ANALYSIS AND IMPLEMENTATION OF SIGNAL PROCESSING STRATEGIES FOR A 3-D DOPPLER LIDAR WIND PROFILER

by:

SAMEH ABDELAZIM

A dissertation submitted to the Graduate Faculty in Electrical Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2012

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Prof. Samir Ahmed

_____ _____

Date Chair of Examining Committee

Prof. Mumtaz K. Kassir

_____ _____

Date Executive Officer

Prof. Fred Moshary

_____

Prof. Mark Arend

_____

Prof. Barry Gross

_____

Dr. Dimitrios Kokkinos

_____

Dr. Robert C. Stirbl

_____

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

ANALYSIS AND IMPLEMENTATION OF SIGNAL PROCESSING STRATEGIES FOR A 3-D DOPPLER LIDAR WIND PROFILER

By:

SAMEH ABDELAZIM

Adviser: Professor Sam Ahmed

Co-advisor: Professor Fred Moshary

A heterodyne detection fiber optic based wind lidar system has been developed and tested, which benefits from unique field programmable gate array (FPGA) signal processing techniques and leverages devices from the telecommunication industry to make it particularly cost efficient. A narrow band stabilized fiber laser, polarization maintaining fiber amplifiers, acousto-optic modulators and an optical circulator comprise the transmitter which is coupled to free space using refractive optics. The collinear propagating lidar return signal that scatters off of atmospheric aerosols and in a heterodyne arrangement beats with a local oscillator and is then detected using a shot noise limited polarization maintaining balanced receiver. The system, which operates at a 20 kHz pulse repetition rate and acquires lidar return signals at 400 MSample/second, accumulates signals that are as much as 20 dB lower than the receiver noise power by using embedded programming techniques. For this reason two FPGA embedded programming approaches are considered and compared. In the first approach, the acquired return signal is gated in time and the square modulus of the fast Fourier transform is accumulated for each range gate, producing a series of power spectra as a function of range. Wind speed

estimates based on numerical estimators can then be made after transferring the range gated accumulated power spectra to a host computer, enabling line of sight wind speed to be calculated as a function of range gate and stored for additional processing. In the second FPGA approach, a digital IQ demodulator and down sampler reduces the data flow requirements so that an autocorrelation matrix representing a pre-selected number of lags can be accumulated, allowing for the process of range gating to be explored on the host computer. The Fourier transform of the autocorrelation produces the power spectrum and, in the same manner as the first approach, estimates can then be made regarding the line of sight wind speed. The added feature of the second approach is that it allows for an additional capability to adjust the range gate period dynamically as the state of the atmospheric boundary layer (e.g. backscatter coefficient and stability condition) changes. A simple manual beam scanning technique is used to sample three line of sight directions and, by making suitable assumptions regarding the coherence of the averaged wind fields, the three dimensional wind field vector (representing both the horizontal wind speed and direction and the vertical wind speed and direction) is calculated and graphically displayed on time-height cross section plots. Precision in the velocity measurements is estimated to be on the order of 0.08 m/sec and the precision in the measured horizontal wind direction is estimated to be to be about 2 degrees, where both of these estimates are made assuming a relatively short 3-beam cycle time (less than 2 minutes) and a typical backscatter coefficient and atmospheric stability condition. A comparison to other observed wind information is presented which indicates that this lidar will open new doors for the practical characterization of microscale meteorology.

PREFACE

To my Father,

I present this thesis and all the accomplishments that I have achieved to my father, the man who always encouraged and motivated me ever since I was in high school until the last time I spoke to him on the phone last year before he passed away. I still remember him waking me up at 5 o'clock in the morning during my final exams of high school to finish my reviews before going to exams, making me breakfast, waiting for me to give me a ride to school.

I always dreamed of having him attend my PhD commencement to make him feel proud of me, as he always was, but it was not meant to happen, as he passed away 10 months before my thesis defense.

I also present this accomplishment to my Mom who always prays for me to succeed, and to my wife who always supported me during the entire journey of the PhD.

ACKNOWLEDGEMENTS

All my thanks are to my advisors; Professor: Sam Ahmed, Fred Moshary, Mark Arend, and Barry Gross for all of their help and guidance. I also would like to thank everyone in the remote sensing laboratory of the City University of New York for their help and co-operation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 CHAPTER I INTRODUCTION

1.1 Introduction

Wind measurements are very important to many fields and applications such as meteorology, aviation, transportation, and power industry. In meteorology wind data are used for weather forecast and to predict long-term global climate changes. Wind speed and direction measurements are very crucial in aviation as it is essential to determine the safety conditions of landing and takeoff. Transportation agencies use wind data for traffic regulations on roads, railways, tunnels, and bridges. Harbors and ships need wind measurements for navigation and safety. In power industry, wind measurements data is needed for wind farm's site selection. When the power plant operates, wind measurements are needed to control wind turbines. Also in nuclear power plants wind measurements are required for safety reasons to provide models of air flow and contamination with radioactive wastes in case of environmental hazards.

The first wind measurement device (anemometer) was invented in 1450 by an Italian architect named Leon Battista Alberti. Four hemispherical cups anemometer was later invented in 1846 by Dr. John Thomas Romney Robinson. Today, wind speed and direction can be measured by using classical anemometers, sonic anemometers, rawinsondes, SODAR (Sonic Detection And Ranging), RADAR (Radio Detection And Ranging), and LIDAR (Light Detection And Ranging). Sonic anemometers determines instantaneous wind speed and direction by measuring how much sound waves traveling between a pair of transducers are sped up or slowed down by the effect of wind. Rawinsonde is a special type of radiosonde (Sonde is a French word that means probe) that is designed to only measure wind speed and direction. Rawinsondes are used in weather balloons to measure atmospheric parameters and transmit them

to a fixed receiver station. Sodar measures wind speed through measurements of the scattering of sound waves by atmospheric turbulence. Both radar and lidar use similar technique such as sodar but instead of using sound waves, radar uses microwave, and lidar uses laser waves.

Sodars and radars are used in wind profilers to measure wind speed and direction at various altitudes above ground level. Wind speed can be estimated by transmitting five beams; one is vertical to measure vertical wind velocity, and the other beams are orthogonal to each other to measure horizontal components of the wind. The profiler's assumption to measure wind speed is that turbulent eddies that scatter probing signals are carried along by the mean wind. Fig. 1-1 shows radar and sodar wind profilers that are mounted at Liberty Science Center, New Jersey, and on top of the Metlife building in the center of Manhattan, New York, respectively, as part of the New York City meteorological network (NYC MetNet). This type of instruments is large and not portable.

Figure 1-1, A radar wind profiler (left) mounted on the Liberty Science Center and a sodar wind profiler (right) mounted on a NYC high rise [1]

Lidar systems can be developed with a low cost while being portable, stable, safe, and reliable. For atmospheric measurements using a lidar, a laser pulse is transmitted into the atmosphere, where the laser radiation is scattered along its propagation path by aerosol particles.

A telescope is used to capture the signals that have been back scattered toward the instrument. Doppler systems can measure wind velocity by estimating backscattered signals' frequency, which is equal to the frequency of transmitted laser pulse plus a frequency shift proportional to the wind velocity along the line-of-sight. Lidar systems for wind measurements have been commercially developed by companies like Leosphere, Halo Photonics, ZephIR. These systems can cost anywhere from $150k to $1M and claim ranges from a few hundred meters to several kilometers.

Coherent Doppler Lidar (CDL) has proven to be a powerful tool for remote sensing of the atmosphere, and has been widely adopted in applications such as measuring atmospheric wind velocity, turbulence, aerosol concentration, cloud height and velocity, and detection of atmospheric constituents and pollutants. The CLR systems have been developed for remote sensing measurements since the late 1960's. The first CDL wind-sensing system was reported by Huffaker et al [1]in 1970, where a 10.6 μm cw CO_2 laser was used. CO_2-laser-based CLR systems have been used for airborne clear air wind and hard target measurements such as ranging and produced valuable results for a long time. Since the late 1980s, CDL systems with newly developed solid-state lasers attracted a lot of researchers due to advantages of size, weight, reliability, and lifetime [2].

Operation at shorter wavelengths allows for higher spectral resolution, which means higher velocity resolution. A lot of effort has been put into 2 μm pulsed systems mainly intended for wind measurements [3] [4] [5]. Kavaya et al [6]developed a 1.06 μm pulsed CDL system that realized a measurable range of a few tens of kilometers used for launch-site wind sensing. Karlsson et al [3] reported a 1.5 μm cw all-fiber wind sensing CDL system, which utilized optical fiber components used in telecommunication systems.

In this study we report on the design, measurements, and performance of a CDL system for wind measurements. Our design involves a very low energy per pulse (12μJ/pulse), because it employs all-fiber optic laser components for availability, cost affordability, robustness, and size compactness. As a result of this low energy per pulse, a very high frequency repetition rate (FRP) is used, which produces a very large volume of returned signals. Acquiring such a large volume of data at a very high sampling rate and processing it cannot be achieved using a classical data acquisition and processing hardware. Therefore, signal pre-processing has to be carried out on hardware level by means of the FPGA, which allows for real time processing and a moderate data transfer rate to from the data acquisition card to the host PC. In our design, signal pre-processing was implemented to produce the power spectrum of time gated received signals by calculating fast Fourier transform FFT, which produces fixed spatial resolution gates. Another pre-processing technique was implemented that involved the calculation of received signals' autocorrelation, which can be used to find the power spectrum at any desired range resolution.

The system consists of a distributed-feedback (DFB) semiconductor laser emitting at a 1.5 μm in addition to an erbium-doped fiber amplifier (EDFA) in a master oscillator power amplifier (MOPA) configuration. A notable feature of our system is that it utilizes polarized maintained (PM) fiber optics, which ensures the maintaining of polarization state between both local oscillator and back scattered fields. The advantage of using a 1.5 μm laser source is the eye-safety feature, which allows for operation in urban areas. In addition to eye-safety feature, the usage of a 1.5 μm source allows the system to benefit from the technology and component development driven by the telecommunications industry, which results in significant cost deductions.

In this study, the development and operation of a CDL system for wind sensing is presented. In chapter 2, the system configuration is presented and system's main components are described. In chapter 3, transceiver noise analysis and coherent lidar signal range dependence are examined. In chapter 4, signal processing and FPGA programming is introduced. In chapter 5, wind measurement results are reported in both vertical pointing and scan modes.

1.2 Coherent Doppler Lidar Theory

In coherent Doppler lidar laser pulses are transmitted into the atmosphere, which interact with aerosols within the atmosphere. As a result of this interaction, laser signals are backscattered towards the laser source, which can be detected and measured through an optical detector. According to the movement of the atmospheric aerosols with respect to the laser source, backscattered signals may suffer a frequency shift (Doppler shift), which is proportional to velocity of moving aerosols. The Doppler shift Δf of laser signals with λ wave length is given by:

$$\Delta f = \frac{2v}{\lambda} \qquad (1.1)$$

where; v is the velocity of the aeroslos, i.e. wind velocity.

1.3 Heterodyne Detection Theory

In heterodyne optical detection, local oscillator and backscattered signals are optically mixed through an optical coupler. The resulting mixed signal is then incident upon a photodetector. Both local oscillator and backscattered fields can be represented as:

$$x_{lo} = A_{lo}\cos(\omega_o t) \qquad (1.2)$$

$$x_s = A_s\cos(\omega_o t + \omega_s t) \qquad (1.3)$$

where; ω_0 is the local oscillator frequency and ω_s is the frequency shift that backscattered signals may suffer. The optical intensity as seen by the heterodyne detector is given by:

$$I_{opt} = [A_{lo}\cos(\omega_o t) + A_s\cos(\omega_o t + \omega_s t)]^2$$

$$= \frac{{A_{lo}}^2}{2}(1 + \cos(2\omega_o t)) + \frac{{A_s}^2}{2}(1 + \cos(2[\omega_o + \omega_s]t)) + A_{lo}A_s[\cos([2\omega_o + \omega_s]t) + \cos(\omega_s t)]$$

$$= \frac{{A_{lo}}^2}{2} + \frac{{A_s}^2}{2} + A_{LO}A_s\cos(\omega_s t) + \text{(High frequency component)} \qquad (1.4)$$

The terms: $2\omega_0$, $2(\omega_0+\omega_s)$, and $(2\omega_0+\omega_s)$ are at higher frequencies than detector's bandwidth and will not be seen by the detector. As a result, the generated photodiode electric current will equal to:

$$I_d = \eta A I_{opt} \qquad (1.5)$$

$$= \frac{\eta A{A_{lo}}^2}{2} + \frac{\eta A{A_s}^2}{2} + \eta A A_{lo}A_s\cos(\omega_s t) \qquad (1.6)$$

$$= \eta P_{lo} + \eta P_s + 2\eta\sqrt{P_{lo}P_s}\cos(\omega_s t) \qquad (1.7)$$

where: A and η are detector's surface area and photo responsivity, respectively. I_d consists of a dc component = $\eta P_{lo} + \eta P_s$ and an ac component = $2\eta\sqrt{P_{lo}P_s}\cos(\omega_s t)$. since $P_{lo} >> P_s$, then:

$$I_{d(dc)} = \eta P_{lo} \qquad (1.8)$$

$$I_{d(ac)} = 2\eta\sqrt{P_{lo}P_s}\cos(\omega_s t) \qquad (1.9)$$

Signal power can be calculated as:

$$< i_s^2 >= \left(I_{d(rms)}\right)^2 \quad (1.10)$$

$$= 2\eta^2 P_{lo} P_s \quad (1.11)$$

Detector's responsivity is related to detector's quantum efficiency through the following relationship:

$$\eta = \frac{e\eta_q}{h\upsilon} \quad (1.12)$$

where; e is electron charge, η_q is the quantum efficiency of the detector, h is Plank's constant, ν is laser's frequency,

$$< i_s^2 >= 2\left(\frac{e\eta_q}{h\upsilon}\right)^2 P_{lo} P_s \quad (1.13)$$

In our system, backscattered signals are sampled at 400 MHz using a 14-bit ADC equipped with an on-board FPGA. The laser pulse frequency rate (PFR) is 20 KHz, which limits the maximum measurement range to 7.5 km. To estimate wind velocity, the frequency shift of scattered signals (Doppler shift) has to be extracted. Backscattered signals are broken into time gates to represent desired range distances and a power spectrum of each range gate is calculated by Fast Fourier Transform (FFT). A gate length of 128 data samples is chosen, which corresponds to 48 m range distance. Due to low pulse energy (14 μJ/pulse), power spectrum accumulation is needed to improve detection probability and velocity estimation accuracy. The wind velocity of each range gate is estimated from the calculated mean frequency of a post processed power spectrum around the peak frequency.

2 CHAPTER II SYSTEM'S CONFIGURATIONS AND TESTING

In this chapter, the main system components are presented and an over view of the system's operation is explained. System's configurations and testing of optical and electrical components are also presented in this chapter.

2.1 System Overview

Our system configuration is shown in Fig. 2-1. The system consists of the following components: i) Laser source (ii) Accousto-optic modulator (iii) Fiber amplifier (iv) Optical circulator (v) Optical antenna (vi) Balanced detector, and (vii) Signal processor. Optical components are connected with a single mode polarized maintained (PM) optical fiber.

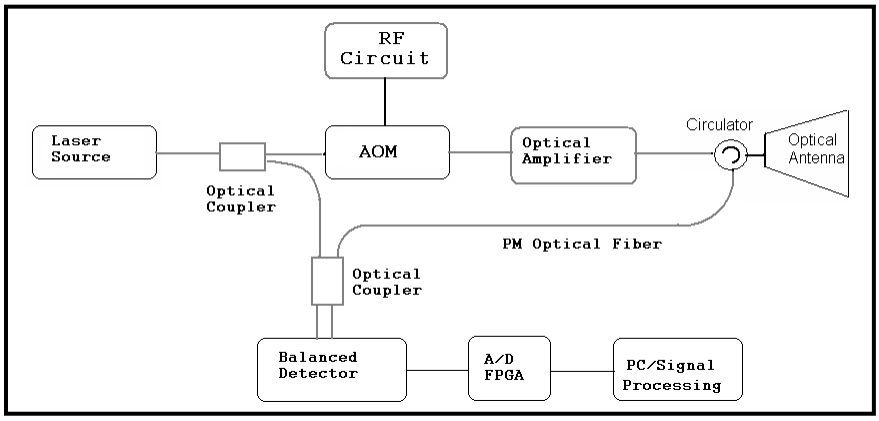

Figure 2-1 Coherent Doppler Lidar system's configuration

Our laser source has two outputs: a low power seed laser that is used as a local oscillator (LO), and a high power output (0.5W) that is modulated, pulsed, and frequency shifted using an acousto-optic modulator (AOM). Electronic circuits drive the AOM to shift laser signals by 84 MHz and generate 200 ns Gaussian shaped laser pulses. These laser pulses are amplified through an erbium doped fiber amplifier (EDFA) then transmitted from port 1 to port 2 of the optical circulator. To minimize the back reflection from port 2 back to port 1, the fiber tip at port 2 is angled and polished. Laser pulses are transmitted into the atmosphere and aerosol particles

scatter the laser signals back into the lens, which in turn are transmitted from the optical circulator's port 2 to port 3. Backscattered and LO signals are optically mixed using an optical coupler. Optically mixed signals are heterodyne detected through an optical balanced detector, which generates RF signals. These RF signals are acquired at a 400MHz sampling rate using an analog to digital converter card (ADC), which is equipped with an on-board field programmable gate array (FPGA) to allow for real time analysis. Digital data is then streamed to a host PC for further processing. More detailed explanations of key components are presented below.

2.2 Laser source

The laser source is a distributed feedback erbium doped fiber laser (DFB-EDFL) from NP Photonics, Fig. 2-2. The laser's wavelength is 1545.2 nm, and it has two outputs; first output, used as a seed laser, and second output has an adjustable output power up to 500 mW. The spectrum of the delayed heterodyne detected signal as measured by a spectrum analyzer has a full width at half-maximum (FWHM) of a few KHz. The laser linewidth is approximately 3 KHz, which corresponds to a velocity estimation accuracy of 0.2 $cm.s^{-1}$, so the laser linewidth is enough for our specification.

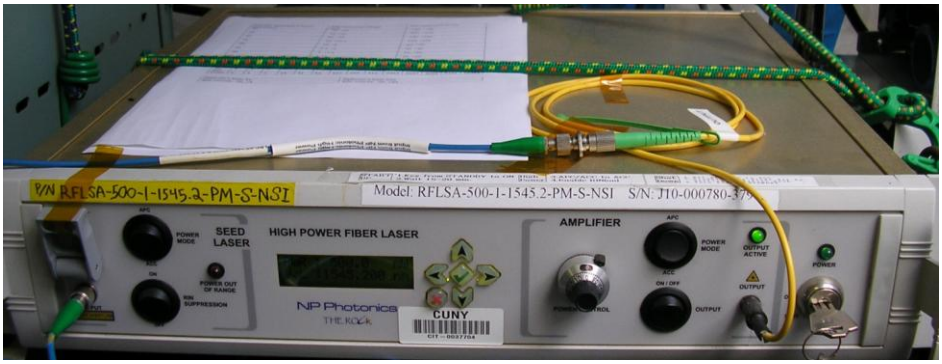

Figure 2-2 The fiber optics Laser source that has two laser outputs; one output is a low power that is used as a seed laser, the other output is a high power.

2.3 AOM

The continuous wave (CW) laser input is frequency shifted and pulsed through the AOMs, where an ultrasonic pulse is generated at a piezoelectric device by driving RF signals. Fig. 2-3 shows one of the two AOMs connected in series to obtain a very low extinction ratio. Each AOM shifts the frequency by 42 MHz, which leads to a total frequency shift of 84 MHz. The purpose of shifting the frequency of transmitted signals by 84 MHz is to shift the frequency of the zero velocity, so that both positive and negative Doppler shifts could be recognized. The driving RF signals, Fig. 2-4, turn the AOMs on during the 300ns of the 20 KHz RF driving pulse to generate a laser pulse with 200ns Full Width at Half Maximum (FWHM), Fig. 2-5.

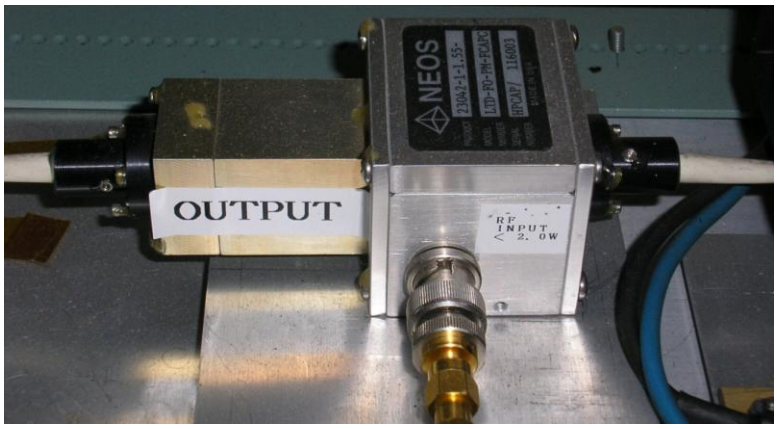

Figure 2-3 One of the two serially connected AOMs used to frequency shift the laser signals by 84 MHz, where each AOM shifts the laser signals by 42 MHz.

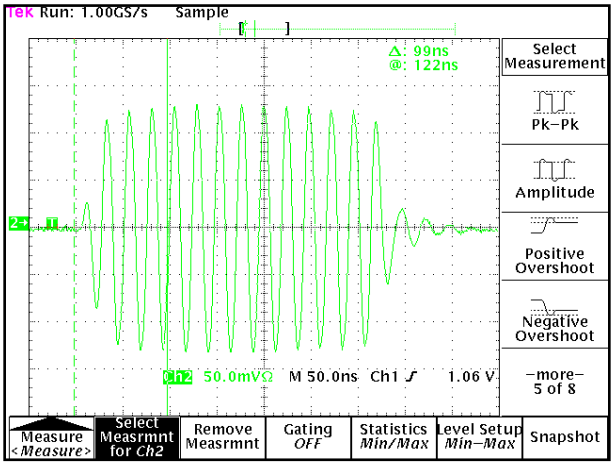

Figure 2-4 The RF signals that drive the AOMs

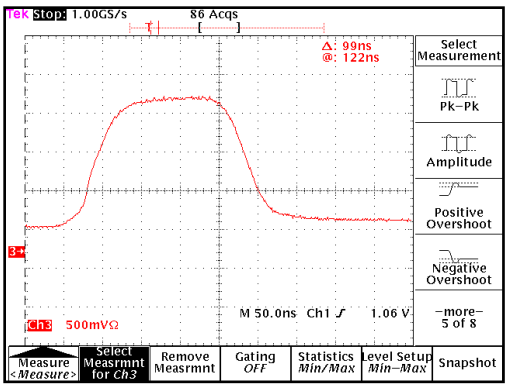

Figure 2-5 The output laser pulse that is generated by the AOM

The experimental setup, Fig. 2-6, shows the AOM frequency shift testing, where the seed laser (159m.W) is fed to a variable optical attenuator to attenuate the laser signal to 1m.W in order not to damage the optical detector, which has a maximum linear range of only 2mW (damage range is 10mW). The output of the attenuator is connected to an optical polarizer followed by a 90˚ beam splitter in order to make sure that the maximum polarized optical signal is selected. The optical signal is then split through a PM (Polarized Maintained) splitter. One signal is fed to an AOM, which shifts the optical signal by 42 MHz. This frequency shifted optical signal is then mixed with the original optical signal through a 50/50 optical coupler. The mixed optical signal is then fed to a heterodyne balanced detector, which is connected to a spectrum analyzer. The spectrum analyzer shows the beat frequency of the RF circuit of 42 MHZ, Fig. 2-7.

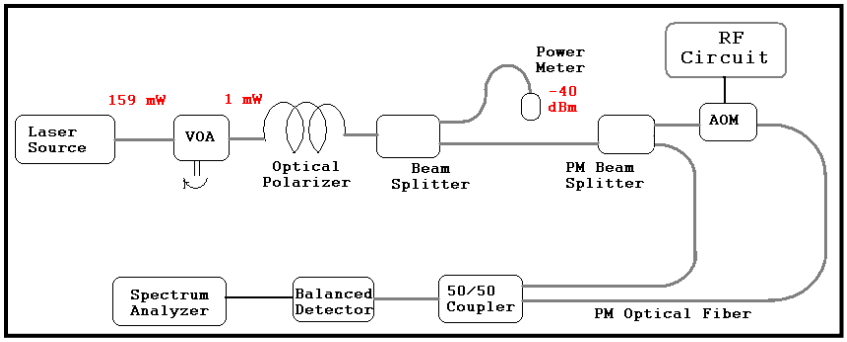

Figure 2-6 Experimental setup to test the frequency shift of the AOM.

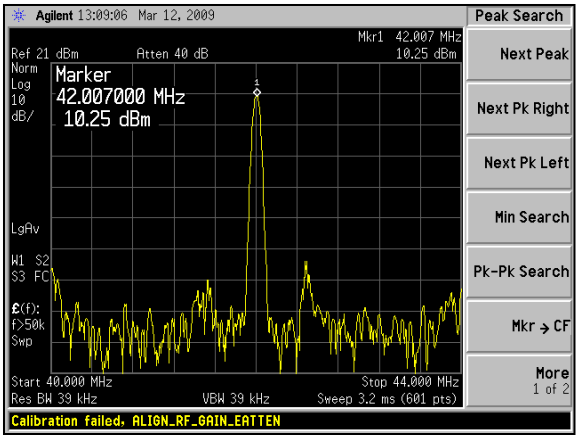

Figure 2-7 The beat frequency of 42 MHz that results from mixing the seed laser with the output of the AOM as examined by the spectrum analyzer.

2.4 Fiber amplifier

An erbium-doped fiber amplifier (EDFA) that has an average power of 340mW and a peak power of 74.6 W is used. This peak value cannot be increased beyond 74.6 W because of the Stimulated Brillouin Scattering (SBS), which can take place when an intense laser beam travels through a medium such as an optical fiber. SBS is generated from the acoustic vibrations in the medium that are caused by variations of the electric field of a traveling laser beam. Usually a laser beam undergoes SBS in an opposite direction to the incoming beam, which in our case can go back to the laser amplifier and cause damage to it. The EDFA has two amplifier stages; pre-amplifier and power amplifier. Output power is adjusted in a current control mode by adjusting the current of the power amplifier stage.

In order to verify that the amplified laser pulse has the prober shape and width, the following test was conducted. An optical power meter was connected to the SBS output port (a probe that monitors SBS power level) to measure SBS power level. As described by the EDFA vendor SBS power should not exceed 40 µw. We found that the SBS starts to have a non linear effect at approximately 1.2 A of the optical amplifier's second stage current. We also examined

the shape of the output laser pulse by using New Focus 1811 optical detector to view the pulse shape. We found that the pulse duration is approximately 100 ns, which is less than the required 200ns pulse width. Therefore, we increased the width of the RF pulse that drives the AOMs from 200 ns to 300 ns in order to widen the amplified laser pulse. After widening the RF pulse, we obtained a laser pulse that has the required width of 200 ns, Fig. 2-8. Widening the laser pulse enables us to reach higher output power from EDFA before start seeing SBS effect (1.4 A of second stage current). The fiber optic's length at the output of the fiber amplifier was shortened to approximately 25cm in order to limit the SBS effect.

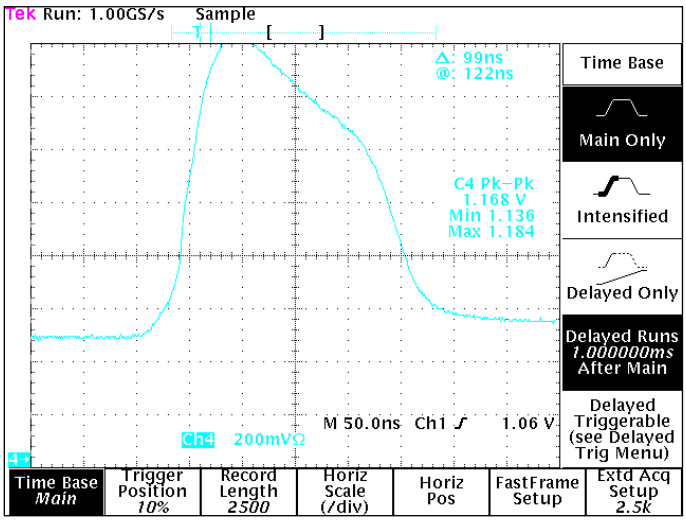

Figure 2-8The laser pulse shape generated by the AOM measured at the output of the optical amplifier using a **New Focus 1811** photo detector.

2.5 Optical circulator

The optical circulator ensures that the amplified output laser pulse is transmitted into the optical antenna and not into the detector. It also ensures that received backscattered signals and signals reflected off the fiber tip from the output pulse are directed into the receiver and not into the fiber amplifier. The back-reflection signal level at the fiber tip of the output port (port 2) of the optical circulator is very critical, because it can damage the optical detector. Therefore, the following

test was conducted to measure this back-reflection signal level to make sure that it is within acceptable levels.

In this experiment, the back-reflection signal power level (power at output-port 3 when the input power source is connected at port 1) was measured at four different input power levels; 100, 50, 25, and 12mw. The laser source was connected to the input of the circulator (port 1), the power level at port 2 and port 3 were measured using an optical power meter, Fig. 2.9. The optical circulator's insertion loss (power loss between port 1 and port 2) was measured and plotted in Fig. 2.10, which shows approximately 1dB insertion loss. Power measurements at port 3 (back-reflection) shows that about -50dB of the power at port 2 is reflected to port 3. This level of reflection is high and can damage the optical detector when the full power is used during operation. Therefore, the optical circulator was sent back to the vendor to have the fiber tip of port 2 polished and angled. Initially port 2 fiber tip was 8˚ angled. However, to achieve < -60dB of back-reflection, the fiber tip was polished and angled at 10˚. The same test was conducted again after port 2 was polished and the angle was made 10˚, which showed that back-reflection was reduced to less than -60dB. Fig. 2-11 shows the output power at port 2 vs. back reflection power at port 3 before and after the polishing of the fiber tip.

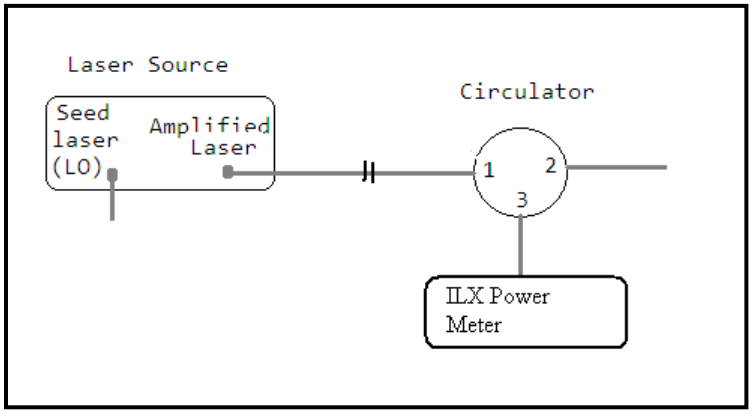

Figure 2-9 Experimental setup to test optical circulator's internal reflection

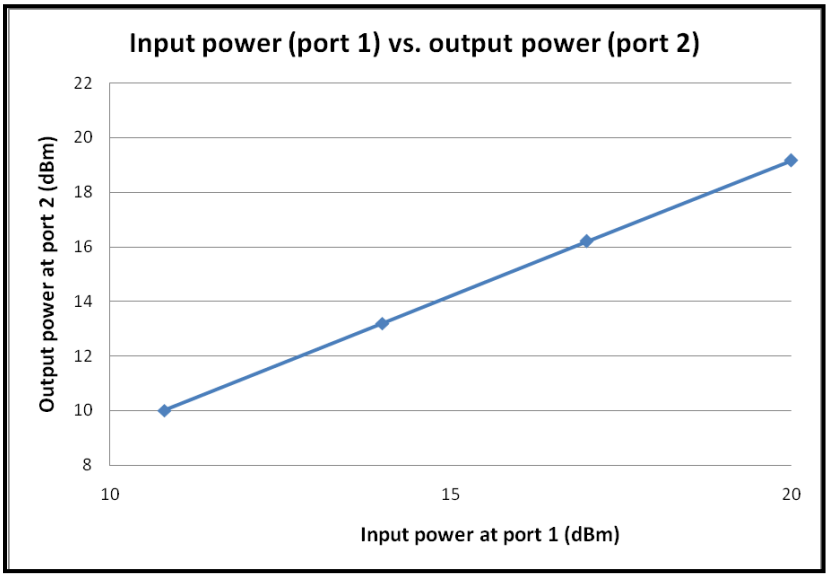

Figure 2-10 The Insertion loss of the optical cirulator measured between port 1 and port 2

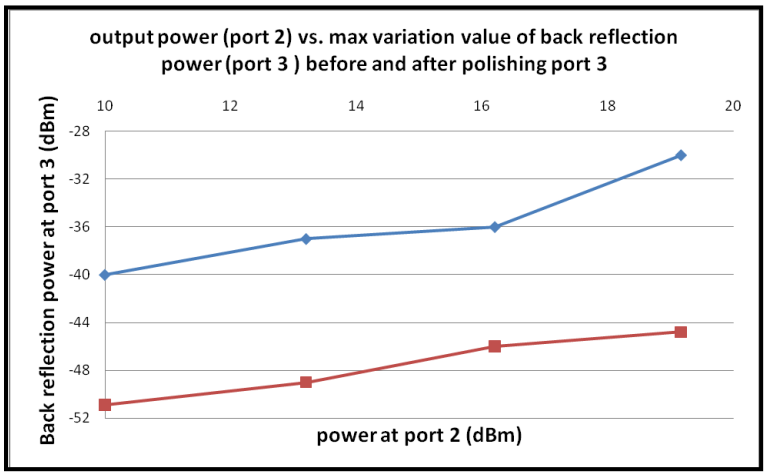

Figure 2-11 output power at port 2 vs. back reflection power at port 3 before polishing fiber tip (top curve) and after (bottom curve)

2.6 Optical antenna

A 4" diameter lens with a focal length of 50cm is used. The truncation ratio is approximately 0.88, and the lens' Rayleigh range is approximately 5km, which means that the laser beam can be collimated for all desired range (100m to 4km). This lens is mounted on an aluminum rail with a fiber holder that houses the optical fiber. A 6" mirror is also mounted on the same rail to steer the laser beam, and the entire setup is mounted on optical table, Fig. 2-12.

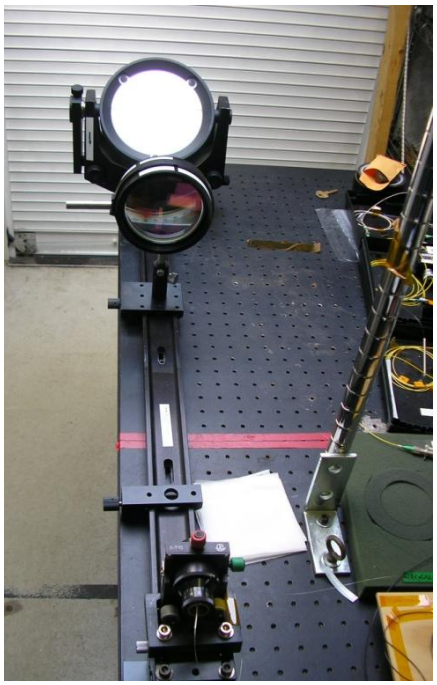

Figure 2-12 the optical setup of a 4” lens mounted on an aluminum rail with a fiber holder housing the fiber optic and a 6” mirror to steer the laser beam all mounted on an optical table.

To test the sensitivity of the fiber position with respect to the lens, the following experiment was conducted using the setup shown in Fig. 2-13. In this experiment, the laser source output is connected to the input port of the optical circulator, and the circulator’s output port (port 2) is mounted on the fiber holder. The lens is placed at a distance from the fiber holder that’s close to the focal distance of the lens. A retroreflector mirror is mounted after the lens to reflect the laser beam on itself and couple it back to the fiber (port 2). This back-reflected laser is then measured at circulator’s port 3. The fiber holder’s position was then varied and the power of back-reflected laser was measured at each time. Fig. 2-14 shows the effect of changing the fiber position on the power coupled back into the fiber. This experiment shows that we can couple back up to 3% of transmitted power. It also shows that moving the fiber holder only 400μm can reduce the coupled back power by 3dB.

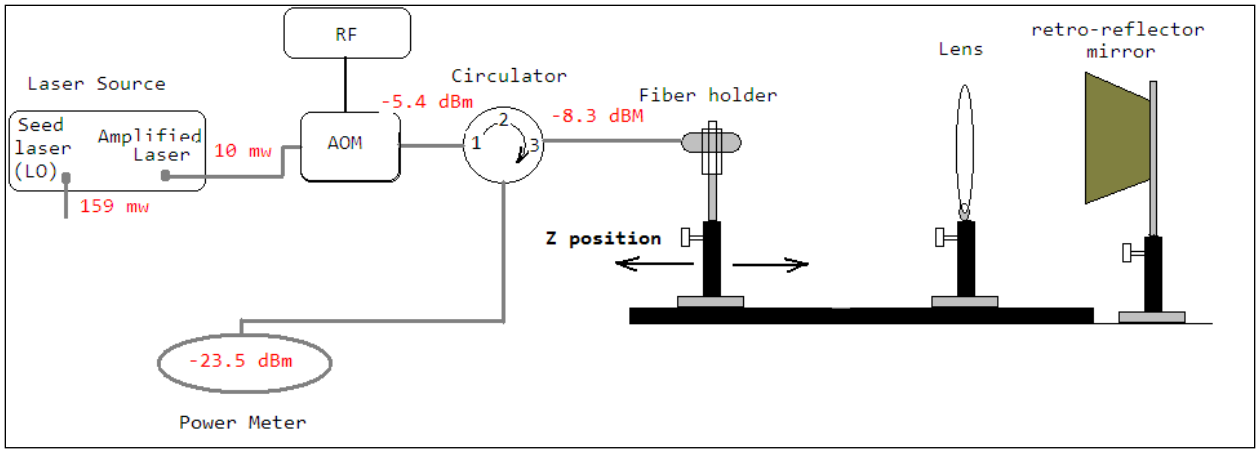

Figure 2-13 Experimental setup to test z-position effect on received power.

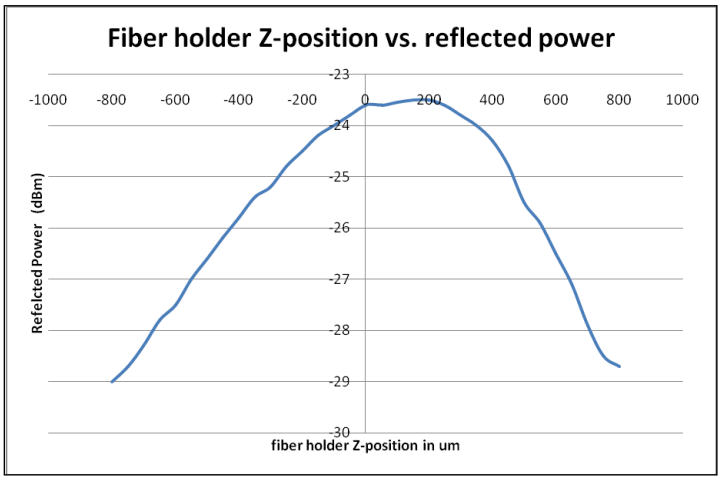

Figure 2-14 Fiber holder's position Vs. power coupled back to the fiber using a retroreflector mirror.

2.7 Balanced Detector

An InGaAs heterodyne balanced detector (from: TeraHertzTechnology) with a bandwidth extending from d.c. to 125MHz is used to retrieve backscattered signals, Fig. 2-15. The benefit of using a balanced detector is to subtract the two optical input signals from each other, which results in the cancellation of common mode noise. This allows for detection of small changes in the signal path from the interfering noise floor. The photo current is converted into voltage through the detector's transimpedance amplifier module. Detector's noise was measured using a

spectrum analyzer while no optical signals were applied to its inputs, gain was set to 1x, transimpedance was set to 1.4 kΩ, coupling was set to DC, and spectrum analyzer's frequency resolution was set to 3 MHz. Detector's noise was equal to -83dBm, which is 1 dB below detector's specification of 3.6 $pW.Hz^{-1/2}$. The detector has a non-flat gain response, Fig. 2-16, i.e. the gain of the detector varies with the frequency of the input signals.

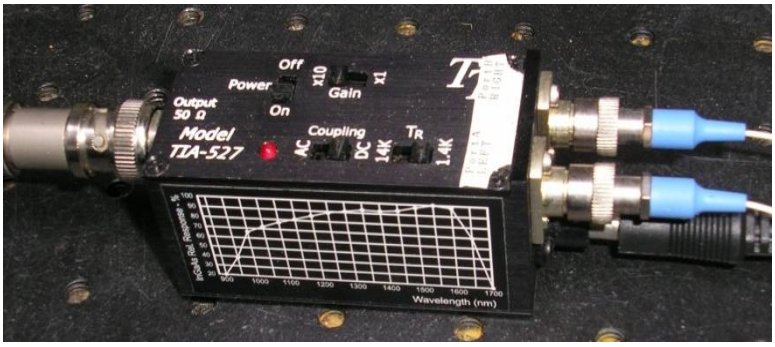

Figure 2-15 The InGaAs optical balanced detector used to retrieve backscattered signals.

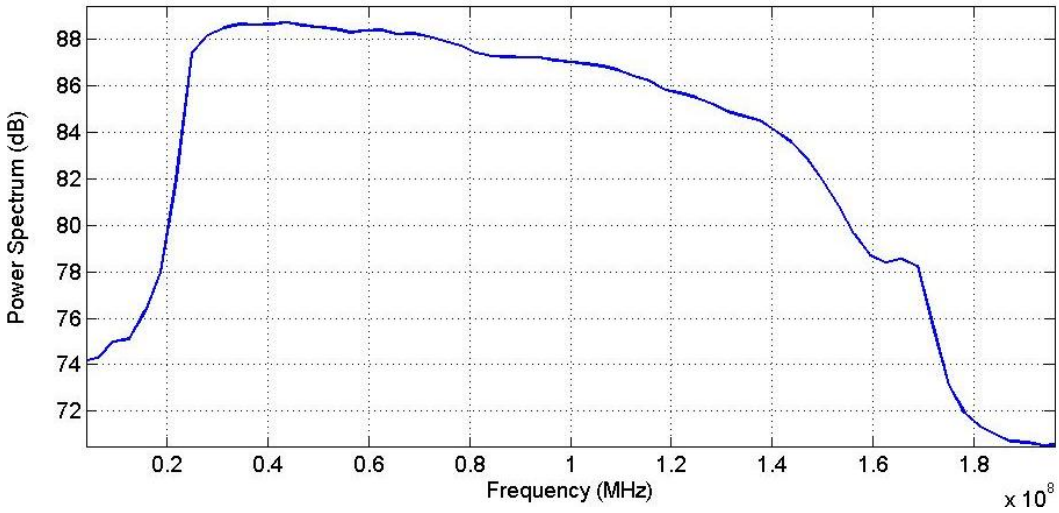

Figure 2-16 Non-flat gain response of the heterodyne balanced detector.

To correct for this non-flat gain shape, received signals' power spectrum is divided by the power spectrum of detector's output while no signal is present. The measured power when signal is present can be represented as:

$P_{total} = P_{sig} + P_{noise}$ (2.1)

Dividing this measured power spectrum by the power spectrum where no signal is present gives:

$$\frac{P_{total}}{P_{noise}} = 1 + \frac{P_{sig}}{P_{noise}} \quad (2.2)$$

$$= 1 + SNR \quad (2.3)$$

Subtracting 1 from equation (2.3), gives the SNR. The previous technique is used in our signal processing to estimate backscattered signal power.

