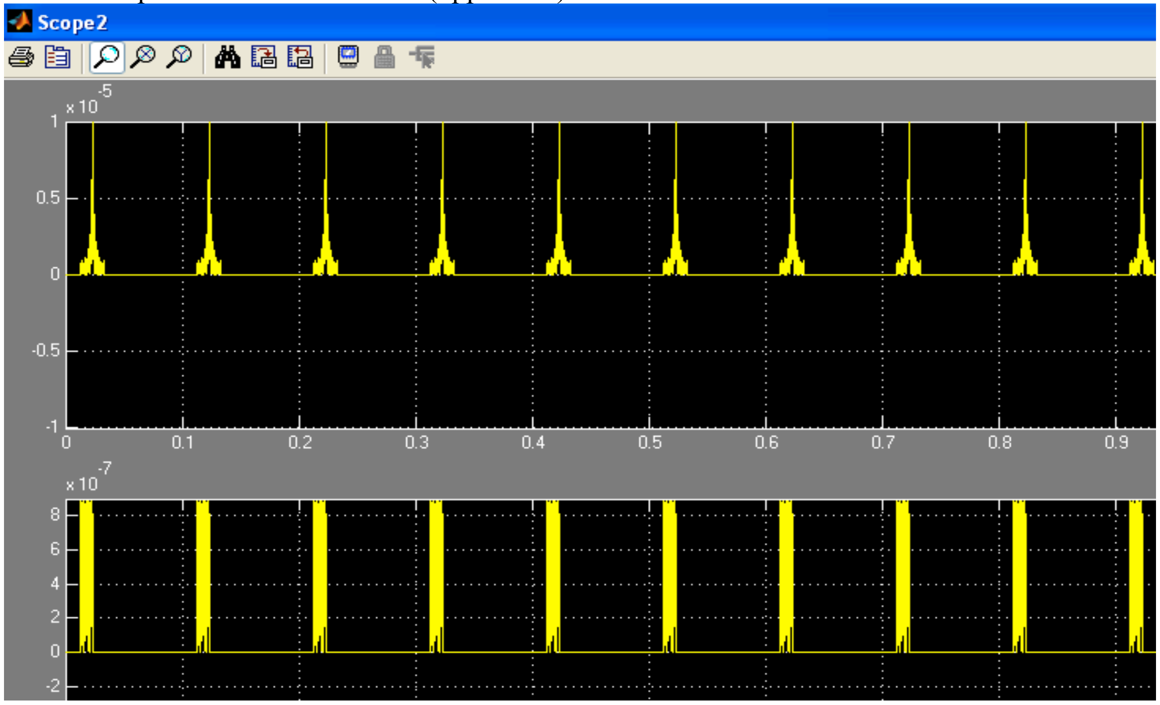


Figure 3.21 Input and Output of the receiver matched filter

- This is the magnified input and output of the matched filter for one pulse.

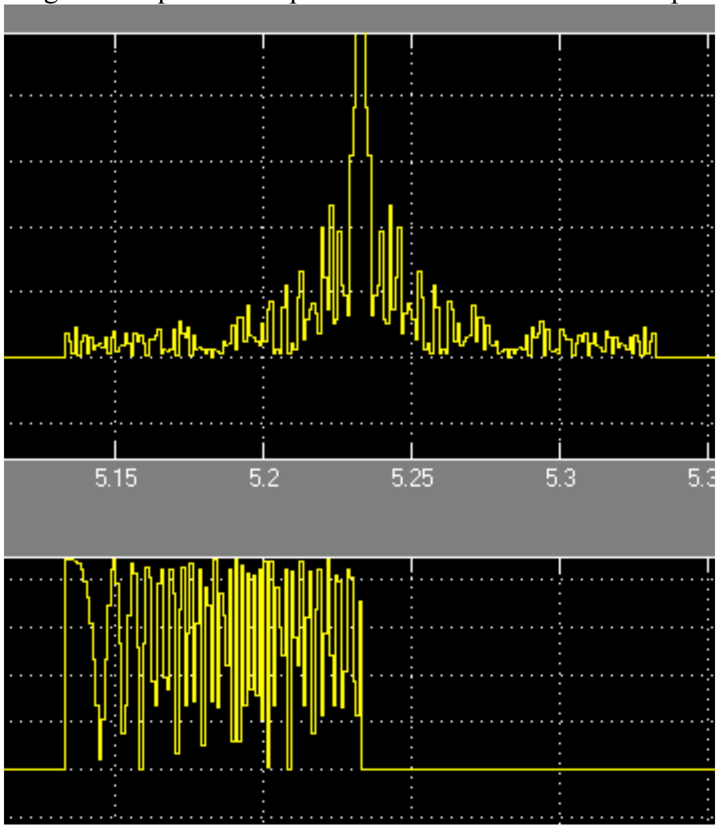
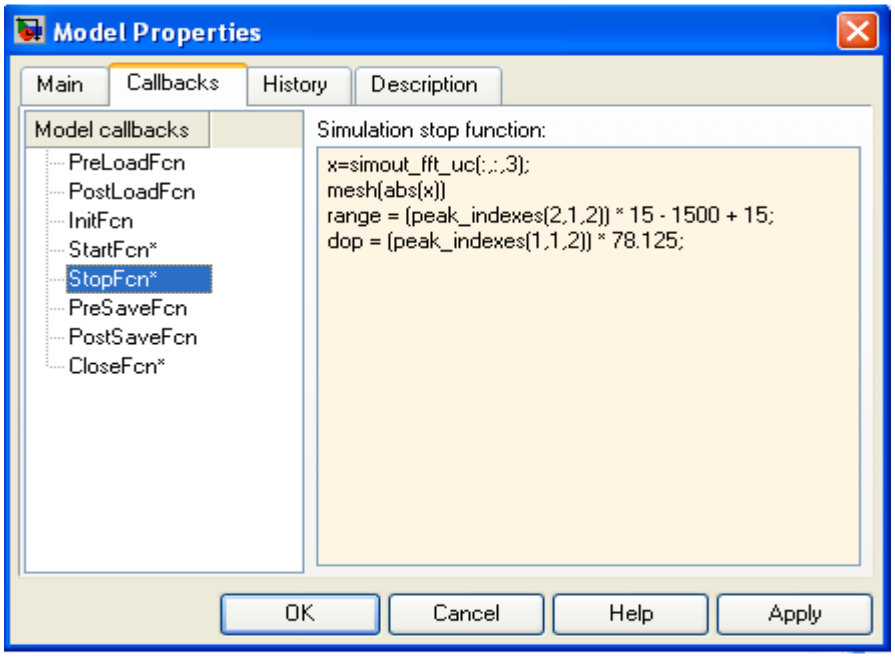


Figure 3.22 Uncompressed return (bottom) and compressed return (top)

- At the end of the simulation, the following function is executed to find the target parameters.



Simulation stop function:

```
x=simout_fft_uc(:,:,3);
mesh(abs(x))
range = (peak_indexes(2,1,2)) * 15 - 1500 + 15;
dop = (peak_indexes(1,1,2)) * 78.125;
```

Figure 3.23 The function to be executed at the end of simulation

- The plot below gives the range and Doppler shift of the target.

