AUTONOMOUS SURFACE DETECTION AND TRACKING FOR FMCW SNOW RADAR USING FIELD PROGRAMMABLE GATE ARRAYS

Aishwarya Bhatnagar

Submitted to the Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Thesis Committee:

Chairperson Dr. Carl Leuschen

Dr. Fernando Rodriguez-Morales

Dr. Christopher Allen

Date Defended: 01/30/2019

The Thesis Committee for Aishwarya Bhatnagar

certifies that this is the approved version of the

following thesis:

AUTONOMOUS SURFACE DETECTION AND TRACKING FOR FMCW SNOW RADAR USING FIELD PROGRAMMABLE GATE ARRAYS

Thesis Committee:

_____

Chairperson Dr. Carl Leuschen

_____

Dr. Fernando Rodriguez-Morales

_____

Dr. Christopher Allen

Date approved: 01/30/2019

Abstract

Sea Ice in Polar Regions is typically covered with a layer of snow. The thermal insulation properties and high albedo of the snow cover insulates the sea ice beneath it, maintaining low temperatures and limiting ice melt, and thus affecting sea ice thickness and growth rates. Remote sensing of snow cover thickness plays a major role in understanding the mass balance of sea ice, inter-annual variability of snow depth, and other factors which directly impact climate change. Researchers at the Center for Remote Sensing of Ice Sheets (CReSIS) at University of Kansas have developed an ultra-wide band FMCW Snow Radar used to measure snow thickness and map internal layers of polar firn from low and high-altitude. This system has shown outstanding performance, but it has some limitations in terms of operational altitude and relies on the operator to make adjustments during surveys to capture radar echoes if the altitude changes significantly. In this thesis, an automated onboard real-time surface tracker for the snow radar is presented to detect snow surface elevation from the aircraft and track changes in the surface elevation. A common technique for an FMCW radar to have a long-range (high-altitude) capability relies on the system's ability to delay the reference chirp signal used for de-chirping to maintain a relatively constant beat frequency. Currently, the radar uses an analog filter bank to condition the received IF signal over discrete altitude ranges and store the spectral power in each band utilizing different Nyquist zones. During airborne missions in Polar Regions with the radar, the operator has to manually switch the filter banks whenever there is a significant change in aircraft elevation. The work done in this thesis aims at eliminating the manual switching operation and providing the radar with surface detection, chirp delay, and a constant beat frequency feedback loop to enhance its long-range capability and ensure autonomous operation.

Acknowledgements

I want to thank my advisor Dr. Carl Leuschen who offered continuous guidance and support to me throughout the course of my master's studies. This work would not have been possible without Dr. Leuschen's vision and continuous feedback on my work. I want to thank my committee members Dr. Fernando Rodriguez-Morales and Dr. Christopher Allen for teaching me the technical concepts and providing valuable suggestions for my thesis. I want to thank the EECS department and all the faculty members who taught me during the course of my studies.

Most importantly, I would like to dedicate this work to my beloved father Mr. Sanjeev Bhatnagar who passed away the previous year. Dad, I want to tell you that whatever I have achieved is all because of your continuous motivation and confidence in me, I love you. I want to thank my mother Mrs. Neelu Bhatnagar and brother, Harsh for their unconditional support. My master's journey has been a worthwhile experience and these two years have taught me a lot. I also want to thank Akhilesh for letting me know about the amazing work done at CReSIS while he was a student at KU and encouraging me to be a part of CReSIS. Finally, I would like to thank all the staff members and students working at CReSIS who extended a helping hand whenever I needed one.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Global mean sea level has been rising over the past century, and the rate of sea level rise has been rapid in recent decades. The overall global rate of sea level rise during the last 100 years has been nearly 2 mm/year [1] and continues to rise at a rate of about one eighth of an inch per year in the 21st century [2]. Globally, this rise has led to thinning of coastlines, loss of agricultural land, abandoned islands, loss of habitat in marshlands and severe storm vulnerabilities [3]. The two major causes of sea level rise are thermal expansion due to warming of the ocean (since water expands as it warms) and melting of ice sheets in Greenland and Antarctica [4]. Therefore, as around 37% of the world's population lives within hundred kilometers of the coast [1], understanding as to why the sea levels continue to increase rapidly is a matter of exigency. By apprehending the reason of this rapid elevation in sea levels, scientists and geologists could develop prognostic and diagnostic models to predict future trends in global sea level rise.

Sea ice in Polar Regions has a covering of snow at the top, which usually varies in thickness between few centimeters to over one meter. This layer of snow over sea ice acts as a thermal insulator which in turn modulates heat exchanges between the ocean and the atmosphere. The thermal conductivity of snow is less than that of the sea ice beneath it, which in turn insulates the sea ice from extremely cold polar winds [5]. Thermal insulation reduces the basal sea-ice accumulation rate and high albedo (reflected solar energy) retards the sea ice melting during the summer [6]. The melting of this insulating snow cover creates ponds on the ice surface which have a low albedo and therefore absorb more heat than snow, which in turn increases the surface melting and initiates fresh water supply into the ocean [7]. Remote sensing of the depth of snow cover over sea ice with the help of

radars provides efficient data for the scientists to understand and investigate the winter inter annual snow depth, sea-ice mass balance, energy and surface heat budget [8]. High quality radar data also helps to determine the inter-annual variability of precipitation rates in the region which discerns the amount of freshwater input in Polar Regions.

The Center of Remote Sensing of Ice Sheets (CReSIS) is a science and technology center at The University of Kansas which was established by the National Science Foundation (NSF) in 2005, with a mission of developing new technologies and computer models to measure and predict the response of sea level change to the mass balance of ice sheets in Greenland and Antarctica [9].At CReSIS, researchers develop various radar sensors which are deployed in polar regions to provide precise measurement of ice bed thickness, mapping of internal layers, sounding and imaging of ice bed, etc. The data acquired from various radar sensors is provided to scientists and geologists and helps them studying the rapidly changing characteristic of polar ice sheets.

1.1 Motivation

Radars operating at a wide range of frequency spectrum possess large signal bandwidth which is essential for efficacious measurement of snow cover depth layers. Transmitting a signal which has a broad range of different frequencies (large bandwidth) enables the radar to be sensitive to minute changes in the snow microstructure. Frequency dependent radar signatures obtained from radars operating at multiple bands identified essential snow cover features such as ice and depth hoar layers [11]. Sub-band data processing utilizes the ultra-wideband nature of the FMCW radar to examine the scattering characteristics of snow within a particular frequency band of interest [10] [12]. A frequency modulated continuous wave radar operating at 2-18 GHz provides a large bandwidth which in turn improves the range resolution of snow cover images to about 1.4 cm from a nominal survey altitude of 500 meters [13] [14]. High operating bandwidth enables unambiguous detection of the underlying stratum of snow cover including the rough, thin and thick layers of snow on sea ice. The unambiguous

detection is further fine-tuned by achieving small range resolution which enables efficient differentiation of the air/snow and snow/ice interfaces.

Currently, as a part of NASA operation IceBridge, the frequency modulated continuous wave (FMCW) radar also called the 'Snow Radar', is used to map near surface internal layers of polar firn [12]. The 'Snow Radar' operates at a frequency of 2-18 GHz and is deployed on aircrafts to carry out airborne measurements of the Polar Regions. A linear, frequency modulated chirp is transmitted which impinges on the target, as a result a backscattered attenuated copy of the transmitted chirp signal is propagated back to the radar receiver after a propagation delay. The transmitted and the received chirps are multiplied together in the hardware and this process is called dechirping or deramping. After dechirping, the signal is bandpass filtered to produce an IF signal known as the beat frequency signal. The radar employs stretch chirp processing which supplies the reference chirp waveform from the radar to the local oscillator (LO) at the same time as the signal transmits from the antenna. Owing to the propagation delay of the received signal, the overlap between the transmitted and received signal has to be significant in order to achieve high signal to noise ratio (SNR). Figure 1.1 shows a picture of sea ice taken from DC-8 aircraft for NASA OIB.

Figure 1.1: Sea Ice as seen from the window of DC-8 aircraft [15]

For the FMCW radar to exhibit a long range capability, the radar must have a reference chirp delaying ability, i.e. the delayed and attenuated return signal should be mixed with a 'delayed' version of the reference chirp signal in order to perform exact match filtering. The amount of time by which the reference chirp signal has to be delayed would depend on the two way propagation time of the transmitted chirp signal. Consequently, the two way propagation time determines the range of the target and an FMCW radar maps range of the target to the IF or beat frequency. Therefore, a way of delaying the reference chirp signal supplied to the local oscillator should be deciphered that results in exact match filtering of the transmitted and received signals.

Currently, in order to achieve a longer range capability, the radar uses an analog filter bank to condition the received IF signal over discrete altitude ranges and store the spectral power in each band, utilizing Nyquist zones 1, 2, 3 and 4 as given in Table 1.1.During the airborne measurements, the radar operator has to manually switch the filter banks whenever there is a significant change in aircraft elevation in order to store the spectral power in each band. One way to bypass the manual switching is to delay the reference chirp by an amount of time corresponding to the increase in the beat frequency of the signal in each band. Thus, by delaying the reference chirp, manual switching of the filter banks could be avoided and the resulting IF signal would only be low pass filtered. Implementing this feature would fully automate the operation of the 'Snow Radar' and would no longer require manual switching of filter banks as the aircraft gains or loses altitude.

Snow Radar (Bandwidth: 2-18GHz, Chirp Length : 240 µsec)			
Nyquist Zone	Range(meters)	Beat Frequency(f_b)(MHz)	Filter Type
1	23 - 281	10 - 125	Low Pass
2	281 - 563	125 - 250	Band Pass
3	563 - 844	250 - 375	Band Pass
4	844 - 1125	375 - 500	High Pass

Table 1.1 Beat frequency range corresponding to different Nyquist zones [16].

1.2 Objectives and Approach

Frequency modulated continuous wave (FMCW) radars use the difference in the frequency of the transmitted as well as the received signal, also known as the beat frequency, to measure the range of the illuminating target. In order to delay the reference chirp fed to the local oscillator (LO), there must be a way of determining the range of the illuminated target in real time in order to determine the amount of time by which the reference chirp signal should be delayed and fed to the local oscillator in order to perform exact match filtering.

The objective of this thesis is to spectrally analyze the returned IF signal in real time and extract the beat frequency information from the signal. The FMCW radar maps the beat frequency directly to the range of the target by the following equation:

$$R = \frac{cT_S f_b}{2B} \qquad \text{..... (1.1)}$$

Where R is the range, f_b the beat frequency, T_S is the sweep time, B is the Bandwidth, and c is the speed of light.

Thus, by determining the range of the target from the beat frequency, one can determine the two-way propagation time of the chirp signal which in turn is equal to the amount of time required to

delay the reference deramping chirp signal sent to the local oscillator.

Moreover, along with calculating the delay for the reference chirp, a robust real time surface tracking algorithm needs to be implemented such that range of the ice surface from the aircraft must be tracked accurately. The tracking algorithm is necessary for the system as it shields the system from tracking and interpreting a wrong waveform peak arising due to a sudden glitch or noise. In order to maintain steady tracking of the ice surface, and to save the system from tracker errors arising due to unforeseen circumstances, implementing a tracking algorithm is a necessity.

In this thesis, to implement signal detection, spectral analysis of the signal was done in real time using fast fourier transforms and a peak tracking algorithm was implemented. The snow radar uses field programmable gate arrays for data acquisition. Field programmable gate arrays (FPGAs) are semiconductor devices which are designed to be configured and reprogrammed by the designer to suit the desired application needs after manufacturing [17].Field programmable gate arrays have an advantage over Application Specific Integrated Circuits (ASICs), as unlike ASICs which are manufactured for a specific application and have a fixed functional aspect, FPGAs can be programmed according to the changing needs of a system. In order to implement a peak detector and tracker unit and make it work in real time, FPGAs are ideally suited due to their high operating speed as well as software controlled reprogrammability [18].

1.3 Contribution

In this thesis, a module is developed which has two basic functions of signal peak detection and consistent peak tracking. The module can be appropriately called the ‘Peak detection and Tracking unit’. The unit will track the returned chirp signal from the ice sheet surface and would consistently update the altitude of the aircraft from the ice surface. The beat frequency obtained from the real time spectral analysis would correspond to the two way propagation time of the chirp and also the amount of time delay by which the local oscillator should receive the reference chirp in order

to perform the dechirp operation.

As mentioned in Table 1.1, different range of beat frequencies would correspond to different range of altitudes of the aircraft and the 'Peak detection and Tracking unit' would automatically delay the reference chirp without the need of manual switching of different filter banks. The radar operator would no longer need to manually switch the filter banks whenever there is a change in the aircraft altitude. This implementation would make the operation of the snow radar more autonomous and free from errors caused in the data acquisition due to manual switching of the filter banks.

1.4 Thesis Organization

The thesis is divided into six chapters. Chapter two presents literature review which comprises of the theory and background of frequency modulated continuous wave radars (FMCW). The next section underlines the theory and principles of operation of field programmable gate arrays. The chapter also reviews different tracking algorithms used for peak detection and tracking.

Chapter 3 discusses the method of peak detection using fourier transforms. It explains how a signal is processed and analyzed in the frequency domain and how spectrum analysis is useful specially in the case of frequency modulated continuous wave radars where the beat frequency of the signal is used to extract the two-way propagation time information.

Chapter 4 presents the importance of surface tracking and how it can be achieved. It also discusses how tracking can be achieved in real time using field programmable gate arrays and the approach used in building a tracker. It also highlights the challenges involved in surface tracking in real time.

Chapter 5 discusses the implementation of the peak detection and tracking module in the lab. It presents how the simulations were done on the host computer followed by the hardware implementation of the module. The chapter also highlights the process of synthesizing and implementing the design on FPGA and tools used for implementation. Finally, Chapter 6 summarizes

the thesis and the challenges faced in the implementation and discusses the future work that can be accomplished.

Chapter 2

Literature Review

Frequency modulated continuous wave (FMCW) radars are capable of performing target detection and ranging in the frequency domain, this makes them different from conventional radars which operate in the time domain. The use of FMCW radars with doppler capability in remote sensing applied to atmospheric applications was first demonstrated by Strauch *et al* in 1976 [19].The radar was designed at Colorado State University and was used to detect atmospheric precipitation (rain and snow).However, the demonstration was based on an analog frequency chirp and the received signal was analyzed analytically in analog form without involving the principles of discretization as well as digitization [20].

With the emergence of digital data acquisition systems (DAQs), the analog to digital converter (ADC) came into existence and this changed the entire perspective of signal analysis. The received signals could now be digitally sampled at a predefined sample rate and sample resolution (number of bits per sample).Digital acquisition of the received signals enabled the signal processing and analysis of the received data using numerical methods like Fast Fourier Transforms (FFT) [21].As the numerical analysis of the signals gained popularity, these methods imposed added restrictions on signal analysis such as sample rate limit and length owing to the digitization. This is how the digital mode of operation came into existence and FMCW radars also exploit this capability. The next section describes the principles of FMCW radars and the theory behind their mode of operation.

2.1 FMCW Radars -Theory and Background

Frequency Modulated Continuous Wave radars (FMCW) are a special category of radar sensors which transmit a continuous wave signal whose frequency content varies with time, therefore the signal is frequency modulated. The frequency of the RF signal changes over time in a sweep across a set bandwidth, i.e. the frequency changes linearly with respect to time. FMCW radars typically have a separate transmitter that transmits the signal continuously and a separate receiver for signal reception during transmission, this feature enables a zero blind range as the receiver is not off during transmission. This differentiates FMCW radars from the conventional pulsed radars which are characterized by their duty factor D given by:

$$D = \tau * PRF \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

Where τ stands for pulse duration and PRF is the pulse repetition frequency. The duty factor for pulsed radar typically ranges from 1% to 20%, whereas for continuous wave radars the duty factor is 100% [17].

The transmit signal typically used in an FMCW radar is a chirp waveform, which is given by equation:

$$s(t) = A\cos(2\pi(f_c t + 0.5kt^2) + \phi_c) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

Where f_c is the starting frequency in Hz, k is the chirp rate in Hz/sec, ϕ_c is the starting phase in radians, t is the time which ranges from $0 \leq t \leq \tau$, τ being the pulse duration, the chirp bandwidth is given by $B=k\tau$.

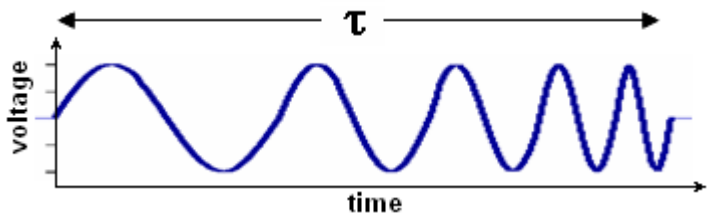

Figure 2.1 A linear frequency modulated chirp waveform [22].

The FMCW radar block diagram is given in figure 2.2, the steps of basic FMCW radar operation are described below:

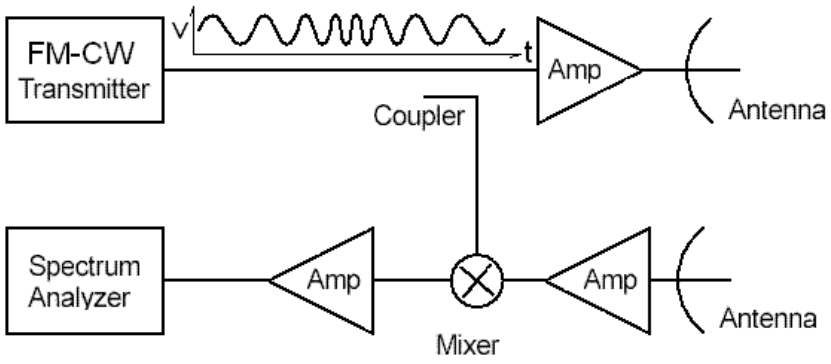

Figure 2.2 FMCW radar block diagram [22].

- The waveform generator generates an FMCW signal (Chirp in this case).
- The signal is radiated into free space by the transmitting antenna.
- Signal propagates through free space towards the target, gets reflected back and travels towards the radar.
- The reflected signal is collected by the receiving antenna and amplified.
- The received signal is then mixed with the transmitted signal, also known as the 'dechirping' operation.
- The difference in frequency between the transmitted and received signal, also known as the IF (Intermediate Frequency) signal is generated by mixing the two signals and then amplified.
- A spectrum analyzer is used to analyze the IF or the beat frequency signal.

The dechirping operation is given by equation 2.3:

$$s(t)s(t-T) = \text{acos}(2\pi(f_c t + 0.5kt^2) + \phi_c)\, acos(2\pi(f_c(t-T) + 0.5k(t-T)^2) + \phi_c) \ldots (2.3)$$

Where T is the two way propagation time of the signal.

After low pass filtering to reject the harmonics, the above equation simplifies to:

$$q(t) = \frac{a^2}{2}\cos(2\pi(f_c T + kTt - 0.5kT^2)) \ldots\ldots\ldots\ldots\ldots\ldots (2.4)$$

The two way propagation time of the signal T (sec) is given by the equation:

$$T = \frac{2R}{c} \quad \text{..... (2.5)}$$

Where R is the Range of the target in meters and c is the speed of light which is 3e8 m/sec. Therefore, if the two-way propagation time of a signal is known, the exact range of the target can be calculated by the above equation.

The beat frequency is the difference of the transmitted and the received signal frequencies and is given by the following equation:

$$R = \frac{cT_s f_b}{2B_{sweep}} \quad \text{..... (2.6)}$$

Where T_s is the sweep time of the signal, f_b is the beat frequency, and B_{sweep} is the sweep bandwidth of the signal. The beat frequency directly maps to the range given by equation 2.6. The details of the mapping are given in the figure:

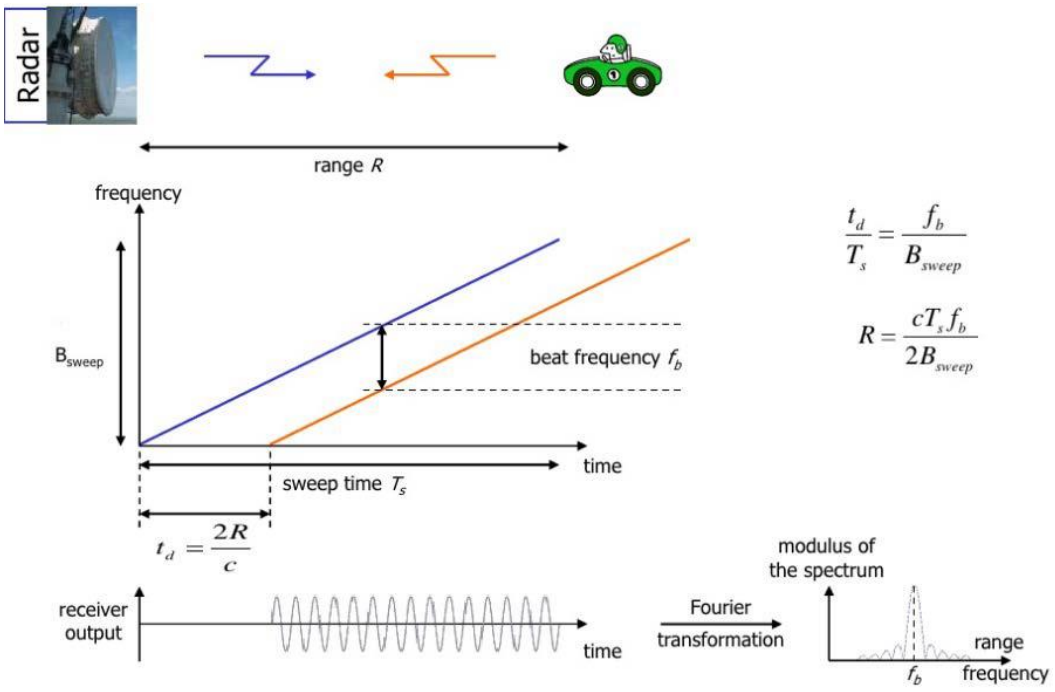

Figure 2.3 Extracting the beat frequency and converting it into range [23].

When a target moves relative to the radar, a Doppler frequency shift is introduced which can also be

calculated by performing several sweeps and storing the data into a matrix and then performing a 2D FFT on the matrix.

To improve my understanding of FMCW radar operation, I simulated an FMCW radar operation in MATLAB capable of detecting the range and velocity of two point targets. The design considerations are given in the table below:

PARAMETER	**VALUES**
Radar Center Frequency	77GHz
Propagation	Free space
Maximum range monitored	100m and 450m for two point targets.
Radial velocity	Kept the first target as stationary and the second target moving at a velocity of 50m/s
Range resolution ΔR	1m
Operating Bandwidth $B=c/2\ \Delta R$	150MHz

Table 2.1 Radar simulation parameters.