In the following experiment, the detector's dynamic range was tested. The backscattered optical signals were simulated by using two serially connected optical attenuators. The two optical attenuators are connected between the AOM and the 50/50 coupler, Fig. 2-17, representing the attenuation that the laser signals will suffer through the atmosphere before being detected by the optical detector. The attenuation adjustment was changed from 0 to 120dB in steps of 20 resulting in attenuation values ranging from 11.2 dB to 131.2 dB due to attenuators' insertion loss. The output power of the detector is shown to be reduced as attenuation value increases until approximately 90dB, where the detector's output power level remains constant, Fig. 2-18. This behavior is because the detector measures the optically mixed signal of the attenuated and non-attenuated signals, and this mix is reduced every time the attenuation value is increased until a certain value (approximately 90dB), where the attenuated signal becomes too small to influence the mixed signal. The mixed signal becomes solely dominated by the non-attenuated signal (shot noise), which causes the detector's output to have a fixed power level.

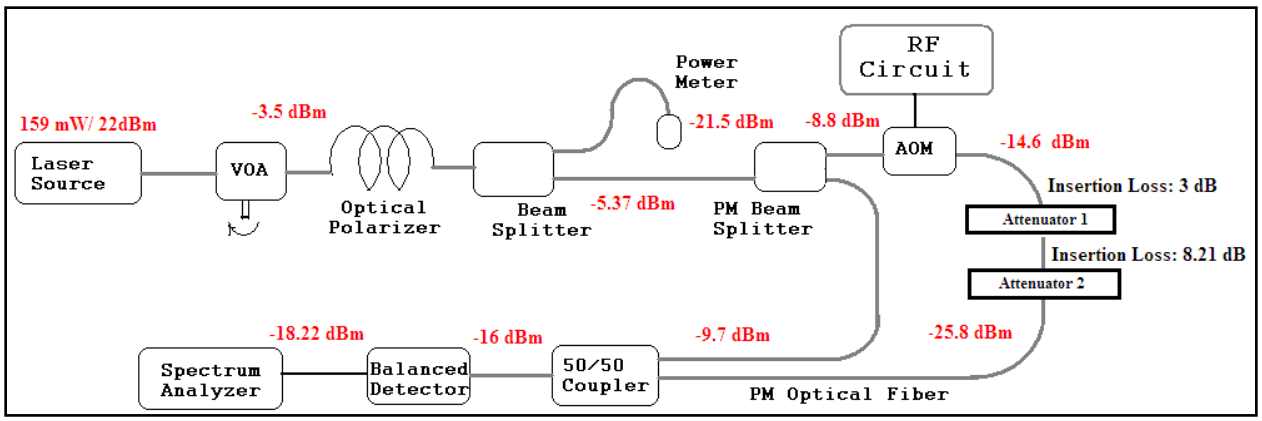

Figure 2-17 Optical detector's dynamic range testing using optical attenuators to represent attenuation caused by the atmosphere.

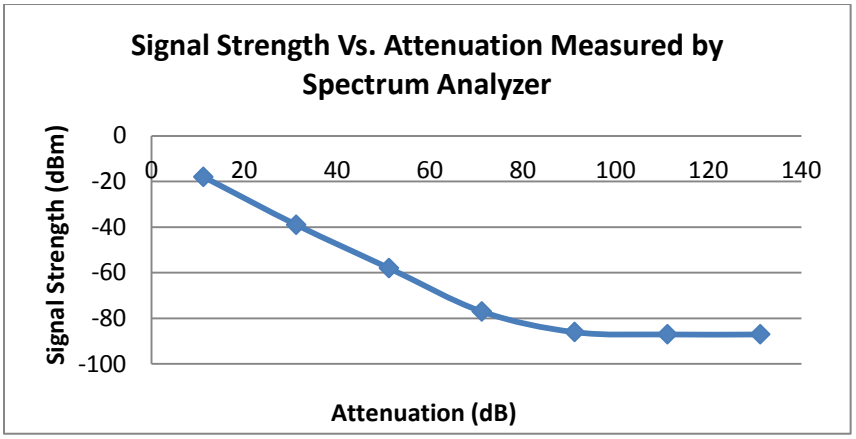

Figure 2-18 Detector's output power level Vs. Attenuation as measured by spectrum analyzer

2.8 Polarized Maintained (PM) Fiber Optic

All optical components are connected through PM optical fibers to ensure that the polarization state of the electric field of the local oscillator and that of the backscattered signals are very close, if not the same. Maintaining the polarization state throughout the different components of the system ensures a high level of the heterodyne detected signals.

The following experiment was conducted to test the influence of changing the polarization state of the laser signals on the power level of the heterodyne detected signals. In this experiment, a polarization controller is connected between the AOM and the 50/50 coupler, Fig. 2-19. A variable optical attenuator is connected at the output of the laser source to attenuate laser power from 159m.W to 1m.W. The optical attenuator is then connected to a half-wave polarization controller, which has two outputs. An optical power meter is connected at one of the polarizer's output. When the power meter reads a very low power level, it means that the power level on the other output is at its highest level. This second output of the optical polarizer is connected to a PM beam splitter, which is connected to an AOM and a 50/50 coupler. The AOM will shift the laser frequency by 42MHz. The output of the AOM is connected to a polarizer controller to change the polarization state of the shifted laser signals. The shifted laser is optically mixed with the non-shifted laser signals (the second output of the PM beam splitter). Now changing the polarization state of the shifted laser signals through the polarizer controller will change the power level of the beat signal that is detected by the heterodyne detector. It was found that signal strength could suffer up to 30 dB attenuation if the polarization state of the two fields is not maintained.

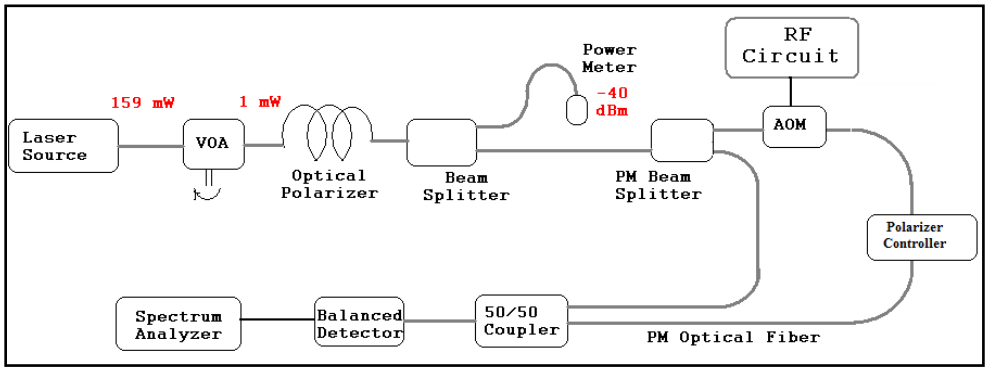

Figure 2-19 Polarization controller test setup

3 CHAPTER III POWER ANALYSIS AND SNR RANGE DEPENDENCE

In this chapter, noise components of the heterodyne photodetector are analyzed, detailed SNR analysis is presented, and the optimum local oscillator power level is determined. The range dependence of SNR is also investigated, and system performance is evaluated. Analytical and experimental wideband SNR was compared. In section (a), we present the SNR at the heterodyne detection and in section (b) we present the range dependence of the wideband SNR.

3.1 Transceiver noise analysis

In optical heterodyne detection, signal to noise ratio (SNR) of the Lidar system determines the system's ability to detect low level backscattered signals out of noise [7] [8]. Lidar heterodyne photoreceiver optimization is required to increase the receiving sensitivity [9]. It was shown that heterodyne detection sensitivity can reach its maximum value if local oscillator power is set to an optimum level [8]. Detector optimization can be achieved when electronics circuitry is optimally matched with detector, optimally match the optical local oscillator phase and shape to the optical backscattered field, and select the optimum local oscillator power level [10]. Many previous works studied effects of critical photoreceiver parameters and provided solutions to optimize heterodyne detection. J. Holmes et al. [8]. investigated the effect of the optical local oscillator power on the heterodyne responsivity of a photodiode detector operating in the photoconductive mode, but his analysis did not take into account the effect of laser's relative intensity noise (RIN). Jean-Pierre Cariou et al. [7] introduced detailed carrier-to-noise ratio (CNR) analysis for both pulsed and CW Doppler Lidar, and determined the optimum local oscillator power level required for maximum CNR when using single detector configuration rather than balanced detection. S. Kameyama et al. [11] showed the relation between local oscillator power and

efficiency on power penalty caused by thermal noise and laser's RIN, which is the ratio of the sum of thermal noise, shot noise, and the RIN to shot noise. The following analysis presents different heterodyne photodetector's noise components and gives an estimate to SNR.

The noise at the output of the optical detector consists of: 1) thermal noise (Johnson noise), 2) shot noise due to local oscillator induced current, and 3) laser's relative intensity noise (RIN).

Thermal noise is related to the detector and does not depend on the local oscillator power (P_{lo}). Thermal noise is expressed as:

$$< i_{th}^2 >= \frac{4kTB}{R_l} \qquad (3\text{-}1)$$

where: k is Boltzmann's constant, T is temperature in degrees Kelvin, B is detector's bandwidth, and R_l is detector's load resistor.

Shot noise, unlike signal powers that cancel through the balanced detector, the uncorrelated shot noise adds [12], resulting a mean-square noise at the output of the detector given by:

$$< i_{sh}^2 >= 2eiB \qquad (3\text{-}2)$$

where: i is the detector's current caused by the local oscillator power. This current can be calculated as follows:

$$i = en_e \qquad (3\text{-}3)$$

where, n_e is the number of electrons, which is given by:

$$n_e = \eta_q e_{ph} \qquad (3\text{-}4)$$

where, n_{ph} is the number of photons incident on the detector, η_q is detector's optical efficiency.

$$e_{ph} = \frac{P_{lo}}{h\nu} \qquad (3\text{-}5)$$

$$\therefore < i_{sh}^2 > = \frac{2\eta_q e^2 B P_{lo}}{h\nu} \qquad (3\text{-}6)$$

Laser relative intensity noise (RIN) is a property of the laser source, which is related to square value of local oscillator power through the following relationship:

$$< i_{RIN}^2 > = (R_{in}) R_b \left(\frac{e\eta_q}{h\nu}\right)^2 B(P_{lo})^2 \qquad (3\text{-}7)$$

where: R_b is RIN suppression ratio through the use of balanced detection. The SNR can now be expressed as:

$$SNR = \frac{<i_s^2>}{<i_{th}^2>+<i_{sh}^2>+<i_{RIN}^2>} \qquad (3\text{-}8)$$

$$= C\left[1 + \frac{2kTh\nu}{\eta_q e^2 P_{lo} R_l} + \frac{\eta_q R_{in} R_b P_{lo}}{2h\nu}\right]^{-1} \qquad (3\text{-}9)$$

where: C is an independent term of local oscillator power = $\frac{\eta_q}{Bh\nu} P_s$. When P_{lo} is small, the second term in the denominator of equation (3-9) dominates, and *SNR* is directly proportional to P_{lo}. Hence SNR increases with increasing P_{lo}. On the other hand, when P_{lo} is large, the third term in the denominator dominates, and *SNR* is inversely proportional to P_{lo}. The *SNR* then decreases with increasing P_{lo}. This means that *SNR* will increase as local oscillator power increases until it reaches a maximum value (where P_{lo} is optimum), after which it starts to decrease. The optimum value of P_{lo} can be determined by plotting the SNR as a function of P_{lo} assuming room temperature, R_{in} = -152 dB as provided by our laser vendor, Fig. 3-1, R_b = -25 dB, η_q=0.8, and R_l= 50 Ω. It is shown that *SNR* is maximum when P_{lo} is approximately = 10 mW, however, we

chose to set P_{lo} to approximately 5 mW to avoid operating the detector near its damage threshold, Fig. 3-2.

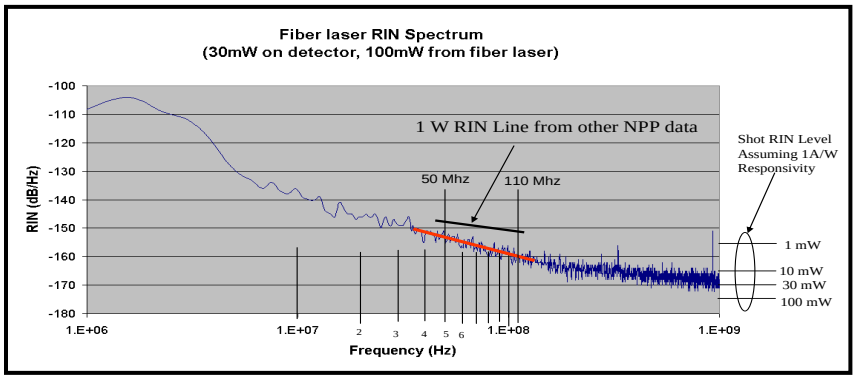

Figure 3-1 Relative Intensity Noise Vs. Frequency

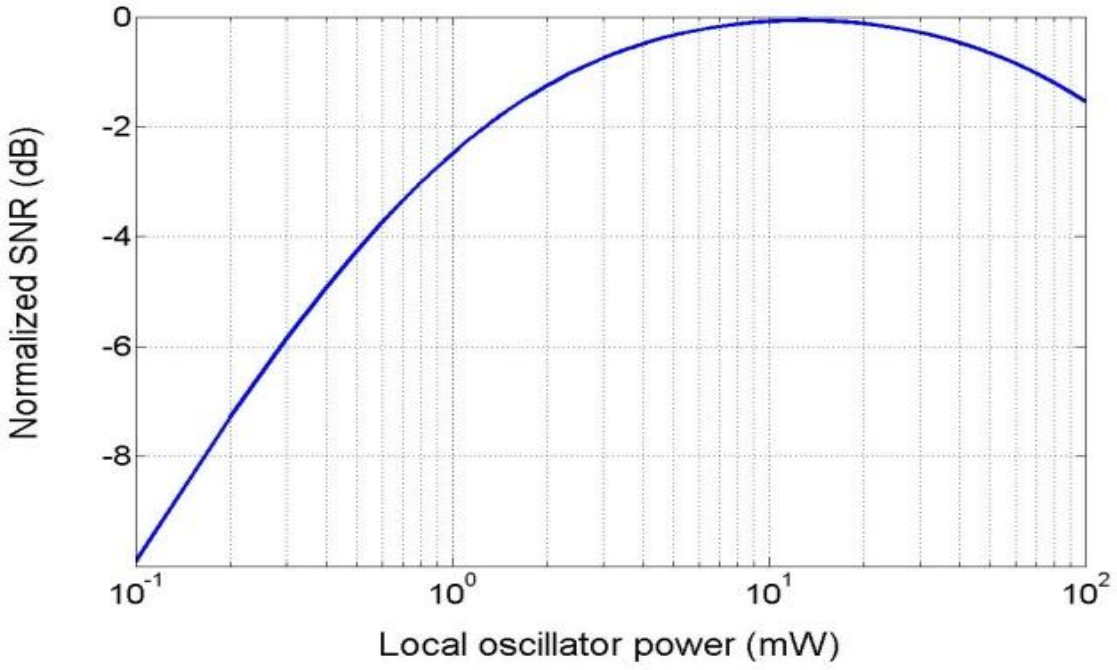

Figure 3-2 Normalized *SNR* as a function of local oscillator power P_{lo}

3.2 Coherent Lidar signal range dependence

In this section, we study the range dependence of SNR for the coherent laser radar (CLR) heterodyne detection using a mono-static configuration, and we compare analytical and experimental results. Mono-static configuration was believed to have an improved performance

due to the correlation of the transmitted and back scattered fields. This correlation is the result of wave-front tilts self correction in a mono-static configuration [13], [14] [15]. The SNR range dependence of a CLR mono-static system is evaluated by using the concept of backprojected local oscillator (BPLO), which is the imaginary local oscillator field distribution projected at the target side of the receiver aperture, receiver lens, originating from the detector [13] [16] [17] . Frehlich et al. [13] derived an equation that describes SNR as a function of range assuming a Gaussian Lidar system i.e., transmitter and LO fields are deterministic, detector response function is uniform, and the detector collects all LO and backscattered power incident on the receiver aperture. The SNR was then found by calculating the overlap integral between the BPLO and the backscattered fields on the receiver plane assuming a distributed aerosol target assuming ideal conditions, i.e. shot noise limited detector and a deterministic beam. To take into account the effects of refractive turbulence on CLR performance, different techniques of wave propagation in random medium were used [18]. Analysis shows that the SNR is proportional to the product of direct detection power and heterodyne efficiency. The calculation of received power and SNR requires mutual coherence function of the backscattered field incident on the receiver. As for natural aerosol targets, backscattered field at each aerosol particle has a random phase, and the mutual coherence function of the total backscattered field is the integration of all mutual coherence functions from each aerosol particle. The SNR range dependence equation [28] is expressed as:

$$SNR(L) = \frac{\eta_D(L)\lambda E \beta K^{2L/1000} \pi D^2}{8hBL^2} \tag{3.10}$$

where; η_D is the system efficiency, given by:

$$\eta_D(L) = \frac{\eta_{total}}{\left\{1+\left(1-\frac{L}{L_F}\right)^2\left[\frac{\pi(A_C D)^2}{4\lambda L}\right]^2+\left(\frac{A_C D}{2S_o(L)}\right)^2\right\}} \qquad (3.11)$$

where; the parameters of equations 10 and 11 are introduced in Table 3-1:

L	Range (m)	K	one way atmospheric transmittance
B	Bandwidth	L_F	Focal Range of Optical Antenna
λ	Wave length	A_c	Correction Factor
E	Pulse Energy	C_n^2	Refractive Index Structure Constant
D	Effective aperture Diameter	η_{total}	Total system efficiency
τ	Pulse width	$S_o(L)$	Transverse coherent length ~ $(1.1\, k_w^2 L C_n^2)^{-3/5}$
β	atmospheric backscatter coefficient	k_w	Wave number = $2\pi/\lambda$

Table 3-1 SNR range dependence parameters

Atmospheric transmittance, system efficiency, and the signal to noise ratio range dependence and aperture diameter dependence based on our system's parameters are shown in Figures: 3-3, 3-4, and 3-5, respectively.

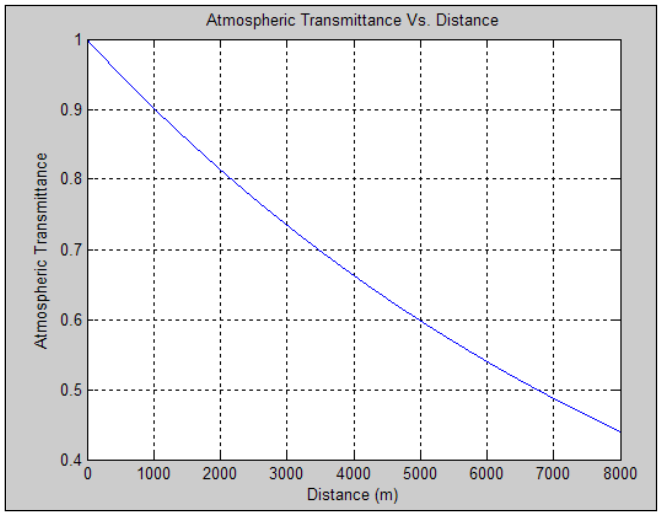

Figure 3-3 atmospheric transmittance as a function of distance

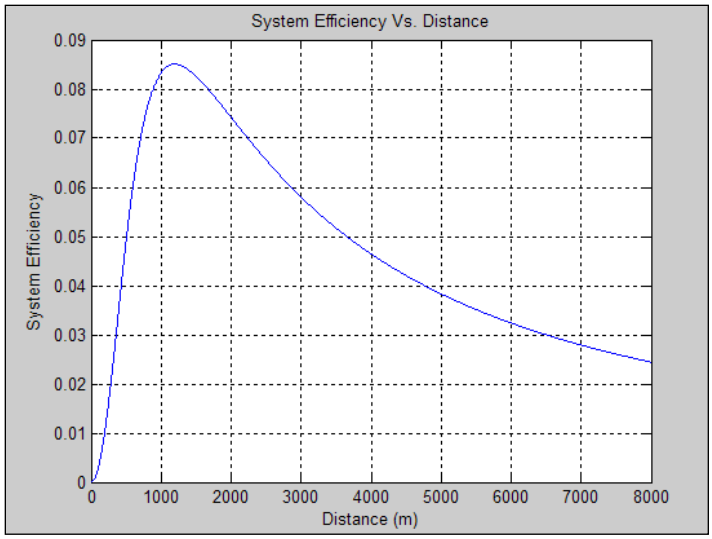

Figure 3-4 System efficiency as a function of distance

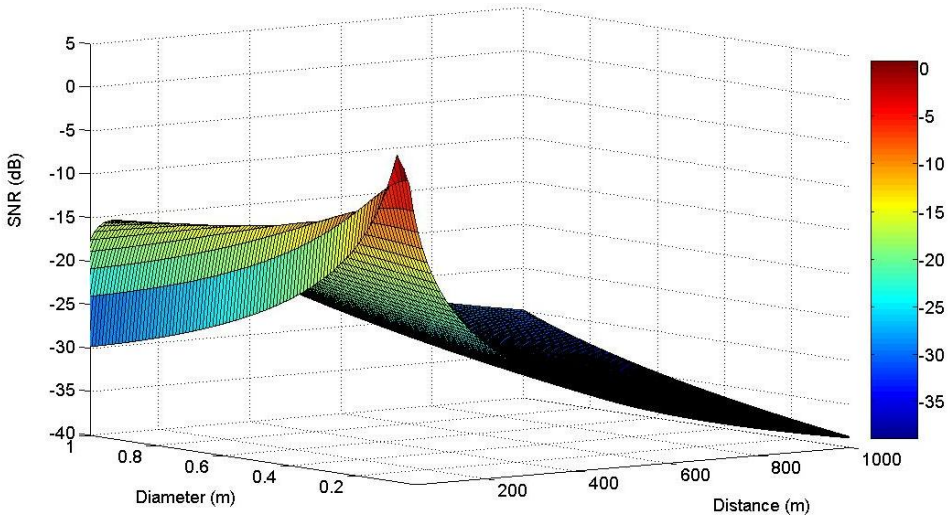

Figure 3-5 received power as a function of distance and apperture diameter

The performance of 6”, 4”, and 3” diameter antennas was evaluated both theoretically and experimentally while focusing the laser beam at approximately 1.8km. Continuous 38 range gates having a length of 0.32 μs (48 m range resolution) were obtained between a minimum range of 128 m and a maximum range of approximately 2km. The power spectra of received

signals from 10,000 laser shots were accumulated and wideband SNR was estimated. Fig. 3-6 shows theoretical and experimental wideband SNR range dependence. It is clear that both measured and theoretically calculated wideband SNR have a very good agreement. It is also shown that small diameter apertures have a large SNR in the near distances, while larger apertures tend to have larger SNR in the far distances. This is because the overlap integral function is higher at short distances for small size aperture; a small diameter aperture cannot focus the laser beam at a very far distance.

The parameters used in this analysis are listed in table 3-2

Parameter	Descriptions	Value
L	Range (m)	
B	Bandwidth	100 MHz
λ	Wave length	1545.2 μm
E	Pulse Energy	7 μJ
D	Effective aperture Diameter	0.15 m
τ	Pulse width	200 ns
β	atmospheric backscatter coefficient	8.3×10^{-7} /m/sr
K	one way atmospheric transmittance	0.95 /km
L_F	Focal Range of Optical Antenna	1.8 km
A_c	Correction Factor	0.76 This value was calculated based on truncation ratio given in Ref.[9]
Cn^2	Refractive Index Structure Constant	$2x10^{-14}\ m^{-2/3}$
η_{total}	Total system efficiency	-2.2 dB
$S_o(L)$	Transverse coherent length	$\sim (1.1\ kw^2 L\ Cn^2)^{-3/5}$
kw	Wave number = $2\pi/\lambda$	

Table 3-2 Parameters corresponding to analytical estimation of wideband SNR range dependence

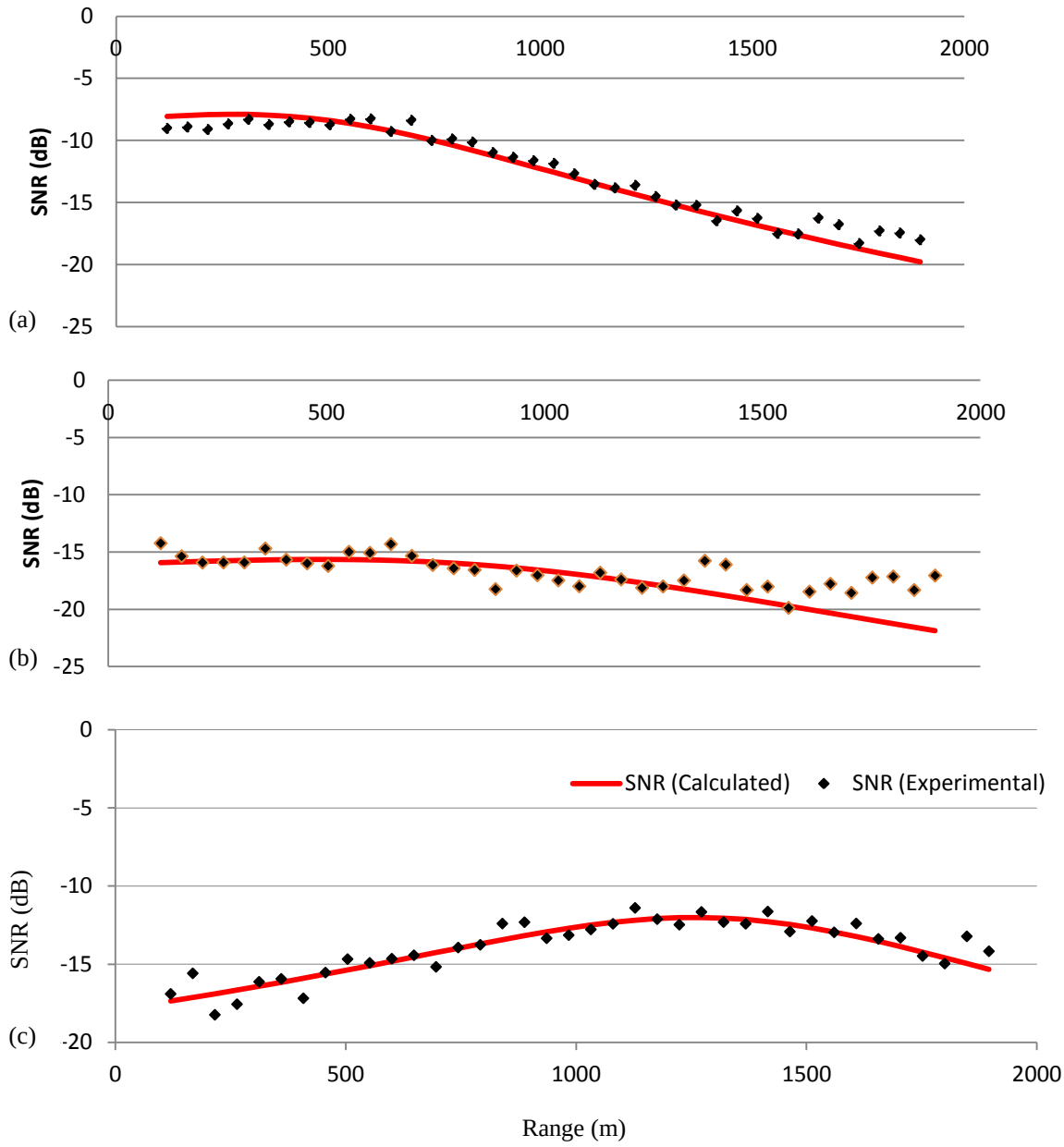

Figure 3-6 Wideband SNR range dependence, experimental and theoretical for: 3” (a), 4” (b), and 6” lens (c).

4 CHAPTER IV FPGA PROGRAMMING AND WIND MEASUREMENTS ANALYZED USING FFT

For a 20 KHz PFR and a 14-bit ADC with a sampling rate of 400 MHz, data transfer rate from the data acquisition card to the host PC will be 800 Mbyte/sec. This high data transfer rate is difficult to be achieved and requires additional hardware and software. Moreover, the amount of data collected in 1 day will be more than 69 Tbyte, which makes data archiving for just a few days nearly impossible. Due to the fast PFR, signal processing on the host computer cannot be achieved in real time, and will cause data to be lost. This means only a small fraction of scattered signals can be analyzed. In this approach, a large chunk of scattered signals is streamed to the host computer after being acquired by the ADC. A signal processing software is then invoked to analyze streamed data chunk. Scattered signal streaming to the host PC must be stopped until the analysis process is completed, which means ignoring new acquired signals. Such a processing technique results in analyzing only about 10% of acquired signals. Therefore, programming the FPGA to calculate power spectra or correlograms of backscattered signals and accumulate the results over a large number of pulses (we chose 10K pulses) will not only take the burden off the host PC and allow for real time analysis, but will significantly reduce data transfer rate across the PCI express bus to the host PC. In this approach, a signal processing algorithm is implemented and programmed onto the FPGA so that backscattered signals time gating, power spectrum calculation, and accumulation will all be simultaneously carried out on the hardware level as soon as signals are acquired by the ADC. Power spectrum of backscattered signals can be estimated directly by calculating the FFT of the time gated signals, or by calculating the FFT of signals' autocorrelation. FFT pre-processing algorithm and wind measurement results using FFT

technique will be explained in the following sections, while autocorrelation pre-processing algorithm and wind measurement results using autocorrelation technique will be explained in details in chapter V.

4.1 ADC Card:

Backscattered signals are sampled at 400 MSPS using a 14-bit ADC card from Innovative Integration, Fig. 4-1. The card model is X5-400M, which features two 14-bit, 400 MSPS A/D and two 16-bit, 500 MSPS DAC channels with a Virtex5 FPGA computing core and a PCI Express host interface, Fig. 4-2. The Virtex5 FPGA can be programmed using VHDL and MATLAB using the Frame Work Logic toolset. The MATLAB Board Support Package (BSP) allows for real-time hardware-in-the-loop development using graphical, block diagram Simulink environment with Xilinx System Generator toolset. Software tools for host PC development can be performed using C++.

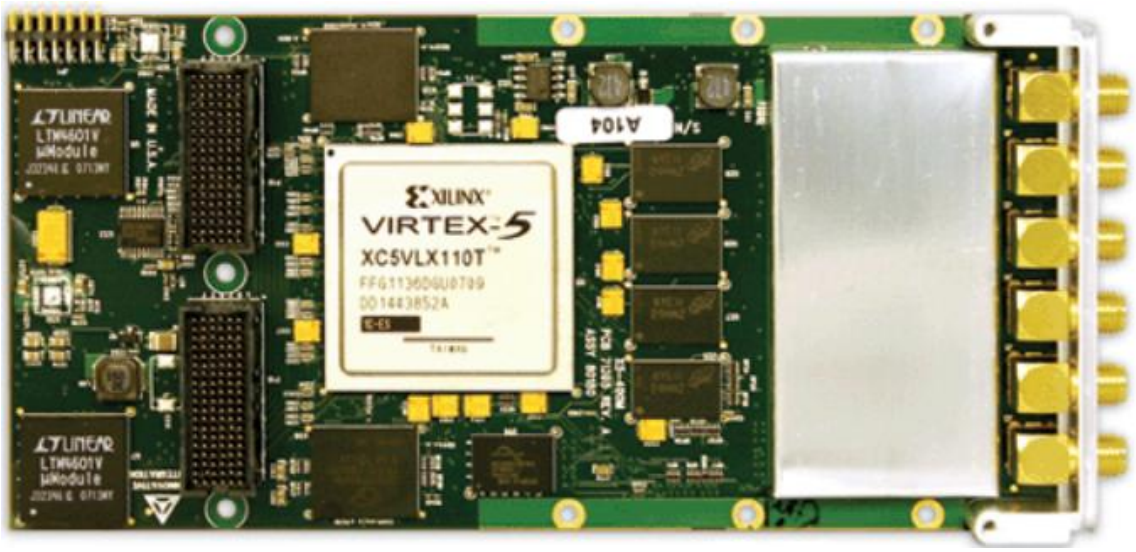

Figure 4-1 The 14-bit ADC used for data acquisition from Innovative Integration (X5-400M)

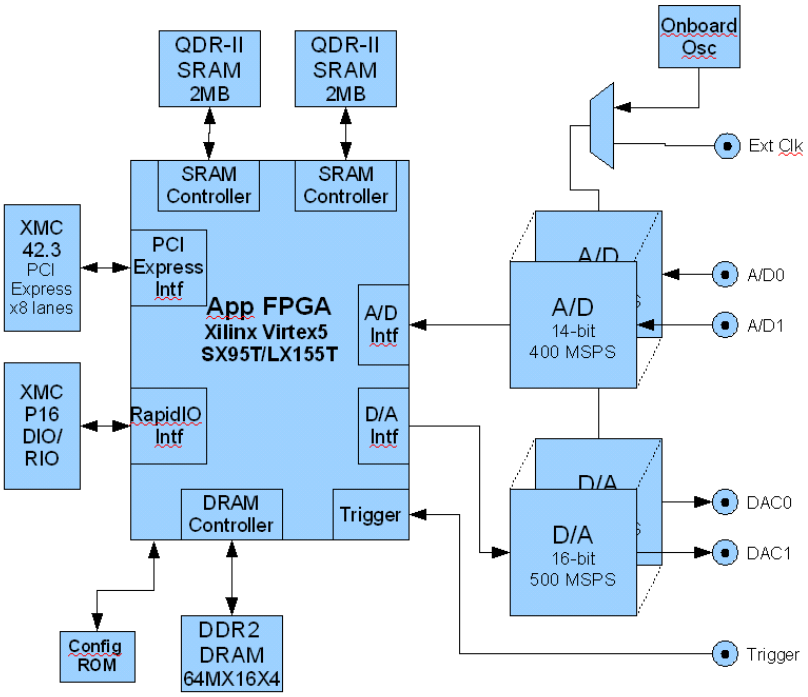

Figure 4-2 The ADC Card (X5-400M) block diagram. It shows the two 400 MSPS A/D and two 16-bit, 500 MSPS DAC channels with a Virtex5 FPGA computing core. It also shows the external trigger and clock input ports.

4.2 FPGA programming algorithm

A signal pre-processing algorithm is initially implemented as a logic design, which can be simulated and tested using Matlab/Simulink software. This logic design is then compiled using Xilinx system generator toolset to produce a hardware VLSI image, which can be downloaded into the FPGA. We chose to pre-process backscattered signals in two different techniques: a) calculate the FFT of time gated signals then accumulate the resulting power spectrum, and b) calculate the autocorrelation of the backscattered signals then accumulate the resulting autocorrelation matrix for 10k laser shots.

4.2.1 FFT Pre-processing Algorithm:

In this pre-processing algorithm, received signals are time gated into portions corresponding to spatial range gates, FFT is estimated for each range gate, and the corresponding power spectra

are accumulated over 10k laser shots. Power spectrum can be calculated using the FFT as follows:

The normalized Fourier transform of a time domain signals *f(t)* can be expressed as:

$$F_T(w) = \frac{1}{\sqrt{T}} \int_0^T f(t)\, \mathrm{e}^{-\mathrm{iwt}}\, \mathrm{dt} \qquad (4\text{-}1)$$

The discrete spectral density can then be found as:

$$PSD(w) = \lim_{T \to \infty} E[|F_T(w)^2|] \qquad (4\text{-}2)$$

Equation 4-2 shows that the squared modulus of the Fourier transform is the power spectrum. Therefore, we program the FPGA to calculate the square modulus of the output of the FFT block. Our ADC vendor provided us with an FPGA logic design that streams digitally converted signals (sampled at 400 MHz rate) across the PCI express bus to the host PC. This logic design accepts an external trigger signal to start data acquisition. A 20 KHz signal synchronized with laser pulses is used to trigger the data acquisition process. ADC card operates in a frame mode in which it acquires a frame of incoming data every time it receives an external trigger's interrupt. A frame size of 8192 samples is chosen, which corresponds to approximately 3.1 km. Xilinx Fast Fourier Transform 7.1 circuit block is used in a pipelined-streaming-io mode to calculate FFT for a vector of 128 samples of time gated scattered signals (corresponding to a 48m spatial resolution). Logic circuits that calculate the modules of the FFT complex output are also implemented and integrated with this design, Fig. 4-4. A first in first out (FIFO) memory circuit of length 8192 is used to accumulate power spectra, Fig. 4-5. In this design the FIFO block is used as a RAM and the whole accumulator circuits acts like a ring, where a power spectrum vector of 8k circles the ring until a new vector arrives, then the stored vector is read out

of the FIFO and added to the newly arrived vector while it is making its way into the ring. The result is then stored into the FIFO until a new vector arrives, and so on. This accumulation process will be performed until the counter circuit (Accumulator block) counts to 81,920,000 samples, which corresponds to the arrival of 10k laser shots, then newly arrived power spectra are ignored and stored accumulated data are streamed out to an output buffer before it is streamed to the host PC. A vector of 8192 zeros makes its way through the multiplexer circuit to the FIFO to flush the previously added power spectrum vector and reset the FIFO. Pre-processing algorithm's designed logic circuits are integrated with the FPGA data streaming circuits, which leads to streaming accumulated power spectra to the host PC instead of raw time domain backscattered signals.

Sampling received signals at 400 MHz and using a 20 KHz PFR generate 20k samples/pulse. FPGA logic circuits run at 250 MHz clock, therefore, two samples are stacked together to form a 32-bit word in order to achieve a 400Msamples/sec flow rate. A 32-bit word has to be broken into its original two 16-bit samples to allow for data analysis. This is accomplished by using a 32-bit to 16-bit converter circuit, Fig. 4-3. Down converting data samples from 32-bit to 16-bit will lower the data flow rate, and as a result, will lead to samples over flow and eventually data loss. To overcome this problem, a frame size of only 8192 samples is acquired at every rising edge of an external trigger signal that is synchronized with the signal driving the laser pulses. As a result, only 8k samples can be acquired by the ADC at each pulse. This allows for data down-conversion without any data loss, however, it limits the range distance to approximately 3.1 km.

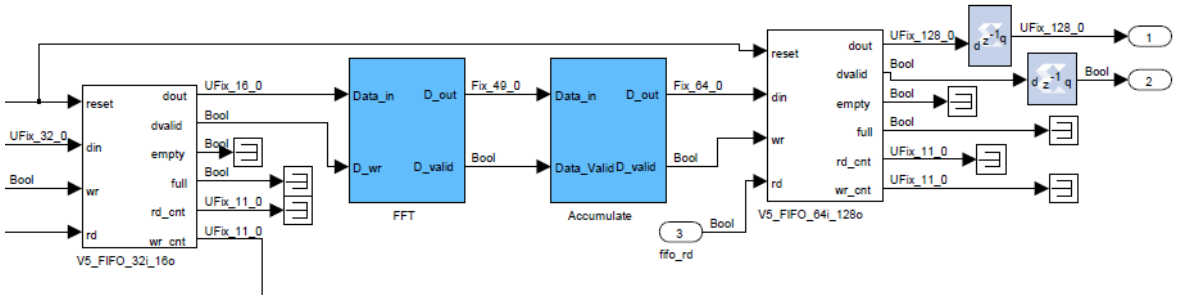

Figure 4-3 The digital circuit as implemented on the FPGA to convert the 32-bit two-words into 16-bit words

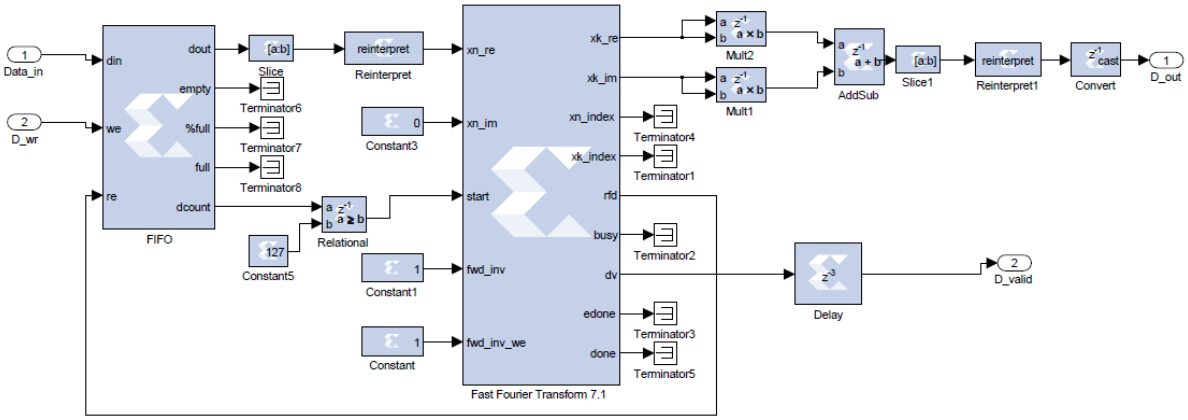

Figure 4-4 FFT computing digital circuit as implemented on the FPGA. The FIFO block streams the time domain signals to the FFT block in bursts of 128 samples, which results in a spatial resolution of 48m. The complex output of the FFT block is then multiplied by its complex conjugate to obtain the square modules of the power spectrum.

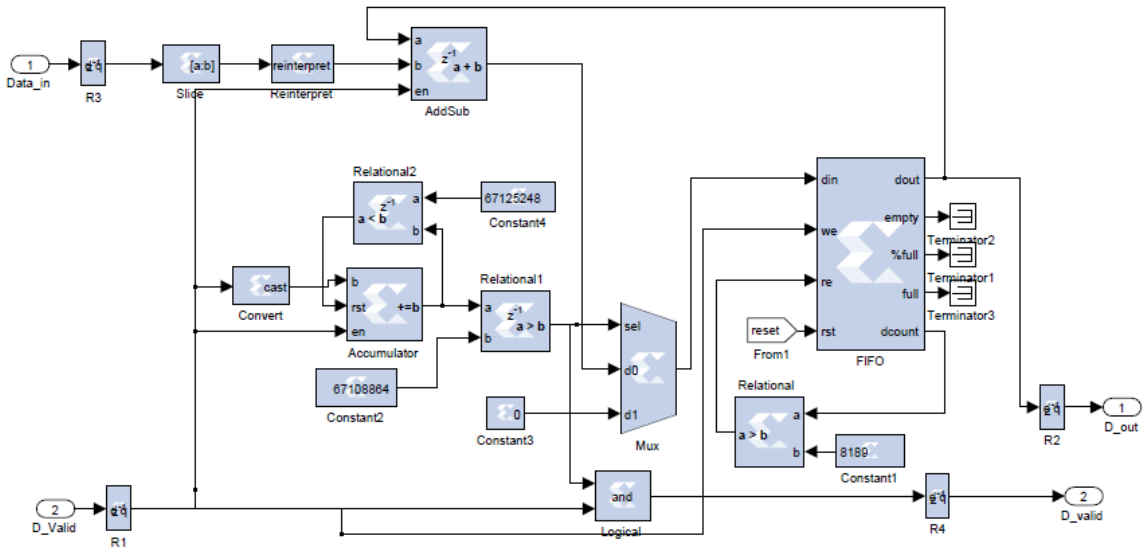

Figure 4-5 Accumulation digital circuit of the FFT output (power spectrum) as implemented on the FPGA.

In this design a FIFO block is used as a RAM and the whole design acts like a ring, where a power spectrum vector of 8k circles the ring until a new vector arrives, then the stored vector is read from the FIFO and added to the newly arrived vector. The result is then stored into the FIFO until a new vector arrives, and so on. This accumulation process will be performed until the counter circuit (Accumulator block) counts to 10k X 8192 samples, which means arrival of 10k laser shots, then newly arrived power spectra are ignored and stored accumulated data are streamed out to an output buffer before it is streamed to the host PC.

Arithmetic calculations using hardware binary bits require special attention to data width change. For example, multiplying two 16-bit numbers results a 33-bit answer, i.e. increasing data width. In our pre-processing algorithm, 16-bit real input data are expanded to 25-bit complex output through the FFT logic circuit. This 25-bit complex output is again expanded to 51-bit when calculating the absolute value. Finally, accumulating these 51-bit absolute values for 10k times can widen their widths to 64-bit. Data width increase requires design modification such as choosing right size buffers and proper interpretation when reading streamed output data.

4.3 Host Computer Signal Processing

Once accumulated power spectra are streamed from the FPGA across the PCI express bus, data post processing is carried out on the host PC to estimate various parameters such as radial wind velocity, backscattered signal strength, and velocity statistics. Data archiving and visualization are also carried out on the host PC.