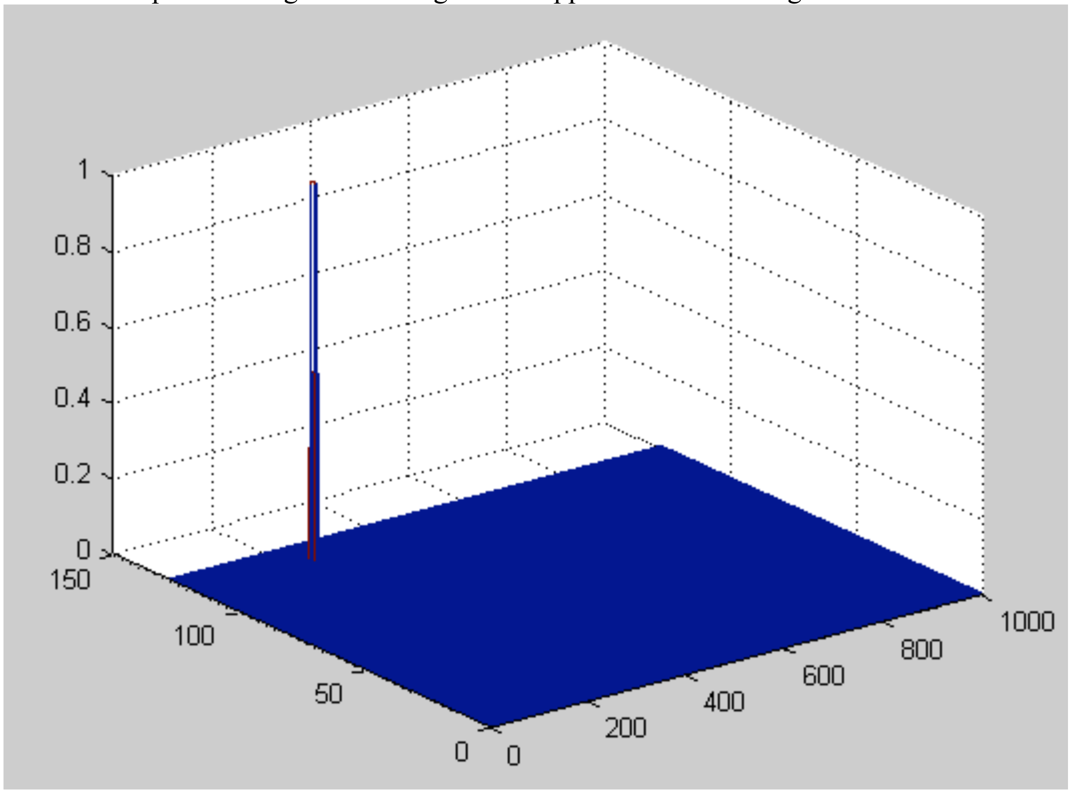


Figure 3.24 Graph displaying the range and Doppler of the target

3.4 System Level Design Parameters for the SAR

In the project software DVD, the parameter calculation application is placed in the folder named 'Section 3.4 SAR Parameter Calculator'. To run it, please run 'SARPARAM.m' found in the folder. A Microsoft Word file containing the system level parameters is also placed in this folder.

3.4.1 Assumptions

- Operating Frequency = 2.7GHz
- Wavelength = 0.11m
- Height of Aircraft = 2km = 2000m Above Ground Level
- Velocity of Aircraft = 65 m/s = 127 Knots (for a helicopter)
- k = Oversampling rate = 2
- Side looking synthetic aperture radar Squint Angle = 90 degrees
- Pencil beamwidth for simplicity i.e. Beamwidth Azimuth = Beamwidth Elevation
- Range Resolution without LFM is 1.5km = 1500m

3.4.2 Gain, Aperture Size & Cross Range Minimum Calculations

- Aperture Size in Azimuth = Aperture Size in Elevation = 0.5m (for a horn antenna)
- Beamwidth in Azimuth = Beamwidth in Elevation = 0.22222rads

=> G=160.007 Watts = 22 dB which is *not acceptable*

25dB <= G <= 30dB = Average Gain of an airborne radar

Thus, changing the aperture size

- Aperture Size in Azimuth = Aperture Size in Elevation = 0.696m (for horn antenna)
- Beamwidth in Azimuth = Beamwidth in Elevation = 0.15964rads

=> G=309.9871=24.9 approx. 25 dB which is now *acceptable*

- Minimum Cross Range Resolution=0.348 m

3.4.3 Bandwidth, Range Resolution, Swath Length, Grazing Angle & Slant Range Calculations

- k = oversampling rate = 2
- height = 2 km
- Range Resolution without LFM is 1.5 km = 1500 m corresponds to *pulse width* of 1e-5 sec = 10 usec
- Bandwidth Time Product = BT = 500.

=> Bandwidth of LFM is 50 MHz which corresponds to Range Resolution with LFM = 3 m

- Ts = 1/kB = 1/(2*50e6) = 0.01usec
- No. of Fast time samples = Pulse width / Ts = 10usec / 0.01usec = 1000 samples (Greater than 1000 results in very high processing)
- Maximum Swath Length covered = 1503m = 1.503km
- Grazing Angle = 1.0882 deg = 0.0190
- Slant Range = 2257.8638 = 2.258km

3.4.4 Synthetic Aperture & Aperture Time Calculations

- Synthetic Aperture = Dsar = (Rs*Lambda) / (2*CRmin) = 360.3473 m

=> This is maximum aperture size.

- Aperture Time = Dsar / Velocity = 5.5438 sec

As aperture time has to be changed therefore assuming aperture time to be 3sec => Dsar = 195m

3.4.5 PRF & Unambiguous Range Calculations

- Number of pulses = 1024 = M = slow time samples (No of pulses slow time samples is directly proportional to processing complexity)
- PRI = Ta / M = 0.0054139 sec = 5msec.
- No of samples in PRI = PRI / Ts = 5msec / 0.01 usec = 500000
- PRF = 1/PRI = 184.7108 Hz which is not alright considering the fact that the PRF constraint is

(2*v*BWaz)/lambda<=PRF<=(2/(Ls*cos(Grazing Angle)) for a sidelooking radar

186.7788Hz <=PRF<=215049.7 Hz or 0.187 KHz <=PRF<=0.21MHz

Thus, we have to change either aperture time or slow time samples but we can't change slow time samples so we change aperture time.

- Assuming aperture time to be 3sec => Dsar = 195m
- PRI = Ta / M = 0.0029297 sec = 3msec.
- No of samples in PRI = PRI / Ts = 3msec / 0.01 usec = 300000
- PRF = 1/PRI = 341.3333 Hz which is alright as

  186.7788Hz <=PRF<=215049.7 Hz or 0.187 KHz <=PRF<=0.21MHz

- Unambiguous Range = 439453.167 m = 439.453km

3.4.6 Resolutions

- Cross Range Resolution Real (Azimuth) = Cross Range Resolution Real (Elevation) = 360.4454m
- Cross Range Resolution Synthetic = 0.64327 metre
- Range Resolution without LFM is 1.5km = 1500m
- Range Resolution with LFM = 3m
- Doppler Resolution = 0.33333
- Angular Resolution = 0.0002849

3.4.7 Bandwidths

- Bandwidth LFM = 50MHz
- Bandwidth Simple Pulse = 0.1MHz
- Bandwidth Spatial = 1.5546Hz

3.4.8 Beamwidths

- Beamwidth Azimuth = 0.15964 rads = 9.1467deg
- Beamwidth Elevation = 0.15964rads = 9.1467deg
- Beamwidth SAR = 2.82e-4rads = 0.0162deg

3.4.9 Aperture Area & Area Coverage Rate & Range Curvature Calculations

- Aperture Area = 0.48443m^2
- Area Coverage Rate = 97695m^2/sec
- Range Curvature = 2.1051 m

3.4.10 Received Power Calculations

If Effective Temperature Te = 290 Kelvin & Thermal Temperature To = 300K than noise figure F=1.9667W

If Atmospheric Attenuation = 0.0005 dB/KM than Atmospheric Loss = 1.00052 Watts

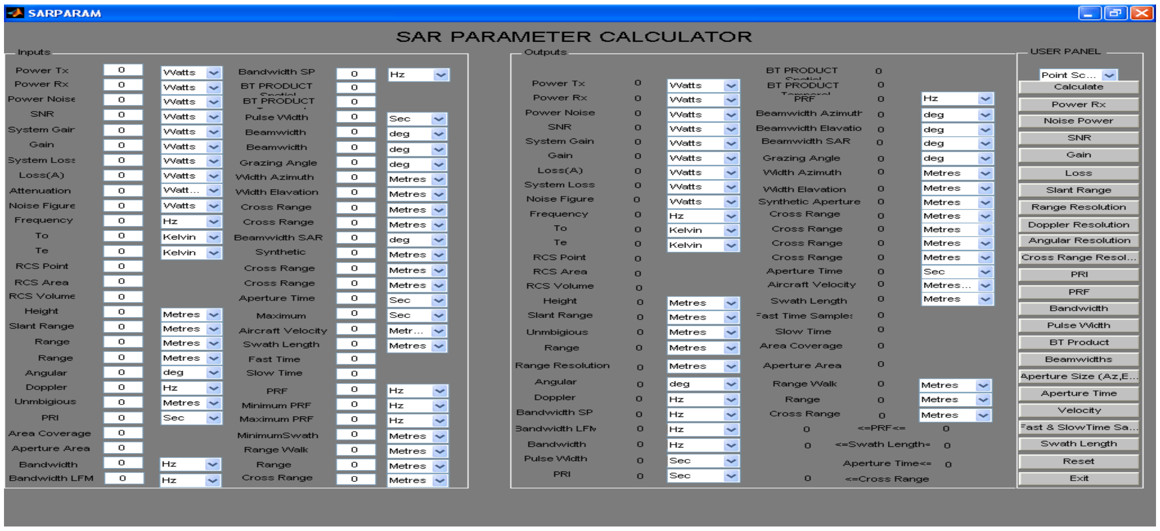
If Gs = 50W than Pnoise = kToBFGs = 2.0355e-11Watts = -106.9dB

If RCS sigma nought = 0.1 (target with small RCS) & Power Transmitted = Pt = 10W & System Loss = 25W than Power Received = Pr = 1.3488e-10Watts = -98.7dB

3.4.11 SAR Parameter Calculator

This section describes the MATLAB based Graphical User Interface (GUI) and the codes involved in the SAR parameter calculator application.