RCS of the 2 targets are given as follows:

Targets	RCS(m^2)	RCS (dB)
Target 1	1	0
Target 2	100	20

Table 2.2 Radar cross section of two targets.

2.1.1 FMCW Radar Operation Simulation

2.1.1.1 Beat Frequency Spectrum

The beat frequency spectrum given in fig 2.4 clearly shows the return coming from two targets in the illuminated scene, first and second point targets return beat frequencies of approximately 13.6 MHz and 61.3 MHz respectively.

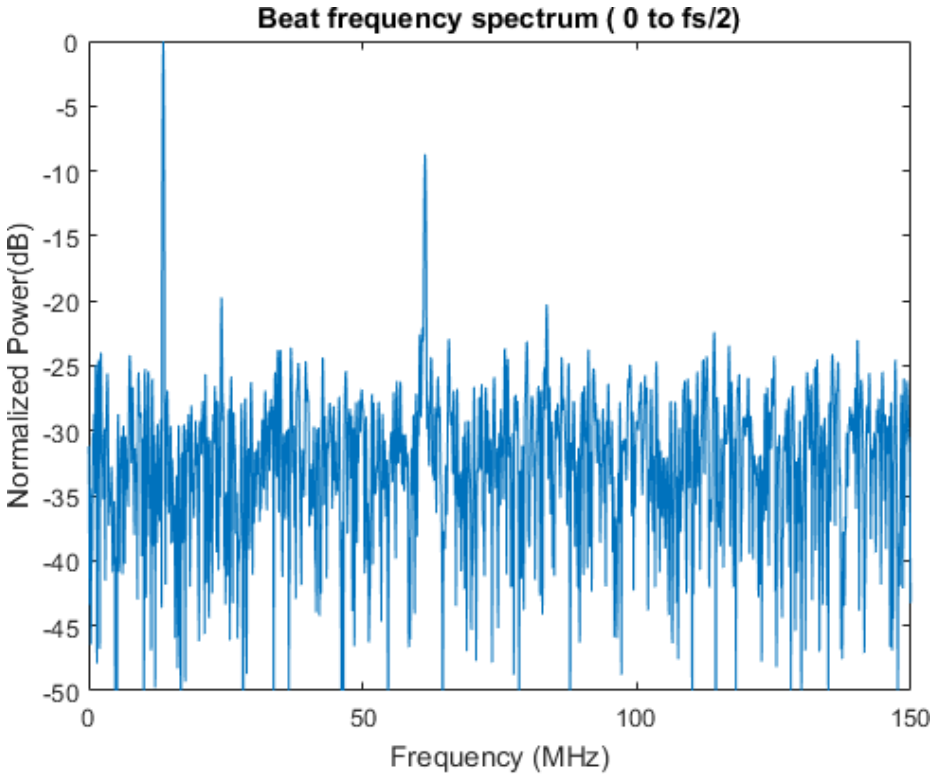

Figure 2.4: Beat frequency spectrum of received echoes.

2.1.1.2 Range Ascope

On converting the corresponding beat frequencies to the target ranges according to the equation 2.6, the target ranges are calculated to be 100 m and 450 m respectively which is in accordance with the plot given in figure 2.5.

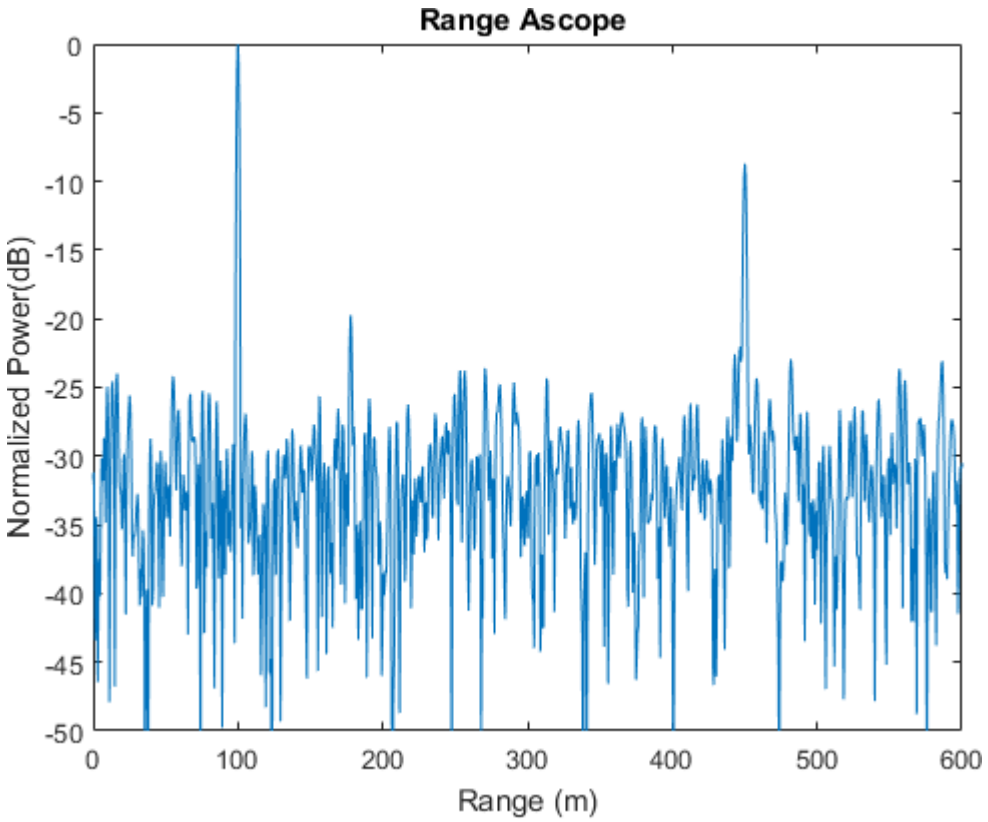

Figure 2.5: Ascope showing the range of targets and the relative strength of their echoes.

2.1.1.3 Range plot-slow time domain

100 slow time samples were taken for the generation of the illuminated scene and figure 2.6 represents the exact ranges of the two targets in the slow time domain. As the displacement is very low with respect to the slow time, the target 2, appears to be stationary in the figure at the range of 450 m.

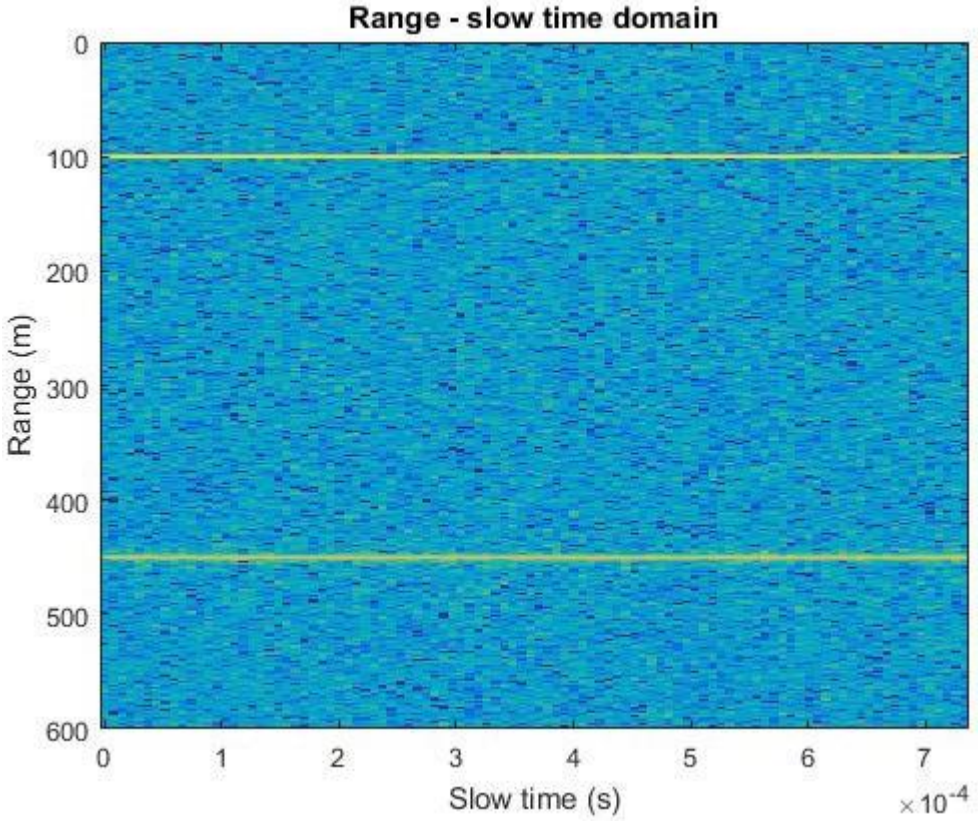

Figure 2.6: Range mapping in slow time domain

2.1.1.4 Range Doppler Spectrum

The range doppler spectrum is obtained by taking a 2D FFT of the received data matrix in which the superimposed signal is stored for several sweeps. A hanning window is also applied before taking the 2D FFT in order to suppress the sidelobes. We can clearly see from figure 2.7 that the target 1 appears to be stationary at the range of 100 m (velocity = 0 m/sec) and target 2 appears to move at a velocity of 50 m/sec which is in accordance to our expected estimation.

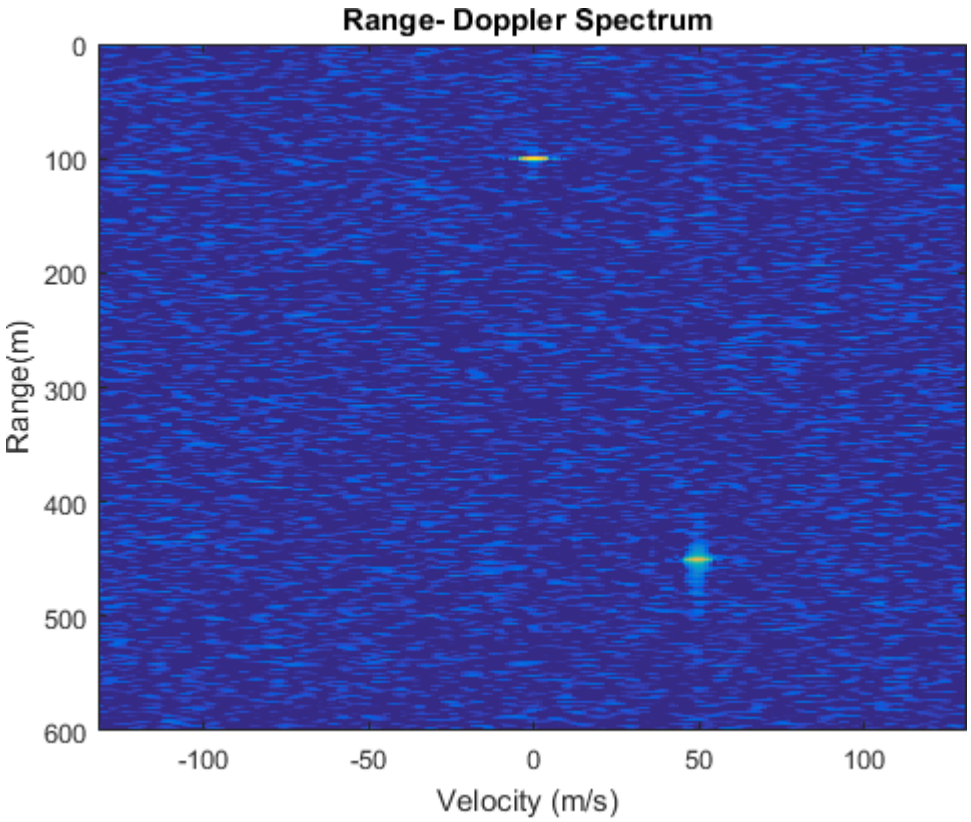

Figure 2.7: Mapping of the received data matrix in range and Doppler spectrum.

Therefore, by creating unique data vectors for each transmit and receive intervals during the course of several sweeps and saving it to a two dimensional data matrix and performing a 2D FFT which maps the signal in both range and doppler domains would give us the exact position and the velocity of the target. The MATLAB code for simulating the 77 GHz FMCW Radar is listed in Appendix A of this thesis.

2.2 Field Programmable Gate Arrays - Background and Principle

The origin of the field programmable gate arrays could be accredited to its precursors, the programmable read only memory (PROM) and programmable logic devices (PLDs). Both of these devices had the ability to be reprogrammed on the field. However, the configurable logic blocks needed to be hard wired to each other [24]. Xilinx Inc. invented FPGA in 1985 [24]. The company built on the programmable logic concept and created a chip which was entirely field programmable.

A field programmable gate array (FPGA) is an integrated circuit which can be programmed

by the designer according to the changing application needs after manufacturing. That is how the term 'field programmable' relates to the name. The reprogramming ability makes them different from the application specific integrated circuits (ASICs) as the functionality of ASICs is fixed and cannot be altered after manufacturing. FPGA's contain configurable logic blocks (CLBs) arranged in an array structure along with interconnects which are reconfigurable in nature to allow the CLBs to be wired together. Figure 2.8 describes the architecture of an FPGA. The CLBs can be configured to perform functions which are from implementing a simple logic gate function like AND or XOR to applications requiring complex combinational logic.

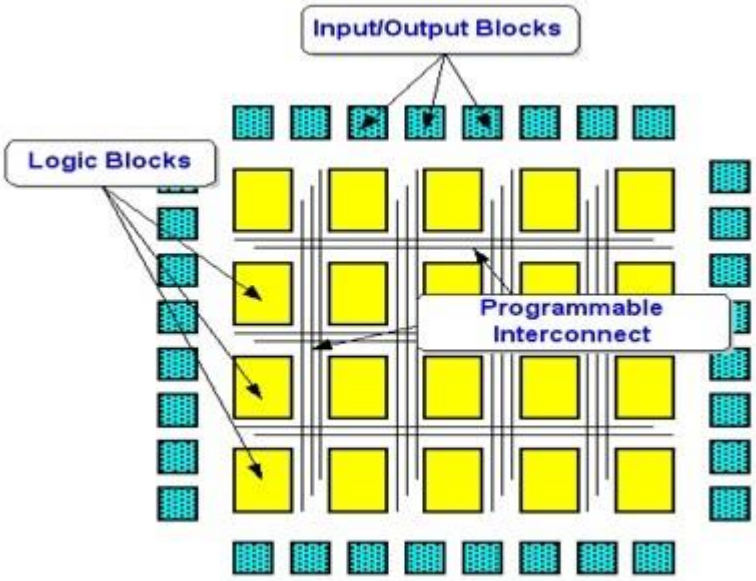

Figure 2.8: Basic architecture of a field programmable gate array [25].

The basic building block of an FPGA is a Look up Table (LUT). LUT is basically a truth table which has different combination of inputs to yield output values. Any combinatorial logic can be implemented in an FPGA by initializing a look up table. FPGAs also have storage elements for storing sequential logic states for every clock cycle such as flip flops (registers) and latches. Apart from LUTs, FPGAs also have dedicated Block RAMs, DSP blocks (are Arithmetic Logic Units), PCI Express supported to provide higher storage and computation speed. Also, FPGAs may have

additional elements such as Multiply Accumulate Blocks (MACs),High-speed serial transceivers, Phase locked loops (PLLs) which are used for operating the FPGA logic at multiple different clock rates. These components help in pipelining the implemented logic as well as providing inherent parallelism across multiple applications.

2.3 Peak Detection

In this thesis, the property of Fast Fourier transforms (FFT) has been utilized for analyzing the beat frequency of the chirp signal in the frequency domain. Fast Fourier transform (FFT) is the name given to an algorithm that computes the Discrete Fourier Transform (DFT) of a sequence. FFT is a fast computational algorithm for DFT which reduces the computational complexity of DFT from O (N^2) to O (NlogN).

In 1965, James Cooley and John Tukey published a journal article named "An Algorithm for the Machine Calculation of Complex Fourier Series" published in the journal Mathematics of Computation [21]. This algorithm is considered to be the most popular and generalized algorithm for calculating the fast fourier transform. Also known as the Cooley-Tukey algorithm dramatically reduced the computational cost of the regular DFT algorithm and became one of the indispensable algorithms in digital signal processing.

2.4 Review of Peak Tracking Algorithms

Implementing a real time surface tracker has to have two components, the first being signal detection, i.e. detection of the beat frequency corresponding to the return signal. Owing to the roughness of the surface snow cover, the backscatter from the snow surface would consist of a superposition of specularly reflected pulses from various points on the snow surface [26]. Therefore, a robust surface tracking algorithm needs to be implemented which can keep track of the return coming from the snow surface and therefore avoid any errors arising due to tracking wrong peak

occurring due to a glitch or noise.

In literature, there are various tracking algorithms used for different radar altimeters and there is no specific universal algorithm used for range tracking. The tracking algorithm depends on operating environment of the radar altimeter and several other factors. Next section gives an overview of some of the tracking algorithms used in radar altimeters.

2.4.1 Adaptive Linear Prediction based Tracking

Adaptive linear prediction is used to estimate and predict the future values of a signal based on its past values. This technique is widely used in speech and image compression. Figure 2.9 shows how the adaptive filter is used to predict the next value of the signal on the basis of its past values.

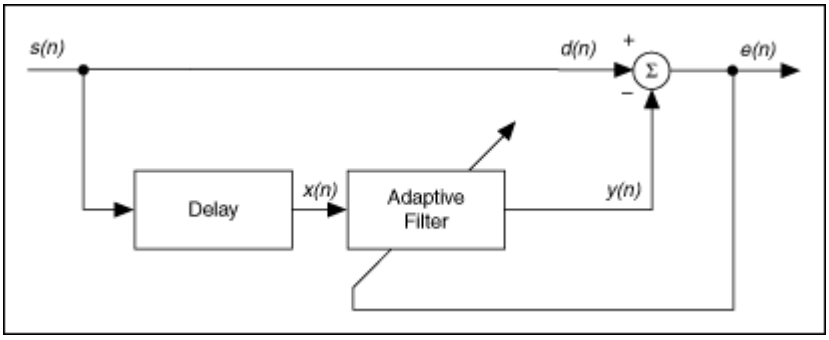

Figure 2.9: Adaptive linear prediction [27].

In figure 2.9, s(n) is a time series signal at time n. x(n) is the delayed version of s(n) and x(n) acts as an input to the Adaptive Filter, y(n) is the output from the adaptive filter. The difference between the output of the adaptive filter y(n) and the input signal s(n) is calculated by the linear prediction system and an error signal e(n) is produced. The prediction system adjusts the adaptive filter coefficients iteratively according to an adaptive algorithm such as to minimize the error e(n). The system predicts the future values based on past signal values when the error e(n) reaches the minimum value [27]. Thus, linear prediction could be used in estimating the next location of snow surface on the basis of past samples.

2.4.2 Kalman Filter Based Tracking

As the name suggests, Kalman filter is named after Rudolf E. Kálmán [28]. The filter works in two steps:

- Prediction
- Correction

It is a mathematical model which uses measurements observed over time which can contain random errors, noise or other variations and predicts values which are closer to the correct values of the quantity under measurement. Kalman filter has been widely used in radar tracking systems and has enormous applications in navigation, guidance and control of spacecrafts, aircrafts, etc.

Kalman filter works as a recursive estimator [29]. It is a set of mathematical equations which recursively estimate the state of the process in such a way that it minimizes the mean of the squared error.

The goal of the filter is to estimate the state $x \in \Re^n$ of a discrete time-controlled process which is given by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \quad \text{(2.7)}$$

And with a measurement $z \in \Re^m$ given by

$$z_k = Hx_k + v_k \quad \text{(2.8)}$$

Where w_k and v_k represent the process and measurement noise respectively [29].

Both are assumed independent from each other with probability distributions

$$p(w) \sim N(0,Q), \quad \text{(2.9)}$$

$$p(v) \sim N(0,R) \quad \text{(2.10)}$$

where Q is the process noise covariance and R is the measurement noise covariance, they are assumed to be constant.

We take $\hat{x}^- \in \Re^n$ to be the a priori state estimate at step k where we assume that we have the

knowledge of the process prior to step k, and $\hat{x} \in \Re^n$ is the a posteriori state estimate at step k with the given measurement z_k. Therefore, the a priori and a posteriori estimate errors are given by,

$$e_k^- = x_k - \hat{x}_k^- \quad \text{(2.11)}$$

$$e_k = x_k - \hat{x}_k \quad \text{(2.12)}$$

The a priori estimate error covariance is given as

$$P_k^- = E[(e_k^- e_k^{-T})] \quad \text{(2.13)}$$

And the a posteriori estimate error covariance is

$$P_k = E[(e_k e_k^T)] \quad \text{(2.14)}$$

The Kalman filter estimates and predicts the state of the process and then acquires feedback in the form of measurements embedded with noise. There are two sets of equations for Kalman filter,

- Time update equations
- Measurement Update equations

The time update equations are basically the equations that predict a value, they project the current state forward in time and error covariance estimates in order to get the a priori estimates for the next time step, whereas the measurement update equations provide feedback (correction), thus incorporating a new measurement into the earlier calculated apriori estimate to obtain an improved version of a posteriori estimate.

Discrete Kalman filter time update equations
$\hat{x}_k = A\hat{x}_{k-1} + Bu_{k-1}$
$P_k^- = AP_{k-1}A^T + Q$

Table 2.3: Discrete Kalman filter time update equations [30].

Discrete Kalman filter measurement update equations
$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$
$\hat{x}_k = \widehat{x_k}^- + K_k(z_k - H\widehat{x_k}^-)$
$P_k = (I - K_K H) P_k^-$

Table 2.4: Discrete Kalman filter measurement update equations [30].

Table 2.1 contains equations that gives the current state and covariance estimates of time step k from time step k-1.Table 2.2 has equations that provide correction, i.e., first equation updates and computes the Kalman gain, second equation gives a posteriori state estimate and the third equation gives a posteriori error covariance estimate. Figure 2.9 shows the complete operation of the Kalman filter by the set of equations. Therefore, the previous a posteriori estimates are used to predict the new a priori estimates.