The Doppler lidar estimate of the radial component v ($m.s^{-1}$) of the velocity vector is obtained from the mean-frequency Δf (Hz) of the Doppler lidar signal as:

$$v = \frac{\lambda}{2}\Delta f \qquad (4.3)$$

where λ (m) is the laser wavelength. As a result, the maximum radial velocity that can be measured is given by:

$$v_{max} = \frac{\lambda}{2} f_{\max} \qquad (4.4)$$

, which is approximately 30 $m.s^{-1}$.

The main parameter of interest in Doppler wind measurement is the mean frequency shift of the backscattered signal, because it is directly proportional to the mean velocity of moving aerosol particles within the atmosphere [19] [20]. Doppler frequency shift can be estimated by finding the centroid of the discrete power spectrum of the backscattered signal after removing the amplifier gain shape [21]. One easily calculated method of finding this frequency from a discrete power spectrum is to find the frequency of the highest power, i.e. the frequency corresponding to the peak power [22]. If the backscattered signal's mean frequency shift and the frequency corresponding to the peak power do not coincide, the velocity estimate can be off by as much as one-half of a frequency resolution.

To accurately estimate the Doppler frequency shift of the backscattered signal, several velocity estimators were developed and thoroughly investigated. An estimator's performance can be analytically tested via simulations to compare its estimation variance against performance bounds such as the Cramer-Rao lower bound (CRLB), which sets the theoretical limit on the local variance of frequency determined from any estimator. Estimator's precision can be determined by its variance especially when the probability density function (PDF) of the estimate is Gaussian, because a Gaussian distribution can be easily characterized by its first and second moments [23]. The first study on performance of a frequency estimator formed from the peak of a discrete power spectrum for a CDL system was reported by Hardesty [21] [11], and performance comparisons with complex-covariance estimators were conducted. Rye and Hardesty investigated the estimation precision difference between discrete- Fourier-transform (DFT)-based estimators and the CRLB [24]. They also compared the performance of DFT-based estimator with that of other estimators such as, pulse pair (PP) or single lag autocovariance, signal matching (SM), and maximum likelihood (ML) estimators [24]. Frehlich and Yadlowsky

presented the performance of the mean-frequency estimators in terms of two basic parameters Φ, the ratio of signal energy per estimate to the spectral noise level, and Ω, which is proportional to the number of independent samples per estimate [25]. They also examined the performance of PP, ML, power spectrum maximum likelihood (PML), and time series estimator of the autoregressive (AR) and minimum variance (MV) estimators and compared their performances with CRLB. The performance of these estimators in the weak-signal regime was studied by Frehlich in [26]. In our analysis we applied the ML estimator, and the following section analytically describes our velocity estimate procedure.

For backscattered signals of a single pulse, the discrete power spectrum of both signals and back-ground noise as obtained by FFT for one range gate can be represented as:

$$x = n_1 + x_1,\ n_2 + x_2, \dots, n_i + x_i, \dots, n_m + x_m \qquad (4.5)$$

,where m is the number of samples per range gate, n_i and x_i are the noise power and signals' power at sample i, respectively. If n is the number of pulses averaged together, then the averaged power spectrum can be represented as:

$$X = N_1 + X_1,\ N_2 + X_2, \dots,\ N_i + X_i, \dots,\ N_m + X_m \qquad (4.6)$$

, where :

$$X_i = \sum_{k=1}^{n} (x_i)_k \qquad (4.7)$$

Next, the power spectrum of a noise sample, which is obtained prior to data acquisition or from a far away range gate where no signal is expected [27], is removed from each range gate's averaged power spectrum. Noise power spectrum removal filters out noise produced by amplifier

irregularities and other noise sources that may introduce velocity bias. The resulting power spectrum can be expressed as:

$$S = \frac{X}{N} \tag{4.8}$$

$$= \frac{N_1+X_1}{N_1}, \frac{N_2+X_2}{N_2}, \dots, \frac{N_i+X_i}{N_i}, \dots, \frac{N_m+X_m}{N_m} \tag{4.9}$$

$$= S_1, S_2, \dots, S_i, \dots, S_m \tag{4.10}$$

$$\text{, where: } S_i = 1 + SNR_i \tag{4.11}$$

Assuming that backscattered signals' power spectrum has a Gaussian profile with a peak value = σ (the wide band SNR), frequency width = ω, and a peak frequency is at f_1, then the power spectrum at any given point i, which is conditional on the value f_1, can be written as [24]:

$$S(f_i : f_1) = \frac{\sigma}{\omega\sqrt{2\pi}} e^{-\left[\frac{(f_i - f_1)^2}{2\omega^2}\right]} \tag{4.12}$$

The frequency corresponding to the averaged power spectrum's peak is a random variable [23]. For accumulated power spectrum, each spectral component can be described by the probability density function (PDF) of a Gamma distribution [24]as follows:

$$P_n(X_i : f_1) = \left(\frac{n}{S(f_i : f_1)}\right)^n \frac{x_i^{n-1} e^{-\left[\frac{nx_i}{S(f_i : f_1)}\right]}}{\Gamma_n} \tag{4.13}$$

The probability that the detected spectral peak is at the i^{th} frequency bin, given that the actual peak is at f_1, can be estimated as the joint probability that the power level at the i^{th} bin is greater than that at all other bins. It can be analytically calculated as the product of the probability that the level in each of other bins is less than x_i. The maximum likelihood estimate

of f_1 is the frequency value that maximizes the log likelihood function of obtaining an averaged power spectrum with a peak value at f_1.

4.4 Setup Procedure to Observe Scattered Signal:

Initially the following procedure was followed when measurements were taken at the remote sensing laboratory of the CCNY to assure the fiber's optimum alignment with respect to the lens. The experimental setup shown in Fig. 4-6 was used. In this setup, the laser pulses were shot at a hard target (~ 100m), and the scattered signal was obtained and monitored on the oscilloscope. Fig. 4-7 shows the scattered signal off a hard target in time domain. Maximizing the magnitude of the hard target's scattered signal through adjusting fiber's x,y, and z positions achieves the optimum fiber's location to focus the beam at about 100m (the hard target's distance). To focus the beam at a different distance, the z position of the fiber holder can be adjusted accordingly. We then direct the laser beam away from the hard target to obtain atmospheric backscattering, Fig. 4-8.

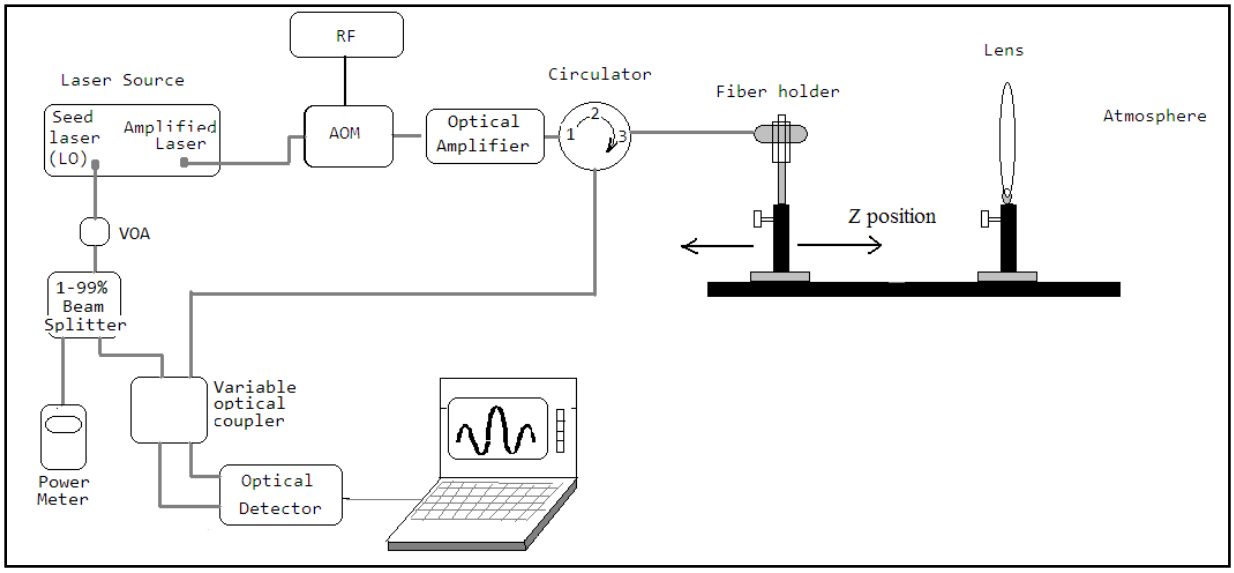

Figure 4-6 Set up procedure to observe scattered signal

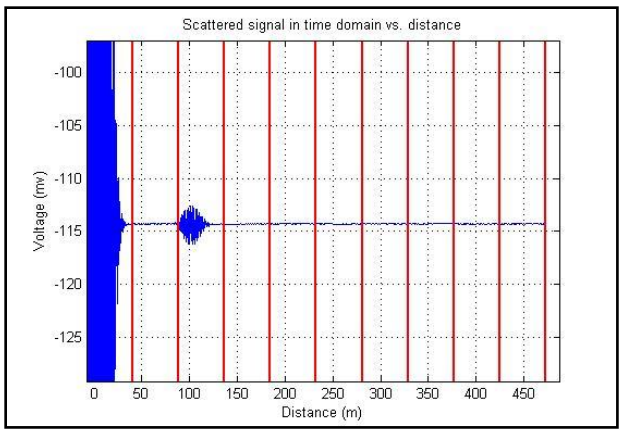

Figure 4-7 Time domain scattered signal off a hard target

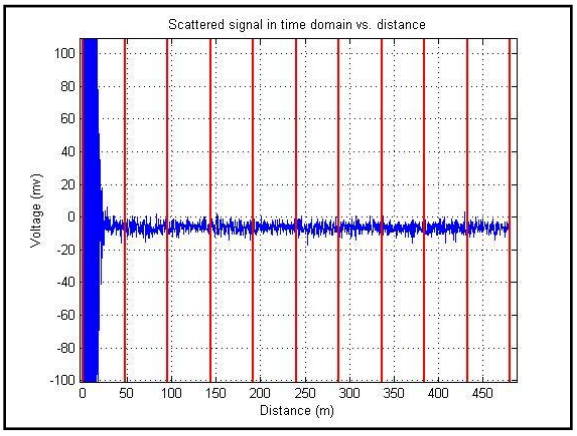

Figure 4-8 Time domain scattered signal off the atmospher

4.5 Non-Real Time Wind Velocity Measurement

In this section wind velocity measurements are presented. The procedure of measuring wind velocity was changed as the development of the instrument progressed. Initially, a backscattered atmospheric time domain data was captured and archived for post processing. This captured data normally represented a short time period of atmospheric data (approximately 0.5 second). Measuring wind velocity by capturing time domain backscattered signals, archiving it, and then processing it could not provide real time measurement because data analysis was a time consuming process. It also required a large data storing volume as 1 second of data

approximately takes about 1GByte. Pre-processing backscattered signals using the FPGA allowed for real time processing and also allowed for continuous wind measurements. FPGA pre-processing involved FFT calculation of time gated backscattered signals and autocorrelation calculation of backscattered signals. Wind measurement was also conducted in two modes; measuring vertical wind velocity by vertically pointing the laser beam during the entire measuring time and measuring horizontal and vertical wind velocity by steering the laser beam in a scanning mode. In this scanning mode, wind velocity was measured along three different directions; the wind vector was then constructed from these three different measurements. A 1μm direct detection measurements were also conducted at the remote sensing lab of the CCNY during our wind measurements, which allowed for atmospheric concentration profile and backscattered signal power comparisons. The following sections present the results of wind velocity measurements using different measurements' modes and procedures.

This section introduces the off-line signal processing algorithm. Backscattered signals were sampled at 400MHz using a 14-bit ADC card. Wind velocity was measured by acquiring approximately 1s worth of backscattered signals, which had approximately 1 GByte size. This 1 GByte was streamed from the data acquisition card and stored at the host PC. Signal processing algorithms were developed to process these archived signals. Backscattered signals were time gated into range gates corresponding to spatial ranges. For example, gating received signals in 128 samples/gate results a spatial resolution of 48m. FFT was calculated for each range gate and the resulted power spectrum was accumulated over many laser shots. Wind velocity was then estimated by determining the Doppler shift of the received signals. Fig. 4-9 shows the power spectrum of the gate where the hard target occurred. Fig. 4-10 shows the FFT of atmospheric scattered signal at gate three, which expands from 100m to148 m. The power spectrum shows a

peak at 87.5 MHz, which means a Doppler shift of 3.5 MHz. Wind velocity can then be estimated from this frequency shift.

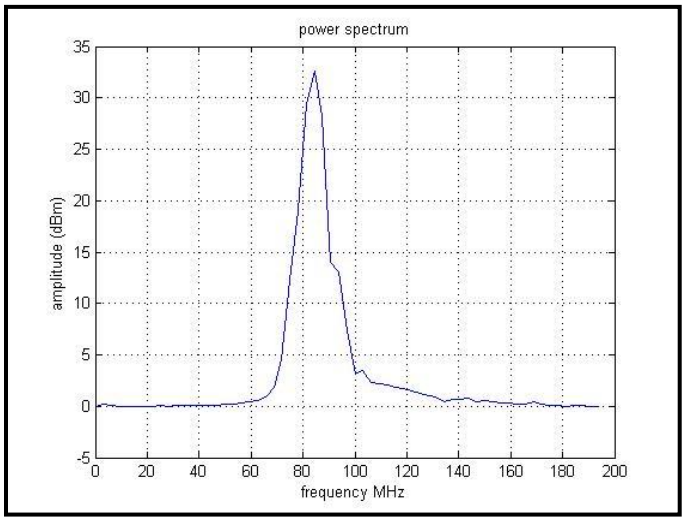

Figure 4-9 Backscattered signal's FFT of the range gate where the hard target occurred

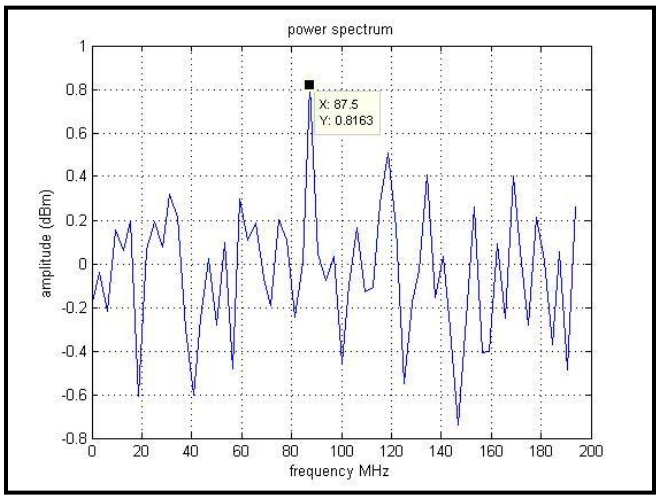

Figure 4-10 Backscattered signal's FFT of gate 3 (100m ~ 148m)

4.5.1 Signals' Distortion Test

In this test, the distortion of received signals when transmitting the laser beam through a glass window was determined. Backscattered signals were measured under two different conditions; first, the laser beam was transmitted, and received, through a glass window (the lab window was

closed), and then compared with the that while the window was opened. The signal strength in the two cases are shown in Fig. 4-11. It is shown that when the window was opened, the signal strength was high and wind velocity could be measured from gate 6 to gate 18 (120m ~ 408m). However, when the window was closed, signal strength suffered a significant distortion and wind velocity could only be measured at gates 8 to 10 (168m ~ 216m). This is because the glass window distorts the laser beam while it's being transmitted, and the backscattered signals are also distorted and backreflected off the window.

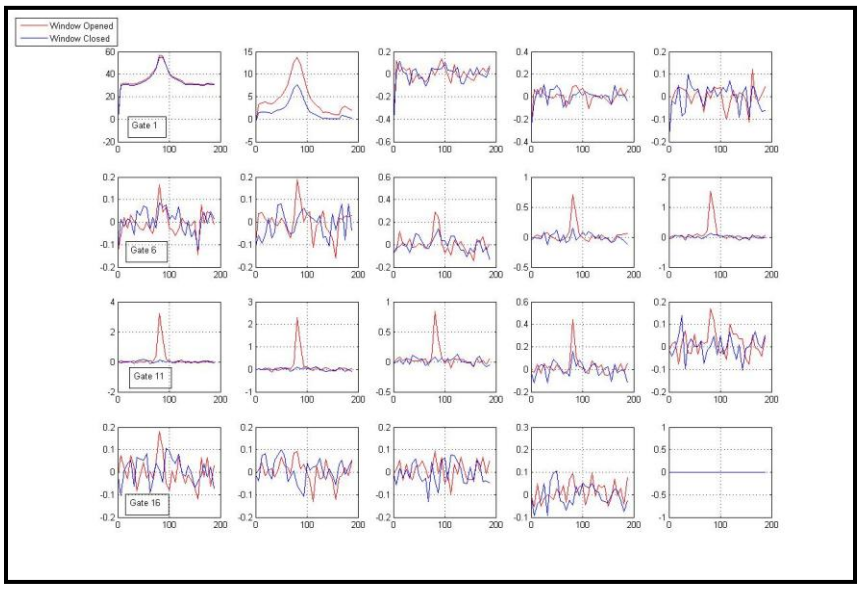

Figure 4-11 Wind speed power spectrum measured through lab window (window closed) and window opened

Horizontal wind velocity measurement along the line of sight was conducted throught our laboratory window. In this measurement the laser power was not at its full scale because of the optical amplifier's SBS limitation. Backscattered signals were processed with range gates of and 128 samples corresponding to a 48 range resolution. Fig. 4-12 shows the power spectrum of gated received signals of which wind velocity can be estimated for up to a proximately of 900m.

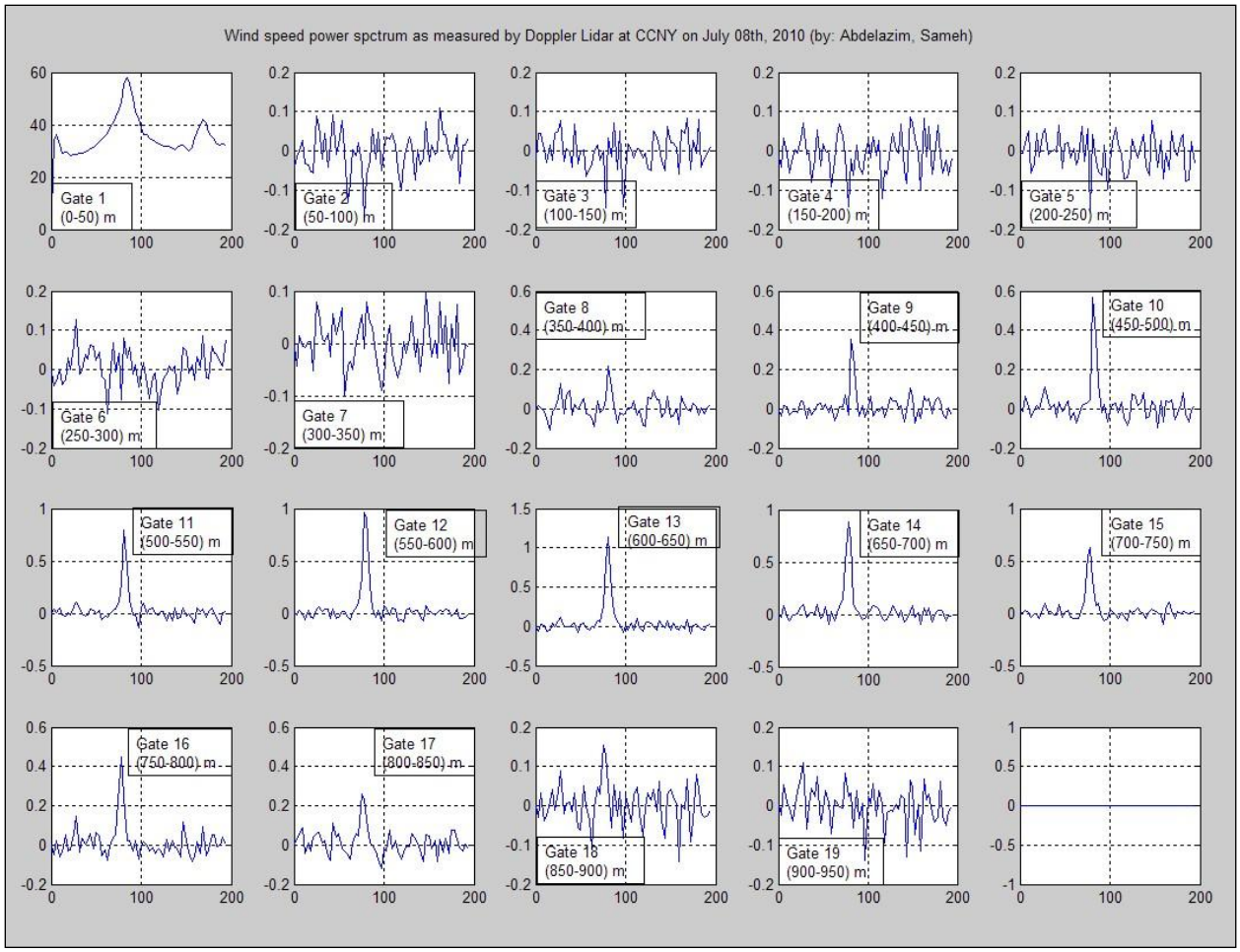

Figure 4-12 Time gated backscattered signals' power spectra of a measurement along the line of sight through the laboratory window. 128 samples per gate are chosen, which correspond to a 48 m range resolution.

4.6 Real-Time Wind Measurement

In this section, real-time wind measurements are reported. Real-time measurements are recoded continuously, thanks to FPGA pre-processing techniques that allow for streaming of either power spectrum or autocorrelation of received signals instead of raw data. The instrument was installed in our research vehicle that is located at the City College of New York at upper Manhattan, New York (latitude: 40.49°N, longitude: 73.56°W), Fig. 4-13. Laser pulses are transmitted into the atmosphere through an opening in the vehicle's roof, Fig. 4-14. The following subsections

introduce vertical wind velocity measurements using FFT and autocorrelation pre-processing techniques. Horizontal wind speed measurement is also introduced.

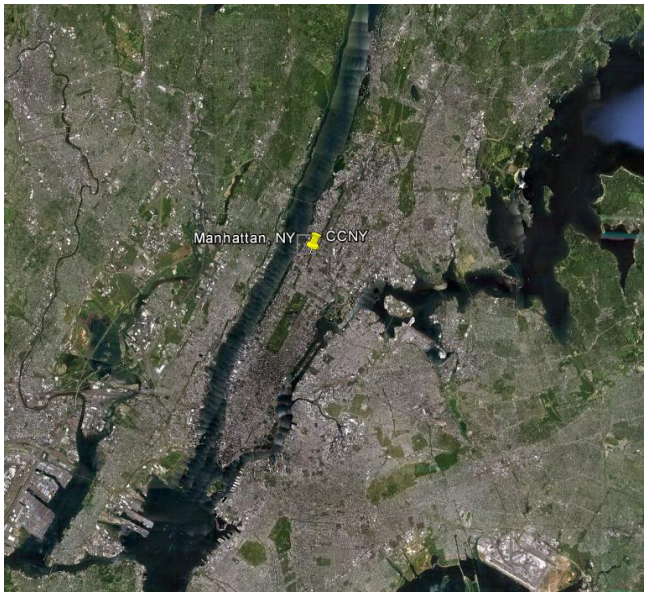

Figure 4-13 The site where wind measurements were conducted at the CCNY

Figure 4-14 The research van where the Doppler lidar instrument is installed for field measurements

4.6.1 Vertical Wind Velocity Measurement Using FFT Pre-Processing Algorithm

Received signals are pre-processed on the FPGA by dividing backscattered signals into 128 samples per a range gate, corresponding to a 48m range resolution, ranging from 96 m to approximately 3km. Power spectra of the gated signals are estimated by calculating the FFT and then accumulating the resulted power spectrum for 10,000 shots. Accumulated power spectra are streamed to the host PC for further processing and archiving. Wind velocities are calculated by estimating the Doppler frequency shift of the received signals' power spectrum. Three different techniques are used for velocity estimate; spectral peak search, power spectrum peak curve fitting, and MLE algorithm.

In the following section, vertical measurement results of wind velocity measured during the month of August of 2011 are introduced and discussed. The vertical wind velocity profile was measured on August 17th, 2011 from 14:35-to-16:35 EDT. Scattered signals' power spectrum is flattened by dividing it by the power spectrum of input signals to the photo-detector when no scattered signals are present. A signal intensity threshold is chosen below which any returned signals power spectrum is ignored. Vertical wind velocity estimated by finding the power spectral peak is shown in Fig. 4-15A. In Fig. 4-15B, vertical wind velocity is estimated by fitting the power spectrum profile around the peak value to a Gaussian curve shape and finding the centroid of the fitted shape. It is clear that fitting the power spectrum to a Gaussian curve shape increases the velocity estimate resolution, because it allows for sub-frequency bin resolution. It is also shown that updrafts and downdrafts are very visible, which is typical weather condition during a summer afternoon.

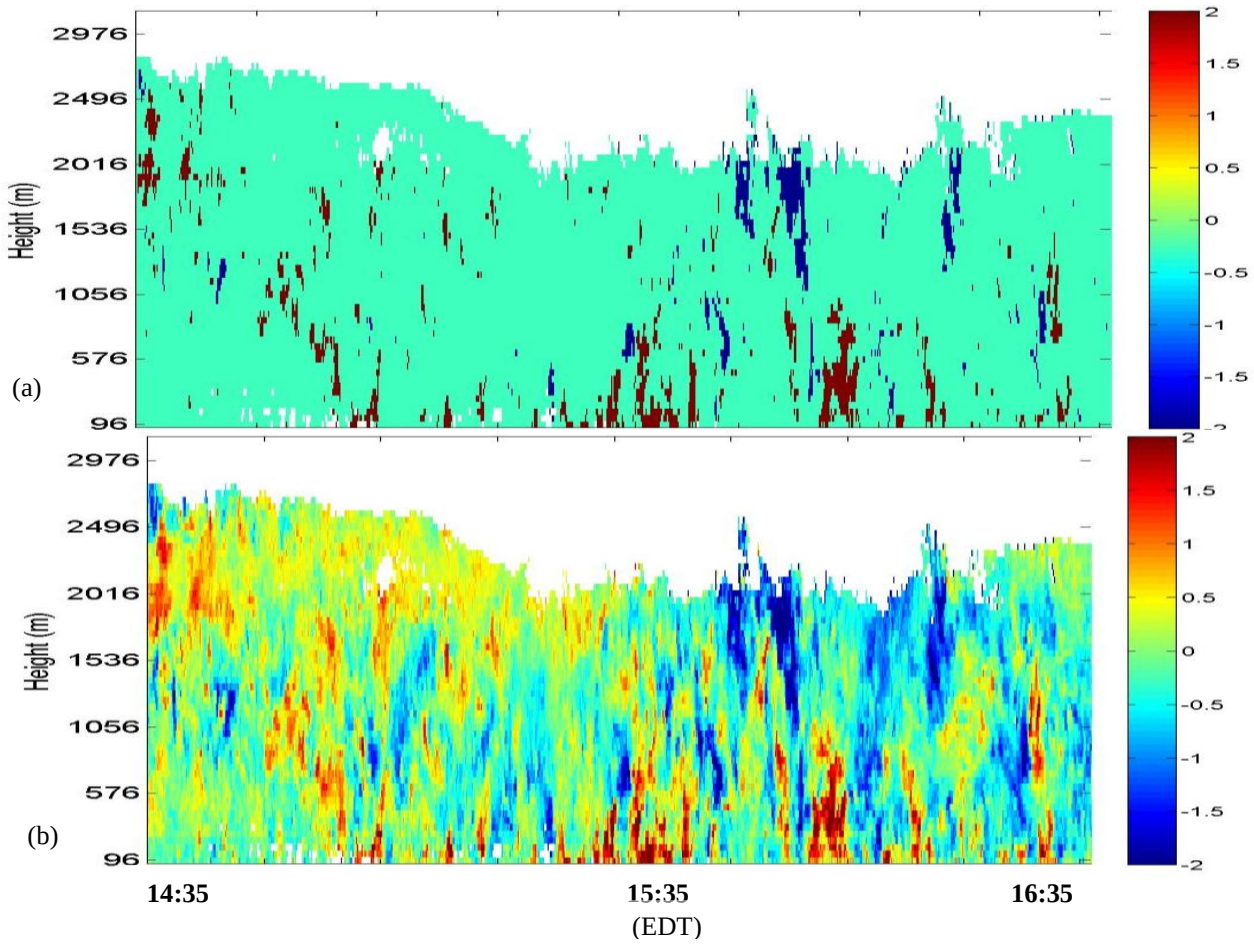

Figure 4-15 Vertical wind velocity profile (m/s) measured at CCNY on August, 17th, 2011 from 14:35-16:35 PM EDT estimated by power spectrum peak search (a), and Gaussian curve fitting (b)

Fig. 4-16 shows wind velocity estimate precision, which is estimated using the CRLB equation given by:

$$\delta_{CRLB} = 0.83 \frac{1+0.32(SNR)}{\sqrt{N}.(SNR)} \qquad (4\text{-}14)$$

where; SNR is the wide band signal to noise ration, and N is the number of accumulated laser shots. It is shown that in the areas where SNR was high, the precision is small as the estimate precision is inversely proportional to the SNR. Overall, a precision of the order of 8cm.s^{-1} or less is observed.

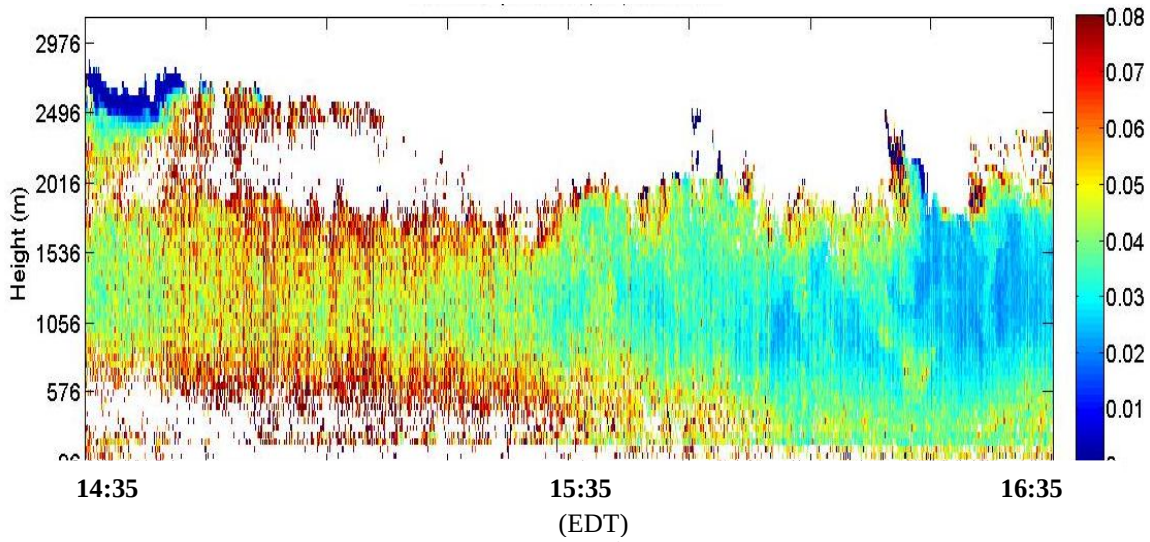

Figure 4-16 Vertical wind velocity estimate precision (m/s) measured at CCNY on August, 17th, 2011 from 14:35-16:35 PM EDT

Backscattered signals' power was estimated by integrating the area under the power spectrum curve, Fig. 4-17. The signal power was then range corrected by dividing estimated power by the overlap integral function given that the laser beam was focused at a known distance (approximately 2km), which was determined by scattering off a hard target (clouds).

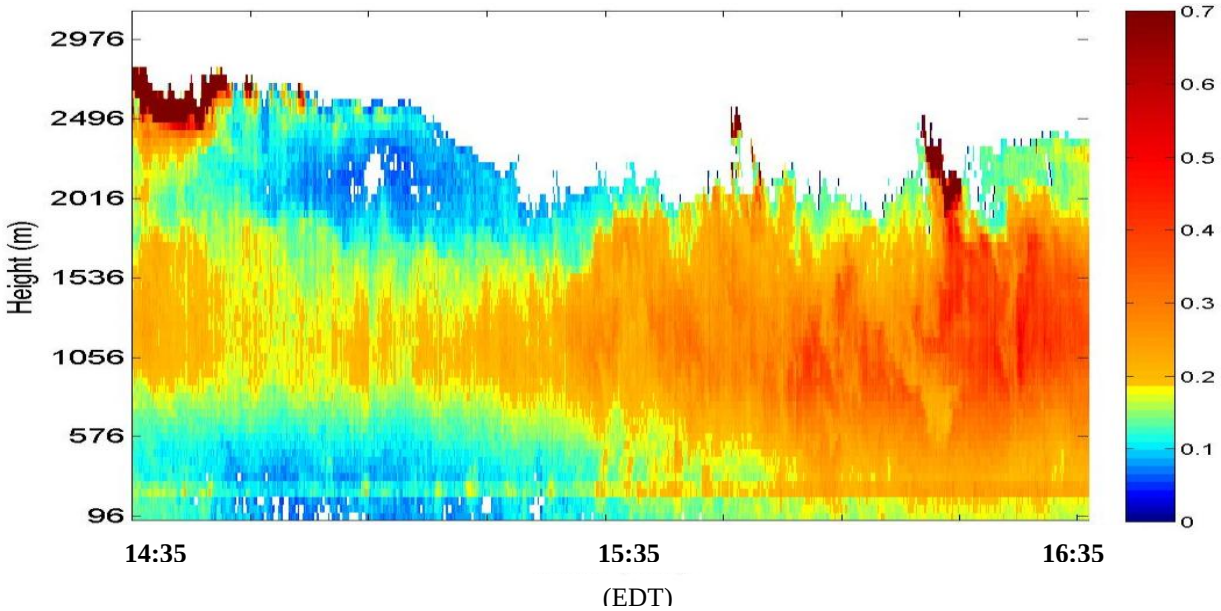

Figure 4-17 ratio of received signals power to shot noise power V.s. time and height measured at CCNY on August, 17th, 2011 from 14:35-16:35 PM EDT

Range correcting received signals' power allows for comparison our results with the results of a 1 µm direct detection lidar, which was operated at the remote sensing laboratory of the City College of New York during the same time of operation. Fig. 4-18A and B show a good agreement of signal intensity profile's and cloud patterns with that of the CDL at 14:30, 15:50 and 16:15. It is also noticed that signal intensity increased significantly at a height of approximately 2700 m at 14:30 and 16:20 due to clouds at that height. Both lidars also show a gradual increase of aerosol concentration as a function of time between 16:00 and 16:30. The Doppler lidar's signals power spectrum comparison with the direct detection measurements proves that the obtained power spectrum is valid, and as a result, the frequency shift has to be due to: a) RF signals going into the AOMs, and b) Doppler shift caused by wind. Since the LO is stable (laser's line width is approximately 3 KHz), and the jitter between the RF signal and the ADC's clock is less than 100 KHz, then measured frequency shifts are solely due to Doppler shifts caused by wind velocity.

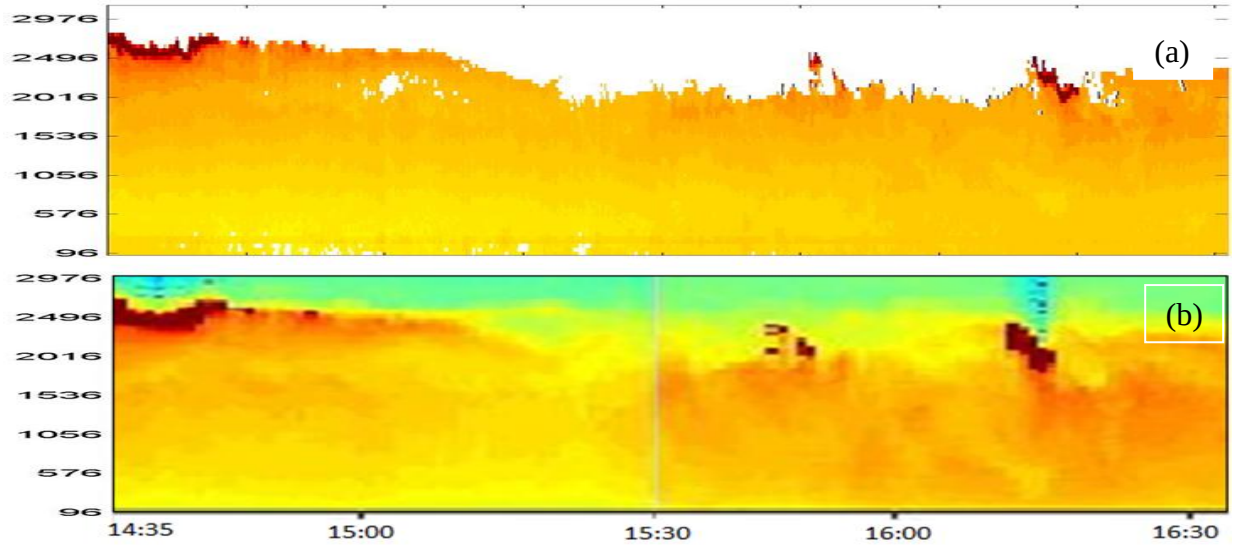

Figure 4-18 Range corrected backscattered signal power V.s. time and height (a) and the 1 µm direct detection lidar signal power vs. height and time (b). Both signals power profiles show a good agreement around 14:35, 15:60, and 16:15, where clouds' patterns are observed at the same heights. Aerosoles concentration profiles also show a good agreement in the two measurements.

As mentioned in chapter II, the power spectrum of received signals needs to be corrected for the detector's non-flat gain shape. This is done by dividing the power spectrum of received signals by a power spectrum of reference signals. The reference signals are obtained using the detector while LO is turned on and no received signals are present (final amplifier is turned off), i.e. this power spectrum only represents shot noise. The SNR can be represented as follows:

$$SNR = \frac{P_{sig}^{(N)}}{P_{nois}} \qquad (4\text{-}15)$$

where, P_{nois}, and $P_{sig}^{(N)}$ are the noise and signal power accumulated over N pulses, respectively. For a shot noise limited receiver system, the noise power is due to shot noise: $P_{shot}^{(N)}$, which can be given by:

$$P_{shot}^{(N)} =< P_{shot} > \pm \sigma_{shot}^{(N)} \qquad (4\text{-}16)$$

where; $< P_{shot} >$ is the average shot noise power, which is a fixed level characterized by the laser source, and $\sigma_{shot}^{(N)}$ is the shot noise variation around its fixed level when averaged N times (standard deviation of the shot noise for N accumulation). The average shot noise fixed power level is given by:

$$P_{shot} = 2eiB \qquad (4\text{-}17)$$

where; *e* is the electronic charge, *i* is the photo-current, and *B* is the detector's bandwidth. Since shot noise follows Poisson's distribution then the averaged shot noise variation $\sigma_{shot}^{(N)}$ can be expressed as:

$$\sigma_{shot}^{(N)} = \frac{P_{shot}}{\sqrt{N}} \qquad (4\text{-}18)$$

Therefore, SNR, equation 4-13, can be re-written as:

$$SNR = \sqrt{N}\left(\frac{P_{sig}^{(N)}}{P_{shot}}\right) \quad (4\text{-}19)$$

Correcting for the non-flat detector's gain involves dividing accumulated measured signals' power spectrum by a reference signal's power spectrum, which can be expressed as:

$$\frac{P_{measured}}{P_{ref}} \quad (4\text{-}20)$$

where; $P_{measured}$ is the measured signals' power, which includes both signals and noise powers, and can be expressed as:

$$P_{measured} = P_{sig}^{(N)} + P_{Noise} \quad (4\text{-}21)$$

$$= P_{sig}^{(N)} + P_{shot}^{(N)} \quad (4\text{-}22)$$

P_{ref} is the power spectrum of LO when no received signals are input to the detector, i.e. only represents shot noise. As a result, the ratio between measured power and ref. power (equation 4-17) can be written as:

$$\frac{P_{measured}}{P_{ref}} = \frac{P_{sig}^{(N)}}{P_{shot}^{\prime(N)}} + \frac{P_{shot}^{(N)}}{P_{shot}^{\prime(N)}} \quad (4\text{-}23)$$

This quantity has an average value $< \frac{P_{measured}}{P_{ref}} >$ and a (fluctuation) standard deviation value σ that can be expressed as follows:

$$< \frac{P_{measured}}{P_{ref}} >= < \frac{P_{sig}^{(N)}}{P_{shot}^{\prime(N)}} > + < \frac{P_{shot}^{(N)}}{P_{shot}^{\prime(N)}} > \quad (4\text{-}24)$$

$$= \frac{P_{sig}}{P_{shot}} + 1 \qquad (4\text{-}25)$$

As for the standard deviation, it can be expressed as:

$$\sigma = \sqrt{\frac{2}{N}} \qquad (4\text{-}26)$$

In our analysis, we calculate the following parameter: $\frac{P_{measured}}{P_{ref}} - 1$ by dividing signals power spectrum by a reference signals power spectrum and then subtracting one, which results to:

$$\frac{P_{measured}}{P_{ref}} - 1 = \frac{(SNR)}{\sqrt{N}} + \sqrt{\frac{2}{N}} \qquad (4\text{-}27)$$

Therefore, when accumulating 10,000 pulses and in order to extract signal out of noise, the signal power should be at least equal to noise power, i.e. SNR =1. As a result, the value we calculate is equal to: $\frac{1}{\sqrt{10{,}000}} + \frac{\sqrt{2}}{\sqrt{10{,}000}}$ = 0.024, which is the value we set as a threshold below which received signals are ignored. It is also worth noting that the signal power we report is not really the signal power, but it's the signal power normalized to the shot noise, in other words, it is the SNR for a single shot.

To verify that the standard deviation (fluctuation) of the result of equation 4-23 (measured signals divided by reference signals) follows: $\sqrt{\frac{2}{N}}$ relationship, the ratio of measured noise to a reference shot noise was characterized by calculating the standard deviation of approximately 250k points. Each point is a measured noise power accumulated 10k times divided by a reference shot noise power also accumulated 10k times. The standard deviation is found to be 0.0155, i.e. 10% more than what we estimated in equation 4-26. This difference can be the result of effect of other noise sources such as thermal noise, i.e. system is not 100% shot

noise limited. In other words, the total noise in the receiver is not solely due to shot noise, but other noises have a small contribution as well. The histogram of the power ratio of the 250k analyzed points is shown in figure 4-19.

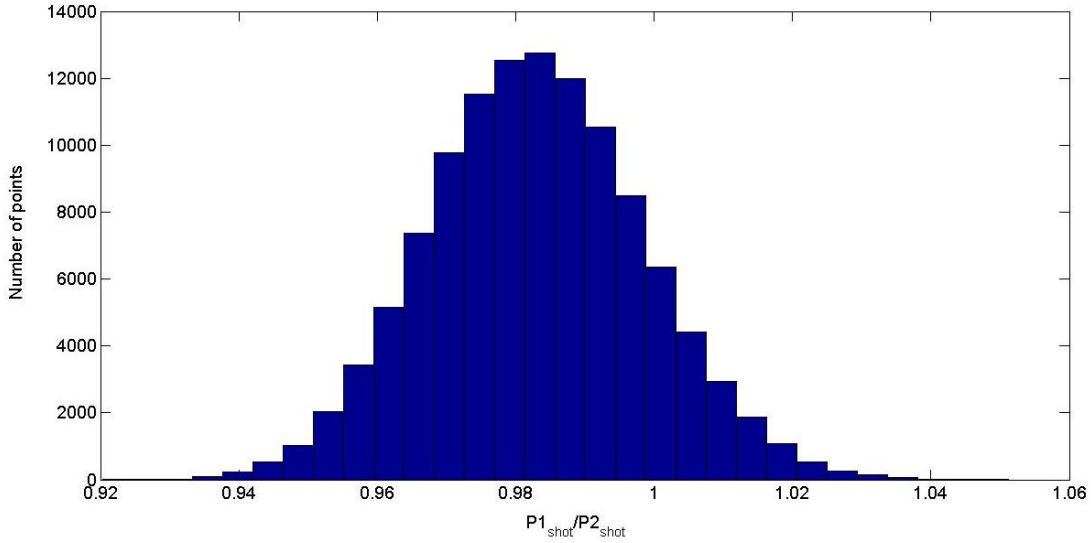

Figure 4-19 The histogram of the ratio of measured noise power (10k accumulations) to a reference shot noise power (also 10k accumulations)

Vertical wind measurements are reported in the following section for the following days: August 2nd, 4th, 5th, 18th, and December 13th, 2011. Figures 4-20, 4-21, 4-22, 4-23, and 4-24 show wind measurement results for these days.