The Graphical User Interface for Parameter Calculation



Point Sc...

Inputs

Power Tx	0	Watts	Bandwidth SP	0	Hz
Power Rx	0	Watts	BT PRODUCT	0	
Power Noise	0	Watts	BT PRODUCT	0	
SNR	0	Watts	Pulse Width	0	Sec
System Gair	0	Watts	Beamwidth	0	deg
Gain	0	Watts	Beamwidth	0	deg
System Loss	0	Watts	Grazing Angle	0	deg
Loss(A)	0	Watts	Width Azimuth	0	Metres
Attenuation	0	Watt...	Width Elavation	0	Metres
Noise Figure	0	Watts	Cross Range	0	Metres
Frequency	0	Hz	Cross Range	0	Metres
To	0	Kelvin	Beamwidth SAR	0	deg
Te	0	Kelvin	Synthetic	0	Metres
RCS Point	0		Cross Range	0	Metres
RCS Area	0		Cross Range	0	Metres
RCS Volume	0		Aperture Time	0	Sec
Height	0	Metres	Maximum	0	Sec
Slant Range	0	Metres	Aircraft Velocity	0	Metr...
Range	0	Metres	Swath Length	0	Metres
Range	0	Metres	Fast Time	0	
Angular	0	deg	Slow Time	0	
Doppler	0	Hz	PRF	0	Hz
Unmbigious	0	Metres	Minimum PRF	0	Hz
PRI	0	Sec	Maximum PRF	0	Hz
Area Coverage	0		MinimumSwath	0	Metres
Aperture Area	0		Range Walk	0	Metres
Bandwidth	0	Hz	Range	0	Metres
Bandwidth LFM	0	Hz	Cross Range	0	Metres

Outputs

			BT PRODUCT	0	
Power Tx	0	Watts	BT PRODUCT	0	
Power Rx	0	Watts	PRF	0	Hz
Power Noise	0	Watts	Beamwidth Azimuth	0	deg
SNR	0	Watts	Beamwidth Elavatio	0	deg
System Gain	0	Watts	Beamwidth SAR	0	deg
Gain	0	Watts	Grazing Angle	0	deg
Loss(A)	0	Watts	Width Azimuth	0	Metres
System Loss	0	Watts	Width Elavation	0	Metres
Noise Figure	0	Watts	Synthetic Aperture	0	Metres
Frequency	0	Hz	Cross Range	0	Metres
To	0	Kelvin	Cross Range	0	Metres
Te	0	Kelvin	Cross Range	0	Metres
RCS Point	0		Cross Range	0	Metres
RCS Area	0		Aperture Time	0	Sec
RCS Volume	0		Aircraft Velocity	0	Metres...
Height	0	Metres	Swath Length	0	Metres
Slant Range	0	Metres	Fast Time Samples	0	
Unmbigious	0	Metres	Slow Time	0	
Range	0	Metres	Area Coverage	0	
Range Resolution	0	Metres	Aperture Area	0	
Angular	0	deg	Range Walk	0	Metres
Doppler	0	Hz	Range	0	Metres
Bandwidth SP	0	Hz	Cross Range	0	Metres
Bandwidth LFM	0	Hz	0 <=PRF<=	0	
Bandwidth	0	Hz	0 <=Swath Length=	0	
Pulse Width	0	Sec	Aperture Time<=	0	
PRI	0	Sec	0 <=Cross Range		

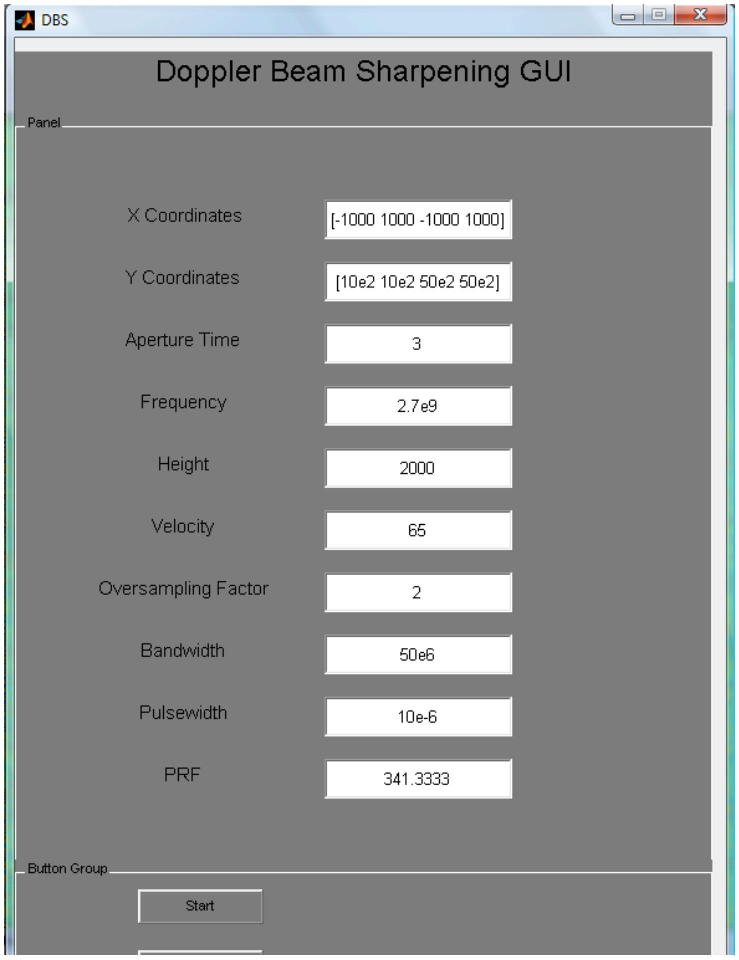
Figure 3.25 GUI for SAR Parameter Calculator

3.5 DBS Simulation using MATLAB

This section describes a MATLAB simulation of image formation using DBS algorithm for four targets.

In the project software DVD, this simulation is placed in the folder named 'Section 3.5 SARDBS'. To run it, please run 'DBS.m' found in the folder.

3.5.1 The Graphical User Interface


DBS

Doppler Beam Sharpening GUI

Panel

X Coordinates	[-1000 1000 -1000 1000]
Y Coordinates	[10e2 10e2 50e2 50e2]
Aperture Time	3
Frequency	2.7e9
Height	2000
Velocity	65
Oversampling Factor	2
Bandwidth	50e6
Pulsewidth	10e-6
PRF	341.3333