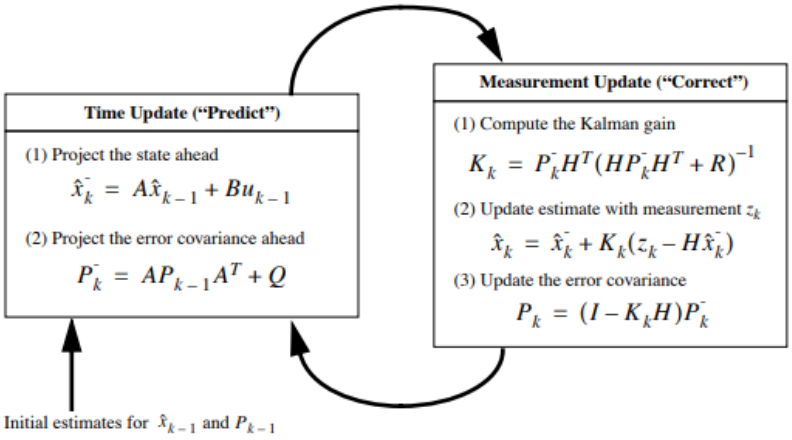

Figure 2.10 Kalman filter Operation [30]

2.4.3 Offset Center of Gravity Algorithm (OCOG)

Offset Center of Gravity Algorithm (OCOG) was developed by Mullard Space science Laboratory [31]. OCOG first tries to estimate the shape of the echo waveform parameters. This algorithm uses an estimation of pulse width to track the return waveform from the target. The width of the pulse is given by W in equation 2.15 as,

$$W = \frac{\left(\sum_{n=0}^{k-1} p_n\right)^2}{\sum_{n=0}^{k-1} p_n^2} \quad \text{(2.15)}$$

Where p_n is the power corresponding to range bin n, i.e the value of n^{th} element in an array. The middle point of the tracking window is assumed to be at half of W, i.e, W/2 and to the left of the gravity center of the return waveform. Assuming a pulse is defined as,

$$p_n = \begin{cases} A, & m \le n < m + j \\ 0, & otherwise \end{cases} \quad \text{(2.16)}$$

Getting the gravity center of the return waveform in a tracking window, the error signal is as shown in equation 2.17. The error output would then be sent to the next processing unit.

$$E = \frac{\sum_{n=0}^{k-1} np_n}{\left(\sum_{n=0}^{k-1} p_n\right)^{-\frac{W}{2}}} \quad \text{(2.17)}$$

In a scenario where the tracking window position changes, a series of errors E can be obtained by

which the tracking compensation could be achieved [32].

The kind of tracking algorithms used for FMCW radars depend on the operating environment, physical properties of the target, for example whether the reflecting surface is ocean or land, etc. Moreover, the algorithm has to be implemented on a field programmable gate array which has limited hardware resources and has computational restraints due to its high operating speed and hardware parallelism. Thus, the tracking algorithm chosen should be computationally efficient such that it does not utilize a large number of resources which may not be possible to implement on an FPGA with limited hardware specifications.

Chapter 3

Peak Detection

Peak detection refers to the detection of the strongest returned chirp signal frequency when analyzing the signal in frequency domain. Frequency domain analysis is one of the tools in signal processing which is of utmost importance. It has wide range of applications in the area of remote sensing, image processing, control systems, communication systems, etc. The difference between time domain analysis and frequency domain analysis is that, time domain analysis gives an idea of how the signal changes with respect to time, whereas frequency domain analysis gives an idea how signal's energy is distributed amongst a range of frequencies. Along with the energy information, frequency domain analysis also provides information on how much phase shift should be applied to each of the frequency components such that the original time signal would be recovered having a combination of all the individual frequency components of the signal. Let us look at the basics of spectral or frequency domain analysis of a signal.

3.1 Spectrum Analysis

A signal could be converted between time and frequency domains using mathematical operators known as transforms. The signal can be processed within these two domains and the process translation of most importance is convolution because convolution in the time domain is equivalent to multiplication in the frequency domain and vice versa. Figure 3.1 describes the various transforms which are used for different applications.

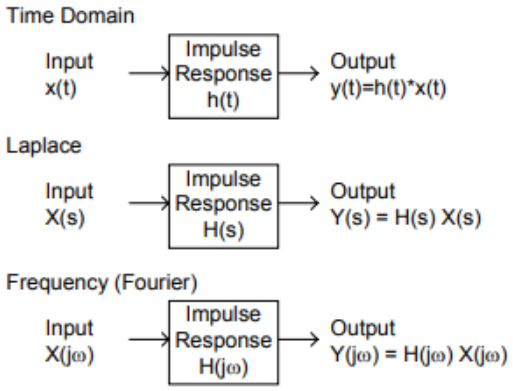

Figure 3.1: Different types of Transforms [33].

3.2 Discrete Fourier Transform

Discrete Fourier Transform (DFT) is a primary tool of digital signal processing. It is used to perform Fourier analysis of many applications. DFT is also used for solving partial differential equations and to perform complex operations such as convolution.

To further understand the DFT, let's look at the numerical representation, the z-transform of a periodically finite sequence is given as:

$$X(z) = \sum_{n=0}^{N-1} x(n) z^{-n} \quad \text{..... (3.1)}$$

If we periodically extend x (n) such that:

$$x(n) = \begin{Bmatrix} \hat{x}(n), & 0 \le n \le N-1 \\ 0, & elsewhere \end{Bmatrix} \text{..... (3.2)}$$

The discrete fourier transform of a periodically extended signal is given by,

$$\hat{X}(k) = \sum_{n=0}^{N-1} \hat{x}(n) e^{-\frac{j2\pi kn}{N}} = \sum_{n=0}^{N-1} \hat{x}(n) \left[e^{\frac{j2\pi k}{N}} \right]^{-n} \text{..... (3.3)}$$

On comparing equations (3.1) and (3.3) we get a relationship between z-transform of the finite sequence and the discrete fourier transform of a periodically extended sequence given as,

$$\hat{X}(k) = X(z)|_{z=e^{j2\pi k/N}} \quad \text{..... (3.4)}$$

The relationship between discrete fourier series (DFS) and discrete time fourier transform (DTFT) is given as,

$$\hat{X}(k) = X(\Omega)|_{\Omega=e^{j2\pi k/N}} \ldots\ldots\ldots\ldots (3.5)$$

Therefore, as we can see, the DFS can be obtained by sampling the DTFT at equal intervals along the unit circle

$$\Omega = \frac{2\pi}{N} \ldots\ldots\ldots\ldots (3.6)$$

Assuming that the input signal is periodic, but we only analyze N points, and moreover, assume that the output signal is also periodic only observing N points, then we obtain a mathematical entity which is known as the Discrete Fourier Transform (DFT).

The DFT is described by equations,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \leq k \leq N-1 \ldots\ldots\ldots\ldots (3.7)$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) W_N^{-nk} \quad 0 \leq n \leq N-1 \ldots\ldots\ldots\ldots (3.8)$$

Therefore, DFT can be given as:

$$X(k) = DFT\{x(n)\} \ldots\ldots\ldots\ldots (3.9)$$

Figure 3.2 shows input signals being transformed using various transforming techniques.

3.3 Fast Fourier Transform

Fast Fourier Transform is basically a fast-computational algorithm for discrete fourier transform (DFT). To understand the FFT algorithm, it is essential to first understand the computational complexity of the DFT. The number of multiplies and adds which are required to compute the discrete fourier transform of a length N signal, equation 3.7 gives the DFT of a signal, which could be further given as,

$$X(k) = \sum_{n=0}^{N-1} [Re\, x(n) + jIm\, x(n)].\left[ReW_N^{kn} + j\, ImW_N^{kn}\right] \quad 0 \le k \le N-1$$

$$\sum_{n=0}^{N-1} \left\{Re\, x(n)ReW_N^{kn} - Im\, x(n)ImW_N^{kn} + jRe\, x(n)ImW_N^{kn} + jImx(n)ReW_N^{kn}\right\}\ 0 \le k \le N-1 \quad (3.10)$$

If we compute the complexity of the DFT algorithm, we see that for each *k* and *n*, we have 4 real multiplications and 2 real additions, and for each *k* there are *N* values of *n*, therefore, we get 4N real multiplications and 4N-2 additions. For N values of k, there are

$4N^2$ multiplications ~N^2,

N (4N-2) additions ~ N^2

Therefore, DFT is computationally intensive algorithm and some other efficient way of performing the transform was required.

Cooley and Tukey invented the FFT algorithm back in 1965, they exploited the conjugate symmetry and periodicity properties of the DFT,

$$W_N^{k(N-n)} = (W_N^{kn})^* \text{ Conjugate symmetry} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.11)$$

$$W_N^{((kn))_N} = W_N^{kn} \quad \text{Periodicity} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.12)$$

So, for a 4-point FFT,

$$X(k) = \sum_{n=0}^{3} x(n)W_4^{kn}$$

When we expand it using DFT algorithm, we have,

$$X(0) = x(0)W_4^0 + x(1)W_4^0 + x(2)W_4^0 + x(3)W_4^0$$

$$X(1) = x(0)W_4^0 + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3$$

$$X(2) = x(0)W_4^0 + x(1)W_4^2 + x(2)W_4^4 + x(3)W_4^6$$

$$X(3) = x(0)W_4^0 + x(1)W_4^3 + x(2)W_4^6 + x(3)W_4^9$$

The above operation requires 64 real multiplies and 56 real additions. Using the properties of symmetry and periodicity we have,

$$W_4^0 = -W_4^2$$

$$W_4^1 = -W_4^3$$

$$W_4^0 = W_4^0$$

$$W_4^6 = W_4^2 = -W_4^0$$

$$W_4^9 = W_4^1$$

From the above properties we have,

$$X(0) = x(0)W_4^0 + x(1)W_4^0 + x(2)W_4^0 + x(3)W_4^0$$

$$X(1) = x(0)W_4^0 + x(1)W_4^1 - x(2)W_4^0 - x(3)W_4^1$$

$$X(2) = x(0)W_4^0 - x(1)W_4^0 + x(2)W_4^0 - x(3)W_4^0$$

$$X(3) = x(0)W_4^0 - x(1)W_4^1 - x(2)W_4^0 + x(3)W_4^1$$

$$X(0) = [x(0) + x(2)]W_4^0 + [x(1) + x(3)]W_4^0$$

$$X(1) = [x(0) - x(2)]W_4^0 + [x(1) - x(3)]W_4^1$$

$$X(2) = [x(0) + x(2)]W_4^0 - [x(1) + x(3)]W_4^0$$

$$X(3) = [x(0) - x(2)]W_4^0 - [x(1) - x(3)]W_4^1$$

If we draw a flowgraph for the above equations, we get a butterfly structure given by,

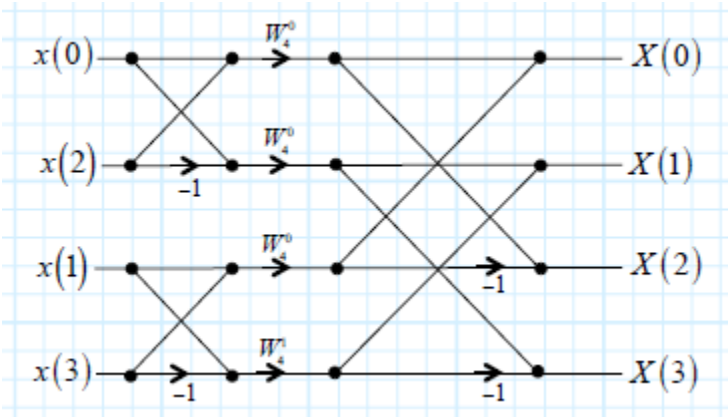

Figure 3.2: A 4-point FFT implementation, note that it requires 8 complex additions and 4 complex multiplies [34].

Thus, the computational complexity in terms of big O notation is given as,

$$\text{DFT - O } [N^2]$$

$$\text{FFT – O } [N \log_2 N]$$

3.4 Spectrum Analysis using FFT

FFT is used to find the components of a signal which is buried in a noisy time domain signal and to analyze the frequency components of a signal. Let us understand how FFT algorithm is implemented and can be used to get signal information from a noisy signal. In order to model a scenario, 2 sinusoidal signals of amplitude =1 and frequency f1=10 Hz and f2=50 Hz are added together and the signal is shown in figure 3.3.

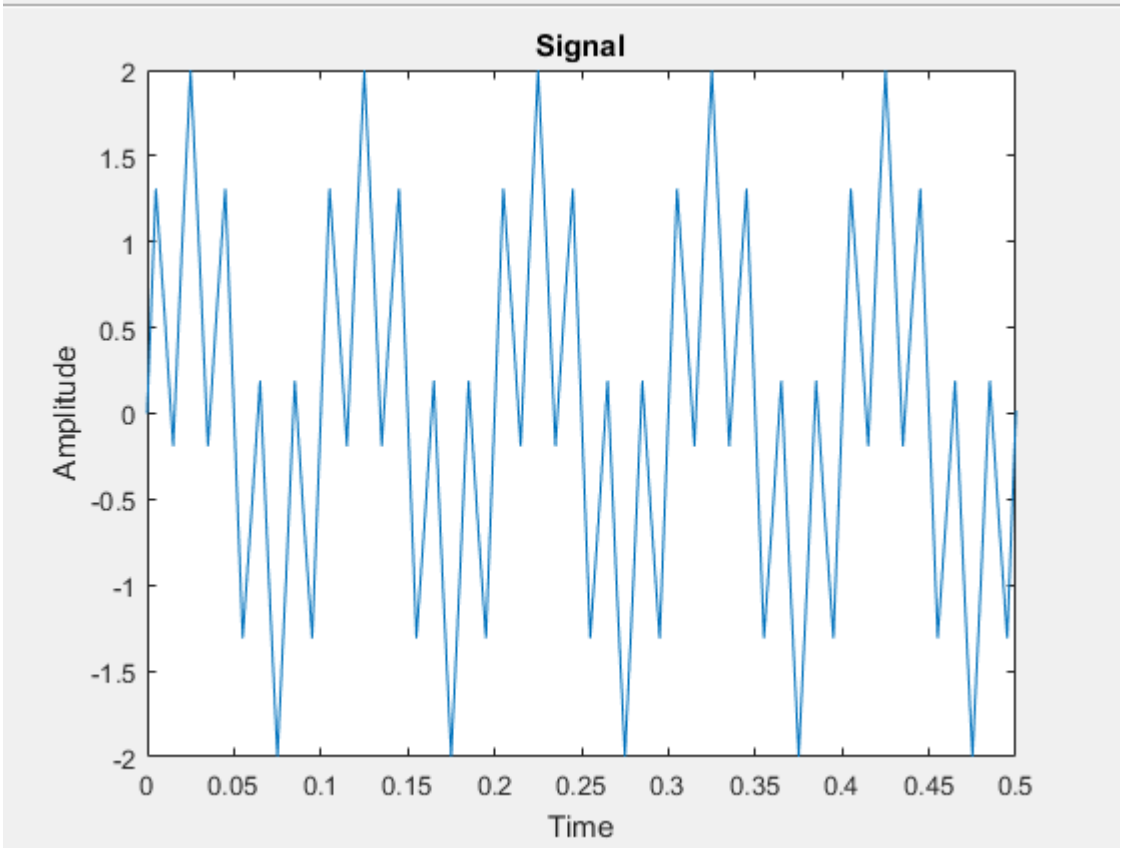

Figure 3.3: Signal containing 2 sinusoids of frequency 10 Hz and 50 Hz.

Next, random noise is added to the above signal and the signal now looks like as given in figure 3.4.

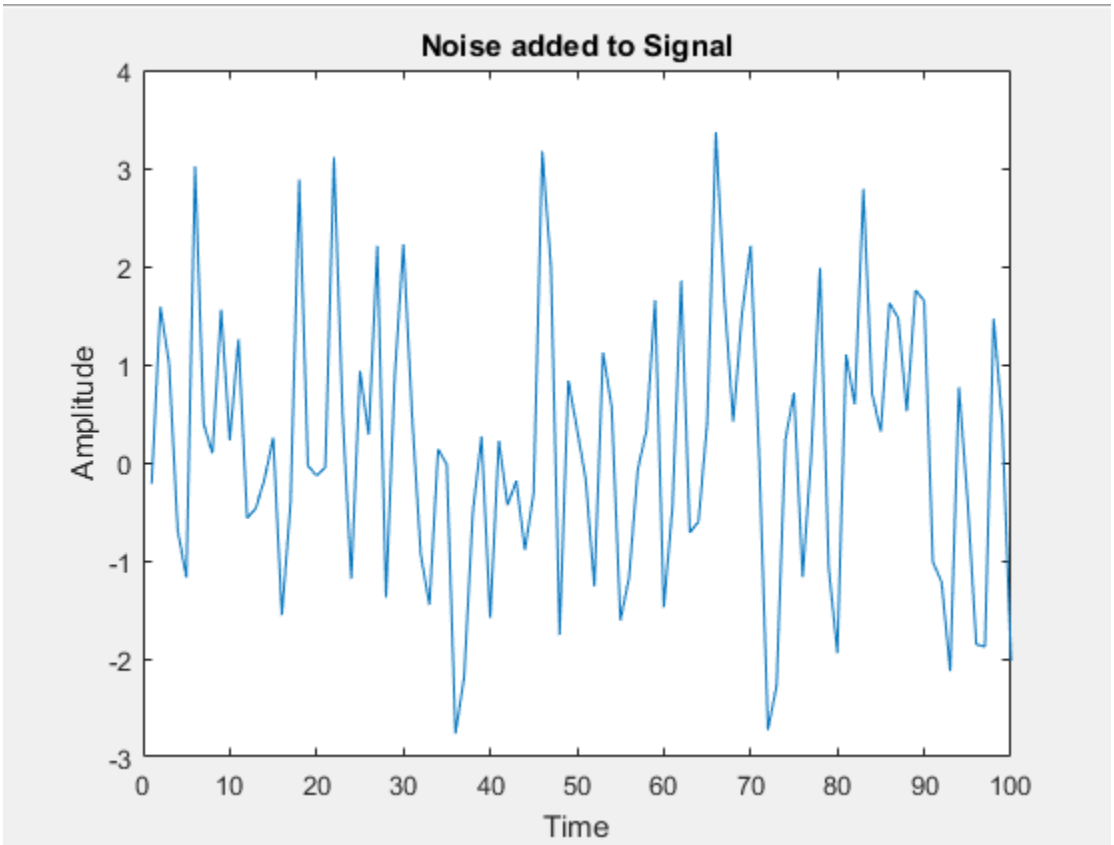

Figure 3.4: Random noise added to signal.

Now, we take the FFT of the signal and plot it on the frequency axis. In the power spectral density plot given in figure 3.5, 2 peaks representing two signals with frequency components 10 Hz and 50 Hz can be clearly seen in the figure and that is how we could effectively determine the frequency components of a signal embedded in noise through the FFT algorithm.

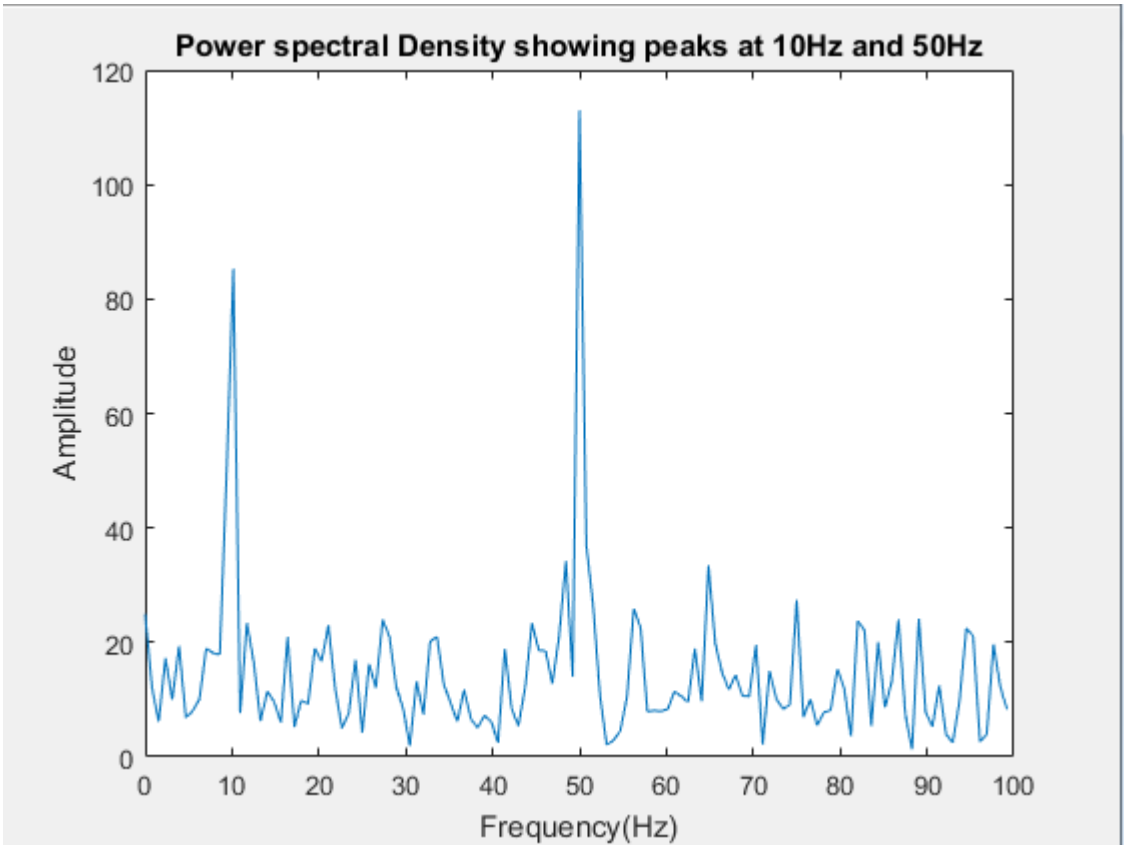

Figure 3.5: Power spectral density plot of the noisy signal.

The MATLAB code for the above implementation is given in table 3.1.

Table 3.1 MATLAB code for FFT implementation

```
%----------%% MATLAB Code for FFT implementation %%------------
clear all;
close all;

A=1; %amplitude

f1=10;%frequency 1
f2 = 50;%frequency 2
fs=200;%sampling frequency
t=0:1/fs:1;%time axis
S=A*sin(2*pi*f1*t)+A*sin(2*pi*f2*t);

figure(1);
plot(t,S);
xlim([0 0.5]);
title('Signal','Color','k');
ylabel('Amplitude');
xlabel('Time');

y = S + randn(size(t));%adding noise
figure(2)
```

```
plot(y)
xlim([0 100]);
title('Noise added to Signal','Color','k');
ylabel('Amplitude');
xlabel('Time');

nfft=2^nextpow2(length(S));
w=fft(y,nfft);%taking fft
w=w(1:nfft/2);
z=abs(w);

fre_axis=(0:nfft/2-1)*fs/nfft;

figure(3);
plot(fre_axis,z);
title('Power spectral Density showing peaks at 10Hz and
50Hz','Color','k');
ylabel('Amplitude');
xlabel('Frequency(Hz)');
```

3.5 Implementing Peak Detection on FPGA

As stated earlier, FFT is a faster way of performing spectral analysis of signals. In order to provide real time measurement analysis, FPGAs offer the desired computational performance required for FFT analysis. The data acquisition system for the Snow radar uses National instruments (NI) software LabVIEW and LabVIEW FPGA. NI FlexRIO FPGA provide high performance Xilinx Virtex class FPGAs which can be programmed with NI LabVIEW design software [35]. The software provides a built-in FFT IP for LabVIEW FPGA which can be used to implement the FFT logic as shown in figure 3.6. However, the inherent hardware parallelism which one gets with field programmable gate arrays (FPGAs), also introduces added complexity of synchronizing the data between operations being performed at different rates and clock cycles. Different types of algorithms need different number of clock cycles to complete, therefore if an IP core is expecting the input data before it is actually made available, all the data might get corrupted without an actual warning [36].Therefore, the FFT IP has different Boolean lines which are used as handshaking signals to pass data between different functions and loops within the FPGA block diagram. These handshaking

signals are named ‘ready for input’, ’input valid’ and ‘ready for output’.