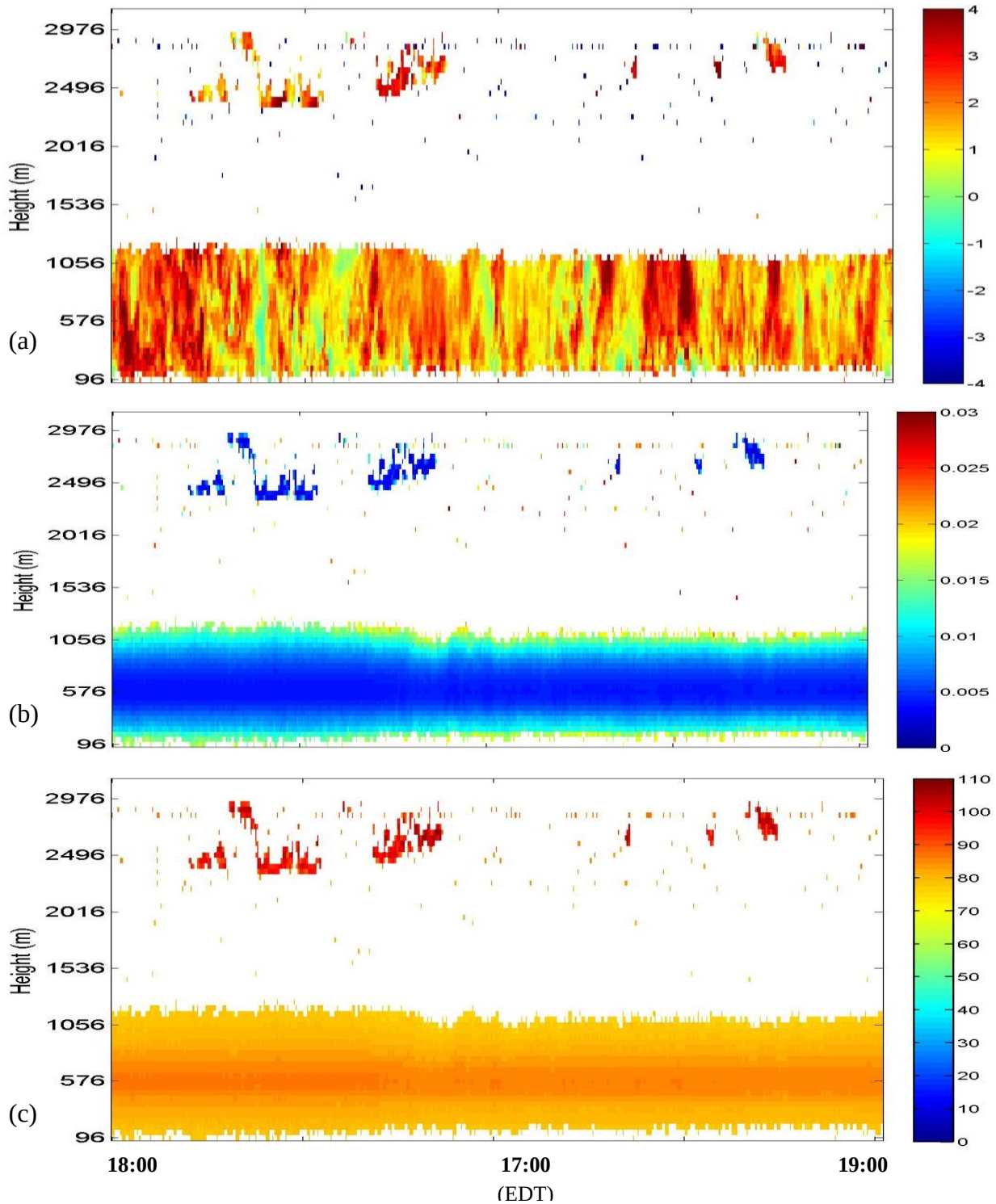

Figure 4-20 Vertical wind velocity (m/s) (a), velocity estimate precision (m/s) (b), and range corrected signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 18:00 – 19:00 PM EDT on August 2nd, 2011. It is shown that at the presence of clouds at heights between 2.5-3km signal power is very high and as a result the velocity estimate precision is very small.

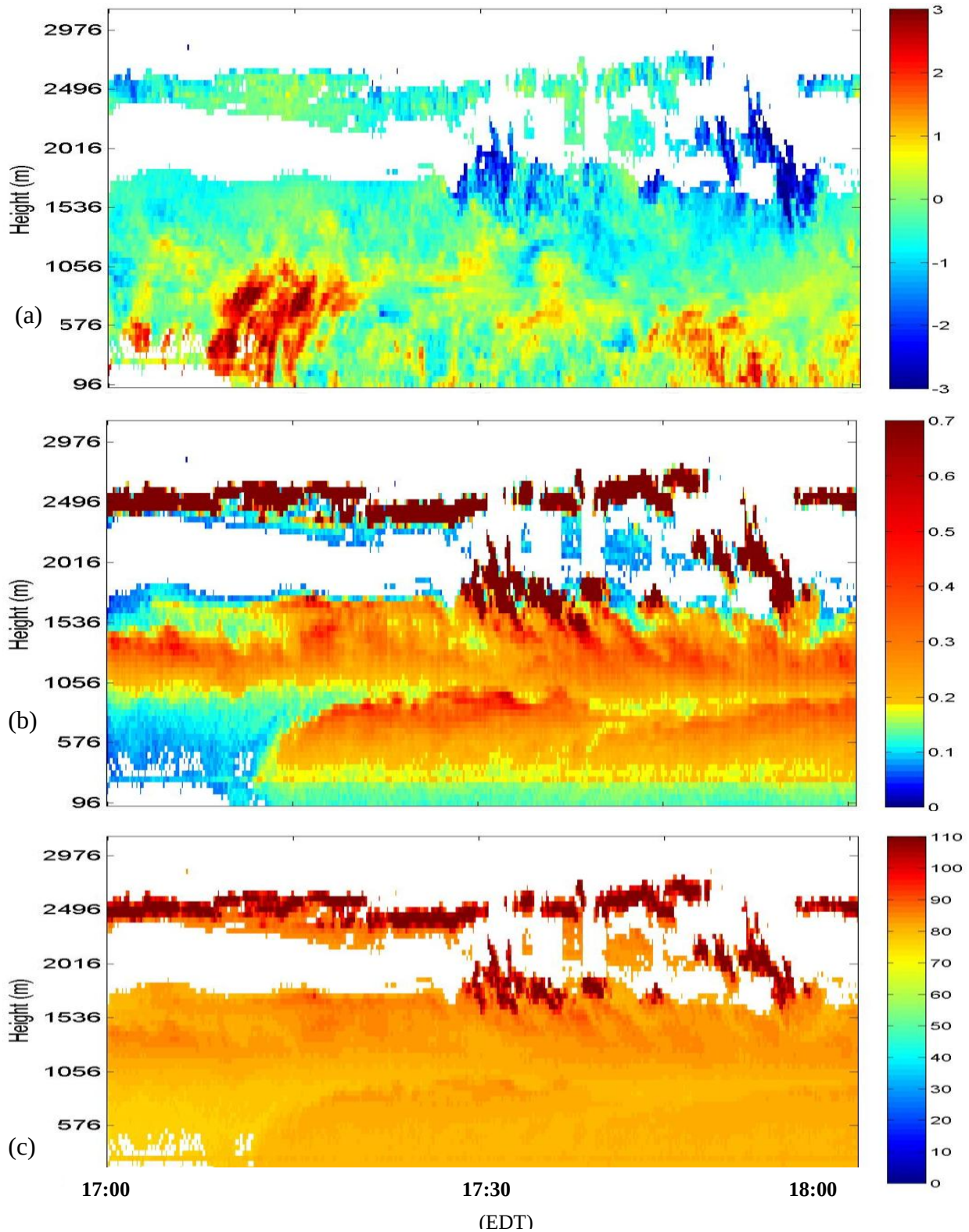

Figure 4-21 Vertical wind velocity (a), and ratio of received signals power to shot noise power (b), and range corrected signals' power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 17:00 – 18:00 PM EDT on August 4th, 2011.

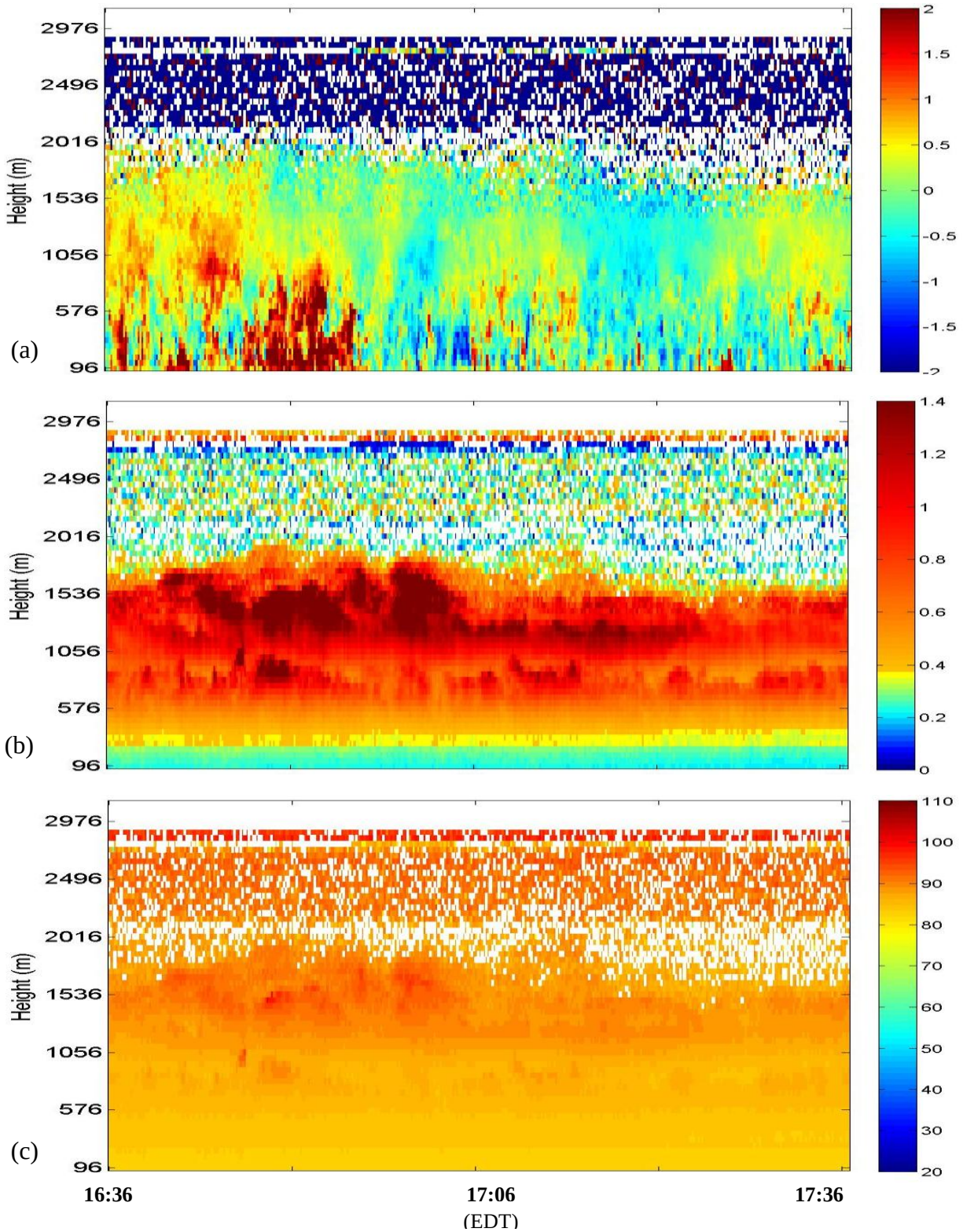

Figure 4-22 Vertical wind velocity (m/s)(a), ratio of eceived signals power normalized to shot noise power (b), and range corrected signals' power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 16:36 – 17:36 PM EST on August 5th, 2011.

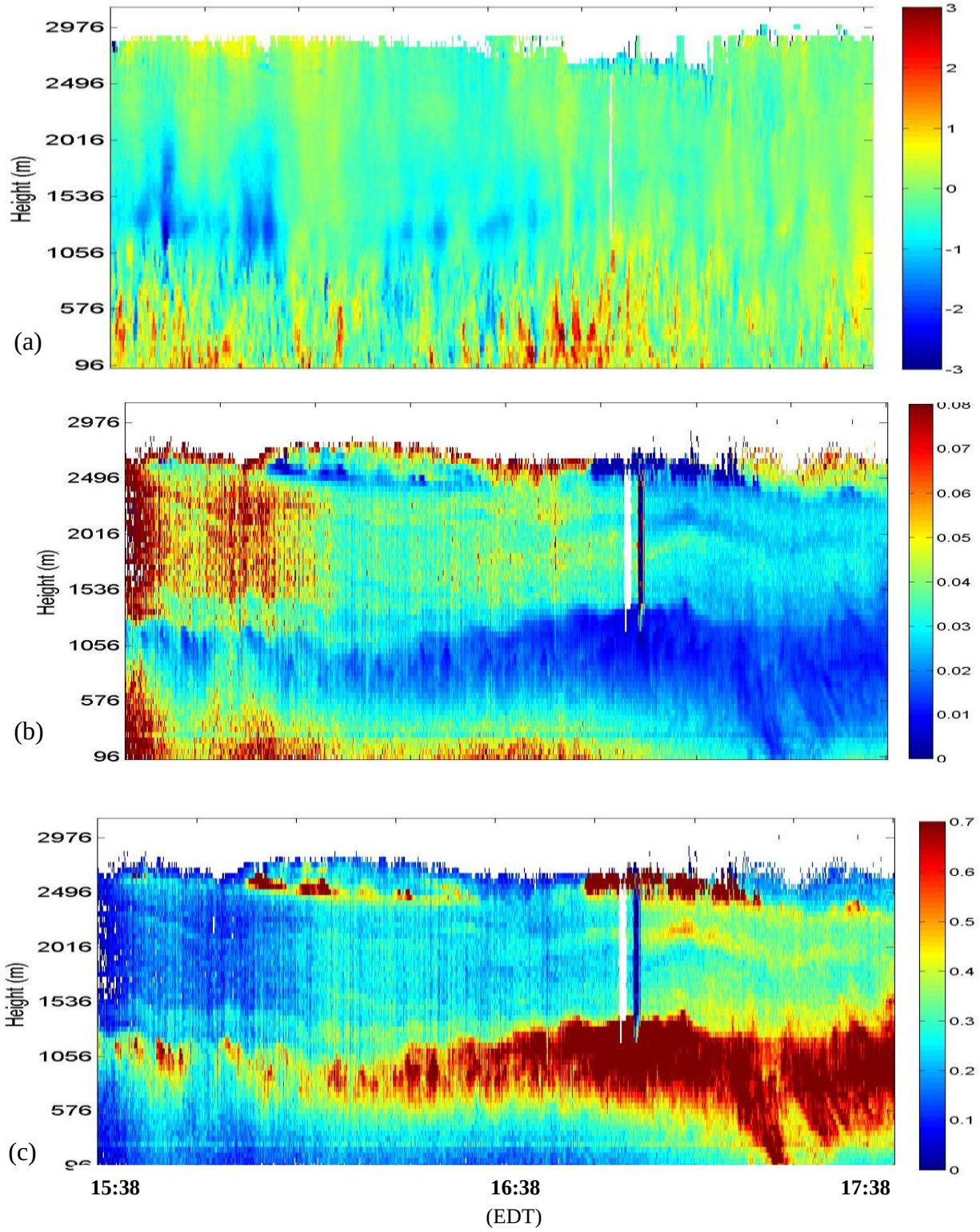

Figure 4-23 Vertical wind velocity (m/s) (a), velocity estimate precision (m/s) (b), and ratio of signals power to shot noise power (c) vs. time and height measured at CCNY remote sensing Laboratory between 15:38 – 17:38 PM EDT on August 18th, 2011.

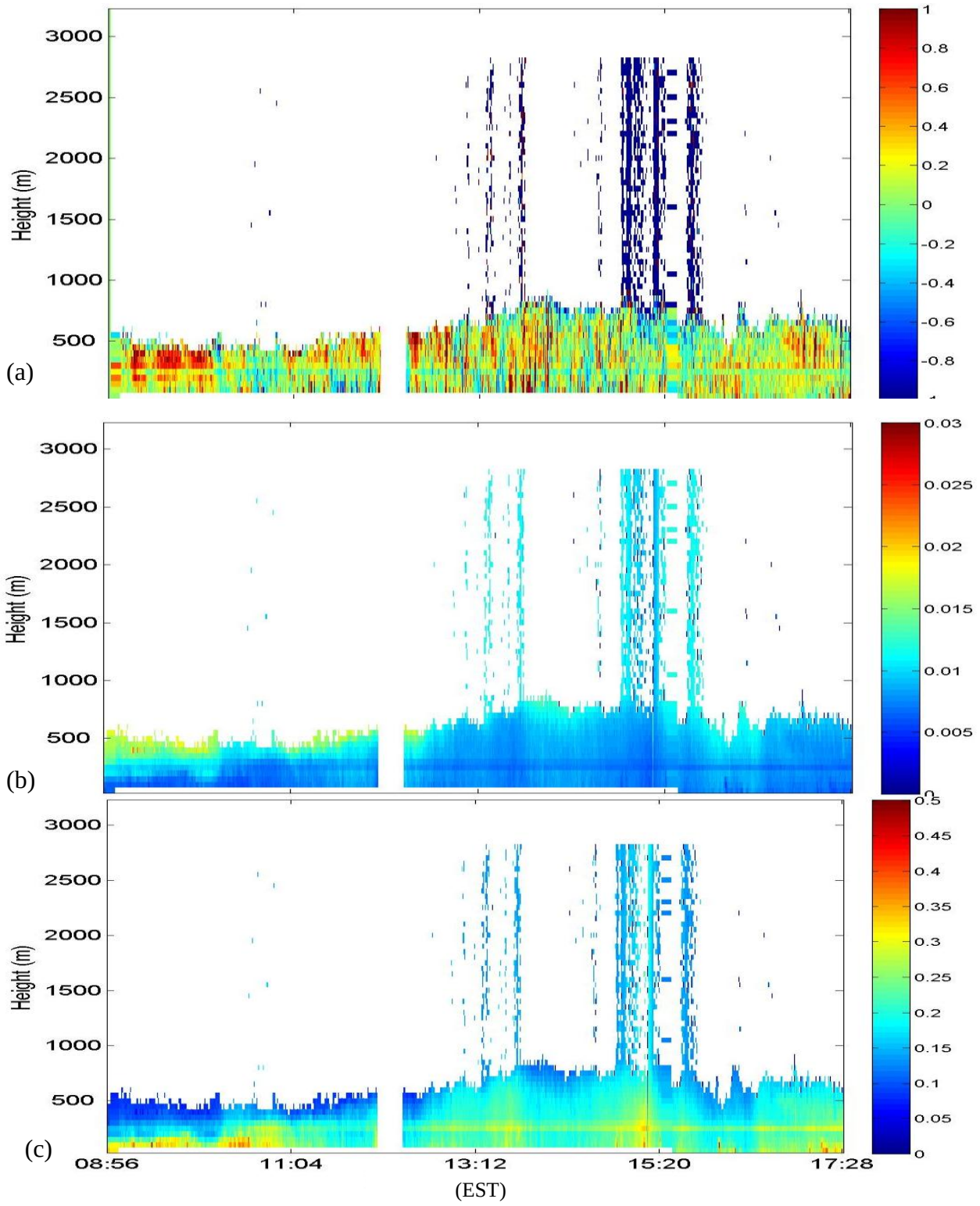

Figure 4-24 Vertical wind velocity (m/s) (a), Velocity estimate precision (m/s) (b), and ratio of signals power to shot noise power (c) vs. time and height measured at CCNY remote sensing Laboratory between 8:56 – 17:28 EST on Dec. 13th, 2011.

4.7 Horizontal Wind Velocity Measurements

3D wind vector components can be constructed from the three line of sight measurements and horizontal wind speed and direction can be estimated (Appendix III), and wind speed is presented using wind barb plots. Wind barbs are a convenient way of representing wind speed and direction. The orientation of the wind barb indicates the direction from which the wind is blowing. The magnitude of the speed is represented in knots (1.15 mph) by the number of barbs, where a full-barb represents 10 knots, a half-barb represents 5 knots, and a flag represents 50 knots. Fig. 4-25 shows a wind barb representing wind speed of 15 knots blowing from north east.

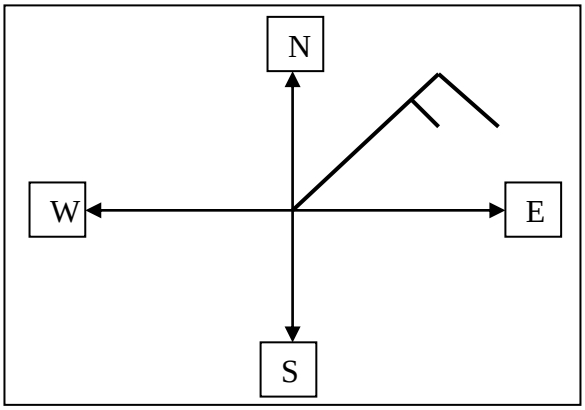

Figure 4-25Wind barb representation of wind speed and direction. Barbs represent speed in knots (A full-barb represents 10 knots, a half-barb represents 5 knots, and a flag represents 50 knots) the direction of the barbs shows where the wind is blowing from.

The following section reports the 3D (horizontal) wind velocity measurements obtained using FPGA FFT pre-processing algorithm. 3D wind vector components were constructed via the three line of sight measurements and horizontal wind speed and direction was estimated. Fig. 4-26 (a) and (b) shows wind barbs plots as a function of time and height measured on Dec 4th, 2011 during approximately 4 hours of scan mode operation. Fig. 4-27 (a), (b), and (c) show vertical wind speed, backscattered signal power, and range corrected backscattered signal power, respectively.

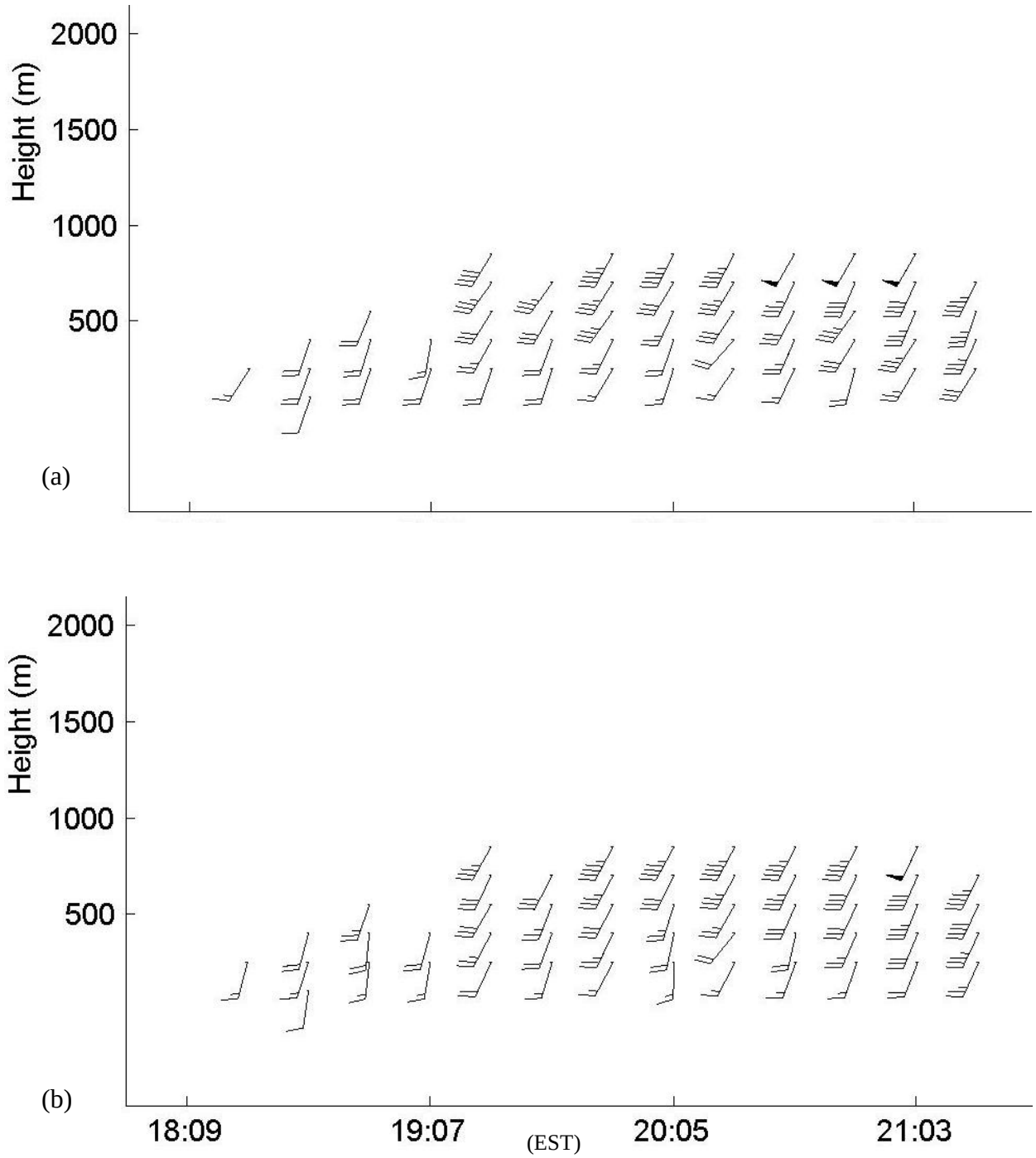

Figure 4-26 Wind barbs estimated using discrete spectral peak (a), and by using power spectrum Gaussian curve fitting (b) vs. time and height measured at CCNY remote sensing Laboratory between 18:09– 22:02 PM EST on Dec. 4th, 2011.

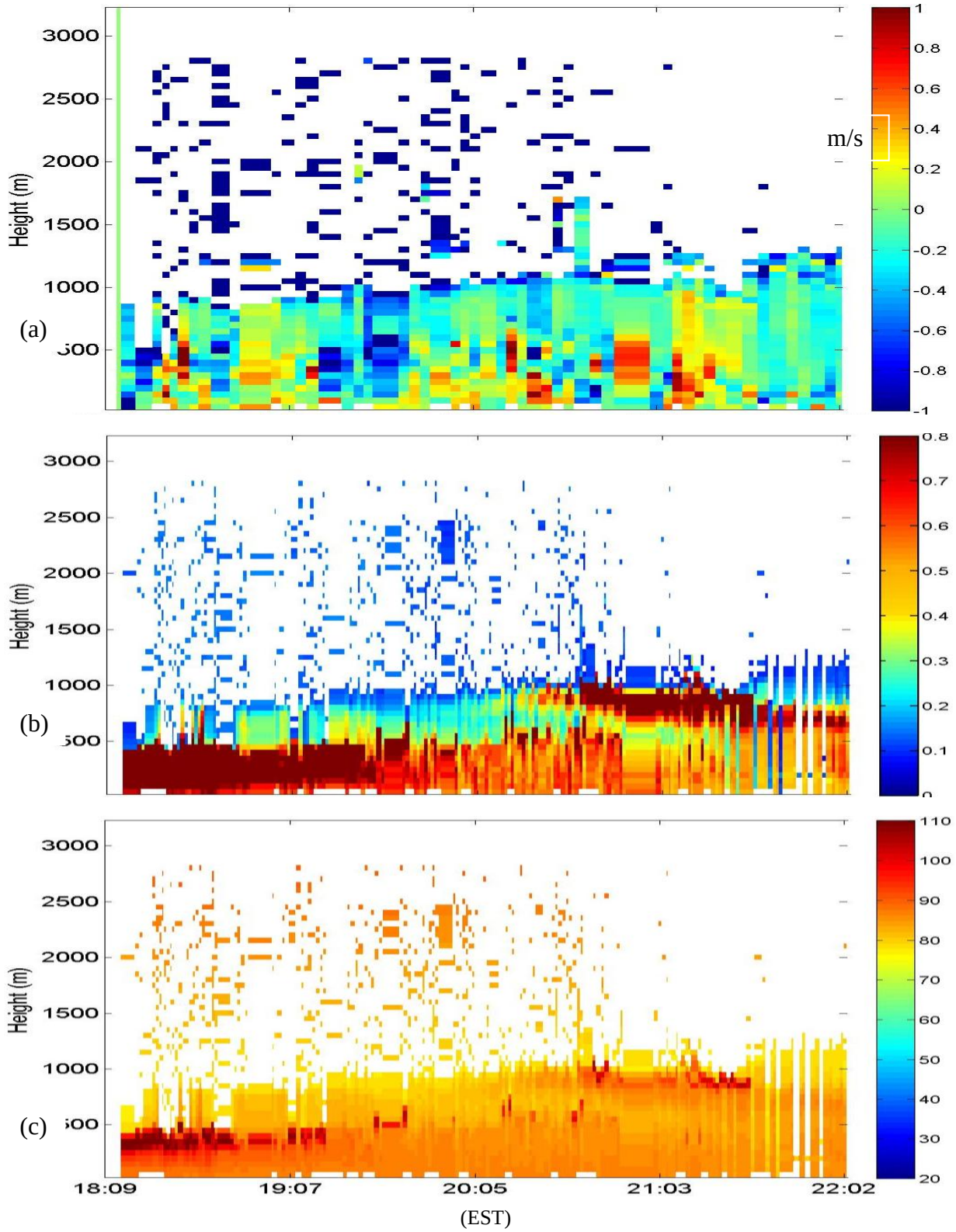

Figure 4-27 Vertical wind velocity (a), ratio of received signals power to shot noise power (b), and range corrected signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 18:09– 22:02 PM EST on Dec. 4th, 2011.

In the following section, horizontal wind measurements are reported for the following days: Dec, 5th, 8th, and 11th of 201, Fig. 4-28, 4-29, 4-30, and 4-31

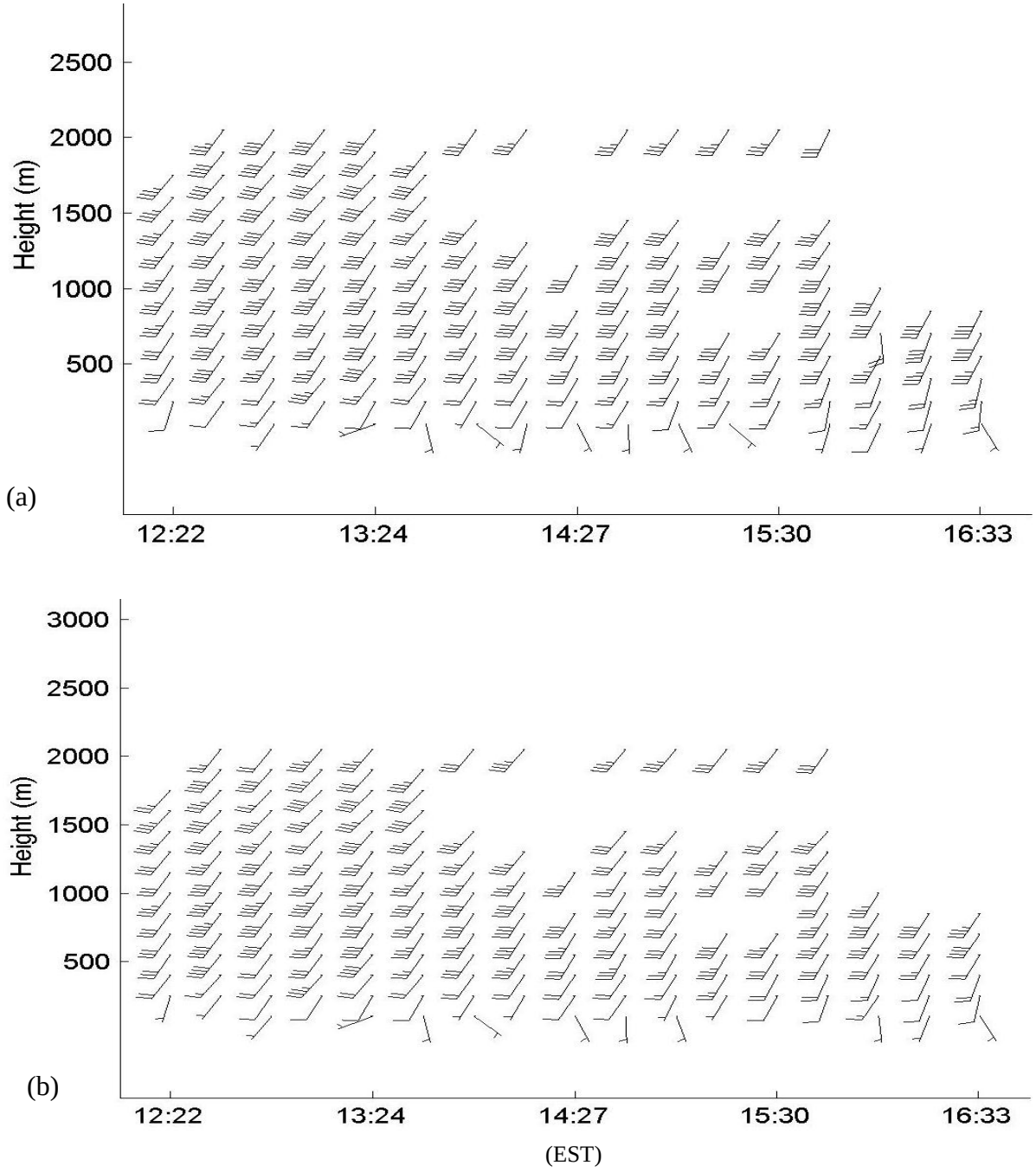

Figure 4-28 Wind barbs estimated using discrete spectral peak search (a), and using power spectrum curve fitting (b) vs. time and height measured at CCNY remote sensing Laboratory between 12:22– 16:33 PM EST on Dec. 5th, 2011.

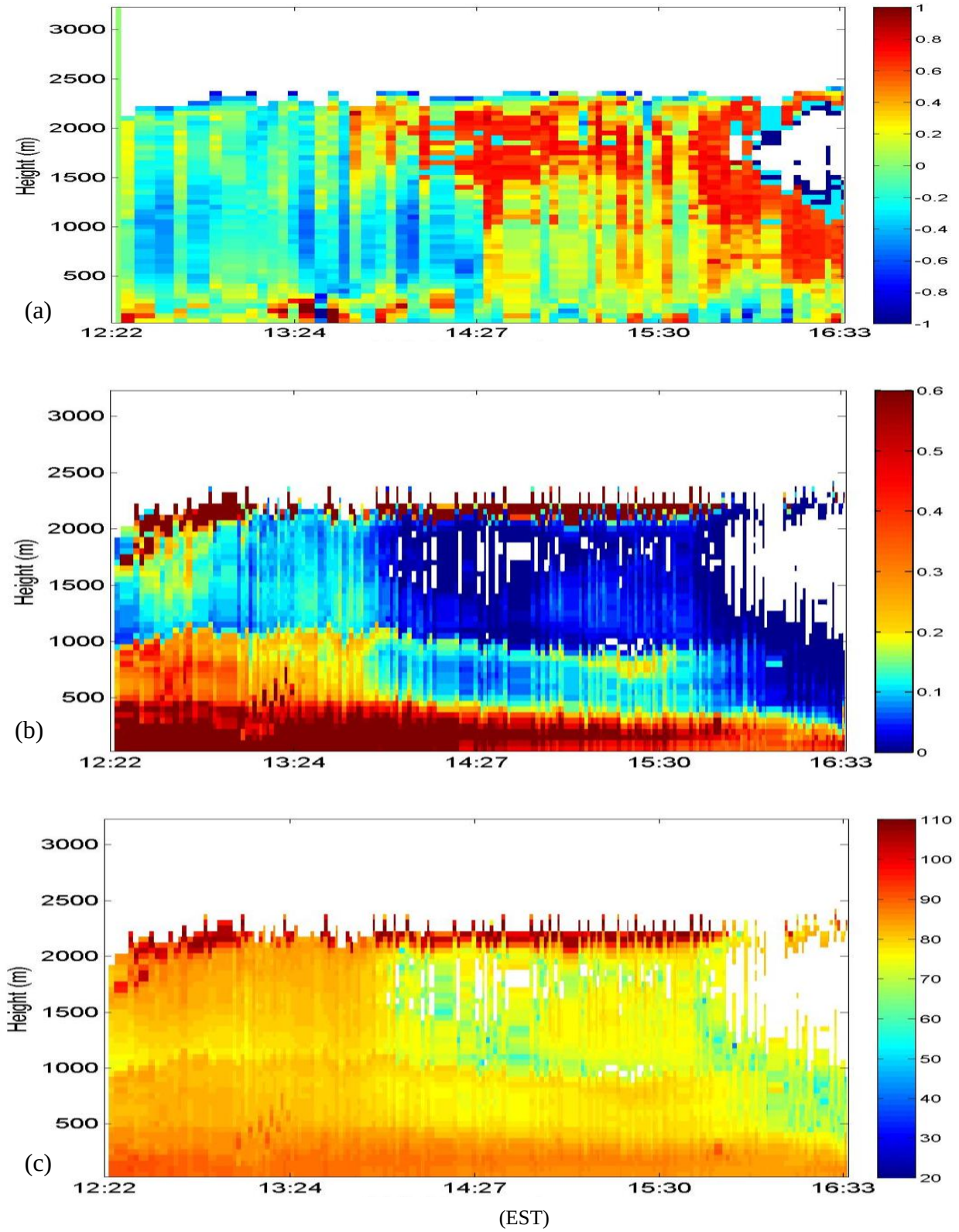

Figure 4-29 Vertical wind velocity (m/s) (a), normalized backscattered signals power to shot noise (b), and range corrected signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 12:22–16:33 PM EST on Dec. 5th, 2011.

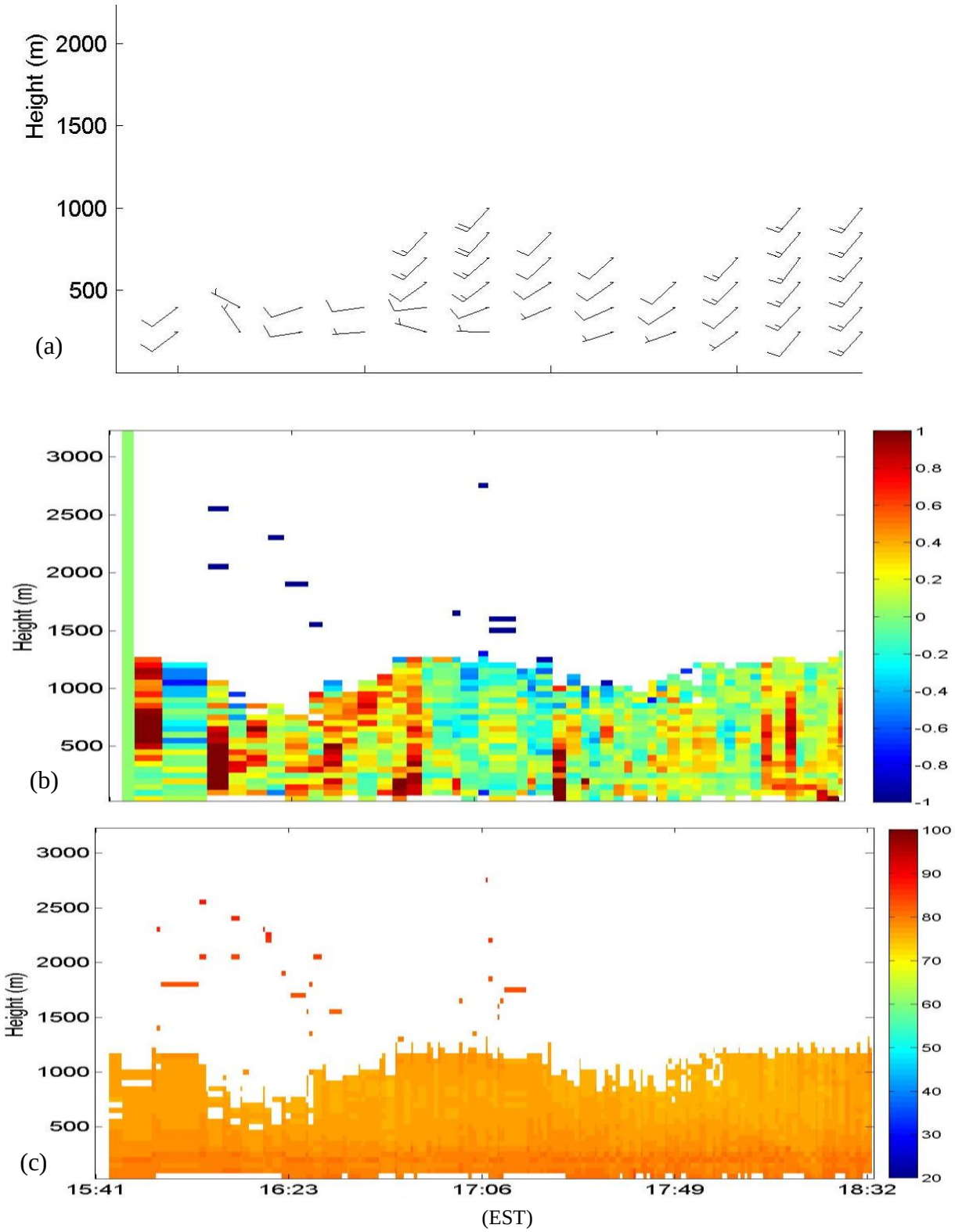

Figure 4-30 Wind barbs (a), Veritcal wind velocity (m/s) (b), and range corrected signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 15:41– 18:32 PM EST on Dec. 8th, 2011.