Button Group

Figure 3.26 GUI for DBS Simulation

3.5.2 The Generated Dataset

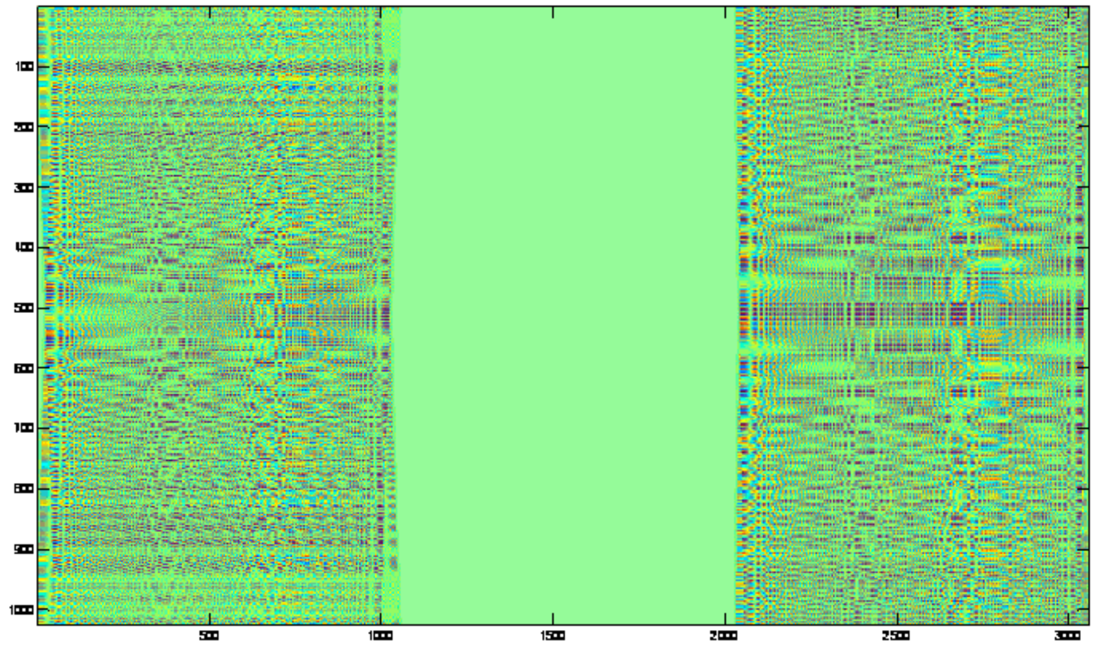


Figure 3.27 Generated Dataset for 4 Scatterers

3.5.3 After Fast Time Pulse Compression

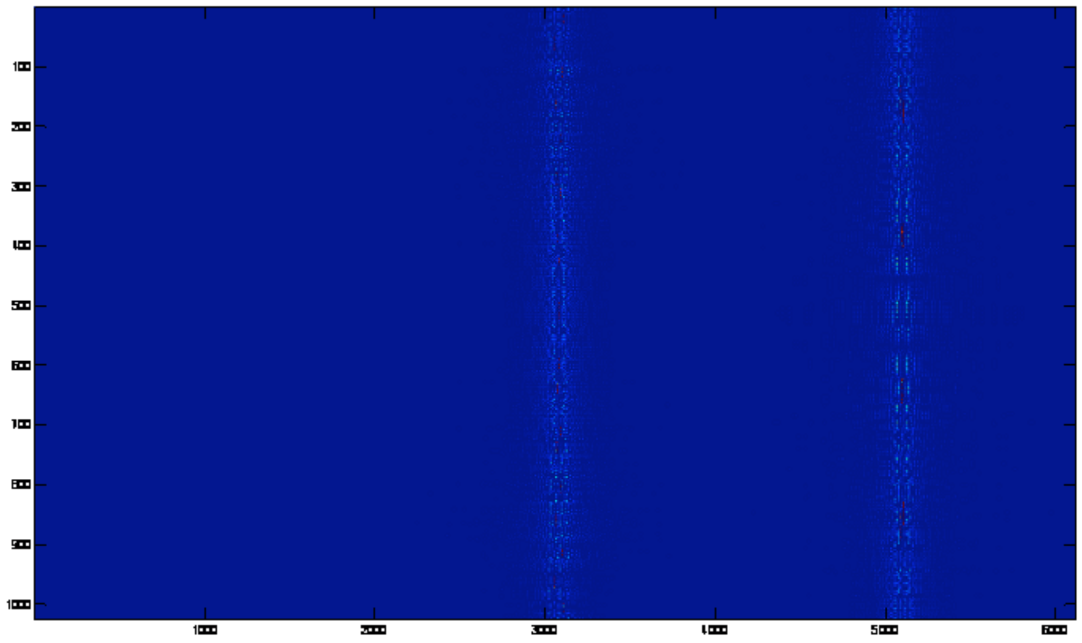


Figure 3.28 After Applying Pulse Compression to the Dataset

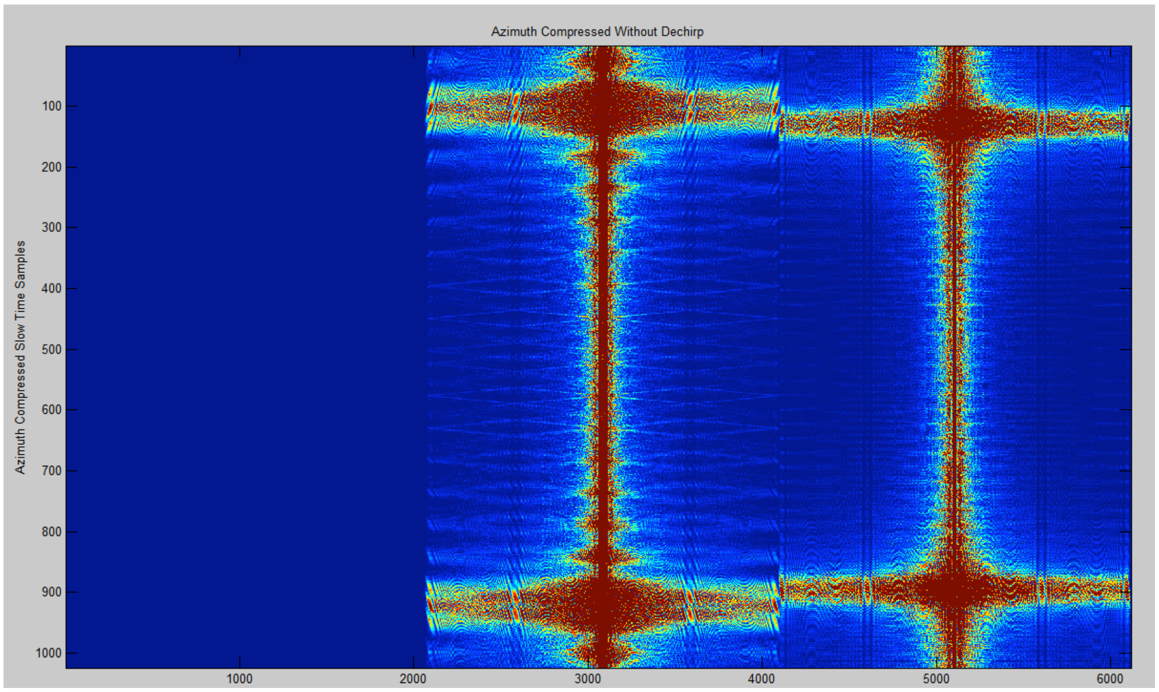


Figure 3.29 Before Dechirping, Azimuth compressed

3.5.4 After Azimuth Compression (The Image)

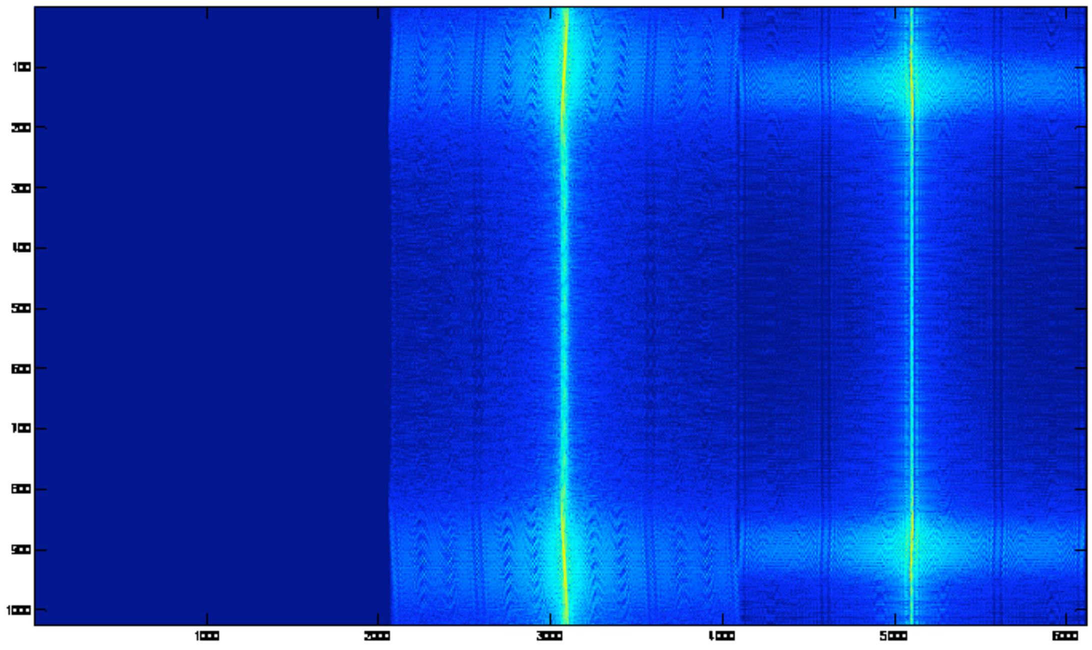


Figure 3.30 The DBS Image after Dechirping, Azimuth Compressed

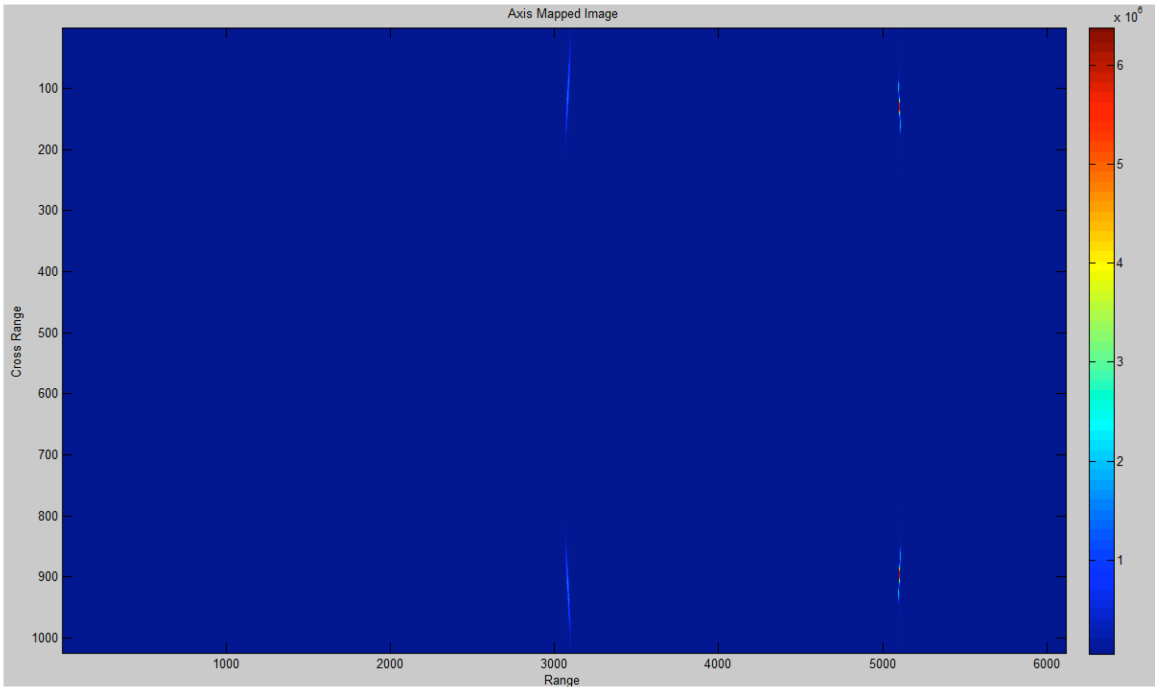


Figure 3.31 The Image after Axis mapped

3.5.5 MATLAB Functions Involved

SAR_DataSet.m:	Generates a data set containing target returns according to input parameters.
FastTime.m:	Applies Fast Time Pulse Compression on Data Set.
SlowTime.m:	Applies Slow Time FFT on Data Set.
DeChirped.m:	Applies DeChirping.
AxisMapping.m:	Applies axis mapping.

Please see Appendix A for MATLAB Codes.

Chapter

4

Conclusions and Recommendations

This chapter contains the future recommendations for this project and highlights any discrepancies left in the project. Possibilities of further design and improvement are also presented.

4.1 Hardware Implementation

This chapter discusses the possibilities and considerations for the hardware implementation of SAR Signal Processor.

4.1.1 DSP vs. FPGA

In this section, a comparison between DSP and FPGA is given. Depending on the comparison, any of the platforms can be adopted for hardware implementation of a SAR signal processor.

In many complex systems, the initial choice of a processing engine and its associated design methodology can have a profound impact on the system reliability. This choice will also have a large affect on the effort required to maintain the system throughout its life cycle.

One fundamental architecture issue is the type of DSP platform. Digital signal processing functions are commonly implemented on two types of programmable platform: ***digital signal processors (DSP)*** and ***field programmable gate arrays (FPGA)***. DSPs are a specialized form of microprocessor, while FPGAs are a form of highly configurable hardware. In the past, usage of DSPs has been nearly ubiquitous, but with the needs of many applications outstripping the processing capabilities (MIPS) of DSPs, the use of FPGAs has become prevalent. Currently, the primary reason most engineers choose to use an FPGA over a DSP is driven by the MIPS requirements of an application. Thus, when comparing DSPs and FPGAs, the common focus is on MIPS comparison; but while this is certainly important, it is not the only advantage of FPGAs. Equally important, and often overlooked, is the inherent advantage that FPGAs have for product reliability and maintainability.

Nearly all engineering project managers can readily quote the date of the next product software update or release. Most technology companies have a long, internal list of software bugs or problem reports, and software release will contain the associated patch or fix. It has generally come to be expected that all software (DSP code is considered a type of software) will contain some bugs and that the best one can do is to minimize them.

By comparison, FPGA designs tend to be much less frequently updated, and it is generally a rather unusual event for a manufacturer to issue a field upgrade for an FPGA configuration file. ***Reliability and maintainability is much better in FPGA implementations compared to those using DSPs***, due to the fact that the engineering development process for DSPs and FPGAs are dramatically different. There is a fundamental challenge in developing complex software for any type of processor. In essence, the DSP is basically a specialized processing engine being constantly reconfigured for many different tasks, some digital signal processing, others more control or protocol oriented tasks. Resources such as processor core registers, internal and external memory, DMA engines, and IO peripherals are shared by all tasks, often referred to as threads (Fig 1). This creates ample opportunities for the design or modification of one task to interact with another, often in unexpected or non-obvious ways. In addition, most DSP algorithms must run in real-time, so even unanticipated delays or latencies can cause system failures.