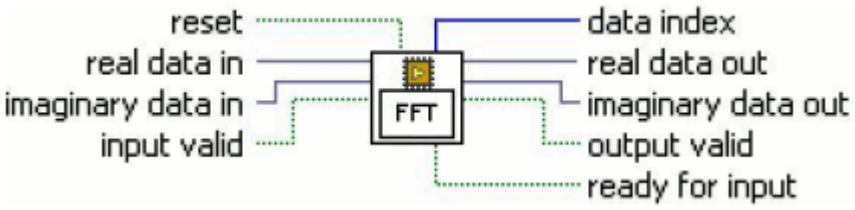

Figure 3.6: FFT IP Core in LabVIEW FPGA [37].

The FFT algorithm uses fixed point number representation (FXP) as inputs to calculate the FFT owing to its limited resources. It uses ‘Continuous input indexes/continuous output indexes’ which makes the FFT continuous to both input and output data, also referred as ‘Single Channel Single Sample’. When implementing FFT inside a single cycle timed loop (SCTL), the FFT works as shown in figure 3.7 where throughput is not equal to one cycle per input.

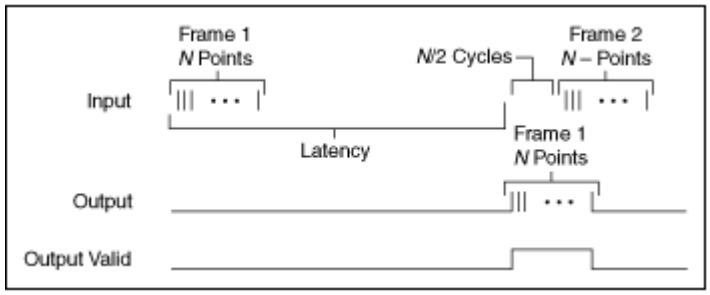

Figure 3.7: FFT implementation mode inside a single cycle timed loop [38].

This is how we utilize the FFT IP core for implementing the fast fourier transform algorithm on the FPGA.As the FPGA has limited resources, and really fast performance speed, the FFT IP core handshaking signals need to be used carefully to ensure smooth data transfer between the input and output FIFOs. Also, another fact to be considered is that the FFT block has a latency which depends on the length of the FFT, i.e., the latency increases with the FFT length. Therefore, while designing an application on the FPGA, the requirements need to be carefully considered beforehand for efficient implementation and execution.

Chapter 4

Surface Tracking

Peak detection is the way through which we detect the backscattered signal of interest and can extract the important information such as the range of the target and the two-way propagation time of the signal. But only peak detection is not enough as the aircraft on which the radar is deployed moves up and down and therefore gains or loses in altitude. In order to continuously acquire the signal, the snow radar currently uses the different Nyquist zones to store the spectral power in each band of interest. Therefore, as the range of the aircraft increases from the surface, the beat frequency also increases and the radar operator has to switch different Nyquist zones one by one so that the spectral energy is stored in different bands without encountering data loss. The next section describes the importance of surface tracking in the radar operation and what would be the benefits of implementing a real time surface tracking algorithm for the snow radar.

4.1 Surface Tracking Importance

Real time signal peak detection also requires continuous peak tracking which implies tracking the surface in real time. Peak detection detects echoes of the target in range against a background of noise. The role of a radar surface tracker would be to monitor consecutive updates from the radar system and to determine that the sequences of plots obtained are from the target that was initially being tracked. Surface tracking also has its own challenges as there is noise along with the signal of interest and there might be glitches in data acquisition, etc. Therefore, a robust surface tracking algorithm is required which would ensure that the radar tracks the ice surface precisely and makes the radar operation fully autonomous, i.e., no more manual Nyquist zone switching would be required by the radar operator.

Surface tracking involves a delayed dechirping operation, i.e. adjusting the delay of the reference chirp going to the local oscillator. By delaying the local oscillator reference chirp, the IF frequency would be constant as the range of the aircraft increases from the surface. Furthermore, the need of storing spectral energy in different bands would not be needed as the IF would not increase with respect to increasing range of the surface because the increase in the IF would be compensated by the delay fed to the local oscillator reference chirp signal.

4.2 Tracking Approach

There is no single fixed algorithm used by conventional radars for surface tracking. The algorithm depends on the kind of surface being tracked, the radar requirements, etc. In this thesis, the tracking approach followed was to analyze the beat frequency IF signal and delay the reference chirp with respect to the acquired IF of the echo from the target. A module was developed which generates a trigger for the transmit chirp, also called as the pulse repetition interval trigger. The PRI trigger is sent to the arbitrary waveform generator (AWG) which transmits the chirp signal. The module assumes a default beat frequency initially and transmits another trigger after a delay amounting to the propagation time calculated from the default initial beat frequency. The delayed trigger goes to the arbitrary waveform generator and triggers the AWG to transmit a reference chirp to the local oscillator. The transmit chirp is received by the receiver after a propagation delay equivalent to the range of the target and finally mixed with the reference chirp through a process known as dechirping. The resultant beat frequency signal is detected by the peak detector or the fast fourier transform algorithm implemented in the module. As the target range varies relative to the radar, the beat frequency of the signal changes, this change is compensated by the adding more delay (if the beat frequency of the signal is greater than the default assumed beat frequency) or reducing the delay (if the beat frequency of the signal is less than the default assumed beat frequency).Therefore, the local oscillator reference chirp trigger is adjusted in accordance to the acquired beat frequency at that

particular instant.

The tracking algorithm works in a way which tries to compensate the increase or decrease in the instantaneous beat frequency with respect to the default assumed beat frequency by adjusting the delay in transmitting the local oscillator reference chirp trigger. The algorithm tries to maintain a constant default beat frequency by adjusting the delay of the LO signal and thereby trying to achieve a constant range which eliminates the need to store the spectral power in different Nyquist zones. Figure 4.1 shows the current operation of the snow radar where the transmit and LO chirps are emitted at the same time, therefore, as range of the target increases, it increases the time delay of the received signal which ultimately increases the beat frequency. Figure 4.2 shows the proposed solution which works by delaying the LO reference chirp signal by which the beat frequency can be adjusted and made constant according the operating range. This way, the FMCW radar can work for increased ranges or altitudes at a constant beat frequency.

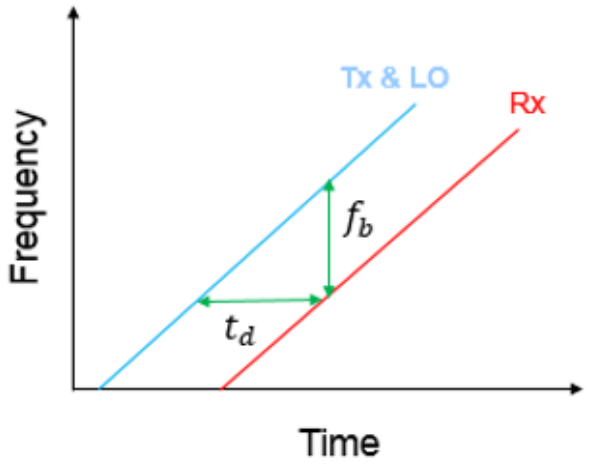

Figure 4.1 Current Radar Operation.

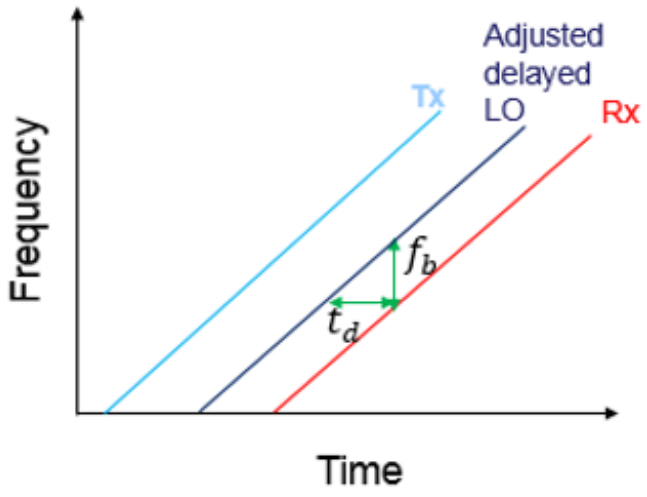

Figure 4.2 Operation with delayed LO reference chirp.

4.3 Implementing Surface Tracking on FPGA

Implementing a surface tracking algorithm real time has its own challenges as there are limited amount of hardware resources available. However, field programmable gate arrays provide the needed hardware resources required for efficiently implementing a surface tracking algorithm.

The algorithm was implemented using a single cycle timed loop (SCTL) and separate counters were used for generating the transmit trigger and the delayed LO trigger signal. The counters were used for calculation of the delay and adjusting and calculating the limit of the counter with respect to the beat frequency at a particular instant. Every operation occurring inside the field programmable gate array is synchronized with respect to the AWG external sampling clock. The steps mentioned below describe how the module was created explaining the requirements one by one:

- First, the Fast Fourier transform algorithm was implemented on the FPGA and all the handshaking signals were implemented for correct real time operation.
- The bin index containing the maximum value in an FFT output block represents the highest

signal energy or the beat frequency of the signal. Maximum value bin index was extracted from the FFT output.

- The logic to convert the bin index to beat frequency was implemented in the module.
- The beat frequency and the two way propagation time of the signal are related by the formula:

$$T = \frac{f_b T_S}{B} \quad \text{..... (4.1)}$$

Where T is the two way propagation time, T_S is the chirp sweep time and B is the chirp bandwidth.

- The propagation time is calculated from the beat frequency and is converted into the number of loop cycles of the single cycle timed loop needed for generating that particular delay with the help of a counter.
- A PRI trigger logic was implemented using a counter which sends a PRI signal continuously to the Arbitrary Waveform Generator in order to transmit a chirp signal.
- The local oscillator reference chirp trigger logic was implemented using counter that counts a fixed number of loop cycles to generate a delay with respect to the instantaneous beat frequency.
- The module is implemented in such a way that every time a PRI trigger is generated, the LO trigger is calculated using the instantaneous beat frequency obtained from the FFT algorithm. Therefore, the LO trigger is synced to the PRI trigger.

Changing the delay of the local oscillator reference chirp signal and adjusting it according to the increase or decrease in the range (altitude) dramatically increases the operating range of the radar. The idea of maintaining a constant beat frequency is achieved by increasing or decreasing the local oscillator chirp delay with the changing range (altitude) of the aircraft with respect to the reflecting surface. Maintaining a constant beat frequency eliminates the limit on the operating range of the radar.

This also eliminates the manual switching of the Nyquist zones which has to be done as the beat frequency increases with increasing range of the radar. Therefore, implementing surface detection and tracking for the snow radar equips the radar with the ability to operate autonomously dramatically increasing the operating range.

Chapter 5

Implementation and Results

5.1 System Information

The snow radar typically operates at 2-18 GHz frequency range. The waveform generator used to generate a chirp waveform is a high-speed Arbitrary Waveform Generator from a commercial vendor [39]. The Data Acquisition System used is NI PXIe-1075 chassis which has the following components:

- System Controller Slot (slot 1)
- Hybrid Peripheral Slots (8: slots 2-5 and slots 15-18)
- PXI Express Peripheral Slots (8: slots 6 to 9 and 11 to 14)
- System Timing Slot (slot 10)
- System Reference Clock

The PXIe-1075 chassis supplies PXI_CLK10, PXIe_CLK100 and PXIe_SYNC100 to every peripheral slot with an independent driver for each signal [40]. FlexRIO is the name of the product from NI which has user-programmable FPGAs along with high speed analog, digital as well as RF I/O. FlexRIO's can be programmed graphically by National Instruments graphical programming language LabVIEW. The architecture of FlexRIO is such that it has modular adapter modules that communicate with PXI FPGA modules over a parallel digital interface [41]. The FlexRIO digitizer module has a mezzanine IO Module which contains high-performance analog-to-digital converters (ADCs) and an FPGA backend for user defined signal processing.

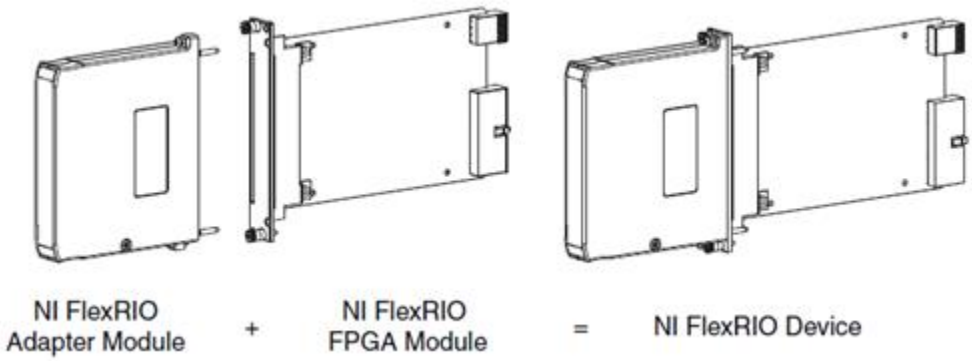

Figure 5.1: NI FlexRIO Device [42].

5.2 High Level Block Diagram

The process steps for surface detection and tracking can be understood by the high-level block diagram shown in figure 5.2. The process starts by giving an initial delay to the local oscillator transmit chirp signal and acquiring IF signal data samples and converting it into beat frequency by performing FFT. Once the instantaneous beat frequency is calculated, the number of loop cycles to be given to the Local oscillator delay trigger counter are calculated by the constant beat frequency loop and the delay is adjusted for transmitting the next LO chirp signal. The PRI Counter is fixed to a number of loop cycles needed to transmit the Tx chirp repeatedly after a fixed amount of time delay.

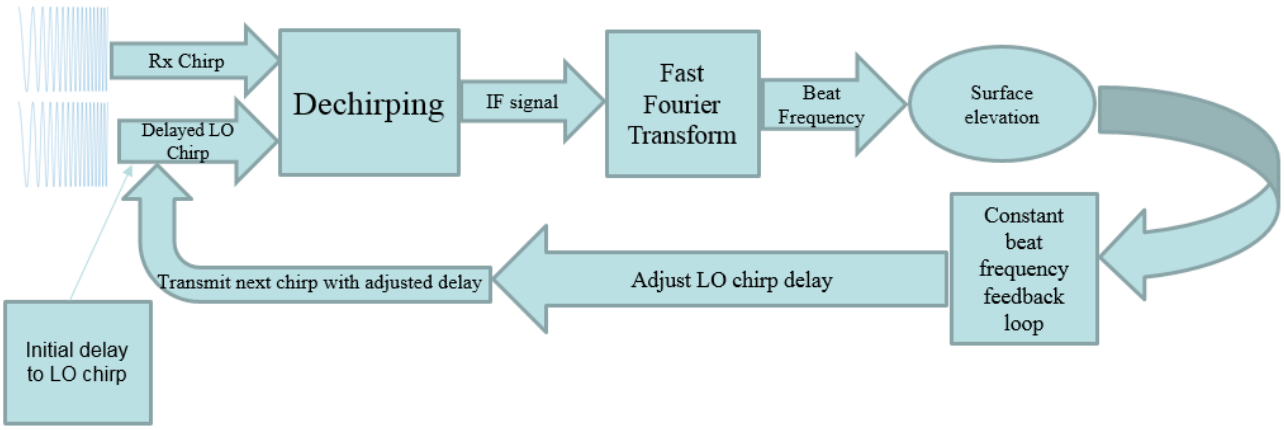

Figure 5.2: High level block diagram showing different processing steps.

5.3 Simulations on Host Computer

5.3.1 FFT Implementation on FPGA

The simulations were done on the FPGA target with simulated IO. For the simulations on host computer, a sinewave generator was simulated on the FPGA target with a frequency of 400 Hz. Keeping in mind the Nyquist sampling theorem which states that the sampling rate must be at least twice the highest frequency component of the signal or greater, the sampling frequency of 1000 Hz was chosen as shown in Figure 5.3.The sinewave generator is capable of generating sine or cosine waveform with user programmable parameters. The samples of the sinewave were stored in FPGA FIFOs (First In First Out).The inbuilt fast fourier transform block was used for performing FFT on the FlexRIO FPGA device. The signal samples stored in the FIFO were read and FFT was performed on them. The FFT block configuration is shown in Figure 5.4.Two types of FIFOs were used to store sine wave samples, one was the target to host direct memory access FIFO for directly transferring the sinewave samples from FPGA target to host computer and the other one was dedicated FPGA FIFO.

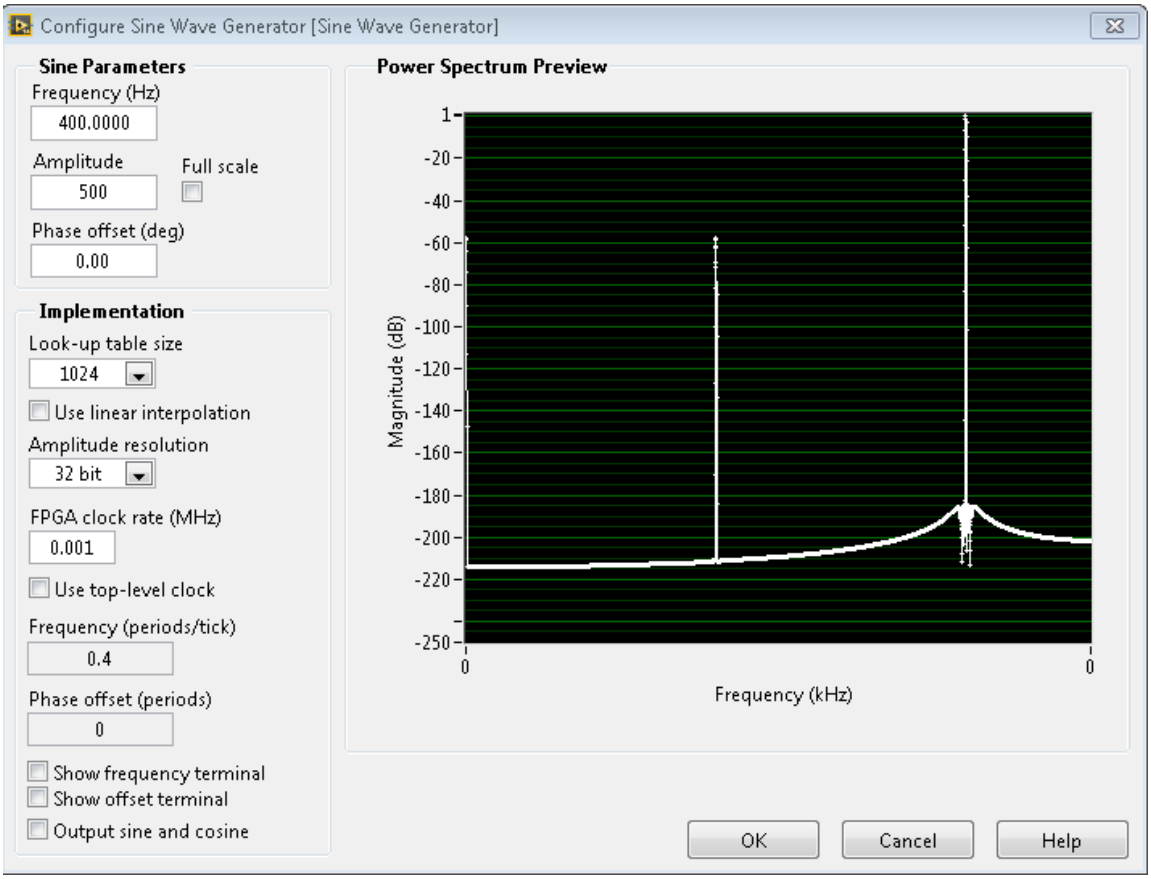

Figure 5.3: Sine Wave Generator [43].

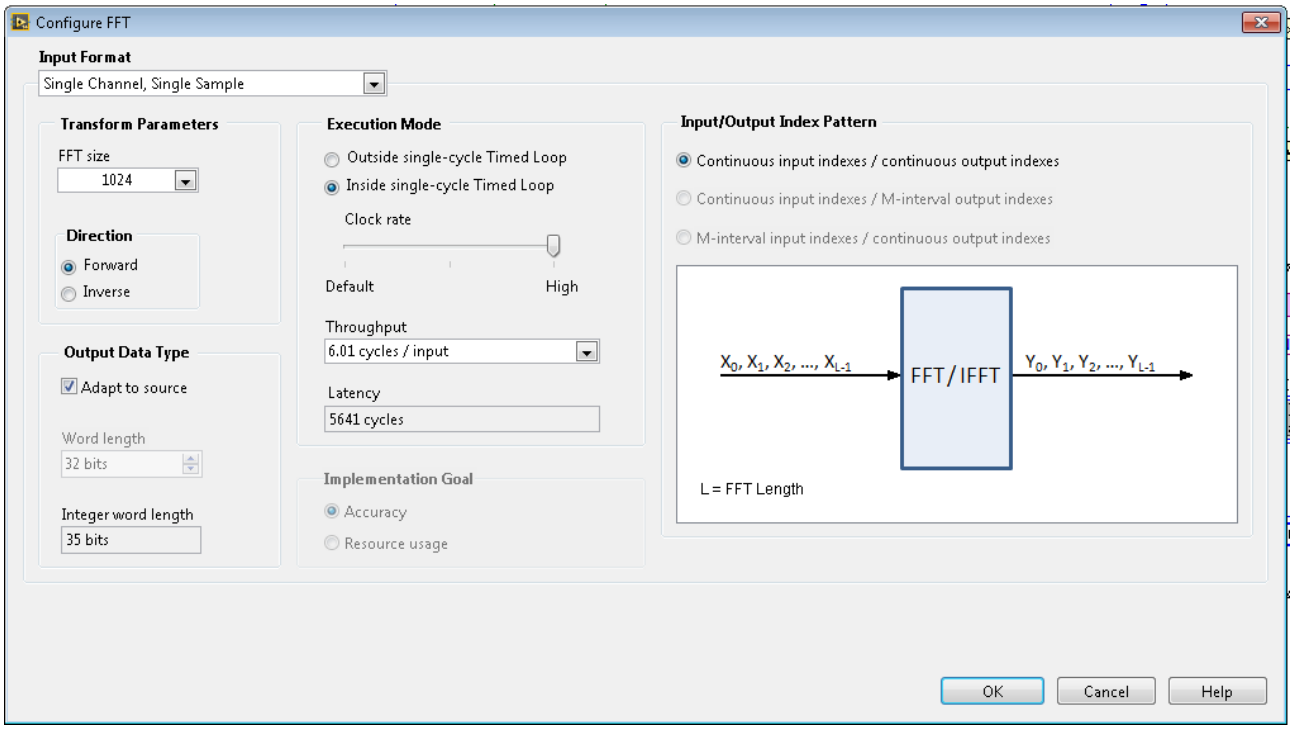

Figure 5.4: FFT IP configuration [44].