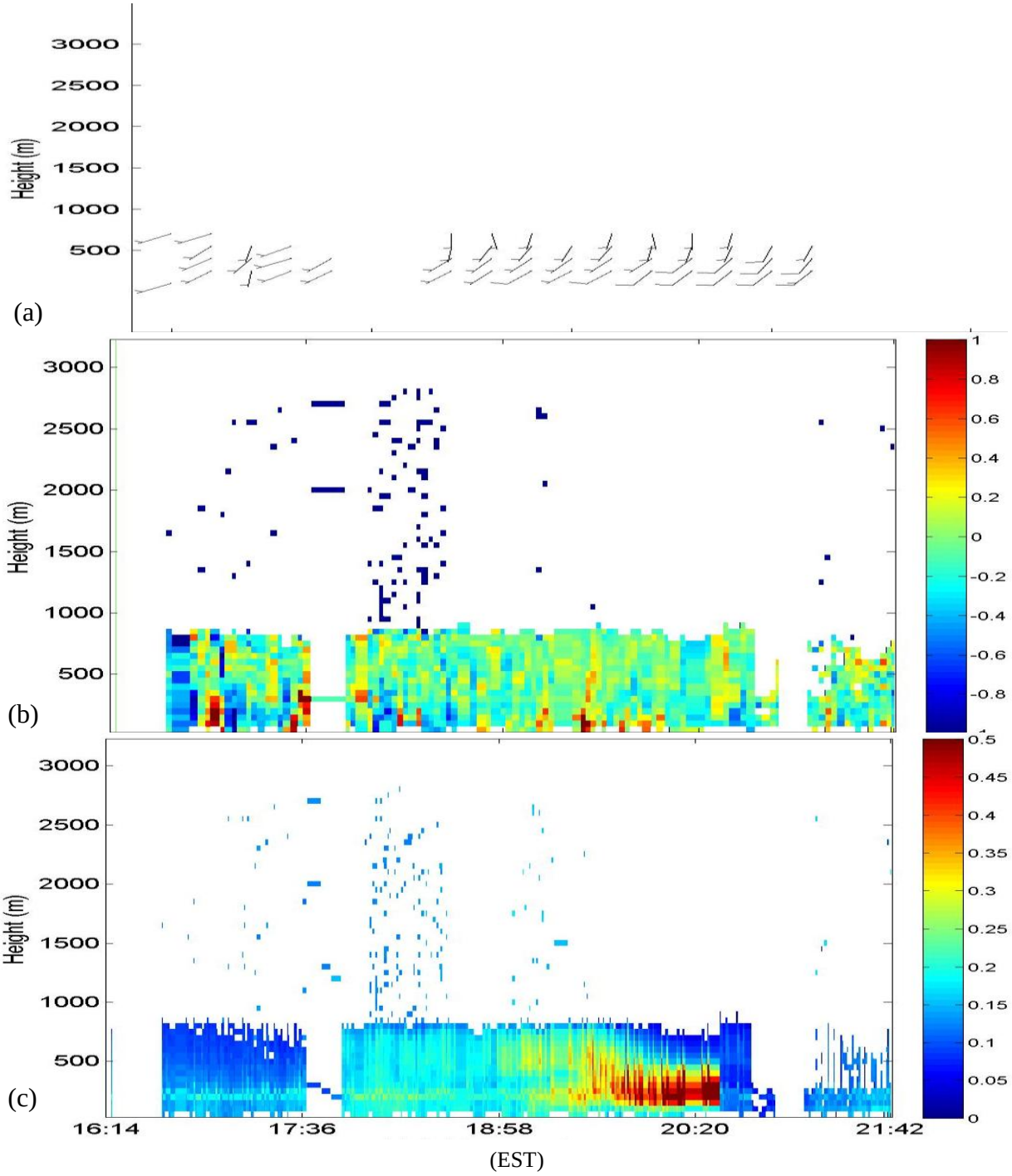

Figure 4-31 Wind Barbs (a), Vertical wind velocity (m/s) (b), ratio of received signals power to shot noise power (c) vs. time and height measured at CCNY remote sensing Laboratory between 16:14– 21:42 PM EST on Dec. 11th, 2011.

4.8 Chirp Analysis

It is noticed that vertical wind velocity changes its direction rapidly at the bottom of a cloud that existed at a height of approximately 2500m between 14:35 to 14:50 of vertical measurements obtained on August 17th, 2011, Fig. 4-15. Having a rapid extinction in the velocity on the order of approximately one pulse's length (two spatial resolution range gates) is an indication of chirped laser pulse. A chirped laser pulse means that the frequency of the laser signals increases (up-chirp) or decreases (down-chirp) with time along the pulse. To investigate the presence of a chirp in the laser pulse, Lidar signals scattered off a hard target, 650m away, were analyzed in two different ways; a) by dividing the hard target's scattered signals into two halves (leading and trailing) and calculating the FFT of both of them, b) by estimating the Doppler shift across the hard target's scattered signals by calculating the FFT of a time domain signals' window that is swept across the scattered signals.

a) Dividing the hard target's scattered signals into two halves.

The power spectrum of both edges is calculated, Fig. 4-32. For a non-chirped laser pulse, both edges should have zero Doppler shift (a stationary target). However, if the laser pulse is chirped, then the Doppler shift of the leading edge and/or trailing edge will have a non-zero value.

10,000 laser shots were analyzed by dividing them into100 groups of 100 accumulated shots. For each group, the difference between Doppler shift of rising and trailing edges is calculated, and a histogram of the 100 groups is estimated, Fig. 4-33 A, B, and C. Laser power was adjusted to a level that doesn't saturate the photo-detector. To verify that the results are solely due to a chirped laser pulse and not signals' noise, the process is repeated multiple times. The laser power level was also reduced to two lower levels (160 mw, and 100 mw), and the difference between the two

Doppler shifts was calculated. It is shown that a frequency difference of about 0.5 MHz between the leading and trailing halves of the laser pulse exists. It is also shown that when the laser power level was reduced, the frequency shift difference is reduced, Fig. 4-34 and 4-35, which indicates that a frequency difference within the laser pulse is proportional to the laser power.

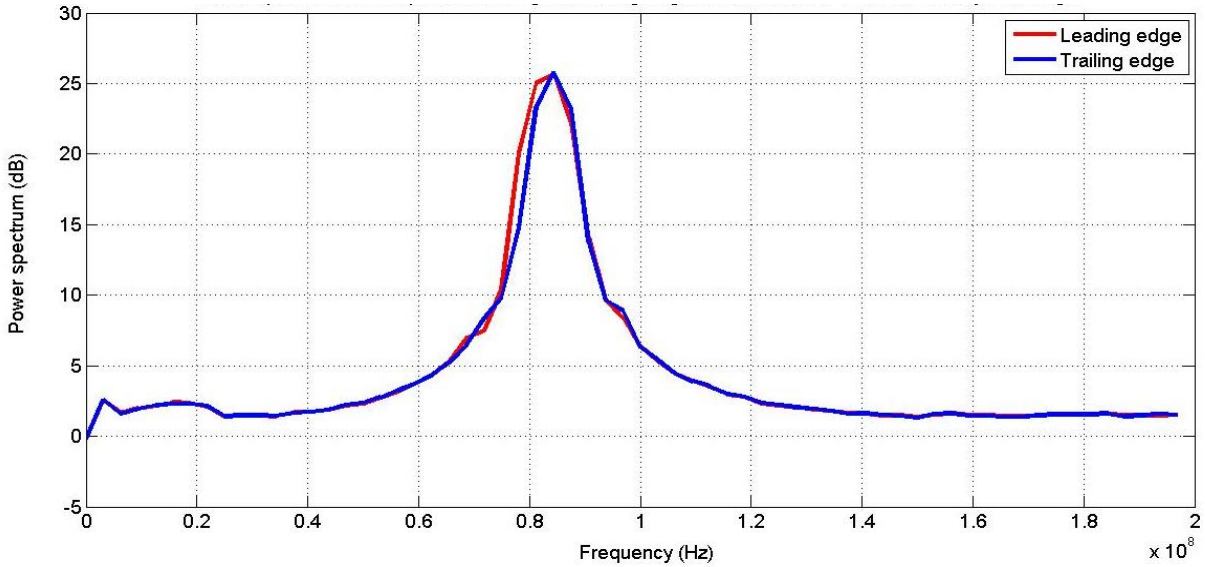

Figure 4-32 Power spectrum of leading and trailing halves of the laser pulse

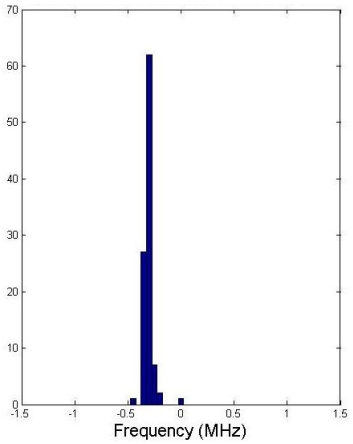

Figure 4-33A Histogram of the frequency shift of the leading edges of the laser pulse

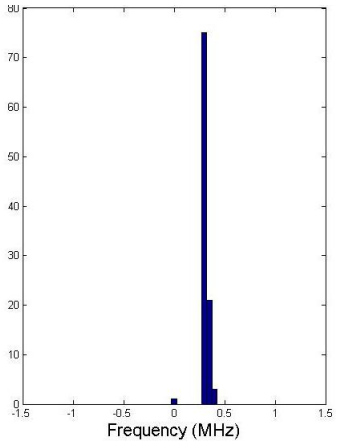

Figure 4-33B Histogram of the frequency shift of the trailing edges of the laser pulse

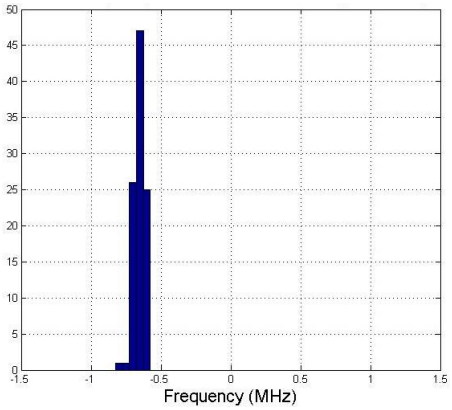

Figure 4-33C Histogram of the frequency shift difference between leading and trailing edges of the laser pulse when operating at laser's full power of 300m.W.

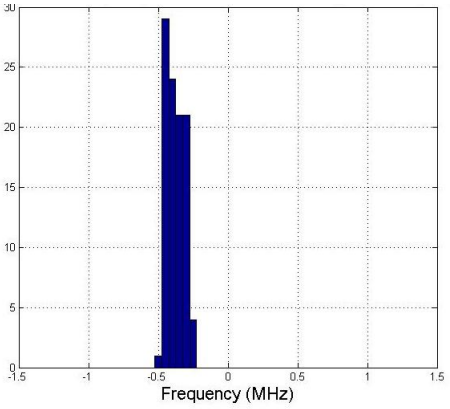

Figure 4-34 Histogram of the frequency shift difference between leading and trailing edges of the laser pulse when operating at 160m.W.

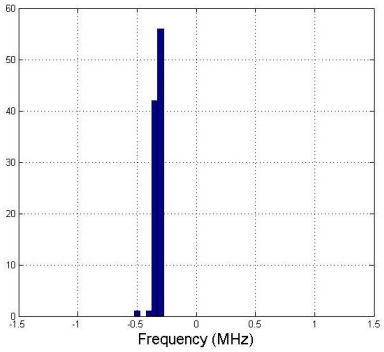

Figure 4-35 Histogram of the frequency shift difference between leading and trailing edges of the laser pulse when operating at 100m.W.

b) Estimating the Doppler shift across the hard target's scattered signals

In this analysis, the Doppler shift is estimated across the hard target's scattered signals by calculating the FFT of a window of time domain signals that is swept across hard target's scattered signals. Both hard target's scattered signals and the estimated Doppler shift of the sweeping window are shown in Fig. 4-36 and 4-37. It is clear that the estimated Doppler shift drifts from approximately -0.3 to 0.35 MHz across the hard target's scattered signals, which indicates that a laser pulse chirp exists. This chirp is what causes the rapid velocity change near the bottom of the cloud.

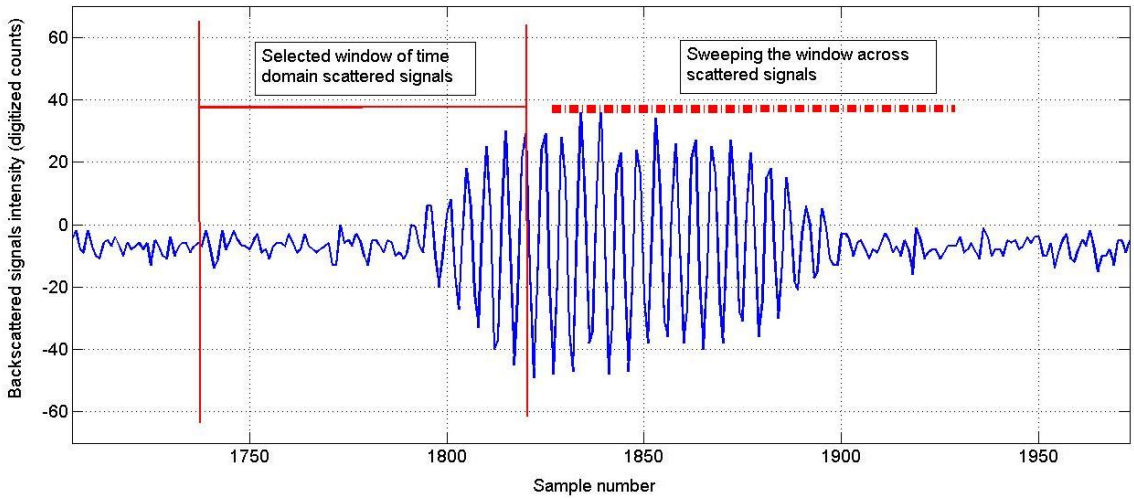

Figure 4-36 selected window of time domain signals is swept across the hard target's scattered signals.

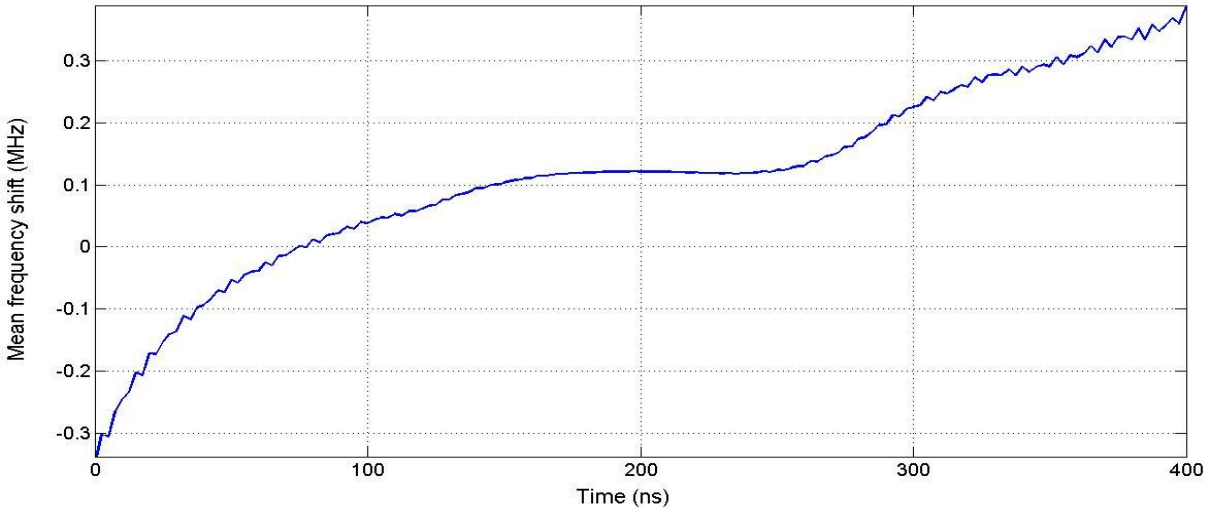

Figure 4-37 Estimated Doppler shift of hard target's scattered signals across time

5 CHAPTER V FPGA PROGRAMMING AND WIND MEASUREMENTS ANALYZED USING AUTOCORRELATION

5.1 Introduction

The objective of processing received signals using this technique is to have the ability to change the spatial resolution. This is achieved by calculating the autocorrelation of received signals and using it to calculate the power spectrum of any desired range gate. The power spectrum of received signals is found by calculating the FFT of the autocorrelation as shown in equations: 5-1 and 5-2.

$$R(\tau) = \int_{-\infty}^{\infty} f(t)f(t+\tau)dt \qquad (5\text{-}1)$$

$$G(f) = \int_{-\infty}^{\infty} R(\tau)e^{-j2\pi f\tau}\, dt \qquad (5\text{-}2)$$

where; $f(t)$ is a time domain signal, $R(\tau)$ is the signal's autocorrelation, and $G(f)$ is the Fourier transform (power spectrum)

Changing range gates (varying spatial resolution) is an advantage that previous FFT pre-processing algorithm does not have. In this technique (autocorrelation), digitized received signals are split into two paths. The first path is mixed with a cosine signal oscillating at 84 MHz to produce an in-phase (I) component; the other path is mixed with a sine signal oscillating at 84 MHz to produce a quadrature (Q) component, Fig. 5-1.

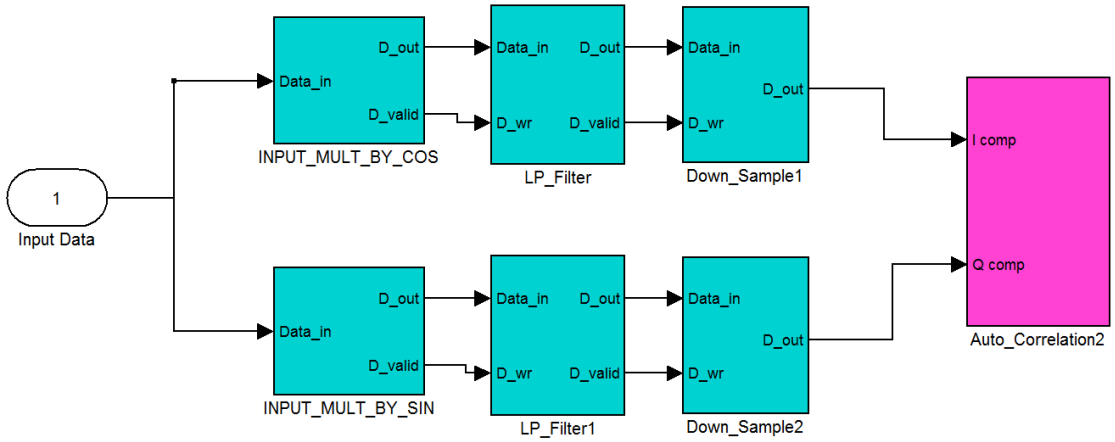

Figure 5-1 Autocorrelation algorithm block diagram as implemented on the FPGA, in which input signal is split into two paths. One path is multiplied by a cosine to produce an in-phase (I) signal and the other path is multiplied by a sine to produce a quadrature (Q) signal. Both (I) and (Q) are filtered out of the high frequencies though an FIR filters then down sampled and fed to the auto-correlator circuit.

5.2 Autocorrelation (Analog Complex Demodulator) Pre-Processing Algorithm

Mixing the received signals (oscillating around 84 MHz +/- Doppler shift) with an 84 MHz cosine and sine waves produces two output signals; a high frequency (sum of the two frequencies) component and low frequency (difference of the two frequencies) component (the Doppler shift). A low-pass, finite impulse response (FIR), filter is used on each path to get rid of the unwanted high frequency. Filtered signals are then down-sampled (decimated) by a factor of 4, which will reduce our original sampling period from 2.5n.s (400 MHz) to 10n.s (100 MHz). This down conversion reduces the maximum detectable frequency (according to Nyquest theorem) to 50 MHz, which corresponds to a radial velocity of approximately 38 m/s. The resulting complex time sequence $d(n) = d_i(n) + j\ d_q(n)$ is input to the (M)-lag autocorrelator circuit, which computes an autocorrelation matrix $D(m,n) = d^*(n).d(n+m)$ for m = 0 to M-1(lags) and n = 0 to N-1 (number of time domain samples, which is 8ksamples/4 = 2k), where $d^* = d_i(n) - j\ d_q(n)$ is the complex conjugate of $d(n)$. The processing repeats for 10k laser shots and the

elements of D matrix are accumulated and then streamed to an output buffer before it is being streamed to the host PC.

Once the accumulated lags' matrix (equation 5-3) is streamed to the host PC, further processing is conducted to calculate the power spectrum of received signals as follows:

$$D = \begin{bmatrix} S_0S_0^* & S_0S_1^* & S_0S_2^* & \dots & \dots & S_0S_{M-1}^* \\ S_1S_1^* & S_1S_2^* & S_1S_3^* & \dots & \dots & \vdots \\ S_2S_2^* & S_2S_3^* & S_2S_4^* & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \dots & \vdots \\ S_{n-1}S_{n-1}^* & S_{n-1}S_{n-2}^* & 0 & \dots & \dots & \vdots \\ S_nS_n^* & 0 & 0 & \dots & \dots & \vdots \end{bmatrix} \quad \text{(5-3)}$$

These samples when accumulated, represent a range gate

where; *M* is the number of lags, *n* is the number of acquired samples, *S* denotes to a sample, and S^* denotes to the complex conjugate of sample *S*.

To calculate the power spectrum of a certain range gate, the columns of the *D* matrix are accumulated from the i^{th} row to the j^{th} row, where *i* and *j* are the first and last corresponding samples of that range gate, respectively. This accumulation process produces an M size autocorrelation vector, which is complex (in-phase and quadrature components). Since the autocorrelation is symmetric, we construct the second half of the autocorrelation vector by making its real part even and imaginary part odd. Finally, we find the power spectrum of that range gate's signals by calculating the FFT of the constructed complex autocorrelation vector.

5.3 Number of Autocorrelation Lags and Down Sampling Factor Selection

In order to determine the optimum number of autocorrelation lags and the down sample factor required to obtain a sufficient SNR, backscattered signals were processed using both FFT and autocorrelation (with different number of lags, and different down sample factors) techniques. Obtained power spectra from autocorrelation cases were compared with that of the FFT.

Atmospheric backscattered signals that showed approximately 1.6 dB of SNR was chosen to make a comparison between two cases; 16-lags with a down sample factor of 4, and 8-lags with a down sample factor of 2 cases with that of the FFT. Down sampling acquired data (originally sampled at 400 MHz) by a factor of 4 will reduce the sampling rate of the data to 100 MHz, which as a result, reduces the maximum detectable Doppler shift to only 50 MHz (Nyquist theorem), i.e. maximum obtained wind velocity is 38 m/s. On the other hand, down sampling by a factor of 2 will only reduce the sampling rate to 200 MHz, which allows for detecting Doppler shifts up-to 100 MHz (wind velocity up-to 79 m/s). On the other hand, choosing the 16-lags case will provide a frequency resolution of 3.2 MHz (i.e. velocity resolution ~ 2.4 m/s), while choosing the 8-lags case will provide a frequency resolution of 13 MHz (i.e. velocity resolution of ~10 m/s). Fig. 5-2 shows the power spectra of processing the backscattered signals using FFT, 16-lags, and 8-lags. It can be shown that the 16-lags' power spectrum has a slightly higher SNR than that of the 8-lags. It is worth noting that even when using the 8-lags case, frequency resolution could be reduced by increasing the time delay between lags, i.e. (τ) of the autocorrelation. Increasing τ improves (reduces) frequency resolution, but on the other hand, decreases the maximum detectable frequency, i.e. reduces measurable velocity range. Table 5-1 summarizes all analyzed scenarios of number of lags, down sampling factors, and different values of τ.

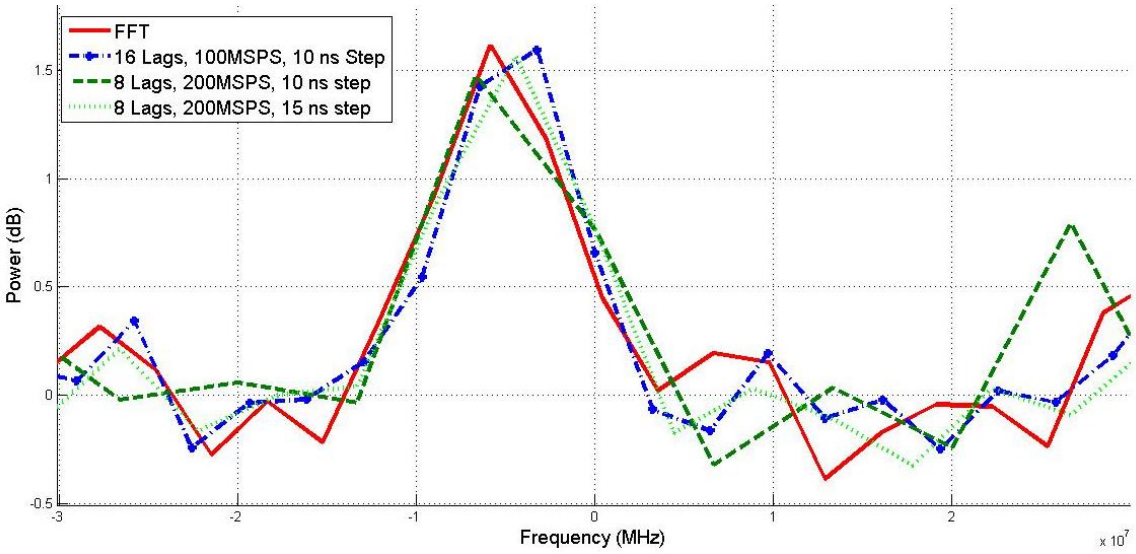

Figure 5-2 Power spectra of backscattered signals obtained by FFT, 16-lags, and 8-lags autocorrelation

Number of lags	τ (ns)	Frequency Res. (MHz)	Velocity Res. (m/s)	Maximum Freq. (MHz)	Maximum velocity (m/S)
16	10	3.2	2.4	+/- 50	38
8	10	6.6	4.8	+/- 52	40
8	15	4.4	3.4	+/- 32	25

Table 5-1 Number of lags and lag delay (τ) influences on velocity resolution and maximum velocity range

It is clear from table 5-1 that 16-lags case with $\tau = 10$ ns provides the lowest velocity resolution with a reasonable maximum velocity range (38 m/s). Therefore, that case was our choice of implementation.

5.4 Logic Design and FPGA Programming Implementation

In this section, I-Q generator, low-pass filter, down sampler, and autocorrelator logic circuits are explained in details.

5.4.1 Input Signals In-phase (I) and Quadrature (Q) Generation Digital Circuit:

This circuit block generates the in-phase signal component by multiplying the input time domain digitized signals (8ksamples vector) by an 8ksample vector consisting of $cos(2\pi f_c)$, where f_c = 84 MHz. This cosine vector is generated using a single port RAM Xilinx block as show in Fig. 5-3. A digital counter circuit is used to drive the address port of the RAM block so that each sample of the input data vector is multiplied by its corresponding indexed point of the cosine vector. The data valid control signal, which is associated with each input sample, is being used as an enable control to the digital counter causing the counter to increment each time a new sample arrives. The output of this circuit is the input signals multiplied by the cosine vector, in-phase (I) component. Similarly, the quadrature (Q) component is generated using a single port RAM Xilinx block storing an 8ksample vector of $sin(2\pi f_c)$, where f_c = 84 MHz.

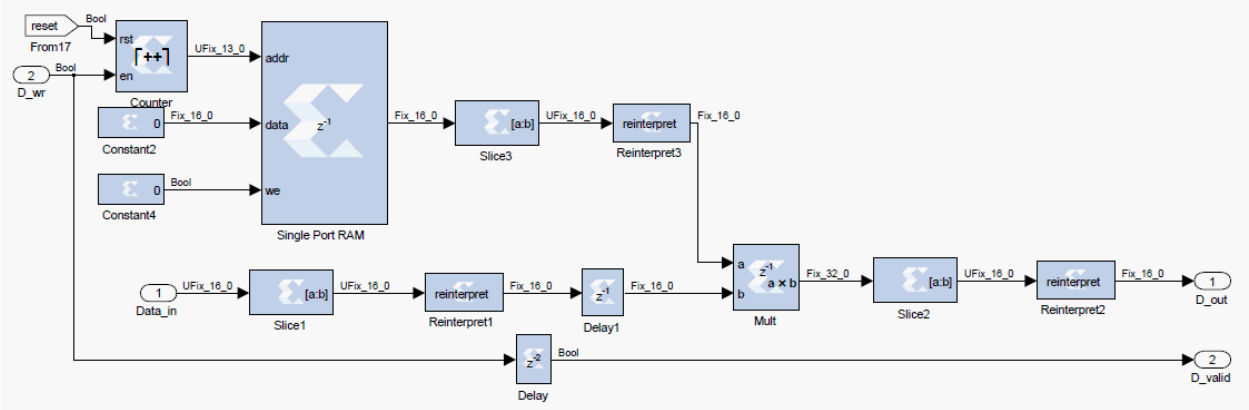

Figure 5-3 In-phase (I) and qaudrature (Q) generation digital circuit as implemented on the FPGA, in which the input time domain digitized signals are multiplied by a cosine (to generate in-phase (I) signal) and sine (to generate quadrature (Q) signal) vectors stored in a single port RAM block.

5.4.2 Low-Pass (FIR) filter Digital Circuit:

The function of this circuit is to filter out the high frequency signals off both in-phase and quadrature components as shown in Fig. 5-4. A Xilinx FIR compiler 5.0 circuit block is being used to perform this task, Fig. 5-5. The FIR filter is designed using MATLAB Simulink toolset

whose frequency response is shown in Fig.5-6. The frequency response shows that the out of band signals (above 50 MHz) will be suppressed by approximately 80 dB.

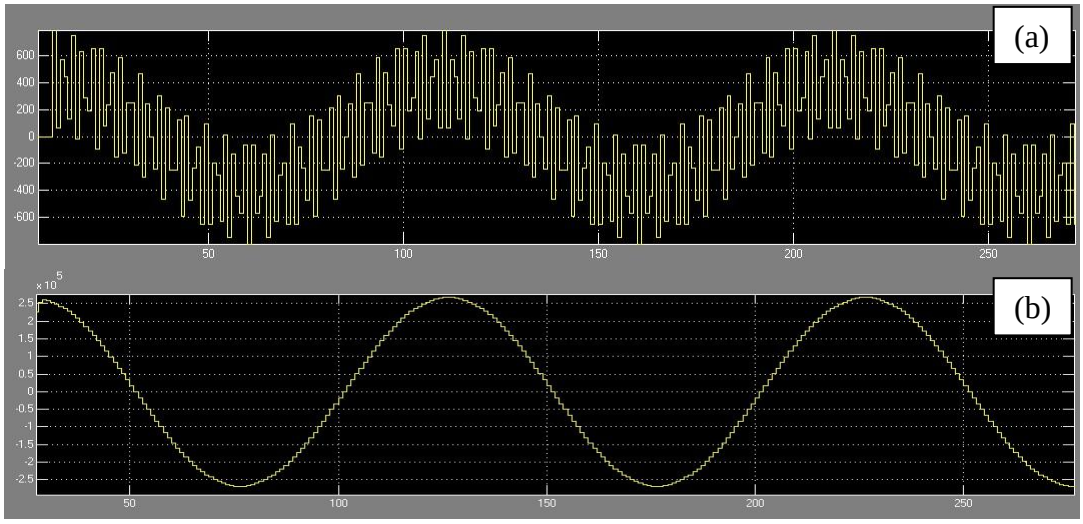

Figure 5-4 In-phase and quadrature signals' shape with high frequency components before being filtered (a), and in-phase and quadrature signals' shape at the output of the FIR low-pass filter (b)

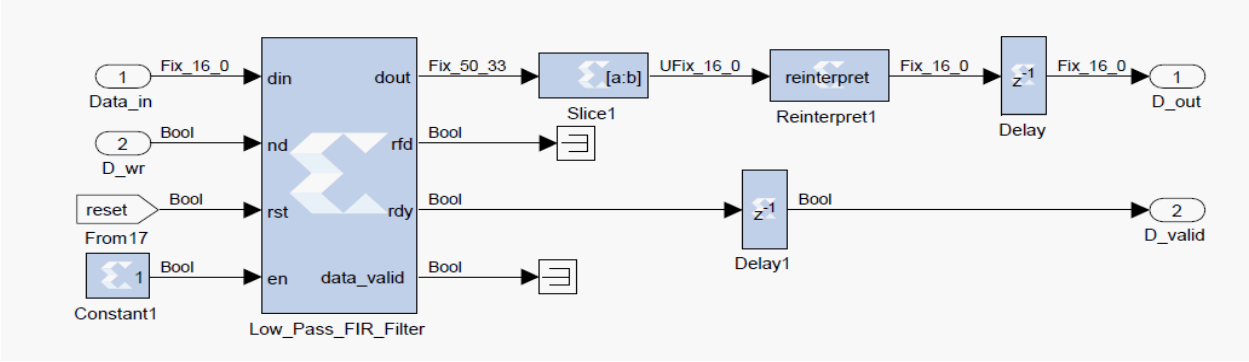

Figure 5-5 FIR low-pass filter digital circuit as implemented on the FPGA

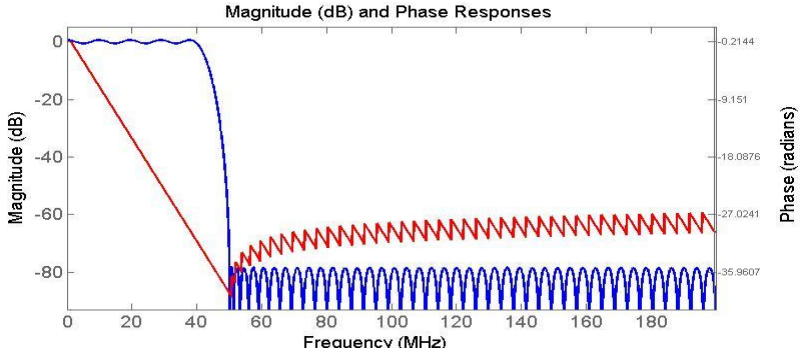

Figure 5-6 FIR low-pass filter magnitude (dB) and phase responses

5.4.3 Down Sampler Circuit

This circuit down converts (decimates) filtered signals by a factor of 4, i.e. for every 4 input samples, it ignores three samples and keeps the fourth one. The down conversion is performed by using a digital accumulator circuit, Fig. 5-7, which accumulates the data valid control signal (D_wr) associated with incoming input samples. When the accumulator output is equal to four, the Relation circuit will output 1, which will act as the new data valid control signal. As a result, the output data valid signal (D_valid) is decimated by a factor of four, which causes the filtered time domain in-phase and quadrature signals (Data_in) to be reduced in size from 8k to 2k (D_out).

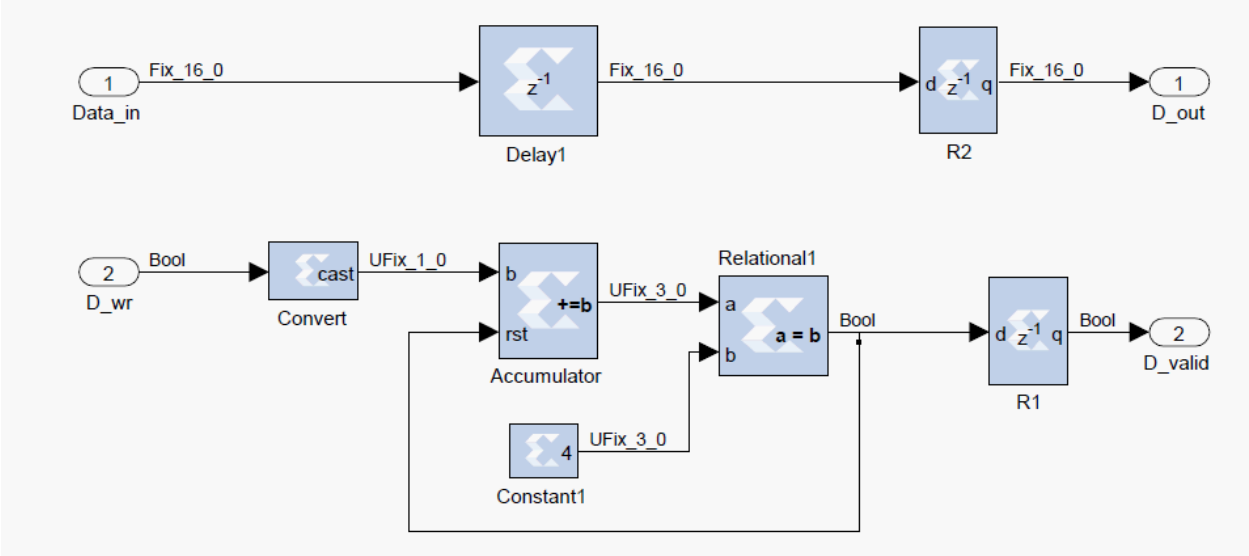

Figure 5-7 The down converter digital circuit as implemented on the FPGA, in which an accumulator circuit is used to count the arrival of incoming samples. Whenever the counter counts to 4, it outputs logic 1 at the data valid control signal to validate only the fourth sample. The previously arrived three samples will be ignored, resulting in a decimation ratio of 3:1.

5.4.4 Autocorrelation Digital Circuit

The function of this circuit is to compute the autocorrelation of the input signals and accumulate the resulting lags-matrix. Down converted, filtered, and digitized input signals arrive to the

autocorrelator circuit as a 2ksamples vector. This vector passes through M-time delay circuits, Fig. 5-8, to calculate 0 to M-1 lags. Input vector is tapped at each delay output. Right after the last delay circuit is a complex conjugate circuit, which outputs the complex conjugate $d^* = d_i(n) - j\, d_q(n)$ of the input signal. This complex conjugate output is then multiplied by tapped output of each delay circuit, resulting $D(m,n) = d^*(n).d(n+m)$, where; n is the sample number, and m is the delay (lag) number. The output of this process is M vectors each of size 2ksample. These 2ksample vectors are feed to accumulator circuits to be accumulated over 10k laser shots. Fig.5-9 shows the actual circuit diagram.

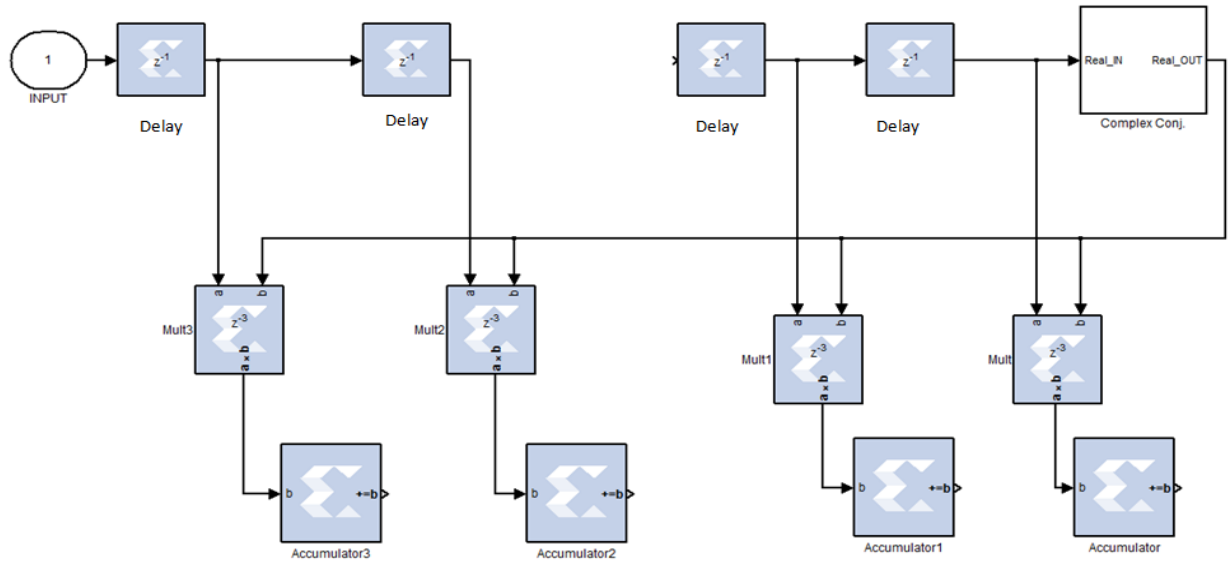

Figure 5-8 Autocorrelator circuit block diagram as implemented on the FPGA in which input vector is delayed through time-delay circuits and then the complex conjugate of the complex I-Q signals is computed and multiplied by the tapped delayed I-Q samples that produces the autocorrelation lags. These lags are then accumulated through the accumulator circuits for 10k laser shots.

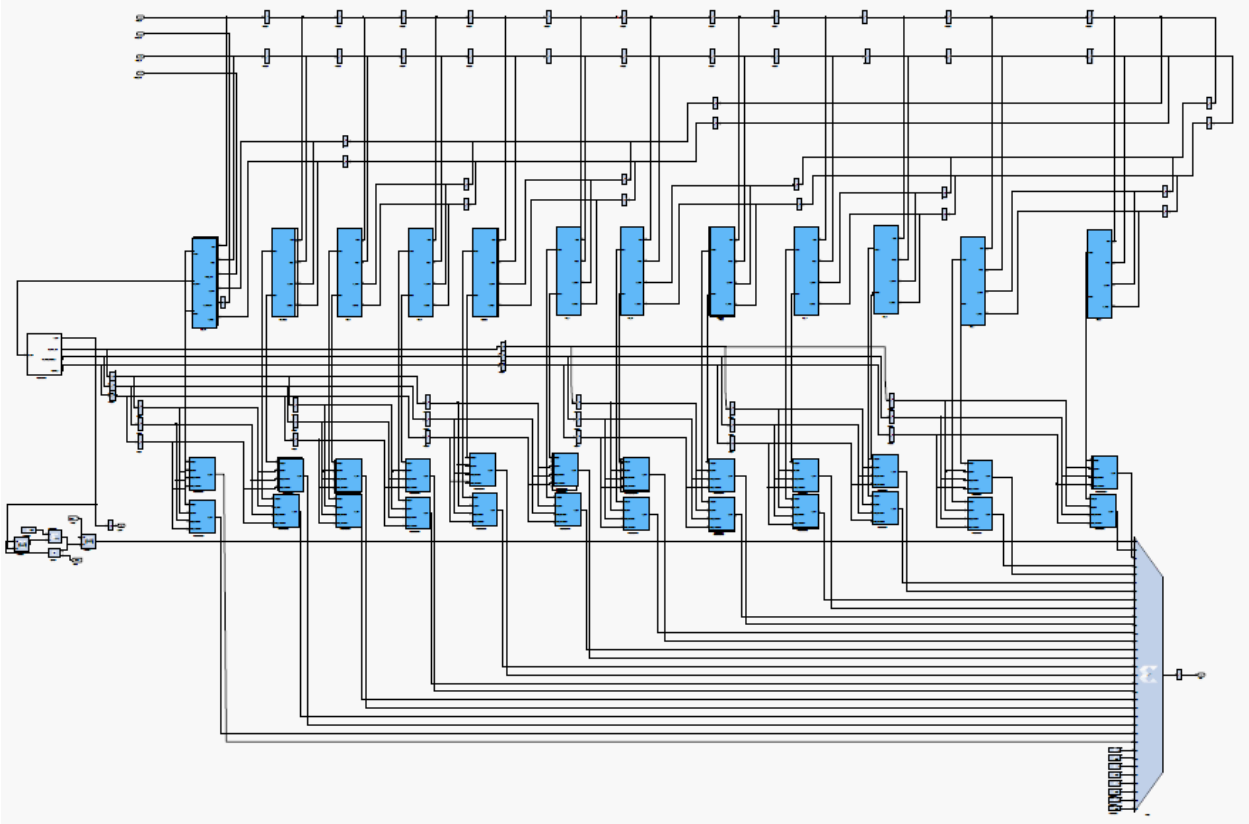

Figure 5-9 Actual circuit design of the autocorrelator as implemented on the FPGA

5.4.5 Accumulator of the Autocorrelation Lags.

The accumulator circuit of the autocorrelation lags acts similarly to that of the FFT pre-processing algorithm. The only difference is that once 10k laser shots are accumulated, control logic circuits ignore newly incoming signals and start to output the accumulated result for 32 pulses'. The reason for this way of operation is that all real and imaginary vectors of the 16 lags need to be streamed out. Therefore, each accumulator outputs its result at the same time while selective logic circuits select which accumulator's output to be streamed out. Once 32 pulses time is elapsed, each accumulator resets its FIFO and start accumulating a new set of 10k laser shots.

5.5 Vertical Wind Velocity Measurements Using autocorrelation Pre-Processing Algorithm

In this section, wind velocity was measured in a vertical mode while pre-processing received signals using an autocorrelation algorithm. The autocorrelation algorithm calculates the autocorrelation of the received signals and streams out the lags matrix that can be gated according to user's range resolution's preference. That feature is what makes autocorrelation technique advantageous over the FFT technique, where range gates are fixed. Wind velocity was measured under this mode of operation for several days during the afternoon time in the summer of 2012. The following section presents vertical wind measurements for June 24th, 26th, 27th, 28th, 29th, July 11th, 12th and 23rd.