Common DSP software bugs are due to:

- Failure of interrupts to completely restore processor state upon completion;
- Blocking of a critical interrupt by another interrupt or by an uninterruptible process;
- Undetected corruption or non-initialization of pointers;
- Failure to properly initialize or disable circular buffering addressing modes;
- Memory leaks or gradual consumption of available volatile memory due to failure of a thread to release all memory when finished;
- Unexpected memory rearrangement by optimizing memory linkers/compilers;
- Use of special mode instruction options in core;
- Conflict or excessive latency between peripheral accesses, such as DMA, serial ports, L1, L2, and external SDRAM memories;
- Corrupted stack or semaphores; and
- Mixture of C or high-level language subroutines with assembly language subroutines.

Microprocessor, DSP, and operating system (OS) vendors have attempted to address these problems with different levels of protection or isolation of one task or thread from another. Typically the operating system, or kernel, is used to manage access to processor resources, such as allowable execution time, memory, or to common peripheral resources. However, there is an inherent conflict between processing efficiency and level of protection offered by the OS. In DSPs, where processing efficiency and deterministic latency are often critical, the result is usually minimal or no level of OS isolation between tasks. Each task often requires unrestricted access to many processor resources in order to run efficiently. Compounding these development difficulties is incomplete verification coverage, both during initial development and during regression testing for subsequent code releases. It is nearly impossible to test all the possible permutations (often referred to as corner cases) and interactions between different tasks or threads which may occur during field operation. This makes software testing arguably the most challenging part of the software development process. Even with automated test scripts, it is not possible to test all possible scenarios. This process must be repeated after every software update or modification to correct known bugs or add new features. Occasionally, a new software release also inadvertently introduces new bugs, which forces yet another release to correct the new bug. As products grow in complexity, the number of lines of code will increase, as will the number of processor cores, and an even greater percentage of the development effort will need to be devoted to software testing.
An FPGA is a more native implementation for most digital signal processing algorithms. Each task is allocated its own resources and runs independently. This intuitively makes more sense, to process an often continuously streaming signal in an assembly-line like process, with dedicated resources for each step. And the result is a dramatic increase in throughput. As the FPGA is inherently a parallel implementation, it offers much higher digital signal processing rates in nearly all applications.

FPGA resources assigned can be tailored to the requirements of the task. The tasks can be broken up along logical partitions. This usually makes for a well defined interface between tasks, and largely eliminates unexpected interaction between tasks. Because each task can run continuously, the memory required is often much less than in a DSP, which must buffer the data and process in a batch fashion. As FPGAs distribute memory throughout the device, each task is most likely permanently allocated the necessary dedicated memory. This achieves a high degree of isolation between tasks. The result is modification of one task being unlikely to cause an unexpected behavior in another task. This, in turn, allows developers to easily isolate and fix bugs in a logical, predictable fashion.

The link between system reliability and design methodology is often underappreciated. Generally, discussions about development tools emphasizing engineering productivity increase. But as product complexities increase, an even greater portion of the overall engineering process is dedicated to testing and verification. This is where FPGA design methodology offers large advantages compared to software-based design verification.