On the host computer virtual interface, two FFT plots were displayed, the first one was named the Direct FFT where the FFT was taken on the host computer and the other one was the FPGA FFT where hardware FFT was taken on the FPGA target and displayed on the host virtual interface. Figure 5.5 shows both the direct as well as FPGA fast fourier transform plots. From the plots it is evident that the 400 Hz sine wave signal is represented by a peak occurring at 400 Hz and a mirror image of the peak at 600 Hz respectively.

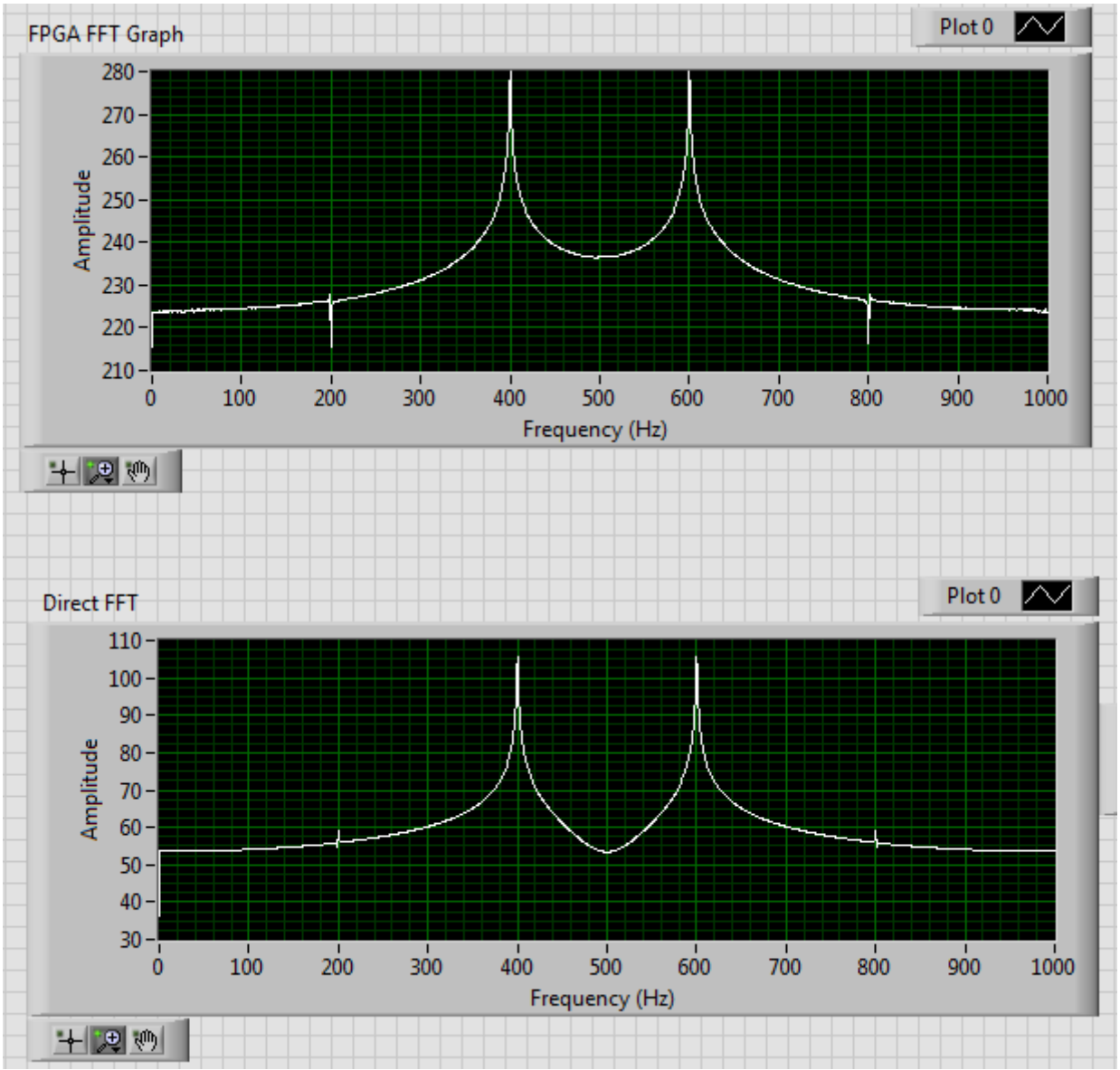

Figure 5.5: 2 plots displaying Direct FFT taken on host computer and FPGA FFT.

5.3.2 Surface Tracking Implementation

For implementing the surface tracking algorithm, a step by step procedure was adopted.

1. The data coming out of the FFT block was scanned for the maximum value. The maximum value was extracted and stored. Figure 5.6 shows the FFT implementation and extraction of maximum values from FFT output.

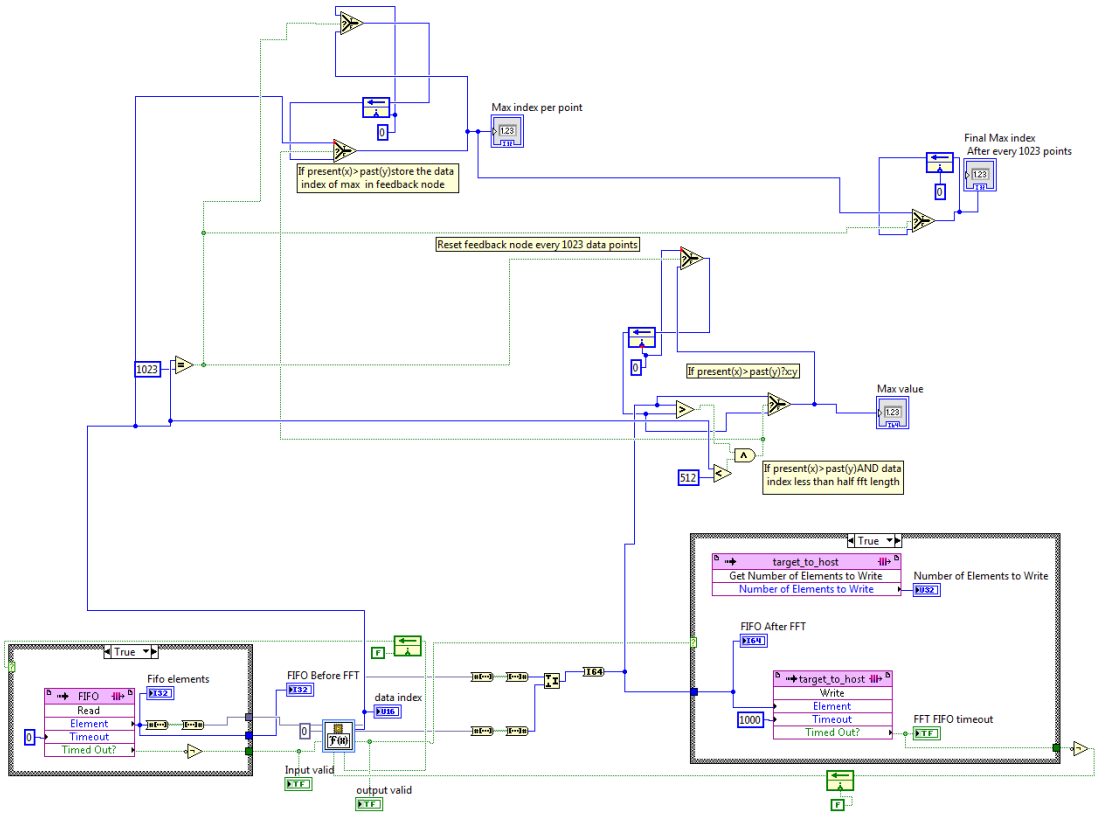

Figure 5.6: LabVIEW code for FFT implementation and extracting maximum FFT output value.

2. The data index corresponding to the maximum value was extracted and stored.
3. After all the samples for the FFT Length are acquired, the maximum data index represents the frequency of the input signal. Figure 5.7 shows the part of the LabVIEW code which stores maximum data index after every FFT output length (Assumed 1023 in the example).

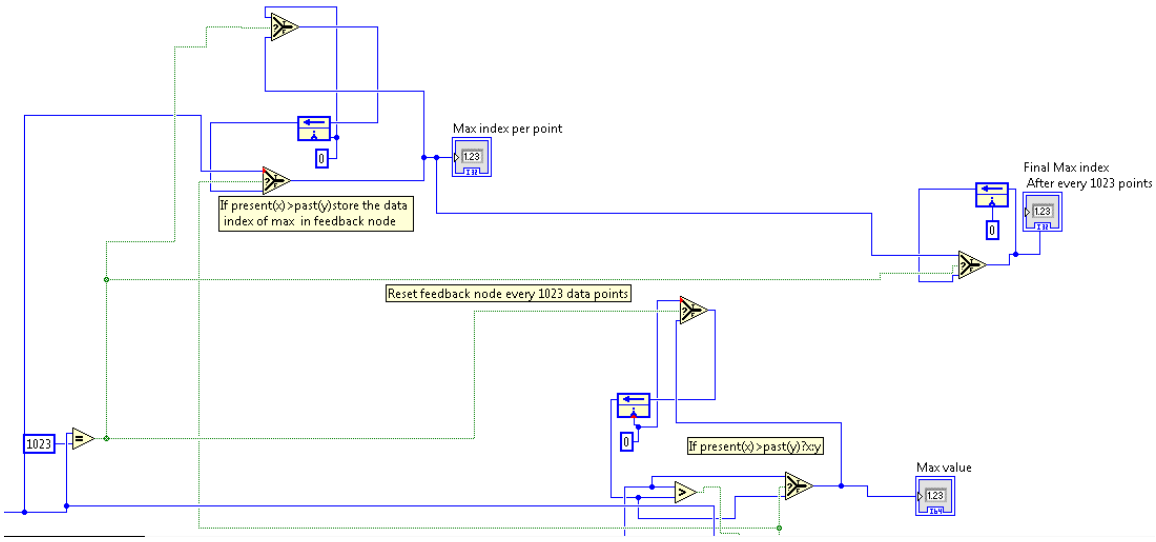

Figure 5.7: LabVIEW code for extracting maximum value every 1023 data indexes.

4. The data index is converted into beat frequency by the formula:

$$Beat\ Frequency(f_b) = \frac{Maximum\ Index * Sampling\ Frequency}{Length\ of\ FFT} \quad \text{..... (5.1)}$$

The LabVIEW code for this conversion is shown in figure 5.8.

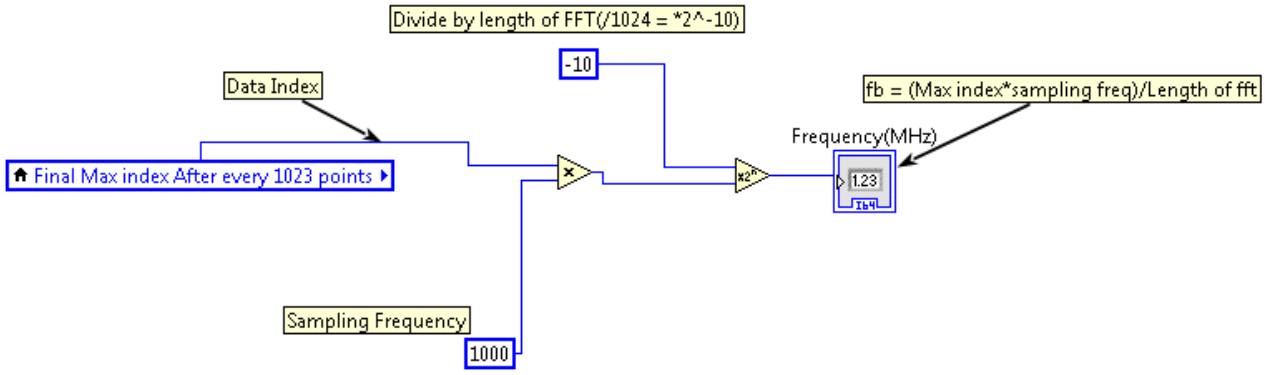

Figure 5.8: LabVIEW code which converts maximum data index into beat frequency.

5. Assuming that the beat frequency obtained is in MHz (which is typically the case with snow radar), the Time delay is obtained by the beat frequency by the following formula:

$$T = \frac{f_b * T_s}{B} \quad \text{..... (5.2)}$$

Where T is the Time delay (in nanosec), f_b is the beat frequency (MHz), T_s is the chirp sweep time (assumed 240 μsec) and B is the bandwidth of the chirp (assumed 16 GHz for a 2-18 GHz chirp).

6. Now we calculate how many loop cycles are required to generate the time delay calculated by equation 5.3. Assuming that the single cycle timed loop (SCTL) works at 250 MHz, a single loop cycle has a delay of 4 nano sec. Therefore in order to calculate the amount of loop cycles needed, we use the following formula:

$$Number\ of\ loops\ required\ to\ generate\ the\ delay = \frac{T}{1\ SCTL\ time\ delay} \ldots\ldots\ldots\ldots (5.3)$$

Where T is the Time delay calculated from equation 5.2.

7. After having calculated the loop cycles, they are fed to the local oscillator counter limit and determines the amount of local oscillator delay that needs to be given.
8. To maintain a constant beat frequency, a feedback loop is added. An initial constant delay is given to the local oscillator, the beat frequency corresponding to the initial delay and the range of the target is obtained and then is adjusted for the constant fixed beat frequency delay, which is converted to loop cycle count and becomes the next limit of the local oscillator delay trigger counter. Figure 5.9 shows the LabVIEW code for performing steps 5, 6, 7 and 8.

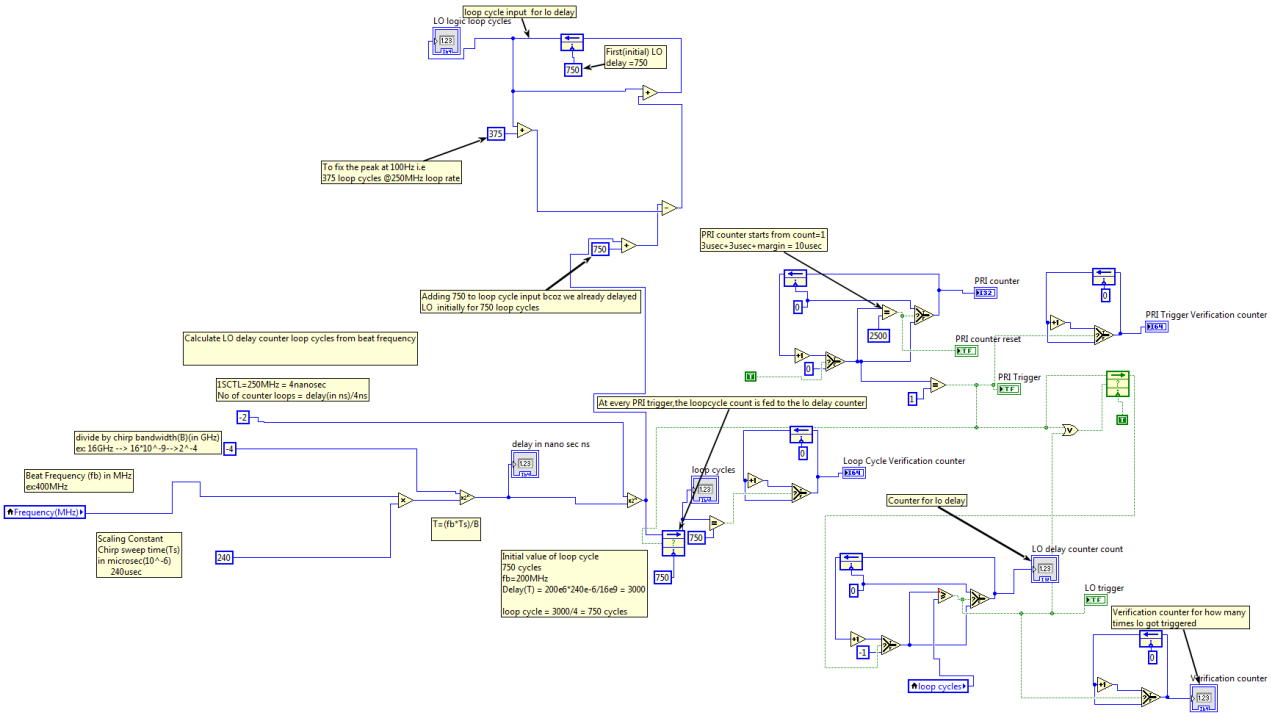

Figure 5.9: LabVIEW code for calculating the delay from the obtained beat frequency and calculating the corresponding loop cycles required.

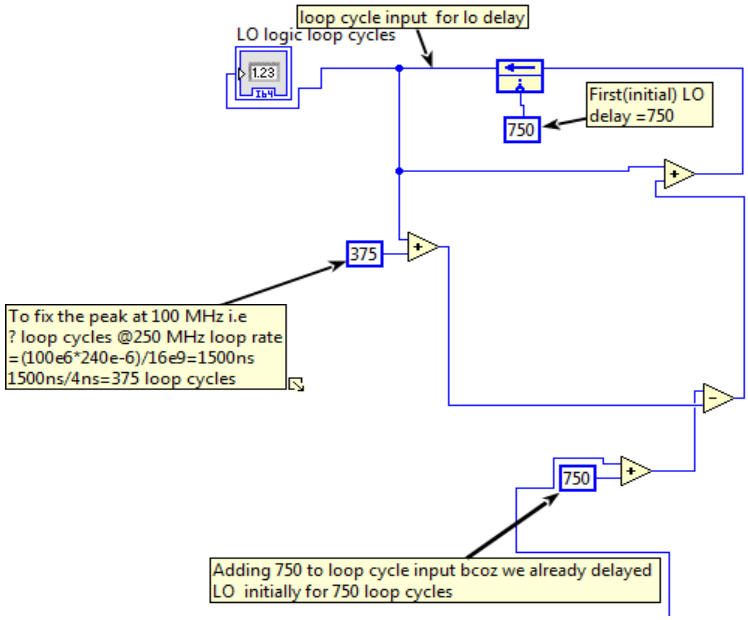

Figure 5.10: LabVIEW code for calculating the error required to fix the beat frequency at 350 loop cycles (100 MHz).

Figure 5.10 shows the constant beat frequency feedback loop which sends an initial delay of 750(Assuming the SCTL to be operating at 250 MHz, 750 loop cycles correspond to 200 MHz), then calculating the error obtained from the acquired beat frequency and calculating the loop cycles required to fix the beat frequency to 100 MHz (i.e. 350 loop cycles).

The working FPGA virtual interface is shown in Figure 5.11.The maximum index of the 1023 point FFT is extracted from the FFT output, i.e. 410^{th} as shown in the figure, the corresponding beat frequency is calculated from the maximum index, shown as 400 Hz. We assume it to be 400 MHz to mimic the real scenario. Having given the initial delay of 750 loop cycles that corresponds to a delay of 200 MHz, the beat frequency peak comes at 400 MHz, therefore 400 MHz corresponds to 600 MHz in real world. In order to keep the peak fixed at 100 MHz (which corresponds to a delay of 350 loop cycles), we have to give a LO delay corresponding to 500 MHz, therefore the number of loop cycles needed would be 375*5 that gives 1875 loop cycles as LO counter limit. This is exactly what is calculated in figure 5.11.

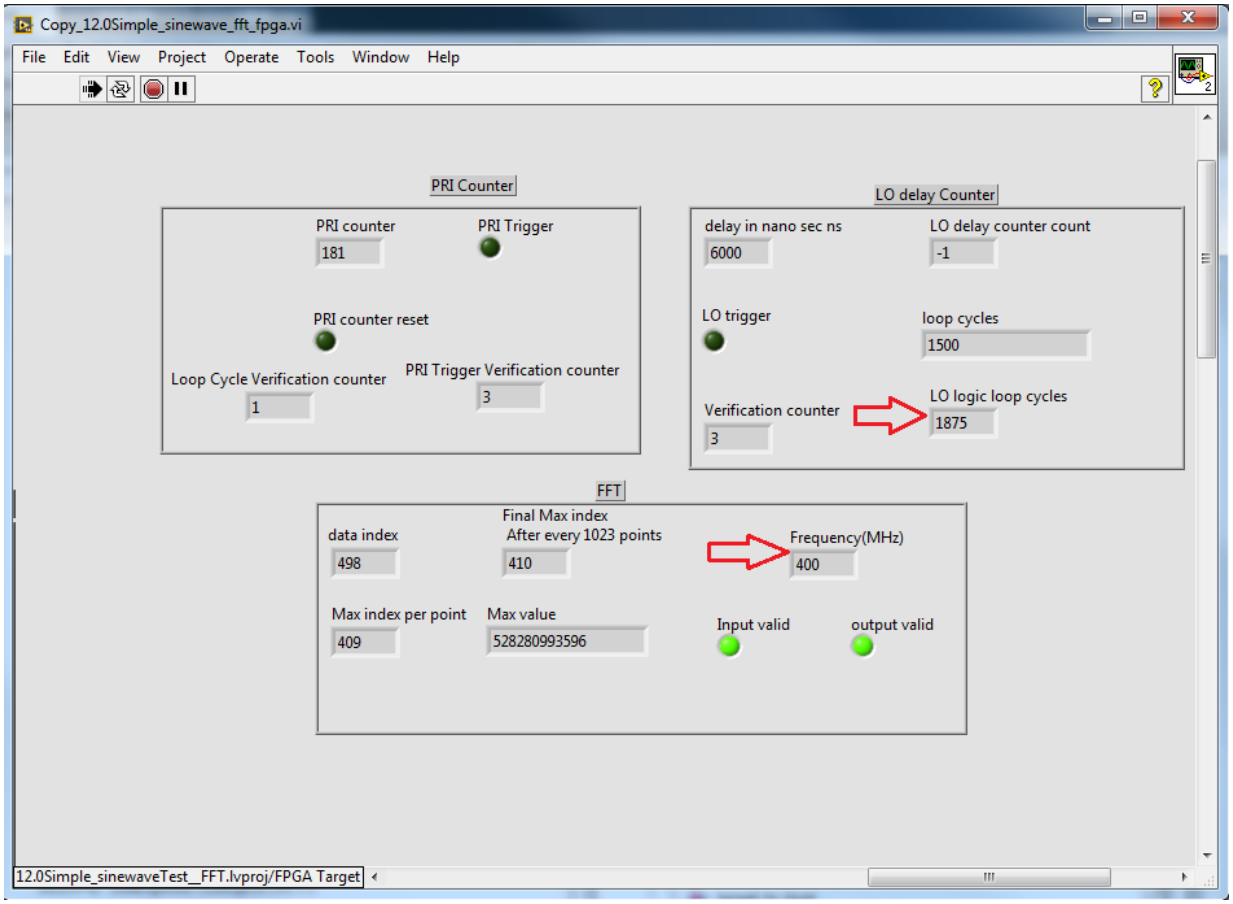

Figure 5.11: Beat frequency 400 MHz corresponding to a delay of 1875 loop cycles to achieve a constant beat frequency of 100 MHz, assuming an initial delay of 750(200 MHz) loop cycles.