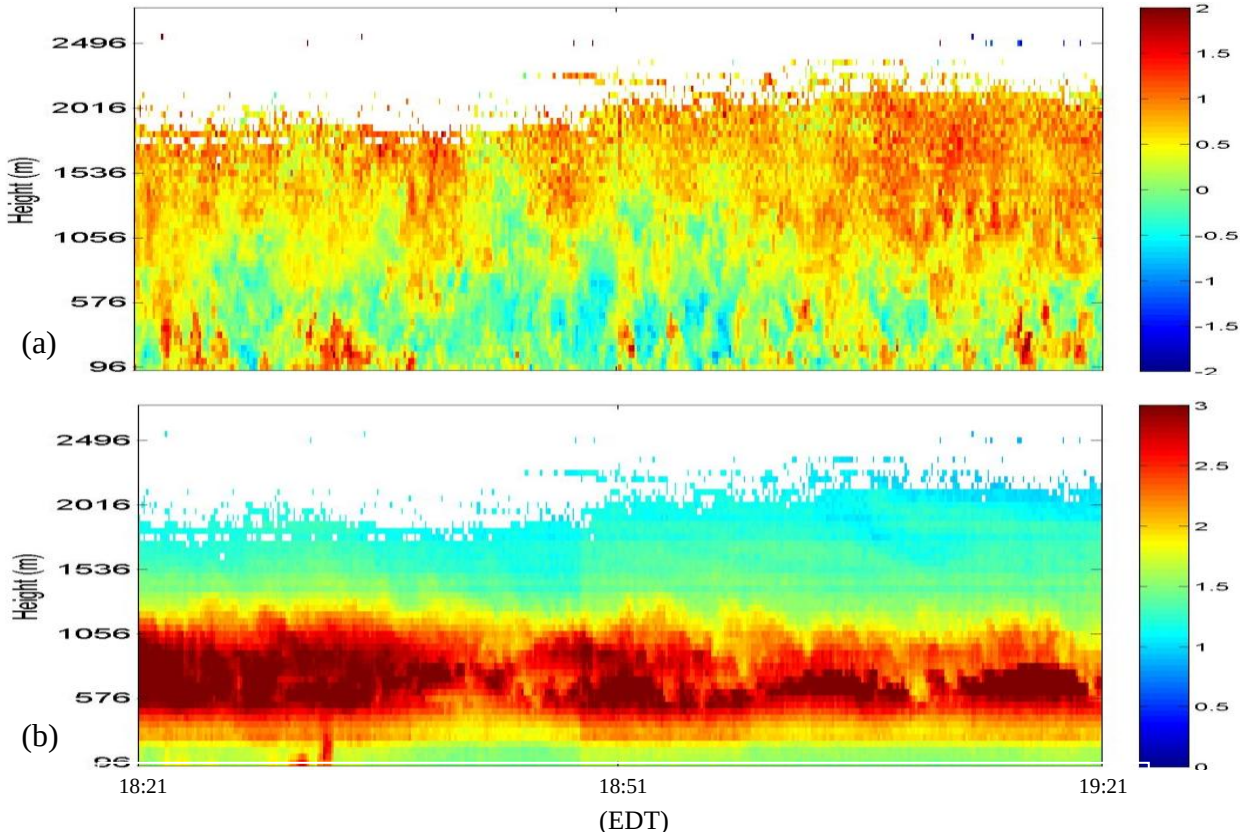

Figure 5-10 Vertical wind velocity (m/s) (a), and ratio of received signals power to shot noise power (b), vs. time and height estimated using autocorrelation measured at CCNY remote sensing Laboratory between 18:21 – 19:21 PM EST on June. 24th, 2012.

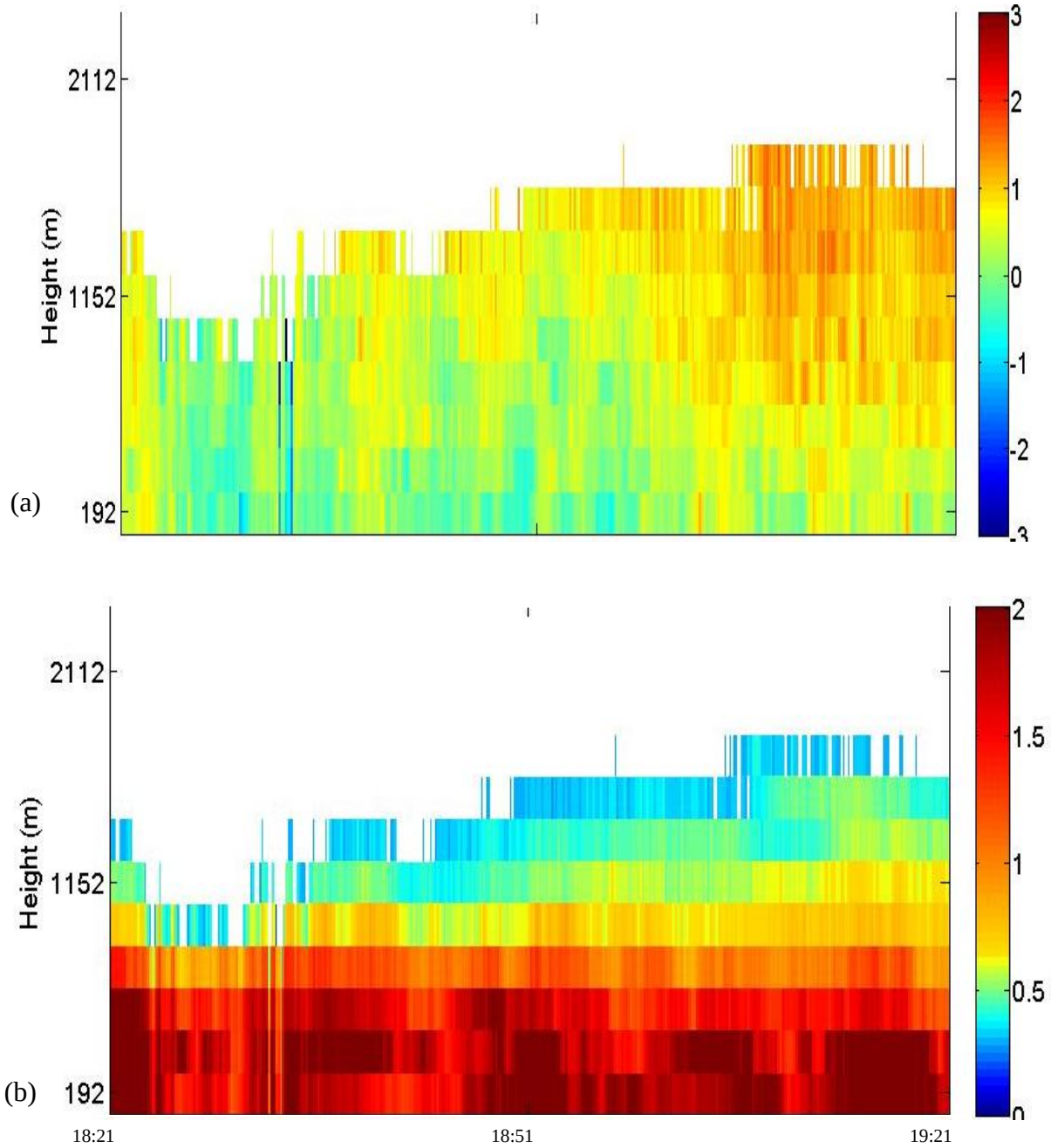

Figure 5-11 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), vs. time and height measured at CCNY remote sensing Laboratory between 18:21 – 19:21 PM EDT on June. 24th, 2012. Processed differently by increasing the range gate (spatial resolution). This could only be analyzed when pre-processing using auto correlation.

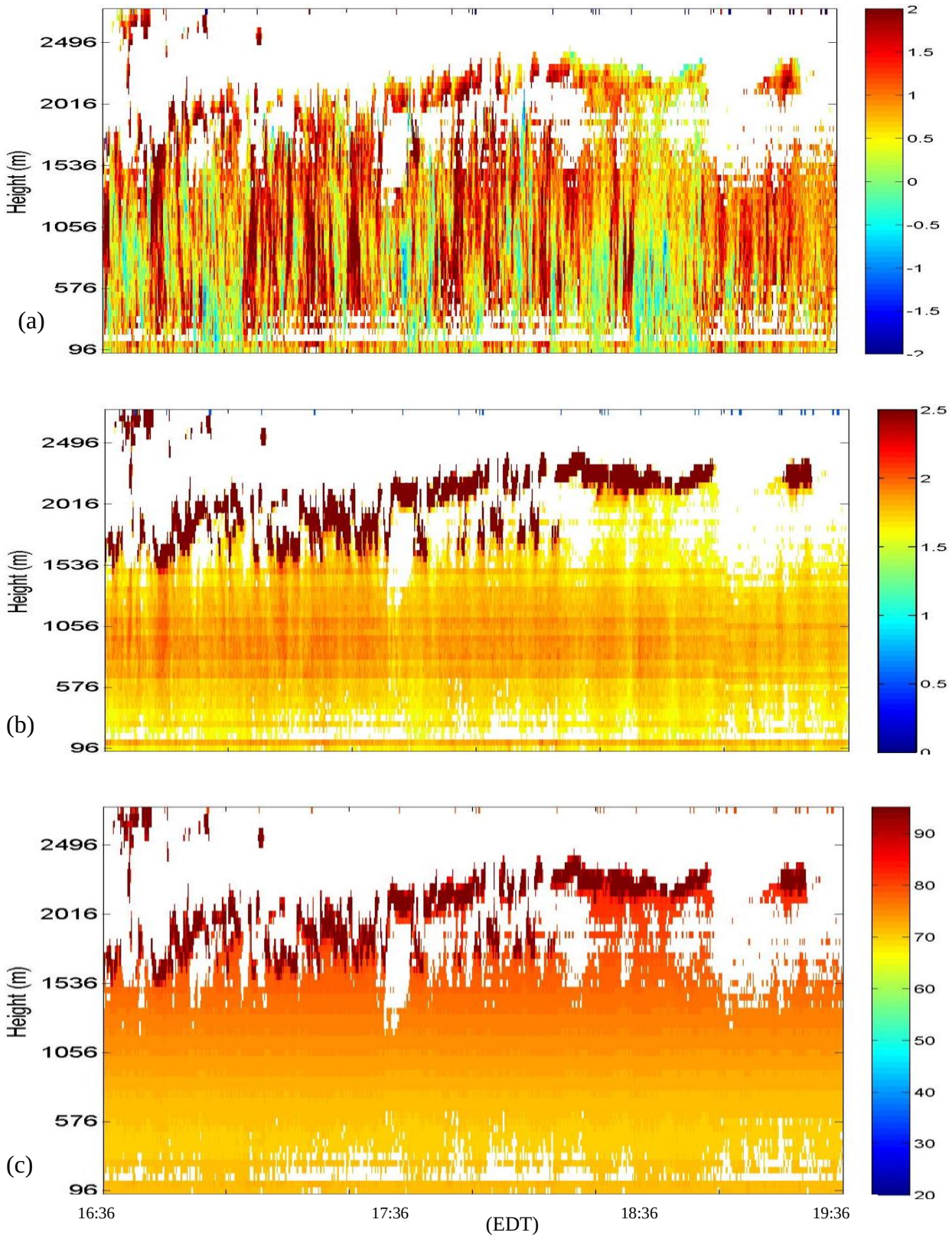

Figure 5-12 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected signals power (c) vs. time and height measured at CCNY remote sensing Laboratory between 16:36 – 19:36 PM EST on June. 26th, 2012.

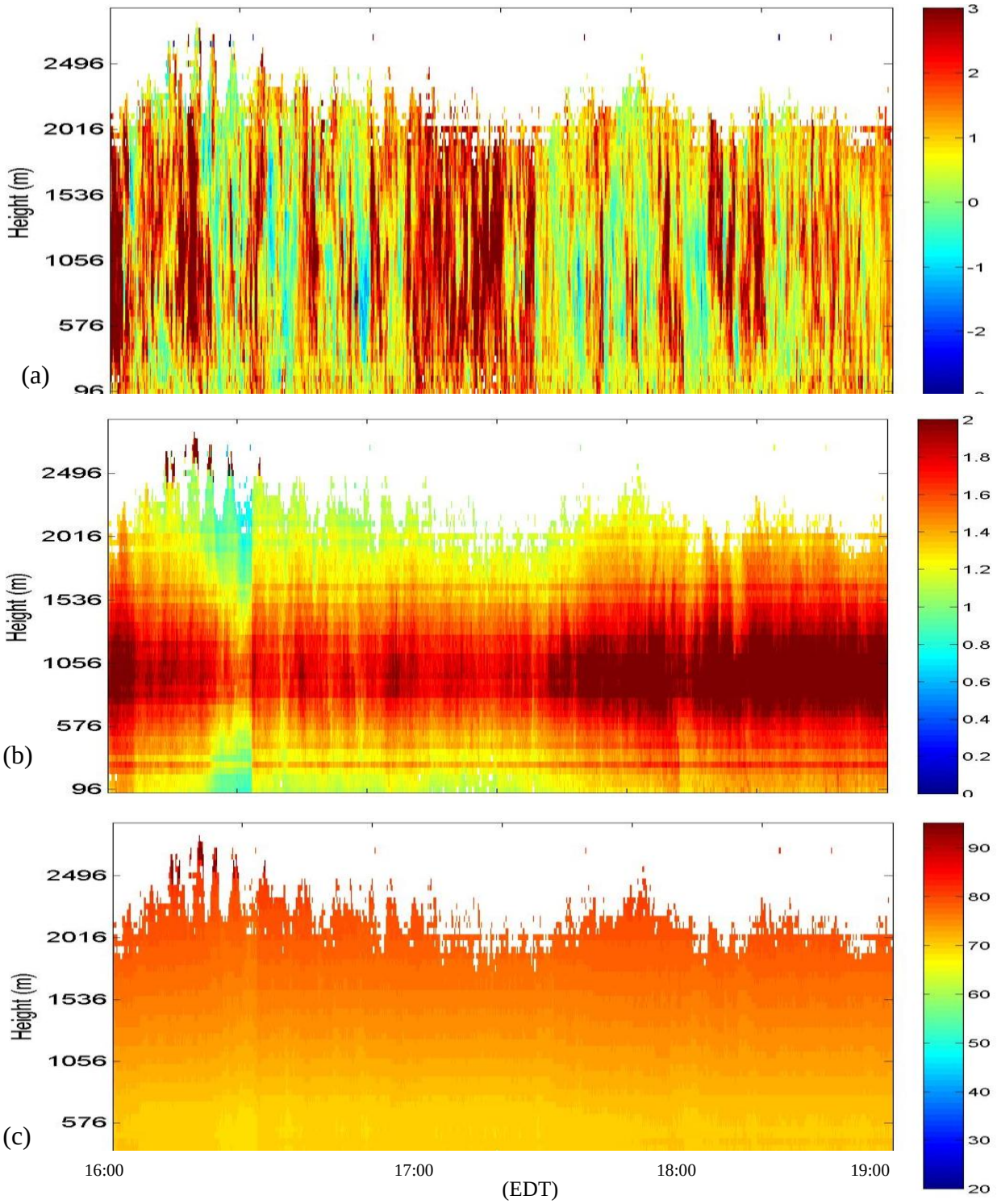

Figure 5-13 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected received signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 16:00 – 19:00 PM EDT on June. 27th, 2012.

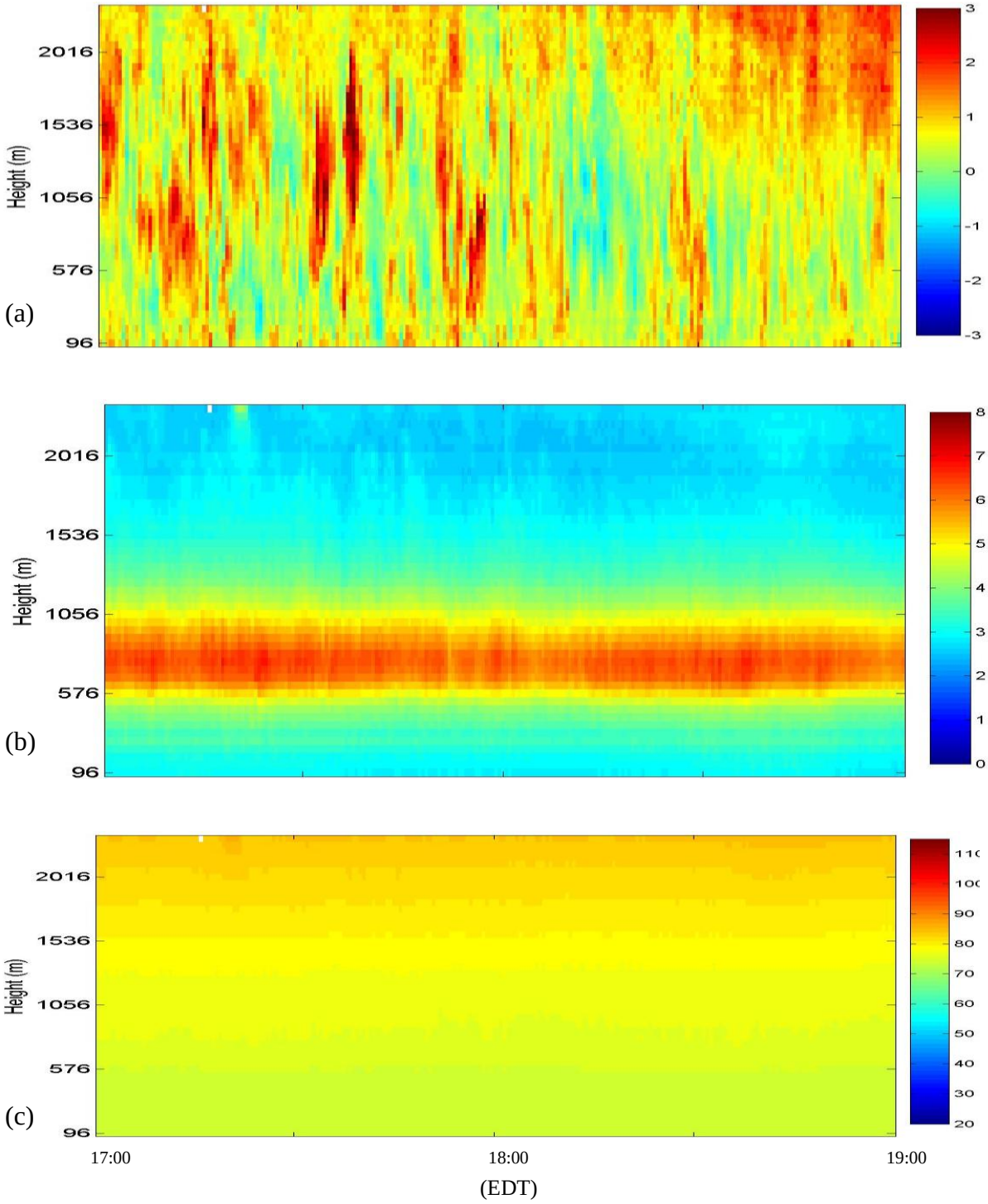

Figure 5-14 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected received signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 17:00 – 19:00 PM EDT on June. 28th, 2012.

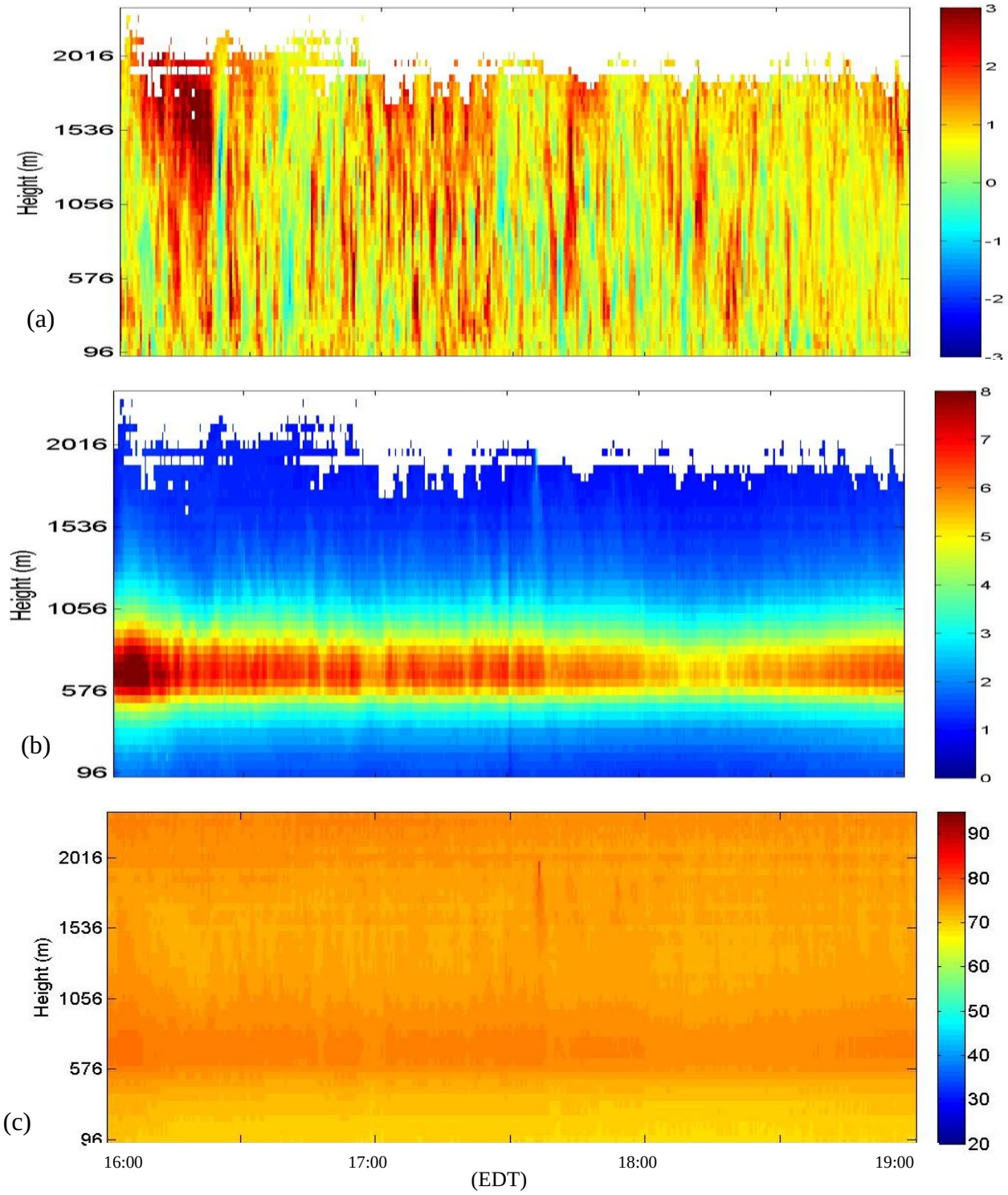

Figure 5-15 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected received signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 16:00 – 19:00 PM EDT on June. 29th, 2012.

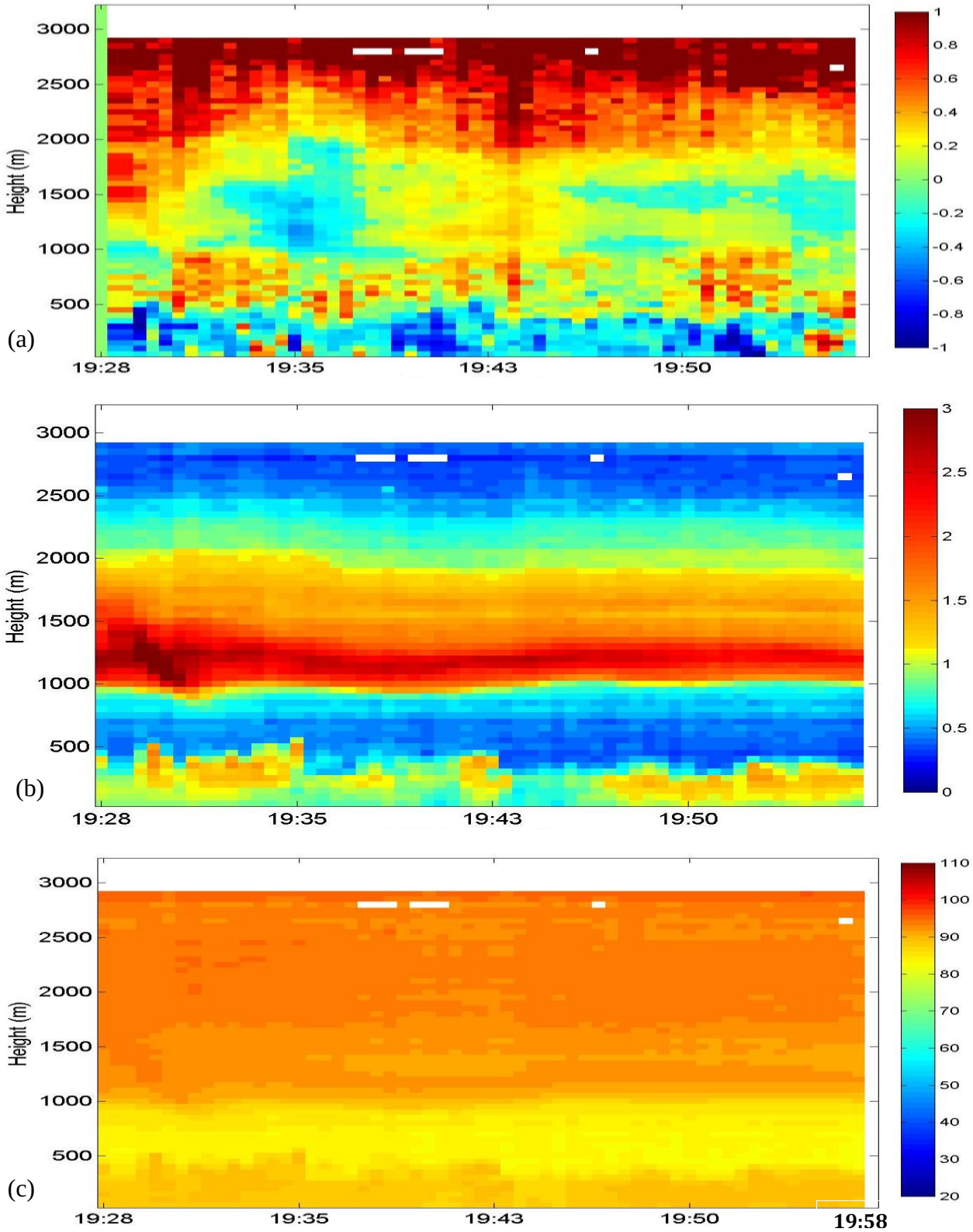

Figure 5-16 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected received signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 19:28 – 19:58 PM EDT on July. 11th, 2012.

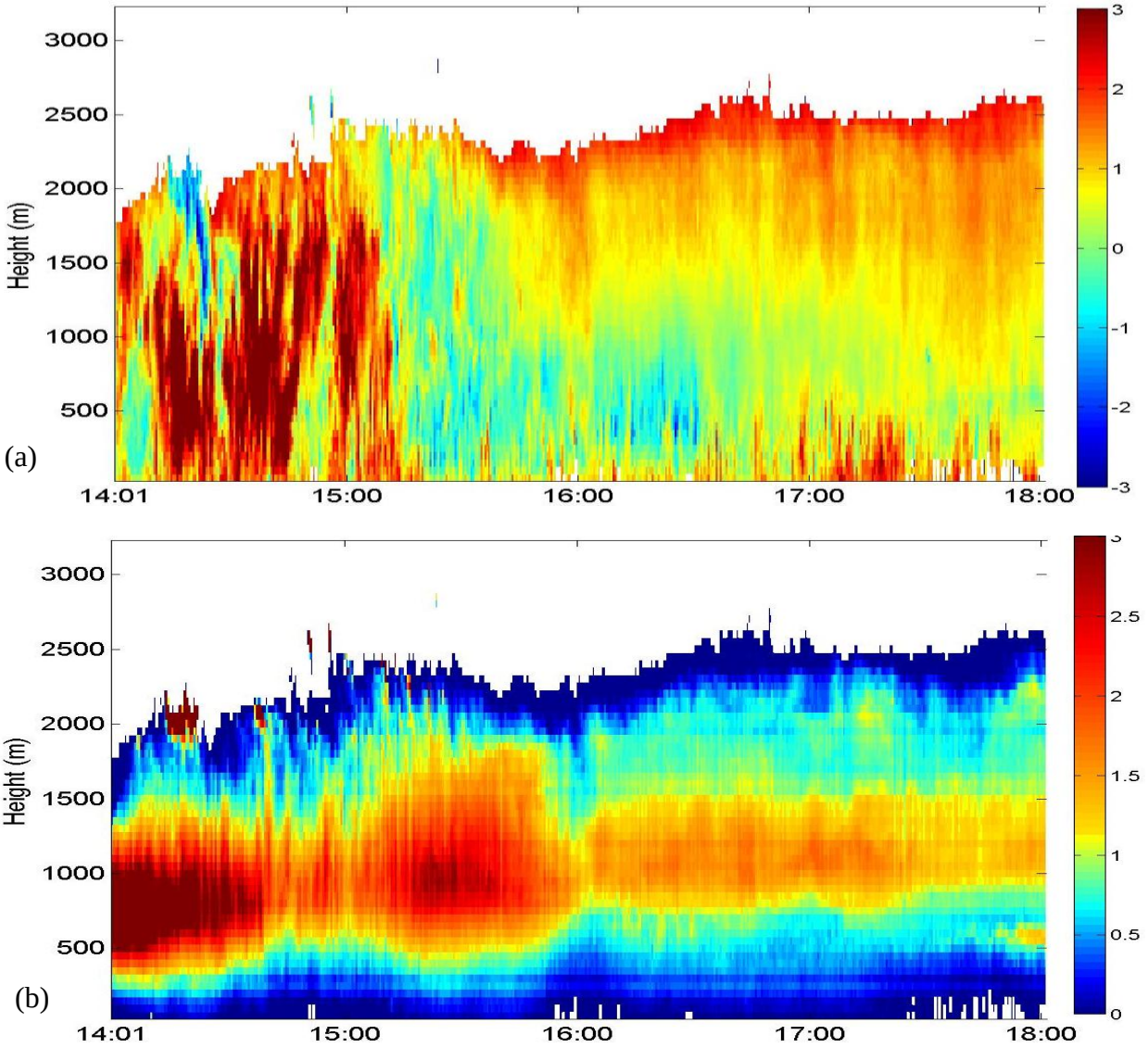

Figure 5-17 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected received signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 14:01 – 18:00 PM EDT on July. 12th, 2012.

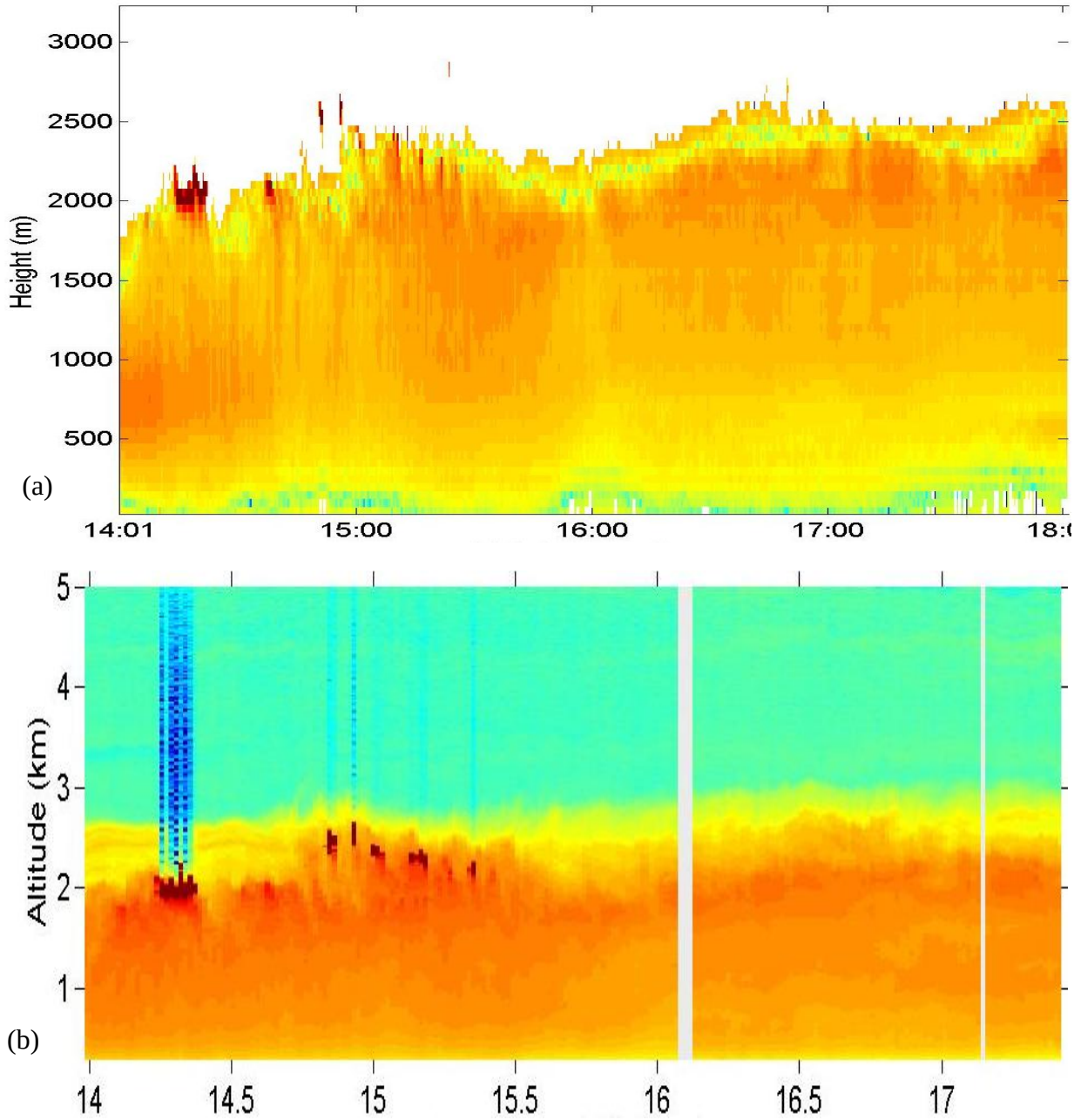

Figure 5-18 Doppler lidar's range corrected backscattered signals power (m/s) (a), Direct detection's range corrected backscattered signals power (bvs. time and height measured at CCNY remote sensing Laboratory between 14:00 – 18:00 PM EDT on July. 12th, 2012.

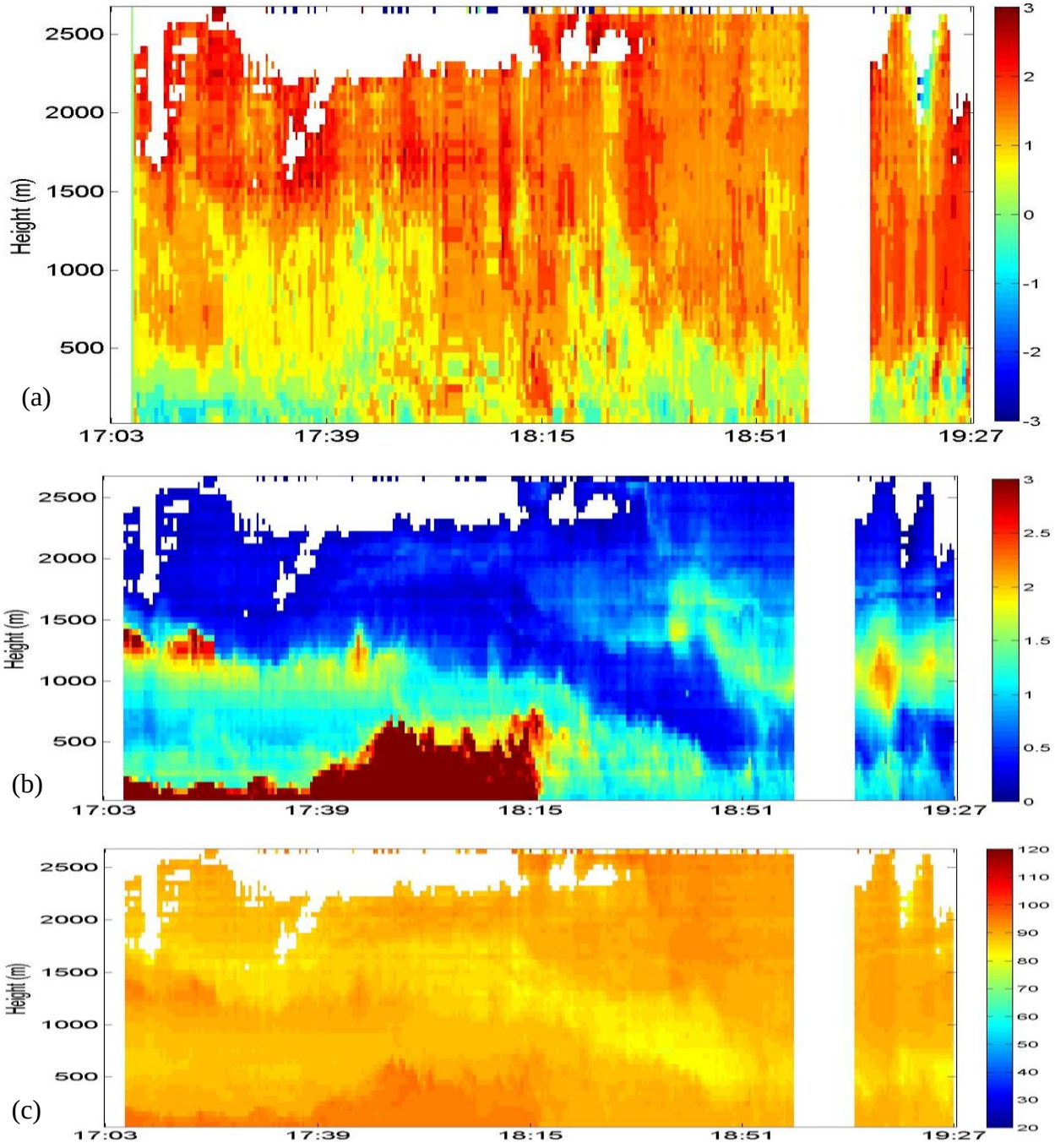

Figure 5-19 Vertical wind velocity (m/s) (a), ratio of received signals power to shot noise power (b), and range corrected received signals power (dB) (c) vs. time and height measured at CCNY remote sensing Laboratory between 17:03 – 19:27 PM EDT on July. 23rd, 2012.

5.6 Horizontal Wind Measurements Using Autocorrelation Pre-Processing Techniques

Horizontal wind measurements were obtained using autocorrelation pre-processing technique similarly as in the FFT technique by constructing a 3D wind vector from three line of sight measurements. Wind speed measured on August 14th, 2011 at the remote sensing Laboratory of the CCNY between 14:12 and 16:15 EDT was verified by checking the local weather reported on weather.com website (wind speed was reported to be 8mph blowing from south and south east), which agrees with the values we measured, Fig. 5-20.

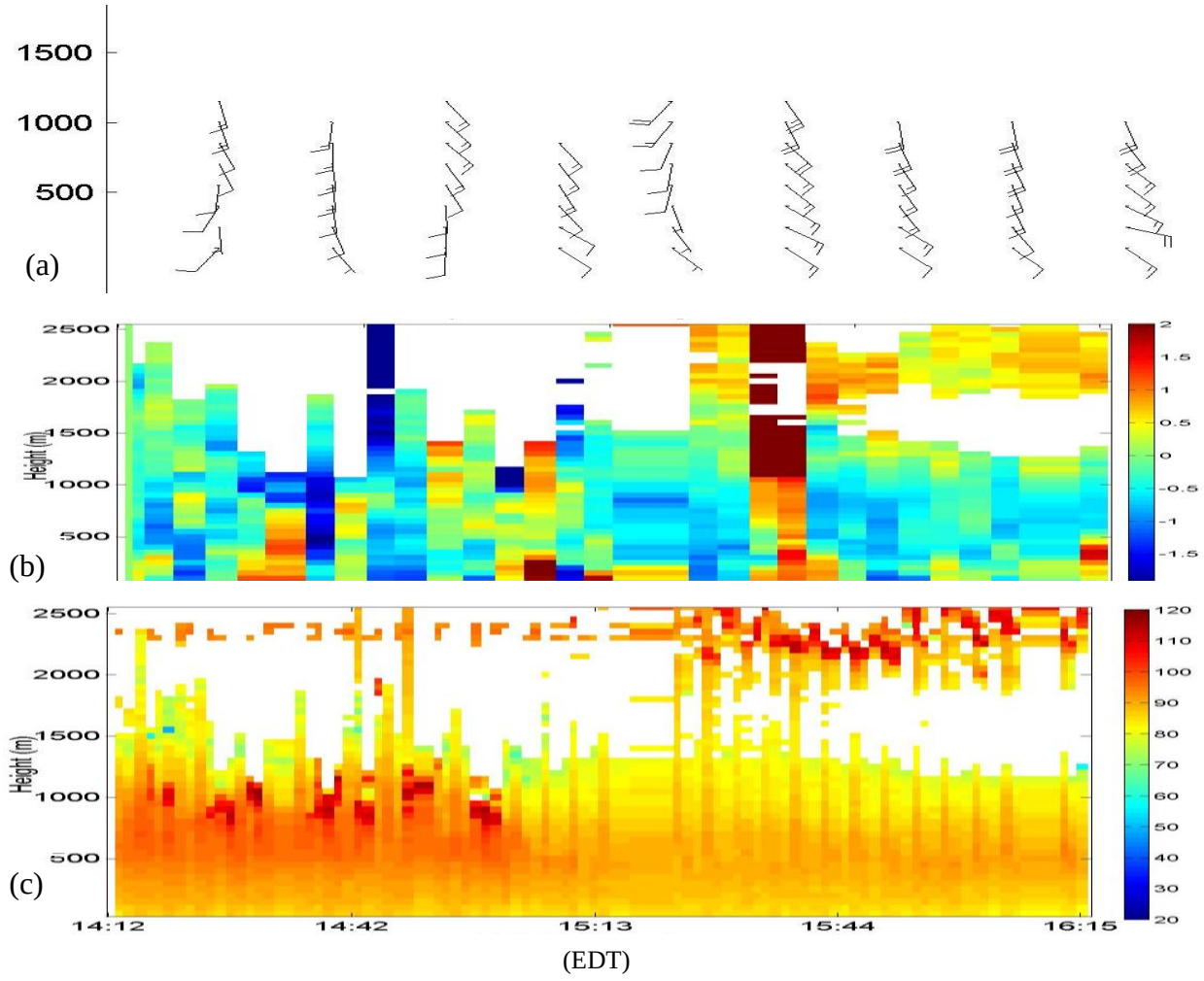

Figure 5-20 Wind barbs (a), vertical wind velocity (m/s) (b), and range corrected backscattered power (dB) (c) measured at CCNY on August 14th, 2012 between 14:12 and 16:15 EDT.

5.7 Autocorrelation and FFT Pre-Processing Techniques Comparison

In this section wind measurements were obtained using autocorrelation pre-processing technique for 1.5 hours followed by 1.5 hours of FFT pre-processing technique, Fig. 5-21. The presence of clouds at 15:35 was observed and verified by taking a photo through the Laboratory van's roof opening, Fig. 5-22. These clouds show in Fig. 5-21(b) at approximately a height of 2200m with a very high backscatterd signal power. The backscattered signal power measured by FFT looks slightly higher than that estimated using autocorrelation as shown in Fig. 5-21(b) at approximately 750 m. The FFT increase in estimated signal power over the autocorrelation was expected, because the power spectrum estimated using autocorrelation would have to use a number of lags equal to that of points used in FFT calculation. However, the number of lags used in autocorrelation was only 12 points while 64 points were used in FFT calculation. Even though the power estimated using autocorrelation seems to be slightly less than that estimated using FFT, but velocity measurements could still be obtained. In addition to that, range resolution can be varied unlike when using FFT, which makes autocorrelation technique more advantageous over FFT.