Fundamentally, FPGA design and verification tools are closely related to ASIC development tools. In practice, most ASIC designs are prototyped on FPGAs. This is a critical point, because bugs are most definitely not tolerated in ASICs. Unlike software, there is remote possibility of field upgrades to remedy design bugs in an ASIC. As time and development costs are very high, ASIC developers go to extreme lengths to verify designs against nearly all scenarios. This has led to test methodologies that provide nearly complete coverage of every gate under all possible inputs, accurate modeling of routing delays within the devices, and comprehensive timing analysis. Since FPGA verification tools are closely related cousins of their ASIC counterparts, they have benefited enormously from the many years of investment in the ASIC verification.

The use of FPGA partitioning, test benches, and simulation models makes both integration and on-going regression testing very effective for quickly isolating problems, speeding the development process, and simplifying product maintenance and feature additions. These are crucial advantages in the FPGA vs DSP development process and will become increasingly important as the complexity of designs and the size of development teams increase.

An FPGA vendor provides a comprehensive set of in-house and third-party tools to provide a unified tool flow for architecture, partitioning, floor planning, facilitated design intent, simulation, timing closure, optimization, and maintainability. In particular, architectural partitioning is integral to the design entry process. This partitioning, which normally includes chip resources required within the partition, is extended during timing closure and ongoing maintenance phases of the development, which guarantees a high degree of isolation. Each logical partition, as well as the overall design, can have independent test benches and simulation models. The development of the test benches during the FPGA design cycle can be reused to verify proper functionality *in later changes. This will make system maintenance much simpler.* The EDA industry is a large industry which continually drives the development of FPGA and ASIC test and verification tools. There is not a comparable counterpart in the software verification space. This may change as the industry realizes the enormous costs and challenges in software verification, but for now, the practical solution in the software world is to keep downloading the latest patch.

Many engineering managers intuitively understand this. The rate of software updates to remedy bugs far exceeds the rate of comparable FPGA updates. It is expected and normal to roll out bug fixes on embedded software on a regular basis. With the availability of both low-cost and high-end DSP-optimized FPGA devices, extensive IP cores, as well as the availability of high-level design entry methods and the inherent robustness of the design and verification processes, FPGAs will increasingly be the preferred choice for implementing digital signal processing. (Source: Michael Parker of Altera Corporation)

4.2 Discrepancies in the Project

This section mentions the weak areas of the project where further improvement is possible. Anyone continuing this project may consider removing these discrepancies.

4.2.1 Simulink Model of Pulse Doppler Radar

The simulation of pulse doppler radar was meant to be extended to a synthetic aperture radar simulation but it could not be accomplished. The doppler beam sharpening algorithm could not be simulated in Simulink.

4.2.2 Peak-to-Sidelobe Ration in Pulse Doppler Radar

The PSLR achieved in the simulation of pulse doppler radar was 25dB whereas a PSLR of 30dB or more is adequate for radar systems.

4.3 Future Recommendations

Here are some of the SAR applications for which, this project can be specialized.

4.3.1 Range Doppler Algorithm

The range-Doppler algorithm is another algorithm for processing continuously collected SAR data into an image. It is computationally efficient and, for typical spaceborne imaging geometries, the range-Doppler algorithm is an accurate approximation to the exact SAR transfer function. The future development in this project can include the range Doppler algorithm in the signal processing part of the system.

4.3.2 Spotlight SAR

Spotlight SAR is a mode of SAR operation for obtaining high resolution by steering the radar beam to keep the target within the beam for a longer time and thus form a longer synthetic aperture. This project can be extended to include spotlight SAR mode.

Compared to conventional SAR strip mapping mode, which assumes a fixed pointing direction of the radar antenna broadside to the platform track, Spotlight SAR is capable of extending the high-resolution SAR imaging capability significantly. This is achieved by keeping a target within the spotlight illumination of the radar beam for a longer time through electronic beam steering, resulting in a longer synthetic aperture which leads, in turn, to increased azimuth resolution. Spotlight SAR mode of operation is usually at the expense of spatial coverage, as other areas within a given accessibility swath of the SAR cannot be illuminated while the radar beam is spotlighting over a particular target area. Classified Spotlight SAR systems have been in operation for some time on military reconnaissance aircraft and satellites. (Source: http://www.ccrs.nrcan.gc.ca/glossary/index_e.php?id=581)

4.3.3 Interferometeric SAR (IFSAR)

Another application which can be implemented by improving this project is the IFSAR. Interferometric synthetic aperture radar (IFSAR) data can be acquired using two antennas on one aircraft or by flying two slightly offset passes of an aircraft with a single antenna. Interferometric SAR can be used to generate very accurate surface profile maps of the terrain. IFSAR is among the more recent options for determining digital elevation. It is a radar technology capable of producing products with vertical accuracies of 30 centimeters RMSE. Not only that, but IFSAR provides cloud penetration, day/night operation (both because of the inherent properties of radar), wide-area coverage, and full digital processing. The technology is quickly proving its worth. More about IFSAR is available at http://www.geospatial-solutions.com/geospatialsolutions.

4.4 References

i. White Paper by *Micheal Parker*
ii. http://www.geospatial-solutions.com/geospatialsolutions
iii. http://www.ccrs.nrcan.gc.ca/glossary/index_e.php?id=581

Appendix

A

MATLAB Code Listings

Codes used in Simulation of Pulse Compression

To use the GUI for pulse compression, enter the parameters in the input text boxes, select an option to view from the drop down menu and click start. Use matrix type of input e.g. [1 2 3] for multiple scatterers.

%LFM.m

```
function [xc,Xc,f,t,N]=LFM(taup,b,k)
clc;
N=k*b*taup;
N=round(N);
ts=1/(k*b);
fs=1/ts;
t=linspace(-taup/2,taup/2,N);
f=linspace(-fs,fs,N);
xc= exp(i*pi*(b/taup).*t.^2);
Xc=fftshift(abs(fft(xc)));
```

%matched_filter.m

```
function [out,time,N,dpoints,dist]=
matchedfilter(taup,b,k,scat_range,scat_rcs,scatno,rrec)
clc;
[xc,Xc,f,t,N]=LFM(taup,b,k);
%x(scatno,1:N)=0;
xcr(1:N)=0;
%LFM pulse received
G=1;Pt=10;Lam=0.1;c=3.e8;
for j=1:length(scat_range)
    A=((Pt*(G*Lam)^2*scat_rcs(j))/((4*pi)^3*scat_range(j)^4))^0.5;
    x(j,:)=A*exp(i*pi*(b/taup).*(t+(2*scat_range(j)/c)).^2);
    xcr=x(j,:)+xcr;
end
%Matched filtering of Continuous Time LFM Chirp
xct=xc;
out=xcorr(xct,xcr);
out=out./N;
time=linspace(0,-taup/length(out)+2*taup,length(out));
rres=(c*taup)/2;
dpoints = ceil(rrec * N /rres);
dist=linspace(0,rrec,dpoints);
figure(1);
plot(dist,abs(out(N:N+dpoints-1)));
```

```
xlabel('Range in metres');
ylabel('Matched Filter Output');
figure(2);
plot(time,abs(out));
xlabel('Time in sec');
ylabel('Matched Filter Output');
save matchedfilter.mat;
clc;
```

%windowedoutput.m

```
function [out outf outt time timet]=
windowedoutput(taup,b,k,scat_range,scat_rcs,scatno,rrec)
clc;
%Windowing in Frequecy Domain
[out,time,N,dpoints,dist]=
matchedfilter(taup,b,k,scat_range,scat_rcs,scatno,rrec);
winf=hamming(length(out))';
IN=fftshift(fft(out));
OUTF=abs(IN).*winf;
outf=fftshift(ifft(OUTF,length(OUTF)));
figure(3)
plot(time,20*log10(abs(out)),'r');
hold on
plot(time,20*log10(abs(outf)));
xlabel('Time in seconds');
ylabel('Frequecy Windowed Matched Filter Output in dBs');
hold off
%Windowing in time domain
wint=hamming(length(out));
WINt=fftshift(ifft(wint));
outt=conv(out,WINt);
timet=linspace(0,-taup/length(outt)+2*taup,length(outt));
figure(4)
plot(time,20*log10(abs(out)),'r');
hold on
plot(timet,20*log10(abs(outt)));
xlabel('Time in seconds');
ylabel('Time Windowed Matched Filter Output in dBs');
```

Codes used in Simulation of Pulse Doppler Radar

%Pulse_Compression.m

```
Chirp_Template=squeeze(Chirp_Template);
Chirp_Template=Chirp_Template(1:100);
Chirp_T_FFT1=fft(Chirp_Template);
window=hamming(100);
Chirp_T_FFT1=conj(Chirp_T_FFT1).*window;
% Received=squeeze(Received);
% Received=Received(1:1000);
% R=fft(Received);
% Chirp_T_FFT1=R.*Chirp_T_FFT1;
Chirp_Template=ifft(Chirp_T_FFT1);
save Chirp_Template;
```

How to Use GUI of SAR Parameter Calculation

After running the GUI, enter any parameters depending upon which you want to calculate the other parameters and press 'Calculate' button. The GUI will display all the other parameters that could be calculated using the available input parameters.