Now we set the frequency of sinewave generator to 399 Hz with a sampling frequency of 1000 Hz, we get the maximum index at 409th data index of the 1023 point FFT output. We assume the frequency to be 399 MHz for creating a real world scenario. Giving the same initial delay of 750 loop cycles (which corresponds to 200 MHz), a 399 MHz beat frequency corresponds to 599 MHz in real world. In order to achieve a constant beat frequency of 100 MHz, we have to give a delay corresponding to 499 MHz, i.e. calculated as 375*4.99 which comes out to be 1871 rounding to the nearest integer. The result is verified from figure 5.12 where the loop cycles match the calculated number of loop cycles needed. We verified the fixed beat frequency logic with different beat frequency inputs and the results were as expected.

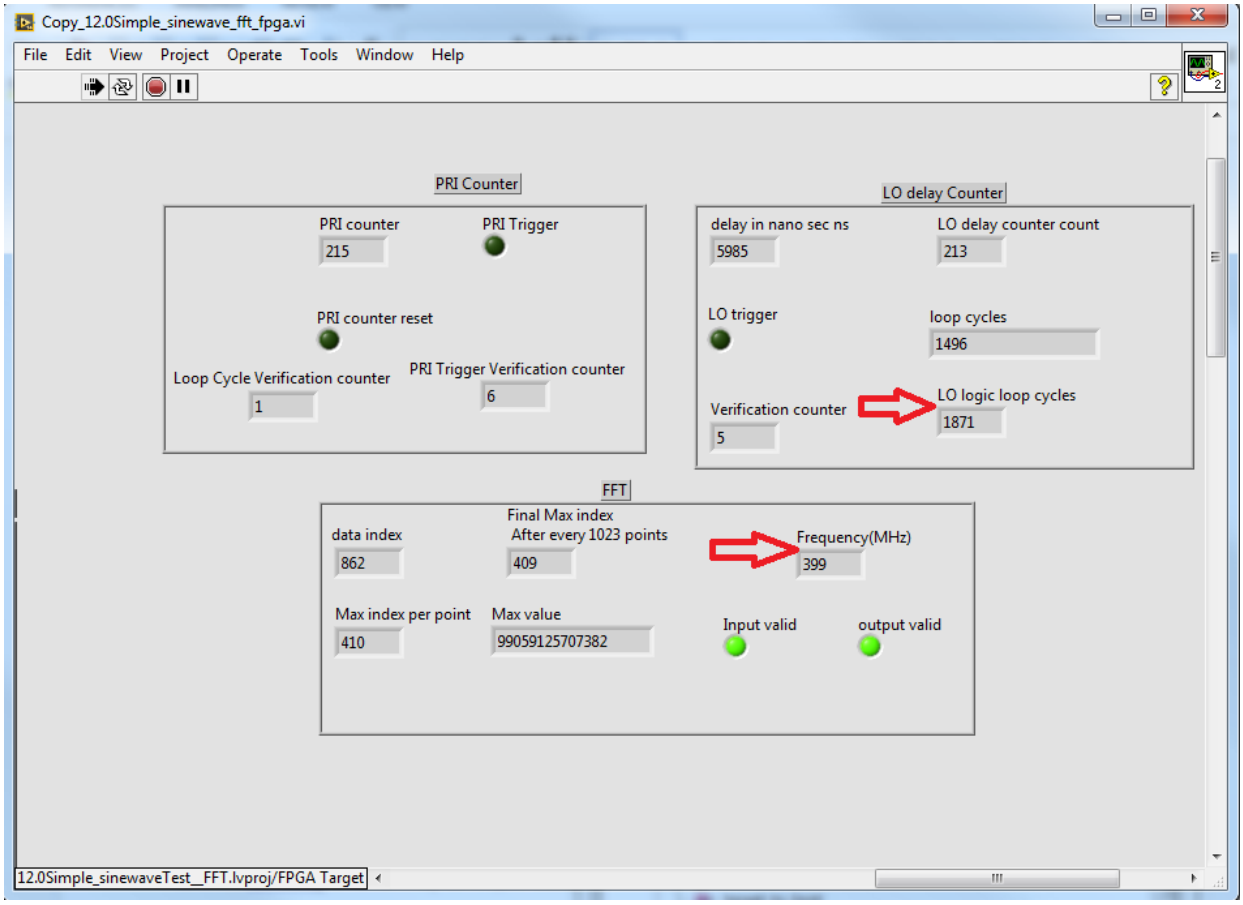

Figure 5.12: Beat frequency 399 MHz corresponding to a delay of 1871 loop cycles to achieve a constant beat frequency of 100 MHz, assuming an initial delay of 750 (200 MHz) loop cycles.

Figure 5.13 shows the counter logic created for the two counters namely the PRI Trigger counter and the LO delay trigger counter. The PRI trigger counter limit was fixed and the LO trigger counter limit was variable according to the fixed beat frequency feedback loop.

Therefore, this is how the simulation logic was created and implemented on the FPGA target through simulations on the host computer.

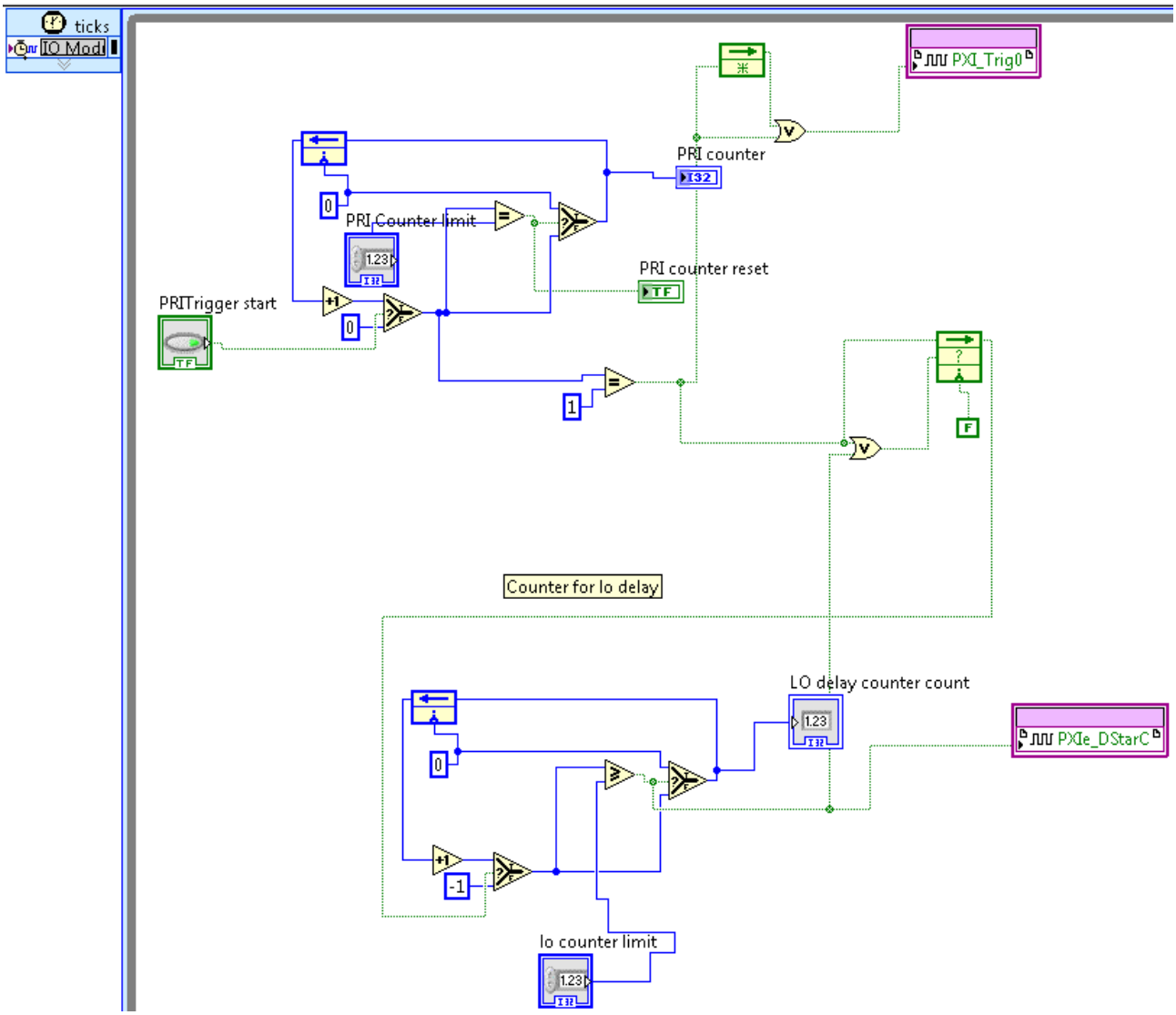

Figure 5.13: PRI Trigger counter and LO Trigger counter logic in LabVIEW.

5.4 Hardware Implementation on FPGA Target

5.4.1 Hardware Implementation Procedure

The LabVIEW FPGA module enables the developers to design their logic and translate the design directly to the hardware. After simulation with simulated I/O as explained in section 5.3, in order to compile the design on the FPGA target, the option shown in Figure 5.14 is selected.

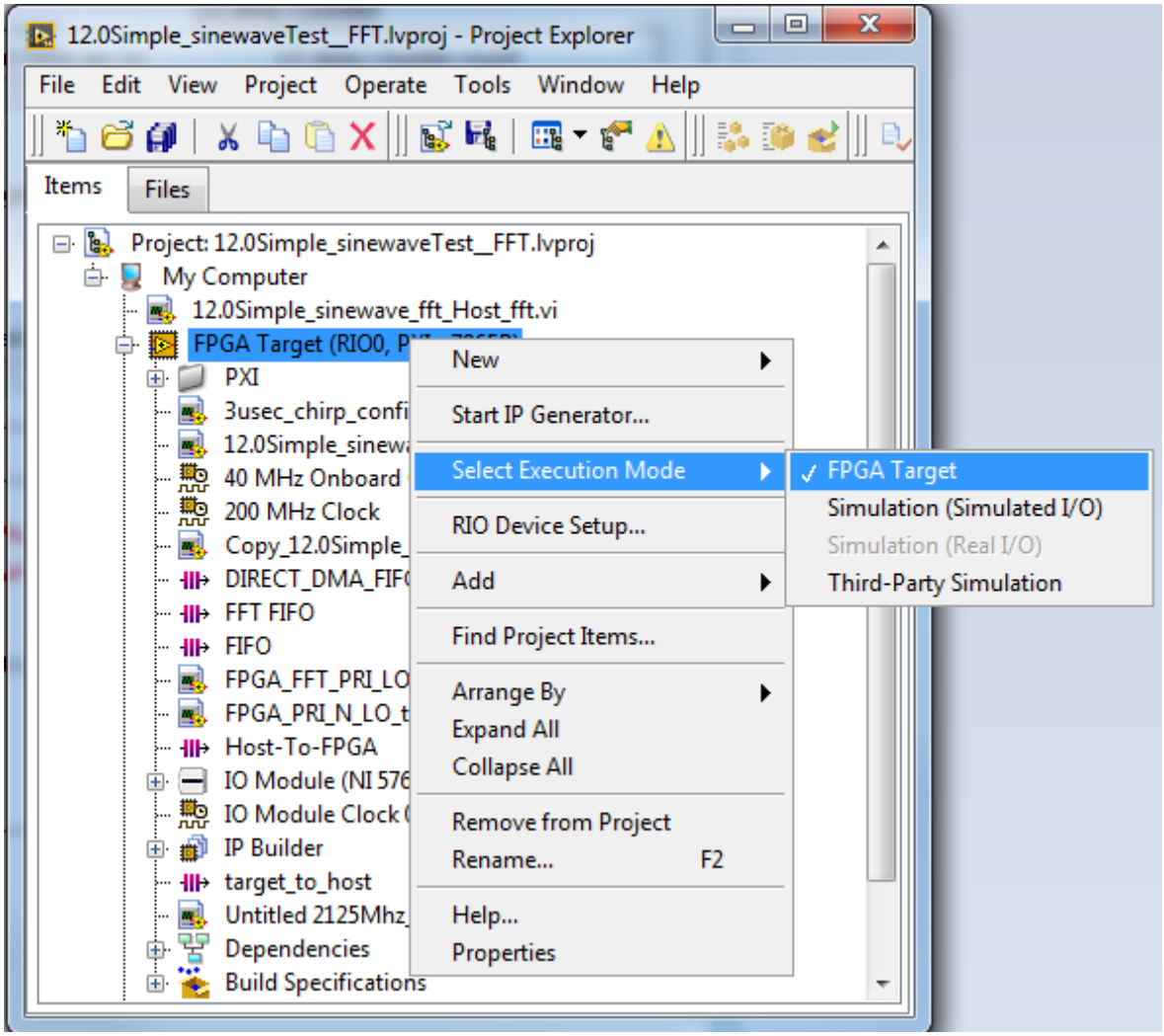

Figure 5.14: Changing the design execution mode to 'FPGA Target' for hardware compilation.

The result of the FPGA compilation is a bit file which is generated for the particular compiled design. Figure 5.15 lists various steps involved in the FPGA compilation process right from starting a compilation to the bit file generation.

Figure 5.15: FPGA Hardware compilation steps [45].

5.4.2 Artificial Target Simulation using Optical Delay Line

An optical delay line was used for simulation of an artificial target. The delay line was used to induce a propagation delay in the signal path. The chirp signal was generated and transmitted by the arbitrary waveform generator, travelling through the optical delay line and then coming back through the receiver and was mixed with the local oscillator chirp. Therefore, the optical delay line simulated a target situated at a range corresponding to the length of the delay line.

5.4.3 Implementation Results and Analysis

Initial tests were done on the mini snow radar, the fast fourier transform algorithm was implemented on the FlexRIO FPGA target, the FFT on the FPGA is resource intensive and the logic was adjusted to fit in a single cycle timed loop (SCTL). The FFT was performed on the presummed chirp data with 8 presums and a sampling frequency of 125 MHz using the NI Low Sample CLIP. Figure 5.16 shows the FFT implementation on host computer as well as on the FPGA target. Both the plots show definite peaks at approximately 43 MHz. After testing initially on mini snow, the implementation was done to the KU Snow Radar. The settings of the KU Snow and the results are listed in the next sections.

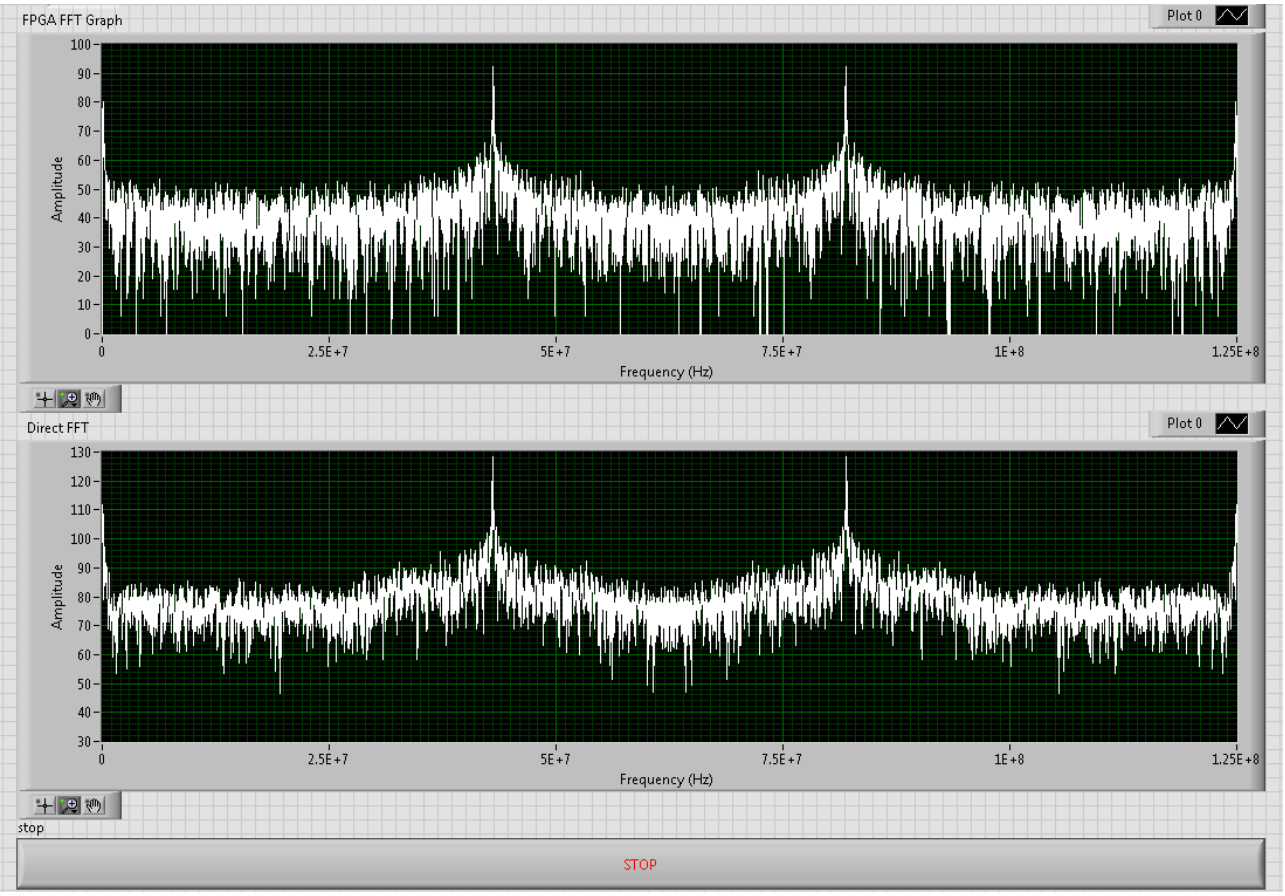

Figure 5.16: Direct and FPGA FFT for 240µsec, 2-18 GHz Chirp with 8 presums.

5.4.4 Arbitrary Waveform Generator Settings

A 3µsec, 2-18 GHz chirp was used as a transmit signal for the high-speed AWG. The AWG was used in the Dual Channel mode where the chirp was loaded in memory segments of Channel 1 and Channel 4. The AWG ports 'Trigger In' and 'Event In' were used for sending the PRI and LO triggers respectively as shown in Figure 5.17. NI Data Acquisition System was used to create PRI and LO counters which generated the two triggers.

Figure 5.17: High-speed AWG ports ‘Trigger In’ and ‘Event In’ were used for sending the PRI and LO triggers [38].

Two 3µsec chirps were loaded in the memory segments of Channel 1 and Channel 4 and the Trigger settings were made such that triggers at “Trigger In” and “Event In” triggered the transmission of the transmit and LO reference chirps respectively. The import waveform settings of the AWG are shown in figure 5.18 and the Trigger settings of used are shown in Figure 5.19 respectively.

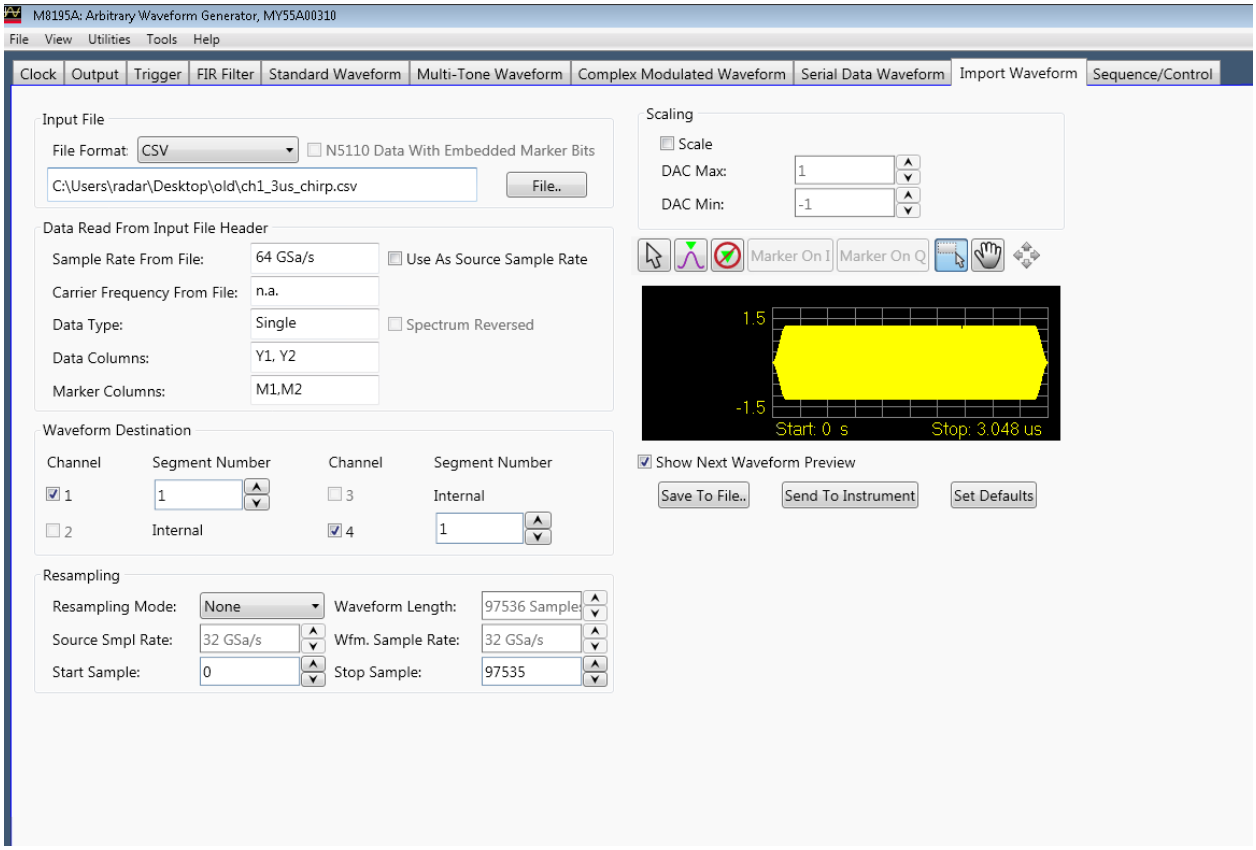

Figure 5.18: AWG import waveform settings [16].