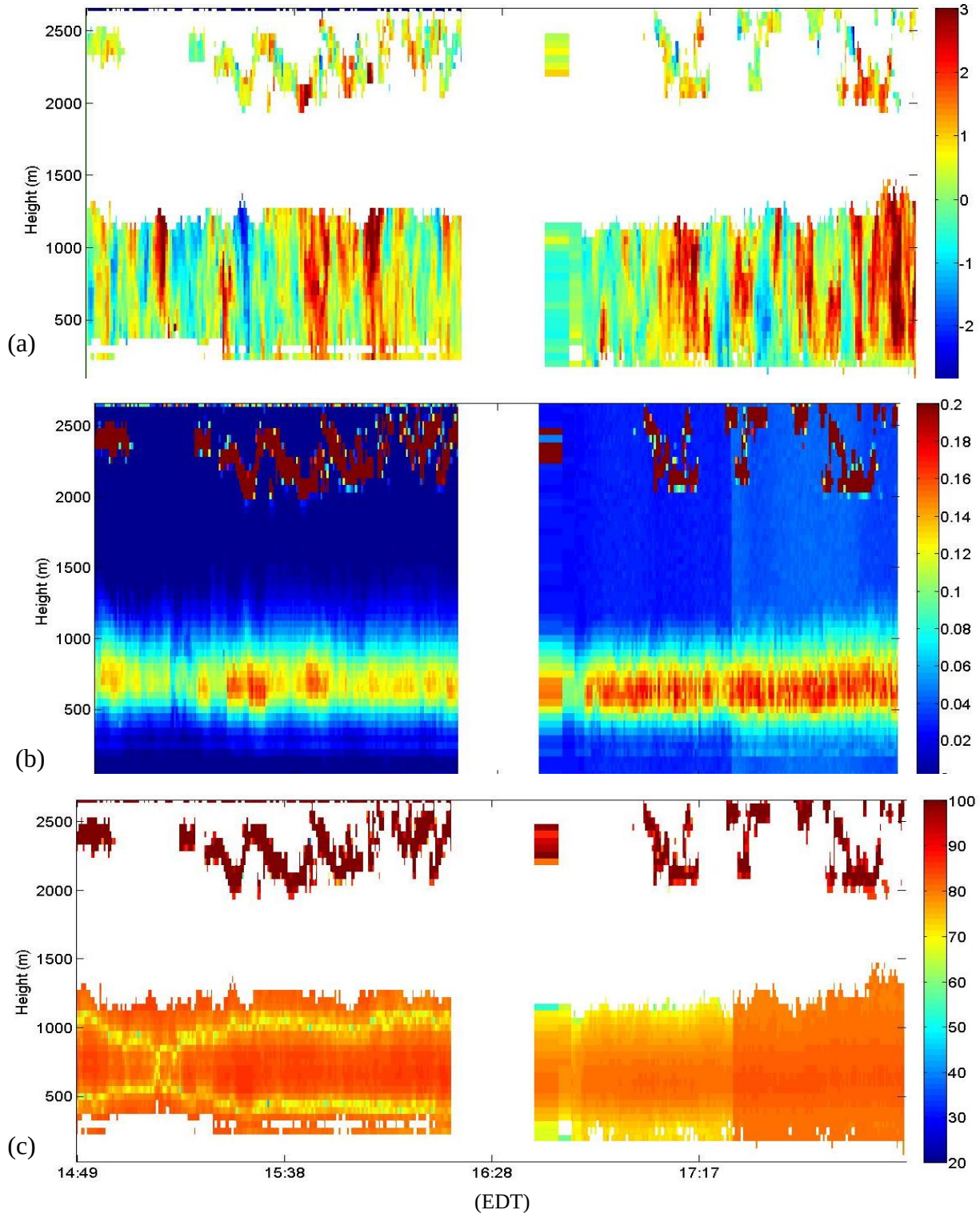

Figure 5-21 Wind barbs (a), ratio of backscattered signals power to shot noise power (b), and rannge corrected backscattered signal power (dB) (c) vs. time and height measured using autocorrelation technique between 14:49–16:19 and FFT technique between 16:40-18:10 on August13th, 2012.

Figure 5-22 This photo of clouds was taken from the Laboratory van's roof opening at 15:35, which appears in Fig. 5-15 (b) and (c) as cloud pattern at approximately 2km with a high backscattere power.

6 CHAPTER VI CONCLOSION

In conclusion, an eye-safe all-fiber CDL system for wind sensing in urban areas was designed, developed, tested, and operated at the remote sensing Laboratory of the City College of New York.

The system utilizes a 1.5μm fiber optics laser, which benefits from the availability and affordability of telecommunication optical components. Two AOMs are connected in series to achieve a high extinction ratio and to shift the laser frequency by 42 MHz each, which produces a total shift of 84 MHz. An optical amplifier amplifies the laser pulse to produce approximately 12μJ/pulse (200 ns FWHM at 20KHz PFR). An optical circulator directs amplified laser pulses to its output port that is connected to the optical antenna, and directs received signals to an optical coupler to be mixed with a LO. Circulator's fiber tip was polished and angled to reduce internal reflection that can damage the detector. Optical mixed signals are detected by a heterodyne balanced detector.

Received signals are sampled at 400 MHz through a 14-bit ADC equipped with an FPGA. Due to the very low energy per pulse (12 μJ/pulse), a high PFR (20KHz) is used to allow for digging the very low signal out of noise. This high pulse rate makes it almost impossible to process the data in real time, therefore, the FPGA was programmed to pre-process received signals at the hardware level as the received signals are being acquired and before streaming to the host PC.

Two different pre-processing algorithms have been simulated and programmed into the FPGA; one algorithm calculates FFT of time gated received signals and accumulates the resulted power spectrum; the other algorithm calculates autocorrelation of the received signals and

accumulates the result. The later algorithm allows for changing range gate (spatial resolution), which can be applied to signals scattered from very high altitudes (where signals are very weak) to improve the SNR.

The system was installed in a research vehicle and wind velocity was measured at the City College of New York. Wind velocity was measured in two different modes; vertical mode, and scan mode. Wind velocity was measured up to 3km in a vertical mode during a very clear day. 3D wind vectors were constructed from three line of sight measurements, and horizontal wind velocity was obtained. The system can be operated to measure wind velocity, processes received signals in real time, and display results while acquiring data.

Improving the system can be achieved by increasing the measured range to 7km instead of 3km. A scanner can automate the scanning operation mode so that a user doesn’t have to scan manually.

7 Appendices

7.1 Appendix I Data Management:

Wind velocity measurement procedures changed according to the development of the instrument. At the very beginning, raw time domain data of received signals was streamed directly from the data acquisition card to the host PC, where it was archived for further processing. Streaming raw data to the host PC produced very large sized files (approximately 1GByte for 1second of data acquisition). Signal processing programming algorithms were developed to process these raw data signals' files.

Programming the FPGA to pre-process acquired data before streaming it to the host PC, not only allowed for continuous data acquisition and real-time processing, but it also changed the size of streamed data significantly. The FPGA was programmed to pre-process backscattered signals in two different modes; FFT, or autocorrelation calculation and accumulation. Wind velocity can be measured in two different modes; vertical measurement mode, or scan (horizontal measurement) mode. The following section explain data management and signal's flow in the previous two modes (vertical and scan).

When measuring wind velocity in a vertical mode, the optical antenna is adjusted to shoot the laser beam in a vertical direction throughout the entire time of operation. Data acquisition software is invoked to capture a certain amount of received backscattered signals that is corresponding to the desired measurement time. For example, when loading the FFT algorithm onto the FPGA, the data streaming software is set to capture 8192 (packet size and frame size, as identified by streaming software) samples of received signals at the receipt of the rising edge of an external trigger (the same trigger that drives the laser pulses). These 8192 samples correspond

to backscattering of signals from 0 to approximately 3km. The FPGA algorithm calculates the FFT of received signals and accumulates them for 10k laser shots (1/2 second). Due to arithmetic operations on the FPGA, the size of the data samples increases from 16-bit/sample to 64 bit/sample, i.e. each sample is quadrupled. As a result, the amount of data streamed out of the FPGA in 1/2 second equals to 4x8192 samples. Therefore, to operate the instrument for a period of time equals to M minutes, one has to enter Mx60(seconds)x2(half a second)x4x8192 as the total number of samples (ceiling) needed to be streamed to the host PC by the acquisition software (SNAP). The streaming software was modified to output three different files; a binary file containing calculated FFT of entire captured data plus some headers, a text file containing calculated FFT of entire captured data without headers, a continuously generated text file containing calculated FFT of the last 30 seconds of captured data without headers (needed for real-time processing and display).

When loading the autocorrelation pre-processing algorithm, on the other hand, data acquisition software is also set to capture 8192 samples upon the receipt of the external trigger, Fig. 7-1. A 12-lags autocorrelation (from 0-to-11) of these 8192 samples is calculated and accumulated for 10k laser shots (1/2 second). Due to arithmetic operations, the size of output data grows from a 16-bit 8192 samples input vector to 32-bit x 12(lags) x 2 (real and imaginary) x 8192 samples. Similarly, to operate the instrument for a period of time equals to M minutes, one has to enter Mx60(seconds)x2(half a second)x2x12x2x8192 as the total number of samples (ceiling) needed to be streamed to the host PC by the acquisition software (SNAP). The streaming software was also modified to output three different files; a binary file containing the autocorrelation of the entire captured data plus some headers, a text file containing entire autocorrelation of captured data without headers, a continuously generated text file containing

the last 30 seconds of autocorrelation of captured data without headers (needed for real-time processing and display).

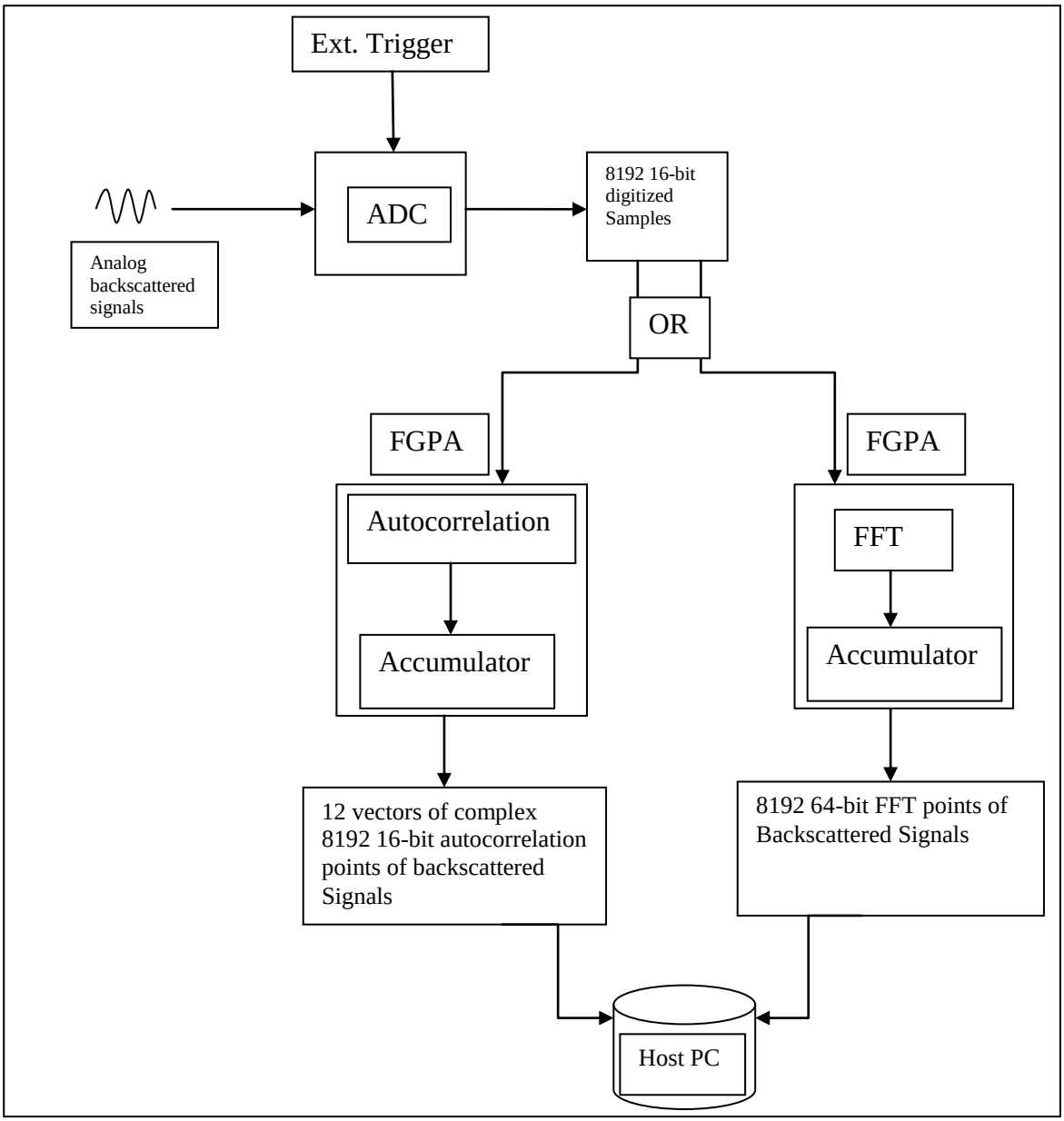

Figure 7-1 Data flow from the ADC to the host PC

7.2 Appendix II MATLAB Programs

In this section, MATLAB programs that were developed for signal processing are presented and explained. These MATLAB programs were changed as data acquisition techniques were developed. For example, Initial data acquisition technique involved streaming raw data to the host PC and archiving it for post processing.

7.2.1 MATLAB Program to Post Process Time Domain Raw Signals

A MATLAB program was developed to process this archived raw data file by finding the beginning of each return pulse, time gate backscattered signals, calculate FFT, accumulate FFTs for all stored pulses, and then estimate and plot wind velocity, signal power, and other statistics. This program was also modified to be run continuously and archive its results.

The program is saved as: ***GO_FFT_23.m***, and the program code is as follows:

```
 clear all;
fprintf('\n////////////////////////////////////////////// ');
 fprintf('\n//////   WELCOM TO GO_FFT_22  program   //////  ');
 fprintf('\n//////   Generated by: Sameh Abdelazim   //////  ');
 fprintf('\n//////   City College of New York        //////  ');
 fprintf('\n////////////////////////////////////////////// \n');

tic
Continuous_Run          = 0; % Run mode (Single/Contiuous)
%Accumulate              = 1;
Accumulation_Time       = 2 ; % Accumulation time in minutes.
%Accumulation_Time_counter = 0;
PRR                     = 20e3; % Pulse Repetition Rate in HZ
No_Gates                = 20;
No_Samples_Per_Gate     = 128;
No_Samples_w_Zero_Pad   = 128;
No_Chunks               = 1;
No_discarded_samples    = 0;
No_Fitted_Points        = 2 ; % The number of points to be fitted from each
                              % Side of the Power Spectrum peak
No_Samples_Per_Chunk    = 10e6;
Power_Scaling           = 10*log10(2*1000/(50*(8192*No_Samples_Per_Gate)^2));
Pulse_Cycle             = (PRR)^-1 ;
Pulse_Width             = 200e-9; % Length of the pulse
Sampling_Rate           = 400e6; % No of samples per second
No_Pulses               = floor((No_Samples_Per_Chunk/Sampling_Rate) * PRR)-3 ;
No_Samples_Per_Pulse    = ( Sampling_Rate / PRR);
c                       = 3e8;
Distance_Per_Pulse      = c * Pulse_Cycle /2 ;  % Distance covered by signal through time of one
pulse
```

```
Time_Per_Sample         = 1/(400e6) ; % time each sample represents
nearest_distance        = No_discarded_samples * 2.5e-9 * 3e8 /2;
Maximum_Range           = c * (No_Samples_Per_Gate * No_Gates + No_discarded_samples ) *
Time_Per_Sample/2  ; % The Maximum Range at which wind speed can be measured in m.
Distance_Per_Gate       = (Maximum_Range - nearest_distance)/No_Gates ;
No_Good_Samples         = No_Samples_Per_Gate * No_Gates ;
No_Bad_Samples          = 0;
Scattered_Signal            = zeros([1 No_Samples_Per_Chunk]);
True_Data_of_One_Gate       = zeros([1 No_Samples_w_Zero_Pad ]);
FFT_True_Data_of_One_Gate   = zeros([1 No_Samples_w_Zero_Pad ]);
Abs_FFT_per_Gate            = zeros([1 No_Samples_w_Zero_Pad ]);
Ave_True_Data               = zeros([1 No_Samples_w_Zero_Pad * No_Gates ]);
Ave_Gate_FFT                = zeros([No_Gates No_Samples_w_Zero_Pad ]);
Old_Ave_Gate_FFT            = zeros([No_Gates No_Samples_w_Zero_Pad ]);
New_Ave_Gate_FFT            = zeros([No_Gates No_Samples_w_Zero_Pad ]);
Peak_Array                  = zeros([2 No_Pulses*No_Chunks ]);
Bad_Sample_Array            = zeros([1 10e3 ]);
Ave_Noise_FFT               = zeros([1 No_Samples_w_Zero_Pad]);
Abs_FFT_noise_floor         = zeros([1 No_Samples_w_Zero_Pad]);
Peak_Freq                   = zeros([1 No_Gates]);
sum1                        = zeros([1 No_Gates]);
sum2                        = zeros([1 No_Gates]);
Peak_Power                  = zeros([1 No_Gates]);
Wind_Speed                  = zeros([1 No_Gates]);
Av_Spec                     = zeros([1 No_Gates]);
MLE_ARRAY                   = zeros([No_Pulses*No_Chunks  No_Gates  2]);
Power_Vector                = zeros([No_Gates 2*No_Fitted_Points +1]);
Frequency_Vector            = zeros([No_Gates 2*No_Fitted_Points +1]);
Wind_Speed_Fitted           = zeros([1 No_Gates]);
Fitted_value                = zeros([1 No_Gates]);
MLE_velocity                = zeros([1 No_Gates]);
Precision                   = zeros([1 No_Gates]);
One_Gate_Frequency_Vector   = zeros([1 10000]);
MLE_Accuracy                = zeros([1 No_Gates]);
j                           = 1;
k                           = 1;
Fs                          = 400e6; % Sampling frequency Hz.
f                           = Fs*(0:No_Samples_w_Zero_Pad/2 -1 )/(No_Samples_w_Zero_Pad);
Freq_Res                    = Fs/No_Samples_w_Zero_Pad ;


if (Continuous_Run == 1)
    source_file = 'C:\Innovative\X5-400M\Examples\Snap\Sameh_Vc9_timer\Debug\HsData.bin';
    dest_file_name = 'Copied_HsData_file.bin';
    file_path = 'C:\Wind_Speed_Archive\' ;
    dest_file = strcat(file_path,dest_file_name);
    copyfile(source_file,dest_file);
    in_file = fopen (dest_file,'r');
    %[year month day hour minute_start sec]=datevec(now);
else
    Group_Wind_Speed      = zeros([Accumulation_Time*3 No_Gates]);
    Group_Precisions      = zeros([Accumulation_Time*3 No_Gates]);
    Group_Peak_Power      = zeros([Accumulation_Time*3 No_Gates]);
    No_Pulses_Counter          = 0;
    No_Runs_Counter            = 0;
    minute_start               = 0;
    file_name = '3in_lens_145A_01202011_2nd.bin' ;
    file_path = 'C:\Data_bin_files\' ;
    data_file = strcat(file_path,file_name);
    in_file = fopen (data_file,'r');
end

Scattered_Signal_1 = (fread(in_file,8,'int16'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%  Reading data from input file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for data_chunk = 1:1:No_Chunks;
        Scattered_Signal_col = (fread(in_file,No_Samples_Per_Chunk,'int16'));

        if (length(Scattered_Signal_col) == No_Samples_Per_Chunk)
            Scattered_Signal = Scattered_Signal_col' ;
```

```
        else
            fprintf('There was a probelm reading from the input file:\n  ');
            fprintf('The size of data read is smaller than assigned array\n  ');
            break ;

        end

         % clear Scattered_Signal_col;

    %%%%%%%%%% Clean the large negative values from the scattered Signal.
        for i =1:1:length(Scattered_Signal);
            if(Scattered_Signal(i)< -1e4 || Scattered_Signal(i) > 1e4)
                 No_Bad_Samples = No_Bad_Samples + 1 ;
                 Bad_Sample_Array(k) = Scattered_Signal(i);
                 k = k+1;
                 Scattered_Signal(i) = (Scattered_Signal(i+1)+Scattered_Signal(i-1))/2;
            end
        end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       for Pulse = 1: 1 : No_Pulses ;
            Peak = 0;
            peak_pos = 0 ;
            found_peak = 0;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%% Peak Search Loop
            %for Sample = Lower_limit: 1: Upper_limit  ;
            for Sample = 1: 1: No_Samples_Per_Pulse  ;
                sample_index = Sample + (Pulse -1)*No_Samples_Per_Pulse;
                    if (Scattered_Signal(sample_index) > Peak)
                        Peak = Scattered_Signal(sample_index);
                        peak_pos = sample_index ;
                        found_peak = 1;
                    end
            end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%  Finding peaks and gating data then calculate FFTs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            if (found_peak == 1)
                    previous_peak_pos = peak_pos;
                    Peak_Array(1,j) = peak_pos + (data_chunk -1)* No_Samples_Per_Chunk;
                    Peak_Array(2,j) = Peak ;
                    j = j+1;

                noise_floor = Scattered_Signal(peak_pos + 10e3: peak_pos + 10e3 +
No_Samples_w_Zero_Pad -1);
                FFT_noise_floor = fft(noise_floor, No_Samples_w_Zero_Pad);
                Abs_FFT_noise_floor = FFT_noise_floor.*conj(FFT_noise_floor) ;
                Ave_Noise_FFT = Ave_Noise_FFT + Abs_FFT_noise_floor/(No_Pulses * No_Chunks) ;

                for Gate = 1: 1: No_Gates ;
                    index_1 = peak_pos + (Gate -1) *No_Samples_Per_Gate + No_discarded_samples ;
                    True_Data_of_One_Gate(1:No_Samples_Per_Gate ) = Scattered_Signal(index_1 :
index_1 + No_Samples_Per_Gate -1 );
                   % Data_Minuse_Noise = True_Data_of_One_Gate - noise_floor ;
                    FFT_True_Data_of_One_Gate = fft(True_Data_of_One_Gate,No_Samples_w_Zero_Pad )
;
                    Abs_FFT_per_Gate =
FFT_True_Data_of_One_Gate.*conj(FFT_True_Data_of_One_Gate);
                    Abs_FFT_per_Gate = Abs_FFT_per_Gate -Abs_FFT_noise_floor ;
                    [peak_power, peak_freq] = max(Abs_FFT_per_Gate);
                    MLE_ARRAY(Pulse, Gate, 1) = ( (2 * peak_freq/No_Samples_w_Zero_Pad) * max(f)
- 84e6 ) * ( (1.5452e-6)/2) ;
                    MLE_ARRAY(Pulse, Gate, 2) = 10*log10(peak_power) + Power_Scaling;
                        for i =1:1:No_Samples_w_Zero_Pad;
                            Ave_Gate_FFT(Gate, i) = Ave_Gate_FFT(Gate,i) +
Abs_FFT_per_Gate(i)/(No_Pulses * No_Chunks) ;
                            Ave_True_Data(i+(Gate -1) *No_Samples_Per_Gate ) =
Ave_True_Data(i+(Gate -1) *No_Samples_Per_Gate) + True_Data_of_One_Gate(i)/ (No_Pulses *
No_Chunks);
                           % sameh_signal(i+(Gate -1) *No_Samples_w_Zero_Pad) =
True_Data_of_One_Gate(i);
```

```
                    end
                end

            end
        end
  %clear Scattered_Signal;
 end

fclose(in_file);

fprintf('\nFinished getting sampled signal and calculating FFTs and: \n ');
toc
tic
fprintf('\nThe total number of pulses in this run was:  ');
    No_of_Accumulations = No_Pulses * No_Chunks



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fig.(17); hold on
for i = 1: 1: No_Gates;

        %  X = (10*log10(Ave_Gate_FFT(i,1:No_Samples_w_Zero_Pad/2 - 1)) + Power_Scaling);
        X = (10*log10(Ave_Gate_FFT(i,1:No_Samples_w_Zero_Pad/2  )) -10*log10(sqrt(No_Pulses)) -
(10*log10(Ave_Noise_FFT(1:No_Samples_w_Zero_Pad/2 ))));
        [peak_power, peak_freq] = max(X);
        Peak_Freq(i) = peak_freq;
        Peak_Power(i) = peak_power;
        Wind_Speed(i) = ( ( peak_freq* Freq_Res) - 84e6 ) * ( (1.5452e-6)/2)*3600/1600 ;
        %plot(f*1e-6,10*log10(Ave_Gate_FFT(i,1:No_Samples_Per_Gate/2 - 1))+ Power_Scaling), grid
on


        % To make sure that the peak frequency is not on the edge  of
        % power spectrum and could be fitted with nearby points
       if ( ((peak_freq + No_Fitted_Points ) < length(X) ) && ((peak_freq - No_Fitted_Points) >
0))
                index1 = peak_freq - No_Fitted_Points;
                for k = 1:1: 2*No_Fitted_Points +1 ; %peak_freq - No_Fitted_Points : 1: peak_freq
+ No_Fitted_Points ;
                    Power_Vector(i,k) = X(index1 +k -1); %Power Spectrum Value
                    Frequency_Vector(i,k) = index1 +k -1; %(2 * k/No_Samples_w_Zero_Pad) *
max(f);
                   % Power_Fitting_Victor(k) = X(index1 +k -1);
                   % Frequency_Fitting_Victor(k) = index1 +k -1;
                end
           % Fitted_Velocity = fit(Frequency_Vector', Power_Vector', 'gauss1');
           % Wind_Speed_Fitted(i) = ( 2 * Fitted_Velocity.b1/No_Samples_w_Zero_Pad * max(f) -
84e6 ) * ( (1.5452e-6)/2) ;

           counter_1 =0;
           counter_2 =0;
            for m = 1:1:2*No_Fitted_Points +1;
                for n = 1:1:floor(abs((Power_Vector(i,m))*100))+1;
                    One_Gate_Frequency_Vector( n + counter_1 ) = Frequency_Vector(i,m);
                    counter_2 = counter_2 +1 ;
                end
                counter_1 = counter_1 +n ;
            end

            Stat_Array = mle(One_Gate_Frequency_Vector(1:counter_2));
            My_mu = Stat_Array(1);
            My_sigma = Stat_Array(2);
            MLE_velocity(i) = ( 2 * My_mu/No_Samples_w_Zero_Pad * max(f) - 84e6 ) * ( (1.5452e-
6)/2)*3600/1600 ;
            MLE_Accuracy(i) = [My_sigma-My_mu] * 1.5/2;


          end
```

```
end


fprintf('\n Fig. plotting was done and :\n  ');
toc
tic


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Writing Instantaneous Data File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (Continuous_Run == 1)
        [year month day hour minute sec]=datevec(now);

        Prefix_Name = '';
        junk_str = num2str(month);
        if month < 10
            Prefix_Name = '0';
        end
        Prefix_Name = strcat(Prefix_Name,junk_str);
        Prefix_Name = strcat(Prefix_Name,'_');

        junk_str = num2str(day);
        if day < 10
            Prefix_Name = strcat(Prefix_Name,'0');
        end
        Prefix_Name = strcat(Prefix_Name,junk_str);
        Prefix_Name = strcat(Prefix_Name,'_');

        junk_str = num2str(year);
        Prefix_Name = strcat(Prefix_Name,junk_str);
        Prefix_Name = strcat(Prefix_Name,'_');

        instantaneous_file_name_str = strcat('INSTANTANEOUS_WIND_SPEEDS_',Prefix_Name);
        instantaneous_file_name_str = strcat(instantaneous_file_name_str,'.txt');
        Accumulated_file_name_str = strcat('AccumulateD_WIND_SPEEDS_',Prefix_Name);
        Accumulated_file_name_str = strcat(Accumulated_file_name_str,'.txt');

        file_path = 'C:\Wind_Speed_Archive\';
        instantaneous_file_name = strcat(file_path,instantaneous_file_name_str);
        Accumulated_file_name = strcat(file_path,Accumulated_file_name_str);

        FID_instantaneous = fopen(instantaneous_file_name,'a') ;
        FID_Accumulated = fopen(Accumulated_file_name,'a') ;

        if( ( FID_instantaneous) >= 1 )
            fprintf('I am writing into instantaneous file now\n');
        else
            fprintf('I will open a instantaneous file for writingam now\n');
            FID_instantaneous = fopen(instantaneous_file_name,'w') ;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%Accumulating Wind Speeds, Signal Powers and STD
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        No_Pulses_Counter = No_Pulses_Counter + No_Pulses*No_Chunks ;
        No_Runs_Counter = No_Runs_Counter + 1;
        range = Distance_Per_Gate : Distance_Per_Gate: Distance_Per_Gate*No_Gates;
        for i = 1:1:No_Gates ;
            Precision(i)                                =
0.91*(1+2.1*Peak_Power(i))/(Peak_Power(i)*sqrt(No_Pulses));
%             Accumulated_Wind_Speed(i)                 = Accumulated_Wind_Speed(i)+
MLE_velocity(i);
%             Accumulated_Peak_Power(i)                 = Accumulated_Peak_Power(i)+ Peak_Power(i)
;
            Group_Wind_Speed(No_Runs_Counter,i)     = MLE_velocity(i);
            Group_Precisions(No_Runs_Counter,i)     = Precision(i);
            Group_Peak_Power(No_Runs_Counter,i)     = Peak_Power(i);
        end
    current_date = datevec(now);
```

```
fprintf(FID_instantaneous,'================================================================================
============\n ');
    fprintf(FID_instantaneous,'%2d %2d %2d %2d:%2d ', year, month, day, hour, minute);
    fprintf(FID_instantaneous,'\tNo_Gates=% 2d', No_Gates);
    fprintf(FID_instantaneous,'\tNo_Samples_Per_Gate=% 2d',No_Samples_Per_Gate );
    fprintf(FID_instantaneous,'\tNo_Pulses= %2d \n', No_Pulses*No_Chunks);
    fprintf(FID_instantaneous,'\nRange(m)\tVelocity(mph)\tSignal Powre(dB)\tPrecision \n ');
    fprintf(FID_instantaneous,'========\t=============\t================\t=========\n ');
    for i = 1:1:No_Gates;
            fprintf(FID_instantaneous,'%d ', range(i));
            fprintf(FID_instantaneous,'\t\t%+8.1f ', MLE_velocity(i));
            fprintf(FID_instantaneous,'\t\t%8.1f ', Peak_Power(i));
            fprintf(FID_instantaneous,'\t\t%8.1f ', Precision(i));
            fprintf(FID_instantaneous,'  \n');
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%% Writing into Accumulation File
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %Accumulation_Time_counter = Accumulation_Time_counter + minute - minute_start ;

    if(minute < minute_start)
        minute = minute + 60;
    end

    if (  Accumulation_Time <= minute - minute_start  )
        if (minute > 60)
            minute = minute -60 ;
        end
        minute_start = minute  ;

            if( ( FID_Accumulated) >= 1 )
                fprintf('I am writing into Accumulated file now\n');
            else
                fprintf('I will open a Accumulated file for writingam now\n');
                FID_Accumulated = fopen(FID_Accumulated,'w') ;
            end
            current_date = datevec(now);

fprintf(FID_Accumulated,'================================================================================
================================\n ');
    fprintf(FID_Accumulated,'%2d %2d %2d %2d:%2d ', year, month, day, hour, minute);
    fprintf(FID_Accumulated,'   No_Gates=% 2d', No_Gates);
    fprintf(FID_Accumulated,'   No_Samples_Per_Gate=% 2d',No_Samples_Per_Gate );
    fprintf(FID_Accumulated,'   No_Pulses= %2d', No_Pulses_Counter);
        fprintf(FID_Accumulated,'   Accumilation time(minutes)= %2d \n\n', Accumulation_Time);
        fprintf(FID_Accumulated,'Range(m)\tAveraged Velocity(mph)\tSTD of Velocity\tSignal
Power(dB)\tSTD of Power(dB)\tPrecision \n ');

fprintf(FID_Accumulated,'========\t====================\t==============\t==============\t====
============\t========= \n ');
        for i = 1:1:No_Gates;
                fprintf(FID_Accumulated,'%d ', range(i));
                fprintf(FID_Accumulated,'\t\t%+8.1f ',  mean(Group_Wind_Speed(:,i)));
                fprintf(FID_Accumulated,'\t\t\t%8.1f ', std(Group_Wind_Speed(:,i)));
                fprintf(FID_Accumulated,'\t\t%8.1f ',   mean(Group_Peak_Power(:,i)));
                fprintf(FID_Accumulated,'\t\t%8.1f ',   std(Group_Peak_Power(:,i)));
                fprintf(FID_Accumulated,'\t\t%8.1f ',   mean(Group_Precisions(:,i)) );
                fprintf(FID_Accumulated,'  \n');
            %end
            %fprintf(file_name_2,'\n');
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%% Reset all counters and accumulated arrays
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    No_Pulses_Counter           = 0;
    No_Runs_Counter             = 0;
    Group_Wind_Speed      = zeros([Accumulation_Time*3 No_Gates]);
    Group_Precisions      = zeros([Accumulation_Time*3 No_Gates]);
    Group_Peak_Power      = zeros([Accumulation_Time*3 No_Gates]);
```

```
end %%  end of continuous run if statement


    fprintf('\n Writing into files was finished and: \n  ');
    toc
    fclose('all');



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Plotting section
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic

 Fig.(11);
 for i = 1: 1: No_Gates;

subplot(5,6,i),plot(f(5:No_Samples_w_Zero_Pad/2),(10*log10(Ave_Gate_FFT(i,5:No_Samples_w_Zero_Pad
/2 )) -10*log10(sqrt(No_Pulses)) - (10*log10(Ave_Noise_FFT(5:No_Samples_w_Zero_Pad/2 )))),'b'),
grid on
 end


  Fig.(30)
  xtics = [1*Distance_Per_Gate:Distance_Per_Gate:No_Gates*Distance_Per_Gate];
  Peak_Power(1) = Peak_Power(No_Gates);
  plot(xtics,Peak_Power(1:No_Gates)), grid on
  %hold on; scatter(xtics,MLE_Accuracy(3:No_Gates));
  title('SNR vs. Distance','FontSize',16);
  xlabel('distance (m)','FontSize',16);
  ylabel('SNR (dB)','FontSize',16);

  Fig.(4)
  range = Distance_Per_Gate : Distance_Per_Gate: Distance_Per_Gate*No_Gates;
%   for i = 1: 1  : length(Wind_Speed) ;
%       if(Wind_Speed(i) > 30 || Wind_Speed(i)<-30)
%           Wind_Speed(i) = 0;
%       end
%   end
  bar(range,Wind_Speed), grid on
  title('Peak Search Wind Speed vs. Distance','FontSize',16);
  xlabel('Distance (m)','FontSize',16);
  ylabel('Wind Speed (mph)','FontSize',16);



  Fig.(6)
%   for i = 1: 1  : length(Wind_Speed) ;
%       if(MLE_velocity(i) > 20 || MLE_velocity(i)<-20   )
%           MLE_velocity(i) = 0;
%       end
%   end
  MLE_velocity(1) =0;
  bar(range,MLE_velocity), grid on
  title('Gaussian curve fitting Wind Speed vs. Distance','FontSize',16);
  xlabel('Distance (m)','FontSize',16);
  ylabel('Wind Speed (mph)','FontSize',16);

fprintf('Finished plotting and: \n');
toc
```

7.2.2 Fiber Tip Z-Position vs. Focal Location

A MATLAB program was developed to help setting the optical antenna by adjusting the position of the fiber tip with respect to the focal point of the lens. The output of this program is a plot of fiber position displacement from the lens' focal point vs. laser focus distance in space, Fig. 7-2. In order to determine the distance in space where the laser beam is focused, backscattered signals are observed on an oscilloscope and the fiber optics' mount micrometer is adjusted to maximize the backscattered signals. By maximizing received signals, we ensure that the laser beam is focused at that particular distance. To adjust the focus distance to a different distance: d, then the fiber tip is moved to a distance +/- δz, which is given by:

$$\delta z = \frac{f^2}{d}$$

, where f is the focal distance of the lens.

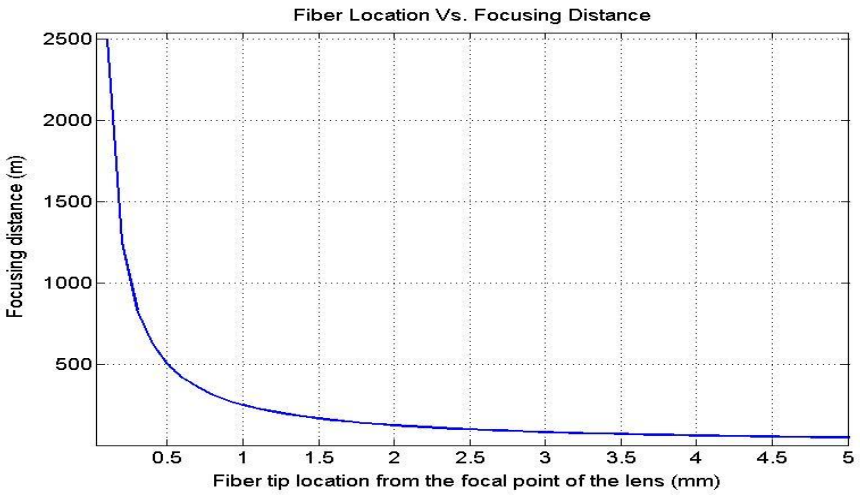

Figure 7-2 Adjusting fiber tip position guiding curve

The MATLAB program is saved as: and the ***Fiber_location_Vs_Focusing_Distance.m*** code:

```
clear all;
f = 0.50 ; % focal length in m
for i = 1:1:100 ;  % Image location in m
    d(i) = i*(1e-4);
    So(i) = f + d(i);
```

```
    Si(i) = So(i)*f/(So(i)-f); % Object location
end
Fig.(1)
plot(d*1000, Si)
xlabel('Fiber location (mm)','FontSize',14)
ylabel(' Focusing distance (m)','FontSize',14)
title('Fiber Location Vs. Focusing Distance','FontSize',14),grid on
```

7.2.3 Analysis of LO power vs. efficiency on power penalty

This MATLAB simulation was programmed to determine the optimum level of local oscillator power with respect to SNR. The program plots the profiles of LO power vs SNR at different levels of RIN, Fig. 7-3.

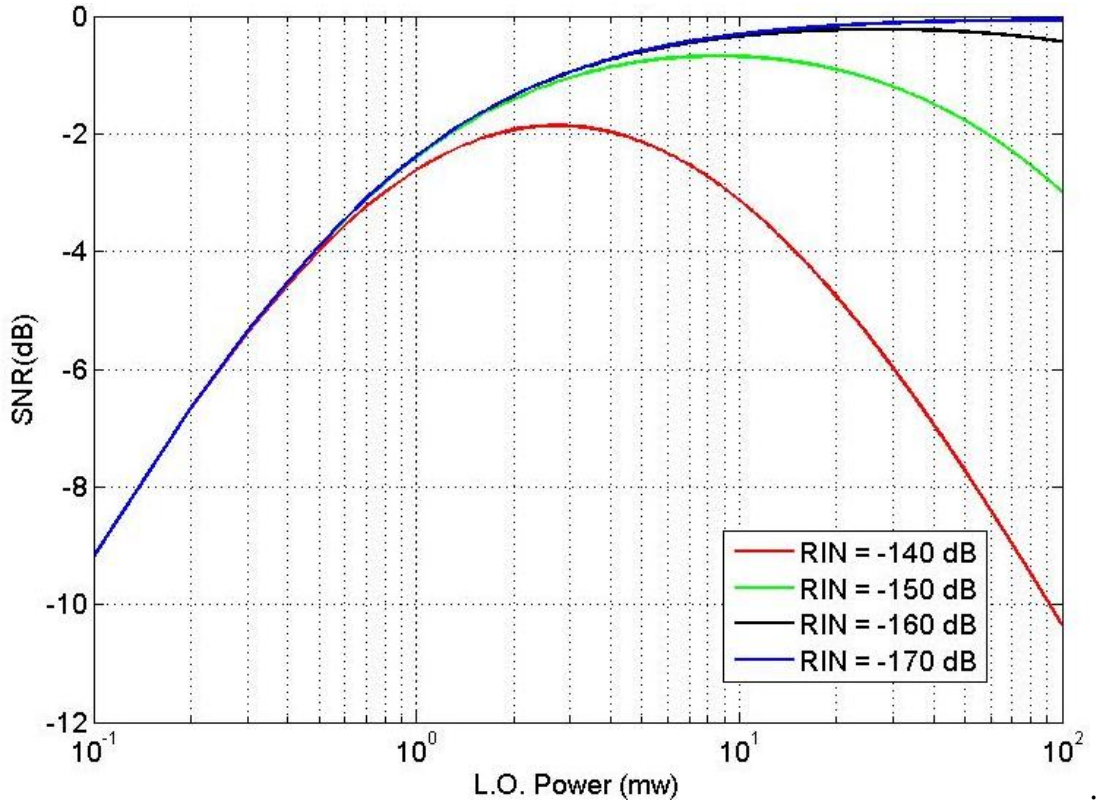

Figure 7-3, Effect of LO power level on SNR

The MATLAB program is saved as: ***SNR_vs_RIN.m***

```
clear all

h=6.63e-34;
k=1.38e-23;
e=1.6e-19;
T=295;
v=1.94e14;
eighta_q=0.8;
RIN= [1e-14 1e-15 1e-16 1e-17] ;
```

```
R_L=70;
F_N=1;
R_B=10^-(2.5);
w = (2*k*T*F_N*h*v)/(eighta_q*(e^2)*R_L);
z = (eighta_q*RIN.*R_B)/(2*h*v);

i=0;
for P_L=0.1:.1:100;
    i = i+1;
    x(i) = P_L;

Efficiency(i,1) = (( 1 + w/(P_L*1e-3)  + z(1) * P_L*1e-3)^-1) ;
Efficiency(i,2) = (( 1 + w/(P_L*1e-3)  + z(2) * P_L*1e-3)^-1) ;
Efficiency(i,3) = (( 1 + w/(P_L*1e-3)  + z(3) * P_L*1e-3)^-1) ;
Efficiency(i,4) = (( 1 + w/(P_L*1e-3)  + z(4) * P_L*1e-3)^-1) ;
end

semilogx((x),10*log10(Efficiency(:,1)),'r','LineWidth',2), hold on
semilogx((x),10*log10(Efficiency(:,2)),'g','LineWidth',2), hold on
semilogx((x),10*log10(Efficiency(:,3)),'k','LineWidth',2), hold on
semilogx((x),10*log10(Efficiency(:,4)),'b','LineWidth',2), hold on
grid on;
title('Effect of Local Oscillator power on SNR','FontSize',14);
xlabel('L.O. Power (mw)','FontSize',14);
ylabel('SNR(dB)','FontSize',14);
```

7.2.4 Analysis of Range Dependence of SNR

In this MATLAB program the SNR range dependence of three different size lenses was investigated under different focusing distances. The program outputs a .avi file that can be played to display the SNR behavior of the three lenses when the focusing distance of the laser beam is changed. The program also outputs the three lenses range dependent SNR, Fig. 7-4.