Codes Used in SARDBS

%git_chirp.m (This function is courtesy of Dr. Mark A. Richards, Georgia Institute of Technology)

```
function    x = git_chirp( T, W, p )
%CHIRP      generate a sampled chirp signal
%    X = git_chirp( T, W, <P> )
%      X:  N=pTW samples of a "chirp" signal
%            exp(j(W/T)pi*t^2)    -T/2 <= t < +T/2
%      T:  time duration from -T/2 to +T/2
%      W:  swept bandwidth from -W/2 to +W/2
%    optional:
%      P:  samples at P times the Nyquist rate (W)
%            i.e., sampling interval is 1/(PW)
%            default is P = 1
%
if nargin < 3
  p = 1;
end
J = sqrt(-1);
%--------------
delta_t = 1/(p*W);
N = round( p*T*W );    %--same as T/delta_t, but rounded
nn = [0:N-1]';
% x = exp( J*pi*W/T * (delta_t*nn - T/2).^2 );  % old version
x = exp( J*pi*W/T * (delta_t*nn - (N-1)/2/p/W).^2 );  % symmetric
version


% even older version
%%%%% alf = 1/(2*p*p*T*W);
%%%%% git_chirp = exp( J*2*pi*alf*((nn-N/2).*(nn-N/2)) );
```

%SAR_DataSet.m

```
function [DataSet x zerosVector]=SAR_DataSet (xp, yg, Ta, F, h, v, K,
B, Taw, PRF)
tic
clc
c=3e8;
lambda=c/F;
fs=K*B;
Dsar=v*Ta;
du=v/PRF;
t=0:1/(fs):Taw;
x=cos(pi*(B/Taw)*t.^2);
u=-Dsar/2:du:Dsar/2;
for SP=1:length(yg)
    Rp(SP)=(h^2+yg(SP)^2)^0.5;
    R(SP,:)=Rp(SP)*( ones(1,length(u))+((u-xp(SP)).^2)/Rp(SP)^2).^0.5;
    Phase(SP,:) = exp(-i*4*pi*R(SP,:)*F/c);
    m=1;
    for n=1:length(R(SP,:))
        tr=2*R(SP,n)/c;
        t1=0:1/fs:tr;
        zerosVector(m)=length(t1);
        m=m+1;
    end
    if (SP==1)
        scale=min(zerosVector);
    end
    zerosVector=zerosVector-scale*ones(1,length(zerosVector));
    for k=1:length(zerosVector)
        Ret=Phase(SP,k)*[zeros(1,zerosVector(k)) x
zeros(1,1+max(zerosVector)-zerosVector(k))];
        if(SP==1)
            DataSet1(k,:)=Ret;
        end
        if(SP==2)
            DataSet2(k,:)=Ret;
        end
        if(SP==3)
            DataSet3(k,:)=Ret;
        end
        if(SP==4)
            DataSet4(k,:)=Ret;
        end
%         if(SP==5)
%             DataSet5(k,:)=Ret;
%         end
%         if(SP==6)
%             DataSet6(k,:)=Ret;
%         end
%         if(SP==6)
%             DataSet6(k,:)=Ret;
%         end
%         if(SP==7)
%             DataSet7(k,:)=Ret;
%         end
%         if(SP==8)
%             DataSet8(k,:)=Ret;
%         end
%         if(SP==9)
%             DataSet9(k,:)=Ret;
%         end
%         if(SP==10)
%             DataSet10(k,:)=Ret;
%         end
    end
end
[r(1),col(1)]=size(DataSet1);
[r(2),col(2)]=size(DataSet2);
[r(3),col(3)]=size(DataSet3);
[r(4),col(4)]=size(DataSet4);
% [r(5),col(5)]=size(DataSet5);
% [r(6),col(6)]=size(DataSet6);
% [r(7),col(7)]=size(DataSet7);
% [r(8),col(8)]=size(DataSet8);
% [r(9),col(9)]=size(DataSet9);
% [r(10),col(10)]=size(DataSet10);
ZC=max(col);
DataSet1=[DataSet1 zeros(r(1),ZC-col(1))];
DataSet2=[DataSet2 zeros(r(2),ZC-col(2))];
DataSet3=[DataSet3 zeros(r(3),ZC-col(3))];
DataSet4=[DataSet4 zeros(r(4),ZC-col(4))];
% DataSet5=[DataSet5 zeros(r(5),ZC-col(5))];
% DataSet6=[DataSet6 zeros(r(6),ZC-col(6))];
% DataSet7=[DataSet7 zeros(r(7),ZC-col(7))];
% DataSet8=[DataSet8 zeros(r(8),ZC-col(8))];
% DataSet8=[DataSet9 zeros(r(9),ZC-col(9))];
% DataSet4=[DataSet4 zeros(r(10),ZC-col(10))];
DataSet=DataSet1+DataSet2+DataSet3+DataSet4;
%
DataSet=DataSet1+DataSet2+DataSet3+DataSet4+DataSet6+DataSet7+DataSet8+
DataSet9+DataSet10;
figure(1);image(real(DataSet));
xlabel('Fast Time Samples');
ylabel('Slow Time Samples');
title('RAW DATA');
save SAR_DataSet.mat;
clc
toc
```

%FastTime.m

```
function [matchFilter]=FastTime(DataSet, x, zerosVector)
tic
clc;
load SAR_DataSet;
maxPoint=2*(max(zerosVector)+150)-1;
matchFilter=zeros(length(zerosVector),(2*length(DataSet(1,:))-1));
for n=1:length(zerosVector)
    %chirp=DataSet(n,1:zerosVector(n)+150);
    chirp=DataSet(n,:);
    FTPC=xcorr(chirp,x);
    matchFilter(n,:)=[FTPC zeros(1,1+maxPoint-length(FTPC))];
    %matchFilter(n,:)=[FTPC];
end
figure(2); image(real(matchFilter));
xlabel('Pulse Compressed Fast Time Samples');
ylabel('Slow Time Samples');
title('Pulse Compressed');
save FastTime;
toc
```

%SlowTime.m

```
function [slowTime]=SlowTime()
tic
clc
load FastTime;
[o,p] = size(matchFilter);
WR = hamming(o);
for I=1:length(matchFilter(1,:))
```

```
    slowTime(:,I)=fftshift(abs(fft(WR.*matchFilter(:,I))));
end
figure(3);image(10*log10(abs(slowTime)));
xlabel('Pulse Compressed Fast Time Samples');
ylabel('Azimuth Compressed Slow Time Samples');
title('Azimuth Compressed');
save SlowTime.mat;
toc
```

%AxisMapping.m

```
function [MappedSet]=AxisMapping()
% Y[l,Fd]<==>Y[R,x]
% l-> R=Ro + c*Ts*l/2
% F-> x=-lamda*R*Fd/2*v
tic
clc
load SlowTime;
lambda=c/F;
Ro=mean(Rp);
for i=1:length(slowTime(1,:))
    Rm(:,i)=Ro+(0.5*c.*slowTime(:,i))./fs;
end
for i=1:length(slowTime(:,1))
    MappedSet(i,:)=0.5.*Rm(i,:).*slowTime(i,:)./v;
end
figure(4);image(real(MappedSet));
xlabel('Range');
ylabel('Cross Range');
title('Axis Mapped Image');
save AxisMapping;
toc
```

%Deshirped.m

```
function [DeChirped]=Dechirped()
tic
clc
load AxisMapping;
u=u';
for j=1:length(MappedSet(1,:))

DeChirped(:,j)=matchFilter(:,j).*exp(i*((2*pi*u.^2)./(Rm(:,j).*lambda))
);
    DeChirped(:,j)=fftshift(abs(fft(WR.*DeChirped(:,j))));
end
figure(5);image(real(DeChirped));
xlabel('Pulse Compressed Fast Time Samples');
ylabel('Range Compressed Slow Time Samples');
title('Azimuth Compressed Without Dechirp');
save DeChirped.mat;
toc
```

Appendix

B

Fast Fourier Transform

What is FFT

The fast Fourier transform (FFT) is a discrete Fourier transform algorithm which reduces the number of computations needed for N points from $2N^2$ to 2NlogN where log is the base-2 logarithm. If the function to be transformed is not harmonically related to the sampling frequency, the response of an FFT looks like a sinc function (although the integrated power is still correct). Aliasing (leakage) can be reduced by apodization using a tapering function. However, aliasing reduction is at the expense of broadening the spectral response.