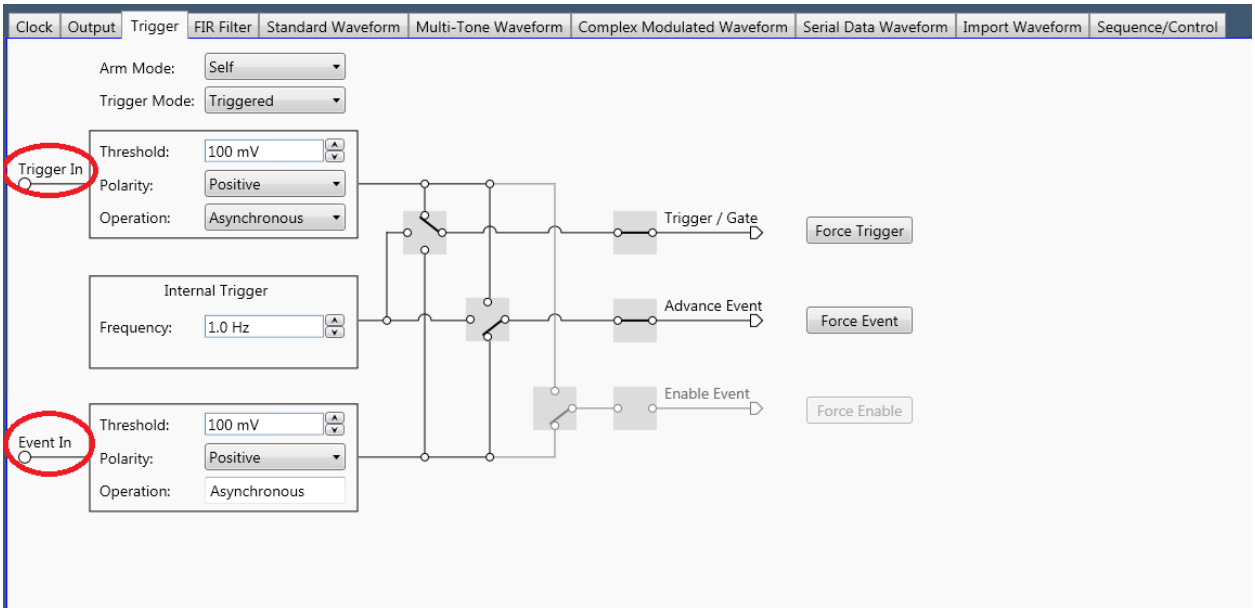

Figure 5.19: AWG Trigger settings [39].

5.4.5 Data Acquisition System Settings

The PRI and LO triggers were routed from the FlexRIO FPGA module through NI PXIe 1075 Backplane to PFI 2 and PFI 5 ports of the PXI Express Timing and Synchronization module NI PXIe-6674T.Figure 5.20 shows the connection settings of NI PXIe-6674T.The DAQ was synchronized with the AWG's 250 MHz external clock. The transmit chirp after getting transmitted from the AWG was fed to the optical delay line which simulated an artificial target at a given range. The delay from the delay line was 3.2µsec which is approximately 480 meters. The KU Snow radar is shown in figure 5.21. All the components of the KU Snow are listed in the figure including the high-speed AWG, the optical delay line used to simulate an artificial target through which the transmit chirp travels, the NI DAQ which sends the triggers to AWG and acquires the dechirped data and the TR chassis where the transmit and LO reference chirps are mixed generating the resultant beat frequency which is low pass

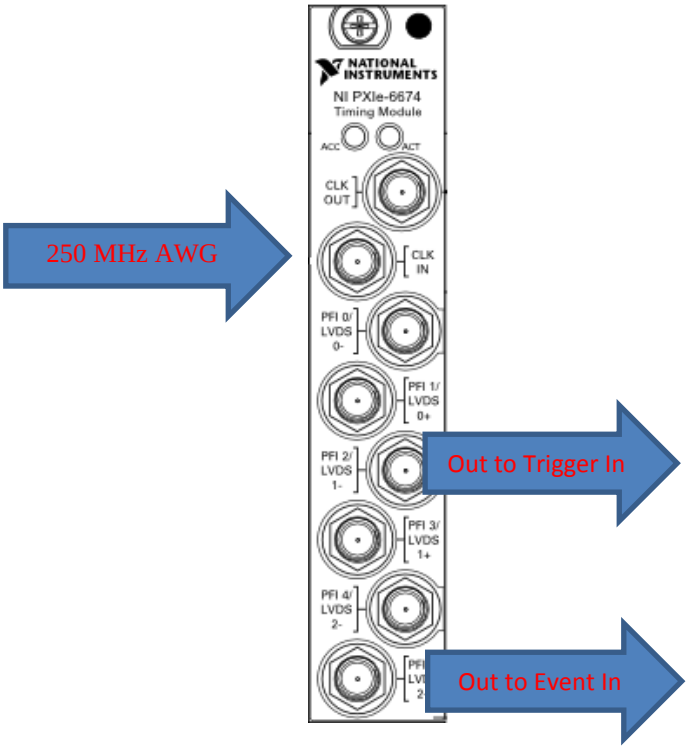

Figure 5.20: NI PXIe 6674T settings [46].

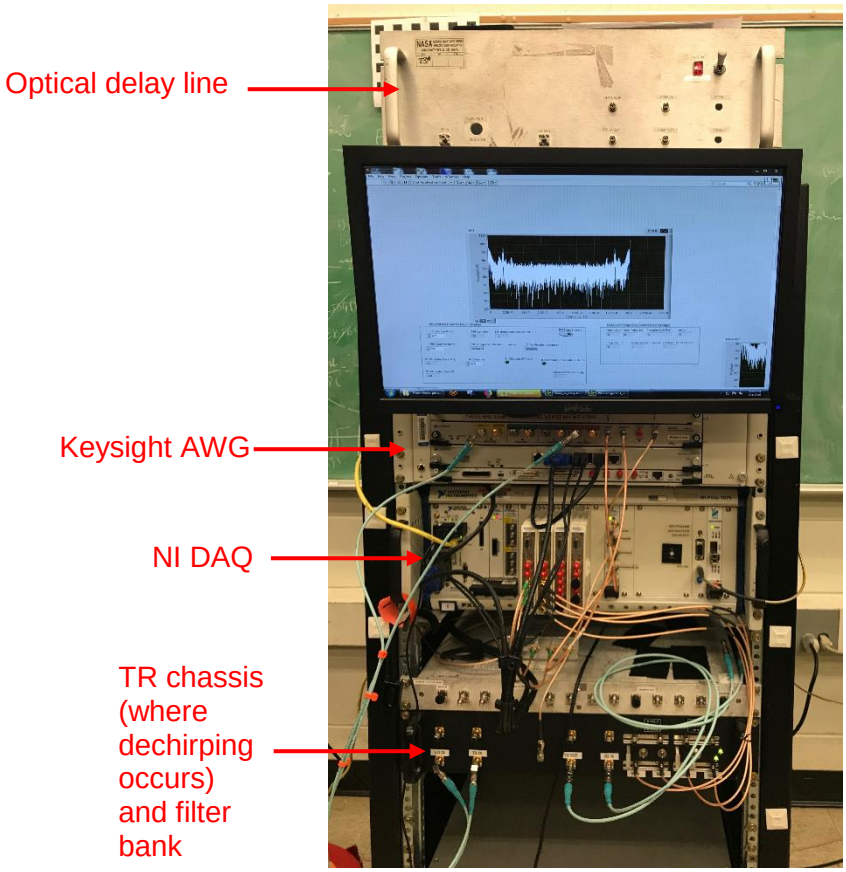

Figure 5.21: KU Snow Radar in operation.

filtered with a cutoff frequency of 125 MHz. The beat frequency signal samples were acquired by the NI Data Acquisition System. Figure 5.22 shows the connection diagram of the AWG with the NI data acquisition system.

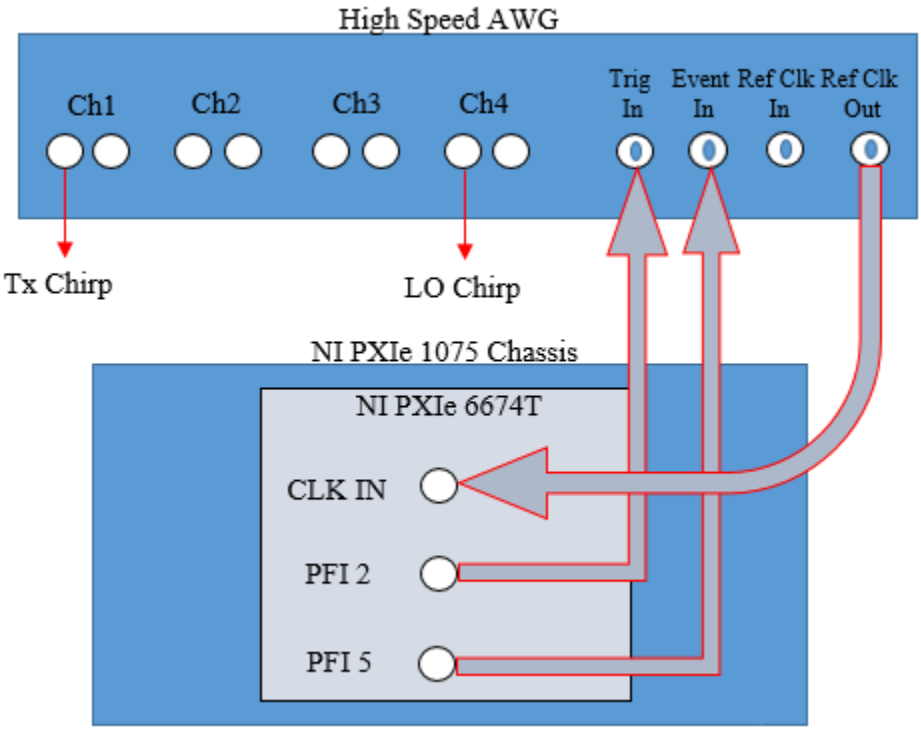

Figure 5.22: Connection diagram of the setup.

5.4.6 Hardware Implementation Results

The FlexRIO ADC generates the beat frequency signal samples at a rate of 250 MHz and the FPGA single cycle timed loop operates at the rate of 125 MHz using the multiple sample clip. Figure 5.23 shows the Data Acquisition LabVIEW virtual interface. The LO counter loop cycles are set to 409,i.e one single cycle timed loop takes 8 nano sec, thus the total delay after which the LO reference chirp is triggered is 409*8 = 3.272μsec. Considering the signal propagation delay, a beat frequency of 75 MHz is obtained at 409 loop cycles of LO counter. Figure 5.24 shows the IF signal obtained corresponding to the beat frequency of 75 MHz.

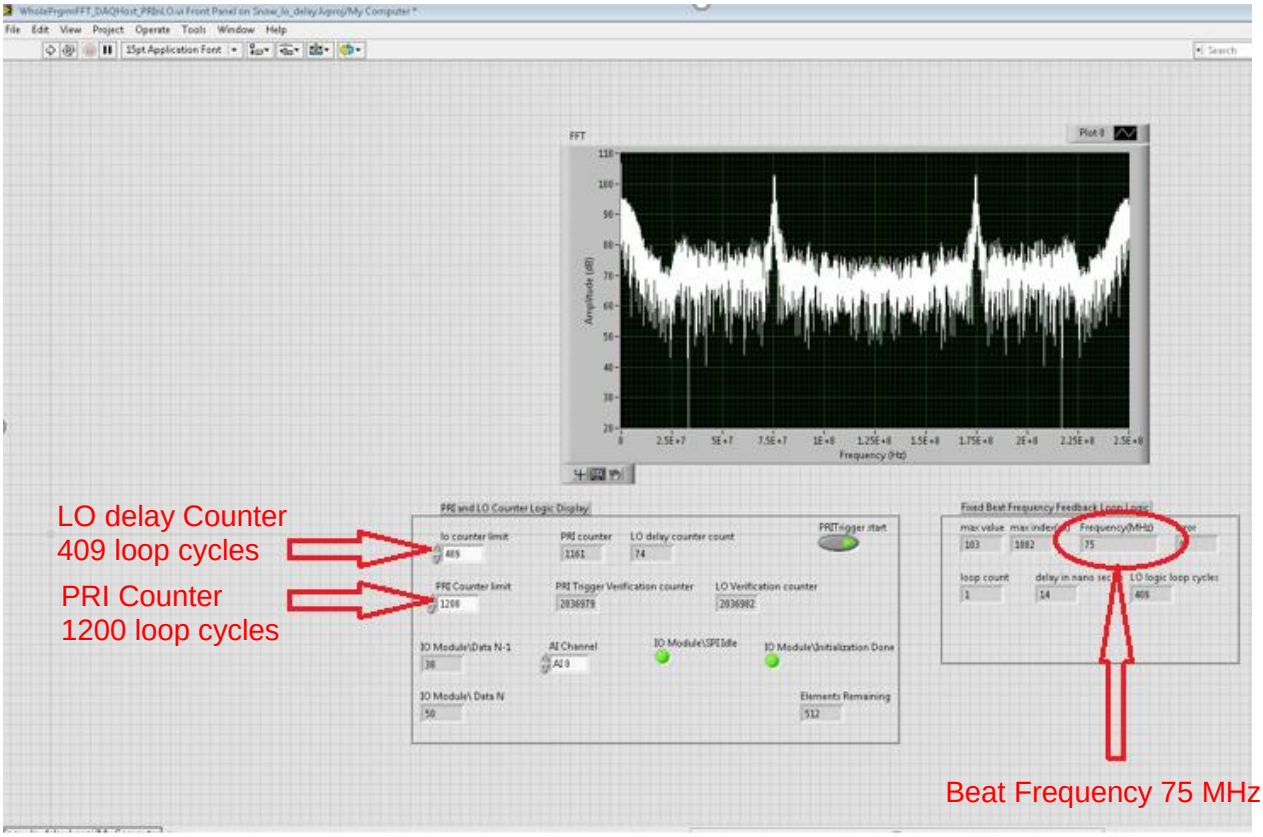

Figure 5.23: NI DAQ showing the beat frequency of 75 MHz at 409 LO loop cycles.

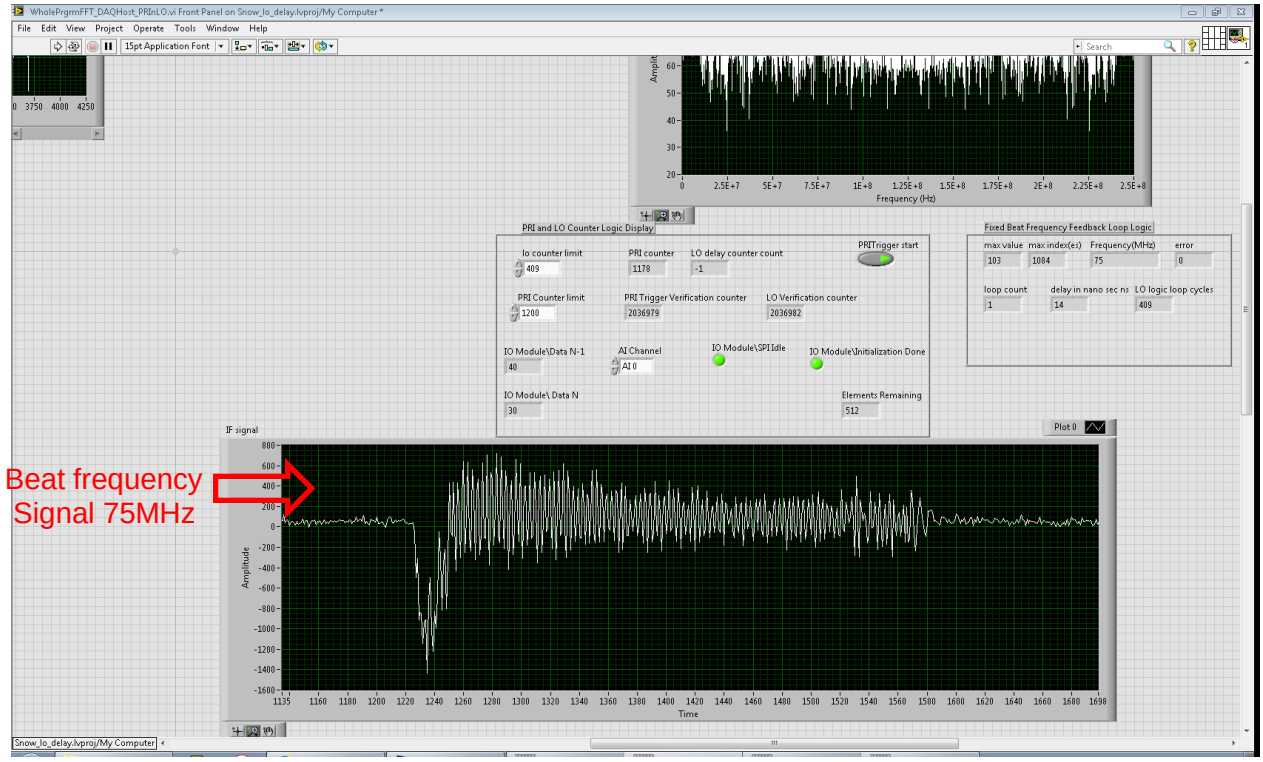

Figure 5.24: NI DAQ showing the beat frequency of 75 MHz.

The constant beat frequency feedback loop was configured such that it would keep the beat frequency constant at approximately 42 MHz. As the limit of LO delay counter is decreased from 409 to 408, the beat frequency becomes 116 MHz. Owing to the short length and high bandwidth of the chirp signal used in the experiment, 1 SCTL loop cycle corresponds to a beat frequency of approx. 42 MHz. Therefore, the decrease of 1 loop cycle in the LO logic loop count implies decreasing the time delay by 8 nano sec which corresponds to an increase in the beat frequency of 42 MHz i.e 116 MHz. Figure 5.25 shows the beat frequency increasing from 75 MHz to 116 MHz on decreasing the LO loop count to 408. As mentioned earlier, the feedback loop was configured to keep the beat frequency constant at 42 MHz, therefore as the beat frequency increases, the 'LO logic loop cycles' indicator value increases to 410 to compensate for increased beat frequency.

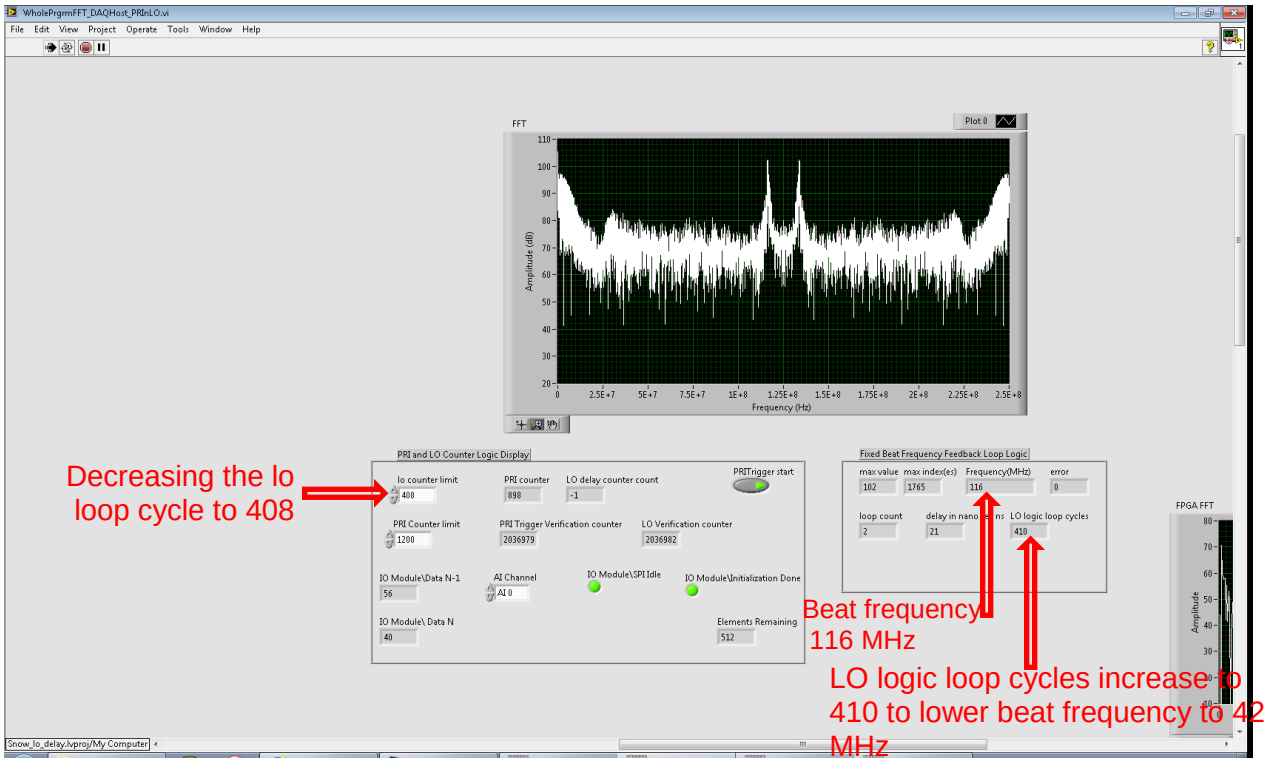

Figure 5.25: NI DAQ showing the beat frequency of 116 MHz at 408 LO loop cycles.

Moreover, following the same logic, increasing the LO counter loop cycle by one causes the beat frequency to decrease by approximately 42 MHz. In order to decrease the beat frequency, the LO loop counter limit is increased to 410 loop cycles. The beat frequency obtained is 33 MHz. The feedback loop wants to again keep the beat frequency constant at 42 MHz, therefore it decreases the LO logic loop cycles to 408.Figure 5.26 shows how by increasing the loop cycles to 410, decreases the beat frequency to 33 MHz and the feedback loop decreases the value of "LO logic loop cycles" to 408 to maintain a constant beat frequency of 42 MHz. The intermediate frequency (IF) signals are also shown in Figure 5.27 and 5.28 for beat frequencies 116 MHz and 34 MHz respectively.