The basic DFT (Discrete Fourier Transform) is

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}$$

Direct computation requires about 4N multiplications and 4N additions for each k (a complex multiplication needs 4 real multiplications and 2 real additions).

For all N coefficients, gives about $\mathbf{8N^2}$ operations.

Generally, we use FFT to refer to algorithms which work by breaking the DFT of a long sequence into smaller and smaller chunks.

Algorithms for computing the DFT which are more computationally efficient than the direct method (better than proportional to N^2) are called *Fast Fourier Transforms*.

Matlab and the FFT

Matlab's FFT function is an effective tool for computing the discrete Fourier transform of a signal. The following code examples will help you to understand the details of using the FFT function.

Example 1: The typical syntax for computing the FFT of a signal is FFT(x,N) where x is the signal, x[n], you wish to transform, and N is the number of points in the FFT. N must be at least as large as the number of samples in x[n]. To demonstrate the effect of changing the value of N, sythesize a cosine with 30 samples at 10 samples per period.

```
n = [0:29];
x = cos(2*pi*n/10);
```

Define 3 different values for N. Then take the transform of x[n] for each of the 3 values that were defined. The abs function finds the magnitude of the transform, as we are not concerned with distinguishing between real and imaginary components.

```
N1 = 64;
N2 = 128;
N3 = 256;
X1 = abs(fft(x,N1));
X2 = abs(fft(x,N2));
X3 = abs(fft(x,N3));
```

The frequency scale begins at 0 and extends to N - 1 for an N-point FFT. We then normalize the scale so that it extends from 0 to 1 – 1/N.

```
F1 = [0 : N1 - 1]/N1;
F2 = [0 : N2 - 1]/N2;
F3 = [0 : N3 - 1]/N3;
```

Plot each of the transforms one above the other.

```
subplot(3,1,1)
plot(F1,X1,'-x'),title('N = 64'),axis([0 1 0 20])
subplot(3,1,2)
plot(F2,X2,'-x'),title('N = 128'),axis([0 1 0 20])
subplot(3,1,3)
plot(F3,X3,'-x'),title('N = 256'),axis([0 1 0 20])
```

Upon examining the plot (shown in figure B.1) one can see that each of the transforms adheres to the same shape, differing only in the number of samples used to approximate that shape. What happens if N is the same as the number of samples in x[n]? To find out, set N1 = 30.

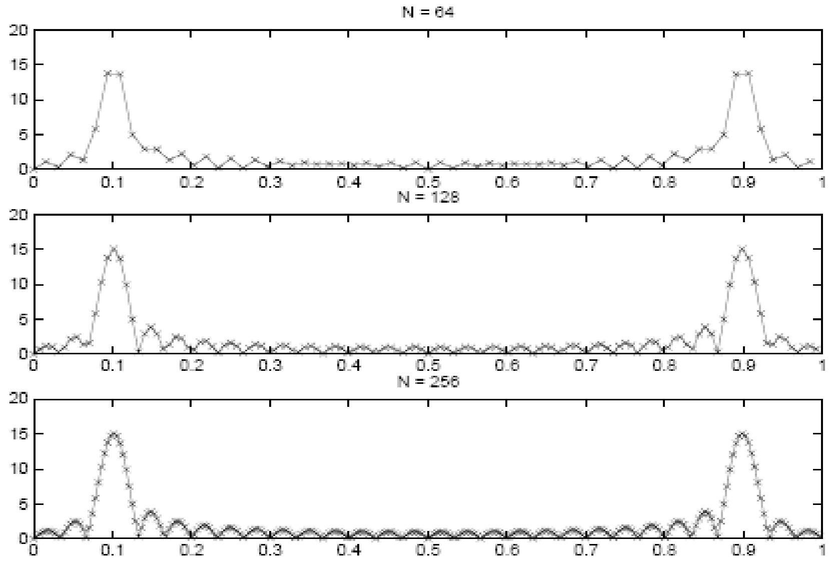


Figure B.1 FFT of a cosine for N = 64, 128, and 256

Fourier Transform Table

$x(t)$	$X(\omega)$
$ARect(t/\tau)$; rectangular pulse	$A\tau Sinc(\omega\tau/2)$
$A\Delta(t/\tau)$; triangular pulse	$A\frac{\tau}{2}Sinc^2(\tau\omega/4)$
$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{t^2}{2\sigma^2}\right)$; Gaussian pulse	$\exp\left(-\frac{\sigma^2\omega^2}{2}\right)$
$e^{-at}u(t)$	$1/(a+j\omega)$
$e^{-a\|t\|}$	$\frac{2a}{a^2+\omega^2}$
$e^{-at}\sin\omega_0 t \quad u(t)$	$\frac{\omega_0}{\omega_0^2+(a+j\omega)^2}$
$e^{-at}\cos\omega_0 t \quad u(t)$	$\frac{a+j\omega}{\omega_0^2+(a+j\omega)^2}$
$\delta(t)$	1
1	$2\pi\delta(\omega)$
$u(t)$	$\pi\delta(\omega)+\frac{1}{j\omega}$
$\text{sgn}(t)$	$\frac{2}{j\omega}$
$\cos\omega_0 t$	$\pi[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)]$
$\sin\omega_0 t$	$j\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]$
$u(t)\cos\omega_0 t$	$\frac{\pi}{2}[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)]+\frac{j\omega}{\omega_0^2-\omega^2}$
$u(t)\sin\omega_0 t$	$\frac{\pi}{2j}[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]+\frac{\omega_0}{\omega_0^2-\omega^2}$
$\|t\|$	$\frac{-2}{\omega^2}$

Appendix

C

Decibel Arithmetic

The decibel, often called dB, is widely used in radar system analysis and design. It is a way of representing the radar parameters and relevant quantities in terms of logarithms.

Gain in dB (in terms of power)	$\log(P_o / P_i)$
Gain in dB (in terms of voltage)	$\log(V_o / V_i)^2$
Gain in dB (in terms of current)	$\log(I_o / I_i)^2$
Definition of dB	$10\log(P_o / P_i) = 10\log(V_o / V_i)^2 = 10\log(I_o / I_i)^2$
Calculating Inverse dB	$P_0/P_i = 10^{dB/10}$ $V_0/V_i = 10^{dB/20}$ $I_0/I_i = 10^{dB/20}$
dB on Multiplication and Division	In general, $10\log\left(\frac{A \times B}{C}\right) = 10\log A + 10\log B - 10\log C$ $10\log A^q = q \times 10\log A$

Appendix

D

Speed Unit Conversions

Speed Conversion Formulae

To convert between miles per hour (mph) and knots (kts):

$$\text{Speed}_{\text{kts}} = 0.868391 \text{ x } \text{Speed}_{\text{mph}}$$

$$\text{Speed}_{\text{mph}} = 1.15155 \text{ x } \text{Speed}_{\text{kts}}$$

To convert between miles per hour (mph) and meters per second (m/s):

$$\text{Speed}_{\text{m/s}} = 0.44704 \text{ x } \text{Speed}_{\text{mph}}$$

$$\text{Speed}_{\text{mph}} = 2.23694 \text{ x } \text{Speed}_{\text{m/s}}$$

To convert between miles per hour (mph) and feet per second (ft/s):

$$\text{Speed}_{\text{f/s}} = 1.46667 \text{ x } \text{Speed}_{\text{mph}}$$

$$\text{Speed}_{\text{mph}} = 0.681818 \text{ x } \text{Speed}_{\text{ft/s}}$$

To convert between knots (kts) and meters per second (m/s):

$$\text{Speed}_{\text{m/s}} = 0.514791 \text{ x } \text{Speed}_{\text{kts}}$$

$$\text{Speed}_{\text{kts}} = 1.194254 \text{ x } \text{Speed}_{\text{m/s}}$$

To convert between knots (kts) and kilometers per hour (km/h):

$$\text{Speed}_{\text{km/h}} = 1.185325 \text{ x } \text{Speed}_{\text{kts}}$$

$$\text{Speed}_{\text{kts}} = 0.539593 \text{ x } \text{Speed}_{\text{km/h}}$$