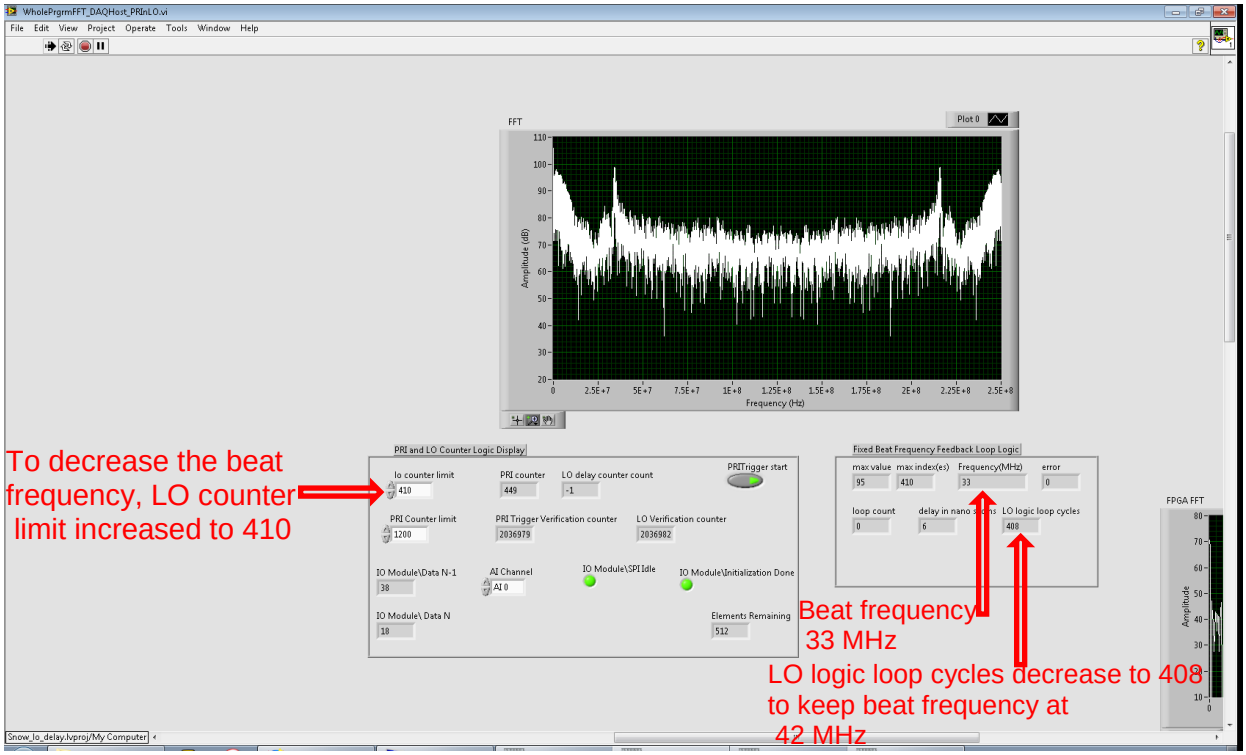

Figure 5.26: NI DAQ showing the beat frequency of 33 MHz at 410 LO loop cycles.

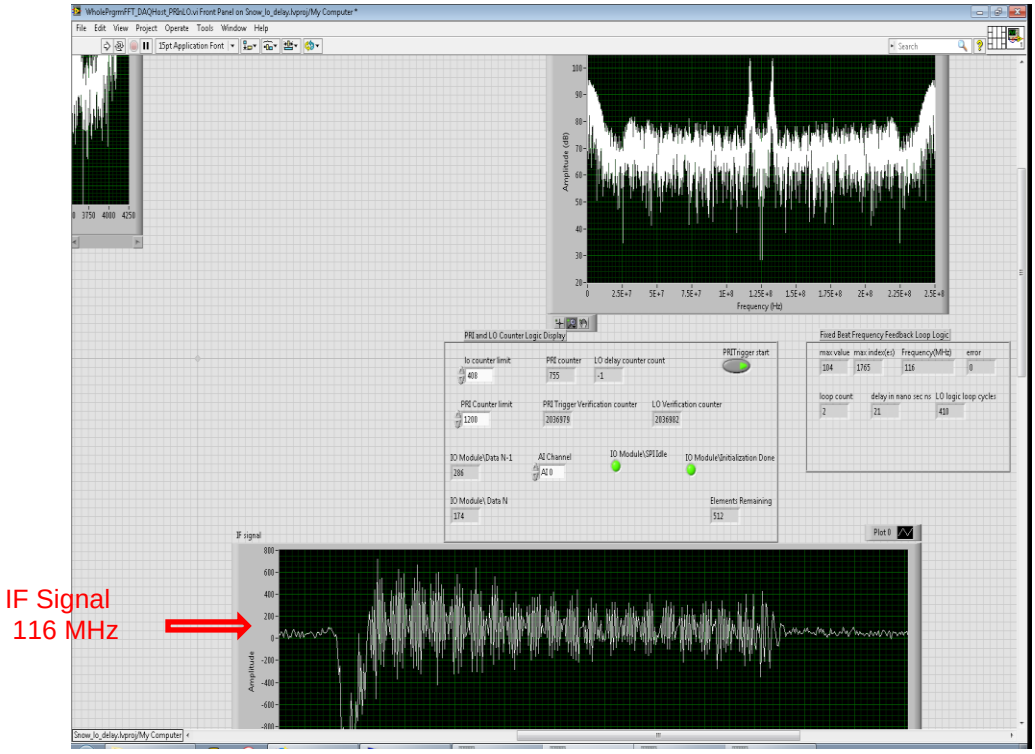

Figure 5.27: NI DAQ showing the IF signal of frequency 116 MHz.

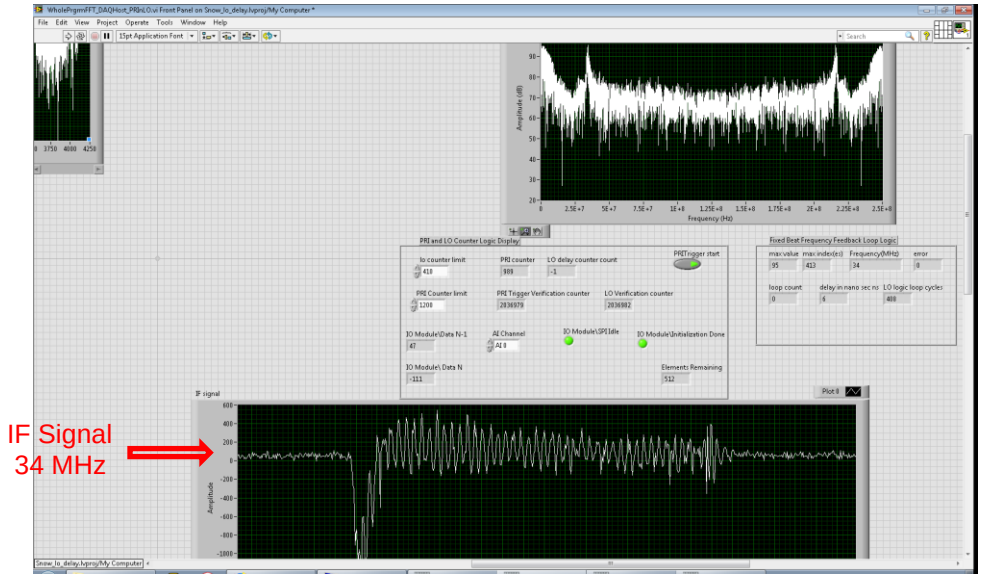

Figure 5.28: NI DAQ showing the IF signal of frequency 34 MHz.

This is how a constant beat frequency can be achieved by calculating the error between the current and the previous beat frequency information and translating them in terms of loop cycles and feeding it as the limit of the LO reference chirp delay counter. Therefore, by utilizing the fast

performance of the field programmable gate arrays, we can efficiently delay the LO reference chirp signal to obtain a constant beat frequency at any desired range.

Chapter 6

Conclusion and Future Work

6.1 Summary

All the test results shown in the previous chapter were done in the Lab with the snow radar. As mentioned in earlier chapters, implementing onboard real time surface tracking and detection for the KU Snow radar would dramatically increase its operating range and would also eliminate the need of using bandpass and high pass filters for storing the spectral power in different Nyquist zones. Moreover, once deployed, it would also eliminate the need for the radar operator to manually switch the Nyquist zones one by one as the altitude of the aircraft increases from the surface. This would make the radar function autonomously eliminating any chance of loss of crucial data due to delay in manual switching operation.

6.2 Future Work

For surface detection and tracking, a separate module was designed and coded. Future work includes integrating the logic of the model into the actual existing snow radar code and making it work for a longer chirp sweep time. In order to increase the signal to noise ratio, presumming of IF signal data samples needs to be implemented. The current snow radar configuration uses presumming to increase the signal to noise ratio (SNR) of the incoming IF signal, likewise during the integration of this module into the actual snow radar code, presumming would be needed.

Moreover, after integration, the logic would be required to be tested in actual field environment. Therefore, the radar needs to be deployed on test flights comprising of various different platforms to ensure that the radar is working as expected and the acquired data has to be field

processed as well as post processed to verify the correctness of the algorithm.

References

[1] Douglas, Bruce C., Michael S. Kearney, and Stephen P. Leatherman, “Sea Level Rise: History and Consequences”, International Geophysics Series Vol. 75, Academic Press, San Diego, 2001.

[2] National Ocean and Atmospheric Administration, United States Department of Commerce: https://oceanservice.noaa.gov/facts/sealevel.html

[3] Douglas, Bruce C., and W. Richard Peltier, “The puzzle of global sea-level rise,” Physics Today, pp. 35-40, March 2002.

[4] The union of concerned scientists (UCS), http://www.ucsusa.org/global_warming/science_and_impacts/impacts/causes-of-sea-level-rise.html#.WWMnPuvyvIU

[5] Matthew Sturm, Glen E. Liston, “The snow cover on lakes of the Arctic Coastal Plain of Alaska, U.S.A”, Journal of Glaciology, Vol. 49, No.166, 2003.

[6] Gary A. Maykut and Nobert Untersteiner, “Some results from a Time Dependent Thermodynamic Model of Sea Ice.”, Journal of Geophysical Research, Vol. 76, No.6, 1971.

[7] Thomas Newman , Sinead L. Farrell, Jacqueline Richter-Menge, Laurence N. Connor, Nathan T. Kurtz, Bruce C. Elder, David McAdoo, “Assessment of radar-derived snow depth over Arctic sea ice”, Journal of Geophysical Research: Oceans ,15 December,2014.

[8] Gary Koh,Norbert E. Yankielun, Ana I. Baptista “Snow cover characterization using Multiband FMCW radars”, Hydrological Processes, Vol. 10, 1996.

[9] Centre of Remote Sensing of Ice Sheets (CReSIS), https://www.cresis.ku.edu/content/about-us

[10] Ben Panzer, Carl Leuschen, Aqsa Patel, Thorsten Markus, Sivaprasad Gogineni ,“ Ultra-

wideband radar measurements of snow thickness over sea ice" , 2010 IEEE International Geoscience and Remote Sensing Symposium.

[11] Jie-Bang Yan, Daniel Gomez-García Alvestegui, Jay W. McDaniel, Yan Li, Sivaprasad Gogineni, Fernando Rodriguez-Morales, John Brozena, Carlton J. Leuschen ,"Ultrawideband FMCW Radar for Airborne Measurements of Snow Over Sea Ice and Land", IEEE Transactions on Geoscience and Remote Sensing (Volume: 55, Issue: 2, Feb. 2017).

[12] Ben panzer,1 Daniel Gomez-García, Carlton J. Leuschen, John Paden, Fernando Rodriguez-Morales, Aqsa Patel, Thorsten Markus, Benjamin Holt, Prasad Gogineni, "An ultra-wideband, microwave radar for measuring snow thickness on sea ice and mapping near-surface internal layers in polar firn", Journal of Glaciology, Vol. 59, No. 214, 2013.

[13] F. Rodriguez-Morales, C. Leuschen, A. Feathers, J. McDaniel, A. Wolf, & S. Garrison. "Packaging and Miniaturization of a 2–18 GHz UWB Radar for Measurements of Snow and Ice: Initial Results." In International Symposium on Microelectronics, 1st ed., Vol. 2017, pp. 36-39, 2017.

[14] F. Rodriguez-Morales, C. Leuschen, A. Feathers, J. McDaniel, A. Wolf, & S. Garrison, "Measurements of Snow Cover Using an Improved UWB 2-18 GHz Airborne Radar Testbed", Proc. 2018 IEEE Radar Conference, Oklahoma City, USA April 23-27, 2018.

[15] CReSIS Newsletters, https://www.cresis.ku.edu/content/news/newsletter/1294

[16] The CReSIS Snow radar user's manual.

[17] Xilinx product website, https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

[18] EDN website https://www.edn.com/design/test-and-measurement/4334609/Choosing-a-Real-Time-Platform-FPGAs-vs-Real-Time-Operating-Systems

[19] R. G. Strauch, W. C. Campbell, R. B. Chadwick, K. P. Moran," Microwave FM-CW Doppler radar for boundary layer probing", Geophysical Research Letters 3(3):193-196 · March 1976.

[20] Dirk Klugmann, "FMCW radar in the digital age", 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS).

[21] James W. Cooley and John W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation 19(90):297-301 · January 1965

[22] Dr. Christopher Allen's powerpoint slides, EECS 725 Introduction to radar systems.

[23] Extraction of range from beat frequency: https://www.slideshare.net/tobiasotto/principle-of-fmcw-radars.

[24] "History of FPGAs", https://web.archive.org/web/20070412183416/http://filebox.vt.edu/users/tmagin/history.htm

[25] FPGA architecture, http://microcontrollerslab.com/fpga-introduction-block-diagram/

[26] Dudley B. Chelton, Edward J. Walsh, and John L. MacArthur, "Pulse Compression and Sea Level Tracking in Satellite Altimetry", Journal of Atmospheric and Oceanic Technology, July 1989.

[27] Adaptive Linear Prediction, http://zone.ni.com/reference/en-XX/help/371988G-01/lvaftconcepts/aft_prediction/

[28] Kalman filter, https://en.wikipedia.org/wiki/Kalman_filter.

[29] Track smoothing, http://www.radartutorial.eu/10.processing/sp23.en.html

[30] Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter", Department of Computer Science University of North Carolina at Chapel Hill, NC.

[31] Ellen J. Ferraro, and Calvin T. Swift, "Comparison of Retracking Algorithms Using Airborne Radar and Laser Altimeter Measurements of the Greenland Ice Sheet", IEEE Transactions on Geoscience and Remote Sensing; p. 700-707; (ISSN 0196-2892); Volume 33; No. 3.

[32] Zhisen Wang, Yunhua Zhang, Jingshan Jiang, "A robust tracking system for imaging radar

altimeter", 2000 5th International Symposium on Antennas, Propagation, and EM Theory.

[33] Frequency domain theory and applications, http://www.numerix-dsp.com/tutorials/DSP/FrequencyDomainProcessing.pdf

[34] Dr. Glenn Prescott's lecture slides, Introduction to digital signal processing.

[35] "Make your Measurements Faster with FPGA Technology", http://www.ni.com/newsletter/51513/en/.

[36] "Tip: FFTs in LabVIEW FPGA", https://www.embedded.com/print/4017633

[37] FFT IP Core in LabVIEW FPGA, https://www.eetimes.com/document.asp?doc_id=1275552

[38] FFT Express VI LabVIEW Help.

[39] Keysight M8195A Arbitrary Waveform Generator user manual.

[40] NI PXIe-1075 user manual," http://www.ni.com/pdf/manuals/372537c.pdf".

[41] FlexRIO from National Instruments, http://www.ni.com/pdf/product-flyers/flexrio-custom-instrumentation.pdf.

[42] NI 5761 user manual, http://www.ni.com/pdf/manuals/375509a.pdf

[43] LabVIEW FPGA, Configure sinewave generator.

[44] LabVIEW FPGA, Express FPGA FFT configuration.

[45] LabVIEW FPGA Compilation Process: From Run Button to Bitfile, http://www.ni.com/white-paper/9381/en/.

[46] NI PXIe 6674T User Manual, http://www.ni.com/pdf/manuals/

Appendices

Appendix A

MATLAB CODE FOR SIMULATION

Matlab File 1: *fmcw_simulation.m*

```
clear all;
close all;
%-------------------------------------------------------------------------%
% Parameter initialization
%-------------------------------------------------------------------------%
c = 3*10^8;
fc=77e9;
lambda=c/fc;
r_max=200;
sweep_time=5.5*2*r_max/c; % 10 times the max range
r_res=1;
B_width= c/(2*r_res); % BW calculated using desired range res
fs=2*B_width; % sampling fre
t=0:1/fs:sweep_time- 1/fs; % sweep time vector
k = B_width./sweep_time;
%%
%-------------------------------------------------------------------------%
% Generating a chirp
%-------------------------------------------------------------------------%
ch =exp(1j*(2*pi*(fc*(t) + 0.5*k*(t).^2)));
%%
%-------------------------------------------------------------------------%
% Generate the scene
%-------------------------------------------------------------------------%
tar_range = [100 450] ; % two target ranges in scene
radial_vel =[0 50]; % m/s taking v large value for memory constraint
RCS_targets = [1 100] ; % RCS of targets
sim_slow_time =100*sweep_time; % slow time in s, total duration of the scene
slow_time_samples = sim_slow_time /sweep_time ;
slow_time = linspace(0, sim_slow_time, slow_time_samples);
[Rx_matrix] = generate_scene(tar_range, radial_vel, slow_time, ch, t,
RCS_targets, lambda, fs, B_width,fc,k);
%%
%-------------------------------------------------------------------------%
% Simulating mixer operation
%-------------------------------------------------------------------------%
nfft = 2^nextpow2(length(ch));
freq_axis=0:(fs/(nfft)):((fs/nfft)*nfft)-1;
range=(freq_axis*sweep_time*c)/(2*B_width);
rec_sig = zeros(nfft, slow_time_samples);
h_b = hann(size(Rx_matrix, 1));
for ii = 1: slow_time_samples
m=dechirp(Rx_matrix(:,ii), ch.');
rec_sig(:,ii) = fft(m.*h_b, nfft);
% rec_sig(:,ii) = fft(m, nfft);
```

```
end
%%
%-------------------------------------------------------------------------%
% Doppler frequency computation
%-------------------------------------------------------------------------%
nfft2 =256;
fs_slow_time = 1/(slow_time(2)- slow_time(1));
f_dopp= 0 :fs_slow_time/ nfft2: fs_slow_time * (nfft2 -1)/nfft2;
rng_dopp = zeros (nfft, nfft2);
h_D = hann(size(Rx_matrix, 2));
for ii = 1: length (range)
ss= rec_sig(ii,:).*h_D.';
rng_dopp(ii, : ) = fftshift(fft(ss, nfft2));
end
%%
%-------------------------------------------------------------------------%
% Generate plots
%-------------------------------------------------------------------------%
figure (1);
k1=20*log10(abs(rec_sig(:,10)./max(rec_sig(:,10))));
plot(1e-6* freq_axis(1:length(freq_axis)/2),k1(1:length(k1)/2));
ylabel('Normalized Power(dB)')
xlabel ('Frequency (MHz)')
title ('Beat frequency spectrum ( 0 to fs/2)')
ylim([-50 0])
figure (2);
k1=20*log10(abs(rec_sig(:,10)./max(rec_sig(:,10))));
plot(range(1:length(k1)/2),k1(1:length(k1)/2));
ylabel('Normalized Power(dB)')
xlabel ('Range (m)')
title ('Range Ascope ')
ylim([-50 0])
xlim([0 600])
figure (3);
imagesc(slow_time, range, db(rec_sig./max(max(rec_sig))));
xlabel('Slow time (s)')
ylabel('Range (m)')
title('Range - slow time domain')
ylim([0 600])
caxis([-50 0])
figure(4)
imagesc( lambda*(f_dopp - fs_slow_time/2)/2, range,
db(rng_dopp./max(max(rng_dopp))))
ylabel('Range(m)')
xlabel ('Velocity (m/s) ')
title ('Range- Doppler Spectrum ')
caxis ([-50 0])
ylim ([0 600])
% xlim([-500 500])
```

Matlab File 2: *generate_scene.m*

```
function [Rx_matrix] = generate_scene(tar_range, radial_vel, slow_time, ch, t,
RCS_targets, lambda, fs, B_width, fc,kk)
%%
c= 3e8;
Rx_matrix = zeros (length (ch), length(slow_time));
R0= tar_range;
v= radial_vel;
for i=1:length(slow_time)
%-------------------------------------------------------------------------%
% code to include all the targets
%-------------------------------------------------------------------------%
for jj= 1: length(tar_range)
R = R0(jj) + v(jj).*t; % time-dep range to tgt (s)
T0 = 2.*R0(jj)./c; % rnd-trip travel time at t = 0 (s)
T = 2.*R./c;
y= zeros(length(ch), 1);
y= exp(1j*(2*pi*(fc*(t- T) + 0.5*kk*(t- T).^2))).*((t-T) >0);
scaling_fact= calculate_return_power (RCS_targets(jj), tar_range(jj), lambda,
ch);
Rx_matrix(:,i)= Rx_matrix(:,i) + (y*scaling_fact).' ; % superimpose the return
R0(jj)= R(end);
end
%-------------------------------------------------------------------------%
% simulate noise - receiver noise and propogation loss
%-------------------------------------------------
---------------------%
k=1.38e-23;
T=290;
F=2;
sigma = k*T*B_width*F ;
rec_noise=sigma*randn(length(ch),1);
%loss_fac = (4*pi*mean(tar_range)/lambda)^2;
prop_noise = 50e-7*randn(length(ch),1);
%-------------------------------------------------------------------------%
% % add uncorrelated noise
%-------------------------------------------------------------------------%
Rx_matrix(:, i) = Rx_matrix (:, i) + rec_noise + prop_noise;
% update the position for the next slow time sample
% tar_range = tar_range + radial_vel.* (slow_time(2)- slow_time(1));
end
end
```

Matlab File 3: *calculate_return_power.m*

```
%-------------------------------------------------------------------------%
% calculating return power
%-------------------------------------------------------------------------%
function scaling_fact= calculate_return_power (RCS_target, range, lambda, ch)
antenna_aperture=6.06*10^-4;
Gain_antenna=(4*pi*antenna_aperture)/lambda^2;
Pt_rms=rms(ch)^2; %Transmit power in watts
Pr_rms=((Pt_rms*(Gain_antenna^2)*(lambda^2)*RCS_target))/((4*pi)^3*(range)^4);
fprintf('\n Power return for target at range = %d is %d W', range, Pr_rms);
scaling_fact=sqrt(Pr_rms*2)/sqrt(Pt_rms*2);
end
```