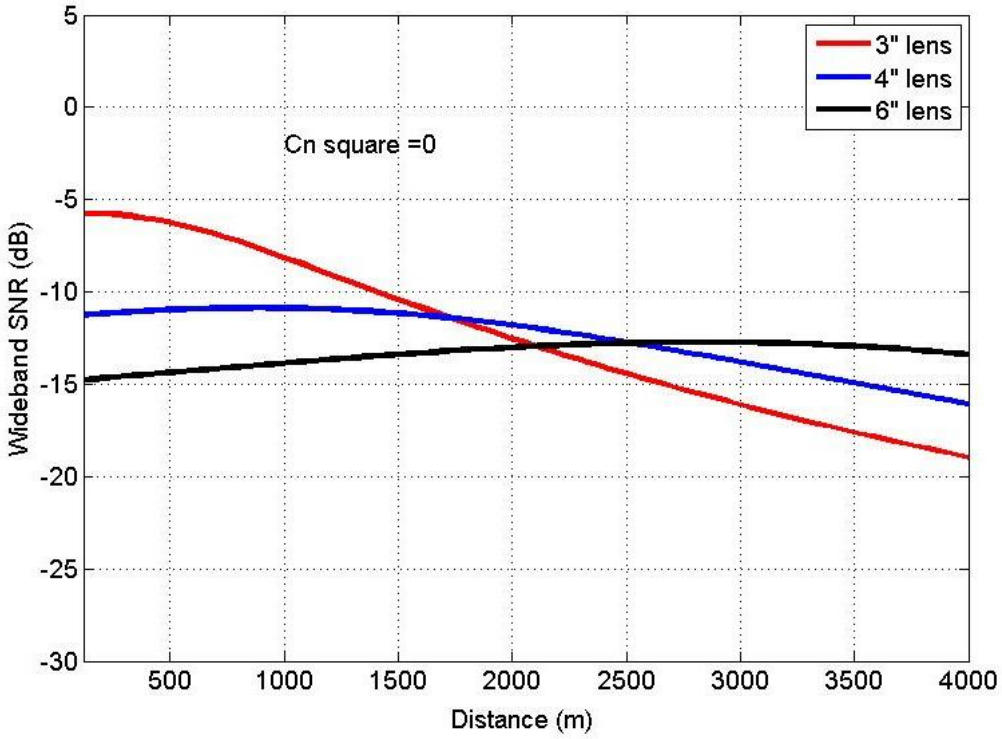

Figure 7-4 SNR Range dependence of 3", 4", and 6" lenses

The MATLAB program is saved as: ***LIDAR_transmission_attenuation3.m***, where the code is written as follows:

```
clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is a SNR simulation of LIdar signal based on equation 3
% of Mitsubishi paper titled compact all-fiber pulsed coherent doppler
% lidar system for wind sensing and it was tested and verified on 4/4/2011
% Generated by Sameh Abdelazim
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

R = 1 ; % reflectivity of a hard target
c = 3e8; % speed of light
lambda = 1.5452e-6; % Wave length (m)
f = c/lambda; % Laser Frequency
h = 6.626e-34; % Planck's constant
B = 100e6; % for wide band SNR use 100MHz, for narrow band SNR use 2MHz Bandwidth .
KW = 2*pi/lambda; %Wave number;
%The following parameter should be checked
Cn_sq = 0;%2e-14; % Refractive Index Structure Constant;
E = 14e-6; % Pulse Energy (J)
D = [0.07 0.1 0.15]; %Effective aperture Diameter (m)
thao = 200e-9; % Pulse width.
beta = 8.3e-7; %R/(f*thao); %atmospheric backscatter coefficiency (/m/sr)
% The following parameter needs to be checked
K =  0.95;   % one way atmospheric transmittance (/km)
F = [500:200:6000]; % Focal Range of Optical Antena (m)
Eta_F = 0.9; %0.355 ; % Far field system efficiency (-4.5 dB)
Ac = [0.66 0.76  0.76  ]; % Correction Factor.
Eta_I = 0.9; %0.676; % Factor for insertion loss of optical components
             % except for telescop (-1.7 dB)
Eta_RE = 0.977; % Telescope absorption and reflection loss (-0.1 dB)
Eta_A = 0.91; % Telescope reflection loss (-0.4 dB)
```

```
Eta_PP = 0.912;
Eta_Q = 0.9; %0.794;
Eta_S = Eta_I * Eta_RE * Eta_A * Eta_PP * Eta_Q ; %

output_file = fopen ('C:\Documents and Settings\Sameh\My
Documents\CUNY\PhD\Matlab_Codes_Sameh\SNR_3_lenses_by_Matlab.xlsx','w');

for i = 1: 1:28 % the focal length of the lens

        for j = 1:1:3; % D: for three aperture diameters
             for L = 1:1:100; % Distance (m)
                 Distance(L) = 120+ (L-1)*48;
                 So = (1.1 * KW^2 * Distance(L) * Cn_sq)^(-3/5);

                 term_a = (1- Distance(L)/F(i))^2 ;
                 term_b = ([pi*(Ac(j)*D(j))^2/(4*lambda*Distance(L))]^2) ;
                 term_c = [Ac(j)*D(j)/(2*So)]^2;

                 Eta_D = ( Eta_S * Eta_F)/(1+ term_a* term_b + term_c) ;
                 Term_1 = lambda * E * beta * [(K)^(2*Distance(L)/1000)]*pi* (D(j))^2/(8 * h* B*
(Distance(L))^2);
                 SNR(j,L) = 10*log10(Eta_D* Term_1) ;
             end
        end
       % xlswrite('output_file.xlsx' , F(i));
        xlswrite('output_file_3.xlsx' , SNR', i);

        Fig.(300), hold off
        axis([0 4e3 -50 25]);
        xlabel('distance (m)');
        ylabel('SNR (dB)');
        grid on;
        str1 = num2str(F(i)) ;
        str2 = strcat('Beam is focused at: ', str1, 'm');
        str3 = num2str(Cn_sq);
        str4 = strcat('Cn square = ',str3);

         axis([Distance(1) 4e3 -30 5]),plot(Distance, (SNR(1,:)),'r','LineWidth', 3)
,text(3000,20,'3" Lens','Color','r','FontSize',12), hold on
         axis([Distance(1) 4e3 -30 5]),plot(Distance, (SNR(2,:)),'b','LineWidth', 3)
,text(3000,15,'4" Lens','Color','b','FontSize',12), hold on
         axis([Distance(1) 4e3 -30 5]),plot(Distance, (SNR(3,:)),'k','LineWidth', 3)
,text(3000,10,'6" Lens','Color','k','FontSize',12), hold on
        text(1000,3,str2,'Color','k','FontSize',12), hold on
        text(1000,-2,str4,'Color','k','FontSize',12), hold on
        grid on;
        U(i) = getframe;
        hold off;
        xlabel('Distance (m)'), ylabel('Wideband SNR (dB)')
end

%fclose(output_file_2.xlsx);
 %movie(U,1,1, [0 0 0 0 ])

 movie2avi(U, 'myPeaks.avi', 'compression', 'None');
```

7.2.5 Analysis of FPGA FFT output

In the beginning stages of developing the FFT algorithm on the FPGA, a .txt output file containing the entire power spectrum of acquired data was saved on the desk of the host PC. A MATLAB program was then developed to post process this power spectrum text file. This

MATLAB program builds a power spectrum array corresponding to range gates, estimate wind velocities and signal intensities, and then plot the results.

The MATLAB program that reads a text file, calculates a back ground noise from the last range gate of backscattered signals, processes the power spectrum, and display results is saved as: ***Range_Corrected_MLE_8192_Signal_Int_W_BK_noise_cal.m***, where the code is written as:

```
clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% This program reads a text file of an FFT return signal
%%%% and calculates the averaged back ground noise, then saves it
%%%% to a new file. You just need to provide the input file name,
%%%% path and file length
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

file_name = 'HsData_FPGA_ACCUM_7_08182011.txt' ;
  file_path = 'D:\Bin_Files\' ;
  Time_Length = 14400;

  data_file_name = strcat(file_path,file_name);
  in_file = fopen (data_file_name,'r');
  back_ground_file_name = strcat(file_path,'BackGround_Noise_',file_name);
  output_file = fopen(back_ground_file_name,'w')
  back_ground = zeros(1, 64);
  new_noise = zeros(4096,1);
%   Signal_Sum = zeros(4096,1);
%   Signal_Intensity = zeros(64,72);

for n =1:1:Time_Length;
      new_noise = (fscanf(in_file,'%f',8192));
      back_ground = back_ground + new_noise(1:64)';
end
 fprintf(output_file,'% g',back_ground/Time_Length);
fclose('all')

    noise_file_name = 'BackGround_Noise_my_output_file_3.txt';

    No_Averages = 1; % No of 1/2 seconds averages.
    File_Length = Time_Length; % Length of the run in 1/2 seconds.
    SNR_Thresh = 0.16; % SNR threshold to make a decision on a measured value
    No_Fitted_Points = 5 ; % No of spectrum points to be fitted in MLE


    %noise_file = strcat(back_ground_file_name);
      in_file = fopen (data_file_name,'r');
      in_file2 = fopen(back_ground_file_name,'r');
      back_ground = zeros(1, 64);
      One_Gate_Spectrum = zeros(1,64);
      Signal = zeros(8192,1);
      Signal_Sum = zeros(8192,1);
      Signal_Intensity = zeros(64,File_Length/No_Averages);
      Freq_Shift       = zeros(64,File_Length/No_Averages);
      MLE_Freq_Shift   = zeros(64,File_Length/No_Averages);
      Range_Corrected  = zeros(60,File_Length/No_Averages);
      Range_Corrected2  = zeros(60,File_Length/No_Averages);
      Precision        = zeros(64,File_Length/No_Averages);
      Power_Vector     = zeros(1,2*No_Fitted_Points +1);
      Frequency_Vector = zeros(1,2*No_Fitted_Points +1);
      MLE_Precision        = zeros(64,File_Length/No_Averages);
      One_Gate_Frequency_Vector = zeros(1,2000);
```

```
      Back_Ground_noise = (fscanf(in_file2,'%f'));
%       output_file_str = strcat(data_file(1:length(data_file)-13),'reorganized.txt');
%       output_file = fopen( output_file_str ,'w') ;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for n =1:1:File_Length;
      Signal = (fscanf(in_file,'%f',8192));
      Signal_Sum = Signal_Sum + Signal/No_Averages ;
      if(mod(n,No_Averages)==0)

      for j =1:1:64;
          One_Gate_Spectrum = 10*log10(Signal_Sum(10+(j-1)*128:(j-1)*128+50))-
10*log10(Back_Ground_noise(10:50)) ;
          [Max_Signal, Max_Freq] = max(One_Gate_Spectrum);
          Signal_Intensity(j,n/No_Averages) = Max_Signal;
         % if j > 5
              Range_Corrected2(j,n/No_Averages)  = 10*log10(10^(Max_Signal/10) * (j*48)^2);
         % else
         % end

          Precision(j,n/No_Averages)= 0.83*(1+0.32*Max_Signal)/(Max_Signal*sqrt(No_Averages *
10e3));
          if (Max_Signal > SNR_Thresh)
                Freq_Shift(j,n/No_Averages) = (84e6 - (Max_Freq + 8)*3.125e6 )* ( (1.5452e-6)/2)
;

                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                  %%%% The following section will calculate MLE
                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                if ( ((Max_Freq + No_Fitted_Points ) < length(One_Gate_Spectrum) ) && ((Max_Freq
- No_Fitted_Points) > 0))
                        index1 = Max_Freq - No_Fitted_Points;
                        for k = 1:1: 2*No_Fitted_Points +1 ; %peak_freq - No_Fitted_Points : 1:
peak_freq + No_Fitted_Points ;
                            Power_Vector(k) = One_Gate_Spectrum(index1 +k -1)'; %Power Spectrum
Value
                            Frequency_Vector(k) = 8+index1 +k -1; %(2 * k/No_Samples_w_Zero_Pad)
* max(f);
                        end

                        counter_1 =0;
                        counter_2 =0;
                        for m = 1:1:2*No_Fitted_Points +1;
                            for i = 1:1:floor(abs((Power_Vector(m))*100))+1;
                                One_Gate_Frequency_Vector( i + counter_1 ) = Frequency_Vector(m);
                                counter_2 = counter_2 +1 ;
                            end
                            counter_1 = counter_1 +i ;
                        end
                        Stat_Array = mle(One_Gate_Frequency_Vector(1:counter_2));
                        My_mu = Stat_Array(1);
                        My_sigma = Stat_Array(2);
                        MLE_Freq_Shift(j,n/No_Averages) = (  84e6 -  My_mu *3.125e6 ) * (
(1.5452e-6)/2) ;
                        MLE_Precision(j,n/No_Averages) = [My_sigma-My_mu] * 1.5/2;
                else
                         MLE_Freq_Shift(mod(j,64)+1,n/No_Averages) = 100;
                         MLE_Precision(j,n/No_Averages) = 10 ;
                end
           else
              Freq_Shift(mod(j,64)+1,n/No_Averages) = 100;
              Signal_Intensity(j,n/No_Averages) = 0;
              Precision(j,n/No_Averages) = 10 ;
              MLE_Freq_Shift(mod(j,64)+1,n/No_Averages) = 100;
              MLE_Precision(j,n/No_Averages) = 10 ;
           end
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
      %%%% End of MLE
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              end

              Signal_Sum  = zeros(8192,1);

      else
      end

      %back_ground = back_ground + Signal(4993:5120)';
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This part is a SNR simulation of LIdar signal based on equation 3
% of Mitsubishi paper titled compact all-fiber pulsed coherent doppler
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

R = 1 ; % reflectivity of a hard target
c = 3e8; % speed of light
lambda = 1.5452e-6; % Wave length (m)
f = c/lambda; % Laser Frequency
h = 6.626e-34; % Planck's constant
B = 100e6; % for wide band SNR use 100MHz, for narrow band SNR use 2MHz Bandwidth .
KW = 2*pi/lambda; %Wave number;
%The following parameter should be checked
Cn_sq = 2e-14; % Refractive Index Structure Constant;
E = 13e-6; % Pulse Energy (J)
D = 0.1; %Effective aperture Diameter (m)
thao = 200e-9; % Pulse width.
beta = 8.3e-7; %R/(f*thao); %atmospheric backscatter coefficiency (/m/sr)
% The following parameter needs to be checked
K =  0.95;   % one way atmospheric transmittance (/km)
F = 1500; % Focal Range of Optical Antena (m)
Eta_F = 0.9; %0.355 ; % Far field system efficiency (-4.5 dB)
Ac = 0.76; % Correction Factor.
Eta_I = 0.9; %0.676; % Factor for insertion loss of optical components
               % except for telescop (-1.7 dB)
Eta_RE = 0.977; % Telescope absorption and reflection loss (-0.1 dB)
Eta_A = 0.91; % Telescope reflection loss (-0.4 dB)
Eta_PP = 0.912;
Eta_Q = 0.9; %0.794;
Eta_S = Eta_I * Eta_RE * Eta_A * Eta_PP * Eta_Q ; %


     for L = 1:1:60; % Distance (m)
         Distance(L) = L*48;
         So(L) = (1.1 * KW^2 * Distance(L) * Cn_sq)^(-3/5);

         term_a(L) = (1- Distance(L)/F)^2 ;
         term_b(L) = ([pi*(Ac*D)^2/(4*lambda*Distance(L))]^2) ;
         term_c(L) = [Ac*D/(2*So(L))]^2;

         Eta_D(L) = ( Eta_S * Eta_F)/(1+ term_a(L)* term_b(L) + term_c(L)) ;
         Term_1(L) = lambda * E * beta * [(K)^(2*Distance(L)/1000)]*pi* (D)^2/(8 * h* B*
Distance(L)^2);
         SNR(L) = Eta_D(L)* Term_1(L) ;
     end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:1:64;
Fig.(11), subplot(8,8,i), plot(10*log10(Signal(10+(i-1)*64:(i-1)*64+50))-
10*log10(Back_Ground_noise(10:50)))
end
```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%55
 Fig.(14), image(Signal_Intensity(9:64,:))
XTIC_VECTOR = [1:3600/No_Averages:File_Length/No_Averages];
XTIC_VECTOR_LABEL = [0:30:30*length(XTIC_VECTOR)-1];
set(gca,'XTick',XTIC_VECTOR);
set(gca,'XTickLabel',XTIC_VECTOR_LABEL)
YTIC_VECTOR = [1:5:59];
YTIC_VECTOR_LABEL = [100:250:3000]*cos(30*pi/180);
set(gca,'YTick',YTIC_VECTOR);
set(gca,'YTickLabel',YTIC_VECTOR_LABEL)
grid on;
ylabel('Height (m)');
xlabel('Time (min)');
title(['Signal Intensity dB ']);

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fig.(15), image(Precision(9:64,:))
XTIC_VECTOR = [1:3600/No_Averages:File_Length/No_Averages];
XTIC_VECTOR_LABEL = [0:30:30*length(XTIC_VECTOR)-1];
set(gca,'XTick',XTIC_VECTOR);
set(gca,'XTickLabel',XTIC_VECTOR_LABEL)
YTIC_VECTOR = [1:5:59];
YTIC_VECTOR_LABEL = [100:250:3000];
set(gca,'YTick',YTIC_VECTOR);
set(gca,'YTickLabel',YTIC_VECTOR_LABEL)
grid on;
ylabel('Height (m)');
xlabel('Time (min)');
title(['Estimate percision (m/s) ']);

Fig.(16), image(Freq_Shift(9:64,:))
XTIC_VECTOR = [1:3600/No_Averages:File_Length/No_Averages];
XTIC_VECTOR_LABEL = [0:30:30*length(XTIC_VECTOR)-1];
set(gca,'XTick',XTIC_VECTOR);
set(gca,'XTickLabel',XTIC_VECTOR_LABEL)
YTIC_VECTOR = [1:5:59];
YTIC_VECTOR_LABEL = [100:250:3000];
set(gca,'YTick',YTIC_VECTOR);
set(gca,'YTickLabel',YTIC_VECTOR_LABEL)
grid on;
ylabel('Height (m)');
xlabel('Time (min)');
title(['Wind speed (m/s) ']);


Fig.(17), image(MLE_Freq_Shift(9:64,:))
XTIC_VECTOR = [1:3600/No_Averages:File_Length/No_Averages];
XTIC_VECTOR_LABEL = [0:30:30*length(XTIC_VECTOR)-1];
set(gca,'XTick',XTIC_VECTOR);
set(gca,'XTickLabel',XTIC_VECTOR_LABEL)
YTIC_VECTOR = [1:5:59];
YTIC_VECTOR_LABEL = [100:250:3000];
set(gca,'YTick',YTIC_VECTOR);
set(gca,'YTickLabel',YTIC_VECTOR_LABEL)
grid on;
ylabel('Height (m)');
xlabel('Time (min)');
title(['MLE wind speed (m/s) ']);

for i = 1:1:59;
    Range_Corrected(i,:) = (Signal_Intensity(i+5,:)-20)-10*log10(SNR(i+1));
end

 Fig.(18), image(Range_Corrected(1:59,:))
XTIC_VECTOR = [1:1200/No_Averages:File_Length/No_Averages];
XTIC_VECTOR_LABEL = [0:10:10*length(XTIC_VECTOR)-1];
set(gca,'XTick',XTIC_VECTOR);
set(gca,'XTickLabel',XTIC_VECTOR_LABEL)
YTIC_VECTOR = [1:5:59];
YTIC_VECTOR_LABEL = [100:250:2900];
set(gca,'YTick',YTIC_VECTOR);
```

```
set(gca,'YTickLabel',YTIC_VECTOR_LABEL)
grid on;
ylabel('Height (m)');
xlabel('Time (min)');
title(['Range Corrected Signal Intensity dB ']);

 Fig.(19), image(Range_Corrected2(9:64,:))
XTIC_VECTOR = [1:1200/No_Averages:File_Length/No_Averages];
XTIC_VECTOR_LABEL = [0:10:10*length(XTIC_VECTOR)-1];
set(gca,'XTick',XTIC_VECTOR);
set(gca,'XTickLabel',XTIC_VECTOR_LABEL)
YTIC_VECTOR = [1:5:59];
YTIC_VECTOR_LABEL = [100:250:2900];
set(gca,'YTick',YTIC_VECTOR);
set(gca,'YTickLabel',YTIC_VECTOR_LABEL)
grid on;
ylabel('Height (m)');
xlabel('Time (min)');
title(['Range corrected Signal Intensity dB ']);


Fig.(2),surf(Signal_Intensity(5:64,:))
Fig.(3), surf(Freq_Shift)

% for j = 1:1:8;
% Fig.(j+10), hold on
% for k = 1:1:8;
% i = k + (j-1)*8; hold on
% subplot(3,3,k),  plot(10*log10(Signal(1+(i-1)*128:(i-1)*128+64))-
10*log10(Back_Ground_noise(1:64)),'b')
% end
% end

%Fig.(10), plot(10*log10(back_ground(1:64)));
fclose('all')
```

7.2.6 Analysis of FPGA autocorrelation output

The analysis of data measurements obtained using the FPGA autocorrelation pre-processing algorithm took different stages. Initially an output text file containing the lags matrix of backscattered signals' autocorrelation is written to the desk for post processing. A MATLAB program is written to read that text file, calculates the autocorrelation of a desired range gate length, calculates FFT of each range gate, and then estimate wind velocity, signal intensity, and other statistical data.

The MATLAB program is saved as: ***Analyze_FPGA_Short_FILE_AutoCorrelation_V2.m***

7.2.7 Real time measurements display

This MATLAB program is written to provide display Fig.s of measured wind velocity and signal intensity while obtaining data. The importance of this program is to allow users to monitor wind velocity results and other estimated quantities while backscattered signal is being acquired and streamed to the host PC, i.e. real-time measurement. The program provides power spectrum plots of each processed range gate, SNR vs. range plot, vertical wind velocity plots, wind barb plot (in scan mode), and two plots of signal power (range corrected, and non-range corrected).

A user can run this program from any directory by starting MATLAB, and then run a program named "Wind_Display_Real_Time.m", which will guide the user through some gui windows to set up running conditions as follows:

1- Select a working directory, where the SNAP program will be running and writing its 30 seconds, Fig. 7-5. The MATLAB program will create a folder in this directory named with the current data (**YYYYMMDD_Output_Files**) where it saves all 30 seconds files after renaming them with the following format:

   (PREFIX_YYYYMMDDTHHMMSS.txt), where PREFIX is:

   a. FFT_Vert (for vertical mode with FFT pre-processing)
   b. FFT_Scanning (for scanning mode with FFT pre-processing)
   c. AutoCorrelation_Vert (for vertical mode with autocorrelation pre-processing)
   d. AutoCorrelation_Scanning (for scanning mode with autocorrelation pre-processing)

And YYYY is four digit year, MM two digit month, DD is two digit day, T is a fixed, HH is a two digit hour, MM is a two digit minute, SS is a two digit seconds, and .txt is the file extension.

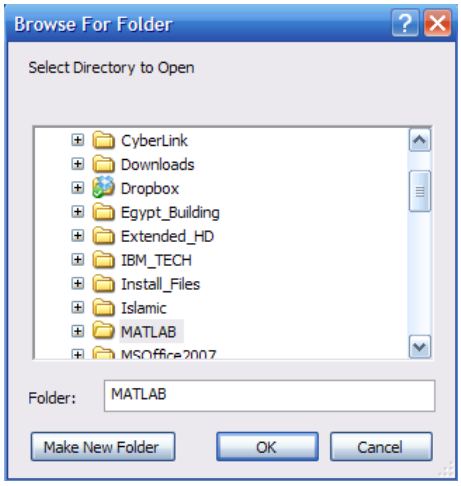

Figure 7-5 Selecting the working folder when running the real-time display MATALB program

2- Enter the following parameters, Fig 7-6:

a. Number of points around spectrum peak used for Gaussian curve fit.

b. SNR threshold value, used to filter out measurement points that are below a certain threshold.

c. Rang resolution, used when running in autocorrelation mode.

d. FPGA pre-processing algorithm: 1 for FFT, 2 for Autocorrelation.

e. Measurement mode: 4 for scan, 3 for vertical.

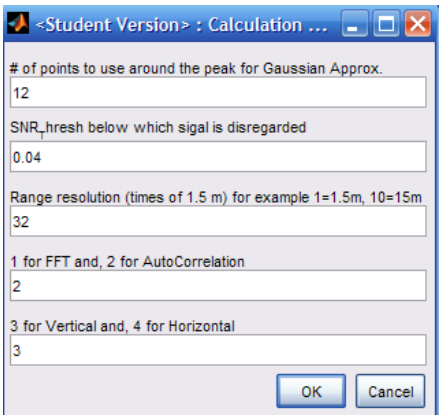

Figure 7-6 Setting the processing parameters for the Real-time measurement display

3- Select a background noise file, Fig. 7-7, that will be used to flatten the detector's gain shape.

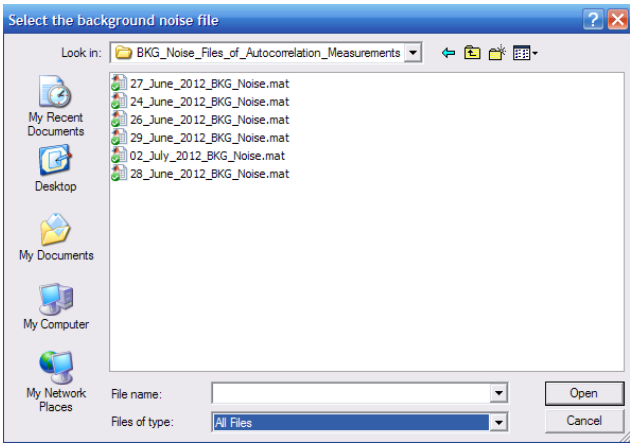

Figure 7-7 Selecting the background noise file

The MATLAB program is saved as: ***Wind_Display_Real_Time.m***, and the code is as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%  This program is developed by Sameh Abdelazim
%%%%%%  To process Power sepctrum files and generate
%%%%%%  Barb plots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

File_Path = uigetdir;

prompt2 = { ,'# of points to use around the peak for Gaussian Approx.'...
            ,'SNR_Thresh below which sigal is disregarded'...
            ,'Range resolution (times of 1.5 m) for example 1=1.5m, 10=15m'...
            ,'1 for FFT and, 2 for AutoCorrelation'...
            ,'3 for Vertical and, 4 for Horizontal'};
dlg_title1          = 'Calculation Factors';
num_lines1          = 1;
def1                = {'12','0.04','32','2','3'};
```

```
options.Resize     = 'on';
options.WindowStyle= 'normal';
options.Interpreter= 'tex';
answer1             = inputdlg(prompt2,dlg_title1,num_lines1,def1,options);

No_Fitted_Points    = str2num(answer1{1});
SNR_Thresh          = str2num(answer1{2});
Range_Res           = str2num(answer1{3});
Is_FFT_or_AC        = str2num(answer1{4});
Is_Vert_or_Horiz    = str2num(answer1{5});
No_Lags             = 12 ;
Theta               = 30*pi/180;
Phi                 = 17*pi/180;
Spectrum_Averages   = 60;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% This section selects a background noise file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cd (File_Path);
[BKG_Noise_FileName,PathName] = uigetfile('*.txt','Select the background noise file');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% This section assigns some arrays
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Signal_Intensity_Peak    = NaN(64,10e3);
Signal_Intensity_Area    = NaN(64,10e3);
Range_Corrected          = NaN(64,10e3);
Precision                = NaN(64,10e3);
Vert_Peak_Vel            = NaN(64, 10e3);
Vert_MLE_Vel             = NaN(64, 10e3);

MV1_Matlab_MLE            = zeros(64,1);
MV2_Matlab_MLE            = zeros(64,1);
MV3_Matlab_MLE            = zeros(64,1);

E_Array_Matlab_MLE       = zeros(64, 60);
N_Array_Matlab_MLE       = zeros(64, 60);
Z_Array_Matlab_MLE       = zeros(64, 60);
Barb_Array_Index         = 1;
barb_count               = 0;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% read background file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  Noise_File_WPath = strcat(PathName,BKG_Noise_FileName);
  Noise_File = fopen(Noise_File_WPath,'r');
  BKGround_Noise = fscanf(Noise_File,'%f',64);
  fclose('all');
 File_Time_Stamp_in_Seconds = 0;
 Current_Time_Stamp_in_Seconds=0;
 First_XTick_flag  = 0;

if (Is_FFT_or_AC == 1)
    file_name = 'my_30_Seconds_file.txt';
elseif (Is_FFT_or_AC == 2)
    file_name = 'AUTOCOR_30_Seconds_file.txt';
else
        printf('Wrong value was entered for recongnizing data type (FFT/Autocorrelation)');
        break
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% This section remaps non-vertical measured signal intensity to its
%%%% corresponding vertical height. The output of this code is a vector
%%%% (Map_Vect) that holds indices of tilted SNR to reallocate them into a
%%%% vertical vector.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
Vert_Height = [96:48:48*64] ;
```

```
Tilted_Height = cos(Theta )* Vert_Height;
Map_Vect = zeros(1,length(Vert_Height)) ;

for i = 1 : length(Vert_Height) ;

    Height_Difference = abs(Vert_Height(i) - Tilted_Height(1));
    Map_Vect(i) = i ;

            for j = 1 : length(Tilted_Height);
                if ( Height_Difference > abs( Vert_Height(i) - Tilted_Height(j)) )
                    Height_Difference =  abs( Vert_Height(i) - Tilted_Height(j));
                    Map_Vect(i) = j;
                end
            end

    if i > 1
        if Map_Vect(i) == Map_Vect(i-1)
            Tilted_Length = i -1 ;
            break
        end
    end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Mirror_Position = 0;
Directory_Name = datestr(date,30);
Output_Directory_Name = strcat(Directory_Name(1:8),'_Output_Files');
%dir_index = [dir_data.isdir];% Check to see if entry is a directory
Does_Output_Dir_Exist = dir (Output_Directory_Name);
if isempty(Does_Output_Dir_Exist)
    mkdir (Output_Directory_Name);
end

for half_a_minute = 1:7200

    if (Is_Vert_or_Horiz == 4)
            Mirror_Position = mod(Mirror_Position,3) + 1;
            % Tell the user to Move the mirror to right position
            gf = warndlg({'Move the mirror to position: ', num2str(Mirror_Position),' ',...
                        'then click OK';...
                        },'','normal');
            waitfor(gf); % Force execution to halt until user presses OK Button

                % Tell the user to click on start stream
            gf = warndlg({'click on Start Stream ' ,'',...
                        'then click OK';...
                        },'','normal');
            waitfor(gf); % Force execution to halt until user presses OK Button

    else
    end


Dir_Data  = dir (file_name);  % Get complete contents of selected dir

wait = 1;

while ( wait ~= 0)
        Dir_Data  = dir (file_name);  % Get complete contents of selected dir
        %dir_index = [dir_data.isdir];% Check to see if entry is a directory
        if ~isempty(Dir_Data)
            [F_year F_month F_day F_hour F_min F_sec] = datevec(Dir_Data.date); % Make list of
file names
            File_Time_Stamp_in_Seconds = F_sec + F_min*60 + F_hour*3600 + F_day *86400  ;
            [year month day hour min sec]=datevec(now);
            Current_Time_Stamp_in_Seconds = sec + min*60 + hour*3600 + day *86400  ;
        else
            File_Time_Stamp_in_Seconds = Current_Time_Stamp_in_Seconds;
        end
```

```
    if ( Current_Time_Stamp_in_Seconds > File_Time_Stamp_in_Seconds +15  &&
Current_Time_Stamp_in_Seconds < File_Time_Stamp_in_Seconds +25)
        if (Is_FFT_or_AC == 1 && Is_Vert_or_Horiz == 3) % FFT and vertical
            dest_file = strcat('FFT_Vert_',datestr(Dir_Data.date,30),'.txt')
        elseif (Is_FFT_or_AC == 1 && Is_Vert_or_Horiz == 4)% FFT and scanning
            dest_file =
strcat('FFT_Scanning_',num2str(Mirror_Position),'_',datestr(Dir_Data.date,30),'.txt')
        elseif (Is_FFT_or_AC == 2 && Is_Vert_or_Horiz == 3) % AutoCorrelation and vertical
            dest_file = strcat('AutoCorrelation_Vert_',datestr(Dir_Data.date,30),'.txt')
        elseif (Is_FFT_or_AC == 2 && Is_Vert_or_Horiz == 4) % AutoCorrelation and scanning
            dest_file =
strcat('AutoCorrelation_Scanning_',num2str(Mirror_Position),'_',datestr(Dir_Data.date,30),'.txt')
        end
            dest_file_str = strcat(File_Path,'/',Output_Directory_Name,'/',dest_file);
            movefile(file_name,dest_file_str);
            wait = 0;
    else
         pause(3);
    end

end
%%


    % The following function is to estimate velocity out of power spectrum
    % using Peak search method and a matlab MLE function out of a power
    % spectrum file
    % It takes a file name, number of points to fit using a matlab MLE
    % technique  number of spectra accumulations to
    % be averaged, back ground noise, SNR threshold value, then returns two
    % wind velocity vectors
    if (Is_FFT_or_AC == 2)
                [Peak_Power  Vel_Vect_Peak Vel_Vect_MLE Area_Power] =
Auto_Correlation2_Peak_Velocity(No_Fitted_Points...
                                                                    ,Spectrum_Averages...
                                                                    ,SNR_Thresh...
                                                                    ,No_Lags...
                                                                    ,dest_file_str);
    else
                [Peak_Power  Vel_Vect_Peak Vel_Vect_MLE Area_Power] =
FFT_2_Peak_Velocity(No_Fitted_Points...
                                                                    ,Spectrum_Averages...
                                                                    ,SNR_Thresh...
                                                                    ,BKGround_Noise...
                                                                    ,dest_file_str);
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%% To make sure that we have assigned three measured beams before
    %%%%% calculating a 3D vector we use the following if statement.
    if (Is_Vert_or_Horiz == 3)    % Vertical wind measurement
        Signal_Intensity_Peak(:,half_a_minute)      = Peak_Power;
        Signal_Intensity_Area(:,half_a_minute)      = Area_Power ;
        Vert_Peak_Vel(:,half_a_minute)              = Vel_Vect_Peak;
        Vert_MLE_Vel(:,half_a_minute)               = Vel_Vect_MLE;
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    elseif( Mirror_Position == 1  ) % means it's in a scanning mode at position 1
            MV1_Matlab_MLE                                  = Vel_Vect_MLE ;
            Signal_Intensity_Peak(:,half_a_minute)          = Peak_Power ;
            Signal_Intensity_Area(:,half_a_minute)          = Area_Power ;
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   elseif ( Mirror_Position == 2)  % means it's in a scanning mode at position 2
            for index = 1: 63 ;
                MV2_Matlab_MLE(index)                       = Vel_Vect_MLE(Map_Vect(index));
                Signal_Intensity_Peak(index,half_a_minute)  = Peak_Power(Map_Vect(index));
                Signal_Intensity_Area(index,half_a_minute)  = Area_Power(Map_Vect(index)) ;
                if ( Map_Vect(index)==Map_Vect(index+1))
                    break
                else
```

```
                end
            end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    elseif ( Mirror_Position == 3  ) % means it's in a scanning mode at position 3
            for index = 1: 63 ;
                MV3_Matlab_MLE(index)                       = Vel_Vect_MLE(Map_Vect(index));
                Signal_Intensity_Peak(index,half_a_minute)  = Peak_Power(Map_Vect(index));
                Signal_Intensity_Area(index,half_a_minute)  = Area_Power(Map_Vect(index)) ;
                if ( Map_Vect(index)==Map_Vect(index+1) )
                    break
                else
                end
            end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    % Any calculated velocity that is greater than 30 will be filtered and
    %%% set to zero

%     for j = 1:64;
%         if (Peak_Vect(j) > 30 || Peak_Vect(j) < -30)
%             Peak_Vect(j) = 0;
%         end
%         if (Matlab_MLE_Vect(j) > 30 || Matlab_MLE_Vect(j) < -30)
%             Matlab_MLE_Vect(j) = 0;
%         end
%         if (New_MLE_Vect(j) > 30 || New_MLE_Vect(j) < -30)
%             New_MLE_Vect(j) = 0;
%         end
%     end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Date_Str = strcat(answer1{1}(5:6), '/', answer1{1}(7:8), '/' ,answer1{1}(1:4));


ylabel_str       = 'Height (m)';
xlabel       = 'U.S. Eastern time';
Date_String = datestr(Dir_Data.date);
Date_Str     = Date_String(1:11);
Distance_Per_Gate = 48;
No_Gates = 64;
range = [96 : Distance_Per_Gate: Distance_Per_Gate*(No_Gates-4)];

File_Time_Stamp = datestr(Dir_Data.date,30);
Min_Stamp  = File_Time_Stamp(12:13);
Hour_Stamp = File_Time_Stamp(10:11);

if (First_XTick_flag == 0)
    Xticks_LABEL            = strcat(Hour_Stamp,':',Min_Stamp);
    XTicks(1)               = 1 ;
    First_XTick_flag        = 1;
    ticks_counter           = 1;
    Barb_XTicks(1)          = 1 ;
end


if ( (F_min == 0 || F_min == 30) && (F_sec > 29) )
    ticks_counter               = ticks_counter + 1;
    Tick_Str                    = strcat(Hour_Stamp,':',Min_Stamp);
    Xticks_LABEL                = [Xticks_LABEL; Tick_Str];
    XTicks(ticks_counter)       = half_a_minute;
    Barb_XTicks(ticks_counter)  = floor(half_a_minute/20)+1;
else
end



YTicks = 1:10:59;
Yticks_LABEL = range(1:10:59);
```

```
time = [1: Barb_Array_Index];
height = [96: 48: 64*48];
[latgrat,longrat] = meshgrid(height,time);
Height_Grid= latgrat';
Time_Grid= longrat';

time_inst = [1: 1];
height_inst = [96: 48: 64*48];
[latgrat_inst,longrat_inst] = meshgrid(height_inst,time_inst);
Height_Grid_inst= latgrat_inst';
Time_Grid_inst= longrat_inst';

ylabel = 'Height (m)';
xlabel = 'U.S. EST';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




 if (half_a_minute >= 3 && Is_Vert_or_Horiz == 4)  % now we can calculate a 3D wind vector
        [Vx_Matlab_MLE    Vy_Matlab_MLE ] = Convert2_3D_V2(MV1_Matlab_MLE, MV2_Matlab_MLE,
MV3_Matlab_MLE, Theta);

        %%%% calculate the east and north wind direction for Matlab MLE
        %%%% values
        VEast_Matlab_MLE  = Vx_Matlab_MLE.*cos(Phi) - Vy_Matlab_MLE.*sin(Phi);
        VNorth_Matlab_MLE = Vx_Matlab_MLE.*sin(Phi) - Vy_Matlab_MLE.*cos(Phi);

        %%%% Populate matrices of wind velocities in east and west
        %%%% direction calculated via MLE to plot wind barbs
        %%%%% This is what I am going to change



        if ( (F_min == 0 || (F_min == 10) || F_min == 20) || (F_min == 30) || (F_min == 40)||
(F_min == 50)&& (F_sec > 29) )
            E_Array_Matlab_MLE(:,Barb_Array_Index) = VEast_Matlab_MLE;
            N_Array_Matlab_MLE(:,Barb_Array_Index) = VNorth_Matlab_MLE;
            Z_Array_Matlab_MLE(:,Barb_Array_Index) = MV1_Matlab_MLE;
            Barb_Array_Index      = Barb_Array_Index + 1 ;
            Fig._number = 1 ;
            title_2 = 'Wind Barb measured by Matlab MLE measured at CCNY ON:' ;
            barbs_scale = 0.5;

Sameh_barbs(E_Array_Matlab_MLE(1:length(height),1:Barb_Array_Index),N_Array_Matlab_MLE(1:length(h
eight),1:Barb_Array_Index), Height_Grid,Time_Grid,...
                        Fig._number, barbs_scale, XTicks,YTicks,Xticks_LABEL,Yticks_LABEL,...
                                  xlabel ,ylabel,title_2,Date_Str )

        end
else
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% %%%%%%%%%%%%%%%%%%%%%%%%%%

%         Fig.(7)
%         title_7a = 'Vertical wind velocity measured by Matlab MLE measured at CCNY ON: ' ;
        first_gate = 1;
        last_gate  = 64;
%         subplot(2,1,1),Sameh_Imagesc(Z_Array_Matlab_MLE(:,1:half_a_minute), first_gate,
last_gate, XTicks,YTicks,...
%                                 Xticks_LABEL,Yticks_LABEL,xlabel,...
%                                 ylabel,title_7a,Date_Str,-2,3 )

        Fig.(7)
        title_7b = 'Vertical wind velocity measured by Matlab MLE measured at CCNY ON: ' ;
        Sameh_Imagesc(Vert_MLE_Vel(:,1:half_a_minute),first_gate, last_gate,XTicks,YTicks,...
```

```
                                Xticks_LABEL,Yticks_LABEL,xlabel,...
                                ylabel,title_7b,Date_Str,-4,4 )

        Fig.(8)
        title_8a = 'Signal intensity peak measured at CCNY ON: ';
        Sameh_Imagesc(Signal_Intensity_Peak(:,1:half_a_minute),first_gate,
last_gate,XTicks,YTicks,...
                                Xticks_LABEL,Yticks_LABEL,xlabel,...
                                ylabel,title_8a,Date_Str,0,0.7 )

        Fig.(9)
        title_8b = 'Vertical wind velocity estimated by peak search measured at CCNY ON: ' ;
        Sameh_Imagesc(Vert_Peak_Vel(:,1:half_a_minute),first_gate, last_gate,XTicks,YTicks,...
                                Xticks_LABEL,Yticks_LABEL,xlabel,...
                                ylabel,title_8b,Date_Str,-4,4 )

    if (half_a_minute >= 3 && Is_Vert_or_Horiz == 4)
            barb_count = barb_count +1;
            Fig._number = 10 ;
            title_2 = 'Wind Barb measured by Matlab MLE measured at CCNY ON:' ;
            barbs_scale = 0.5;

Sameh_barbs(E_Array_Matlab_MLE(1:length(height_inst),barb_count),N_Array_Matlab_MLE(1:length(heig
ht_inst),barb_count), Height_Grid_inst,Time_Grid_inst,...
                        Fig._number, barbs_scale, XTicks,YTicks,Xticks_LABEL,Yticks_LABEL,...
                                xlabel ,ylabel,title_2,Date_Str )
    else
    end
end
```

7.2.8 Post processing atmospheric measurements obtained using FFT and Autocorrelation

This MATLAB program was developed to post process wind measurements files obtained in either vertical or scan mode with FFT or autocorrelation FPGA pre-processing algorithm. The code and usage of this program is very similar to previous program (***Wind_Display_Real_Time.m***), therefore, it will not be listed here.

The program is saved as: ***plotwindbarb_V7.m***

7.3 Appendix III Simulink logic designs

Developing the logic design for pre-processing backscattered signals was a challenging task due to the large number of logic gates required to implement required pre-processing functionality. The main real estate consuming component was the memory required for accumulating the autocorrelation vector. Binary system arithmetic adds another complication to the design process, as data width increases after addition and doubles after a multiplication process. Data width increase causes increase of logic gates and memory blocks. Therefore, a careful attention had to be paid to manage data widths throughout the design.

Backscattered 16-bit signals pass through different logic circuits for different calculations. Each time a sample goes through an addition or multiplication process, its width will change. A previous knowledge of the data range is important not to reserve un-needed bits width. For example, if the 16-bit samples goe through sine and cosine multiplication to generate the in-phase and quadrature signals, its width will grow to 32-bit. The 32-bit will not change by going through the Xilinx FIR Compiler 5.0 circuit. Now the 32-bit will be multiplied by itself to calculate the autocorrelation lags, which will grow the result to 64-bit. Finally accumulating this result for 10k times, means an increase of the width by at least 14 bits, which means the final result needs a 78-bit representation. This large width of signal representation is very expensive in terms of FPGA circuits' placement and wiring. For a design with only 6 lags, it may not be possible to be implemented. Therefore, data widths had to be reduced.

Observing the output of the optical detector while output gain is set to 10x and input is fed by both local oscillator and received signals shows that it is sampled to a range of +/-70 counts on the ADC, i.e. up to 7-bits + a sign bit. The cosine and sine are set such that each has only 3-bits + a sign bit, Fig. 7-8. Therefore, the input samples are truncated at the 8^{th} bit. Now

the result of multiplying the input samples by the sine and cosine results a 10-bit + sign output. This 10-bit + sign goes through the FIR filter and the output is represented in 50 bits, however, the most significant bit (MSB) does not pass the 10^{th} bit. The least significant bit (LSB) is then truncated, which reduces the MSB to the 9^{th} bit. The filtered output is then multiplied by itself to produce the lags, which will make the width equal to 18-bit + a sign bit. Accumulating this 18-bit number 10k times can increase its width by about 14 bits, i.e. the final result can be expressed in 32-bit.

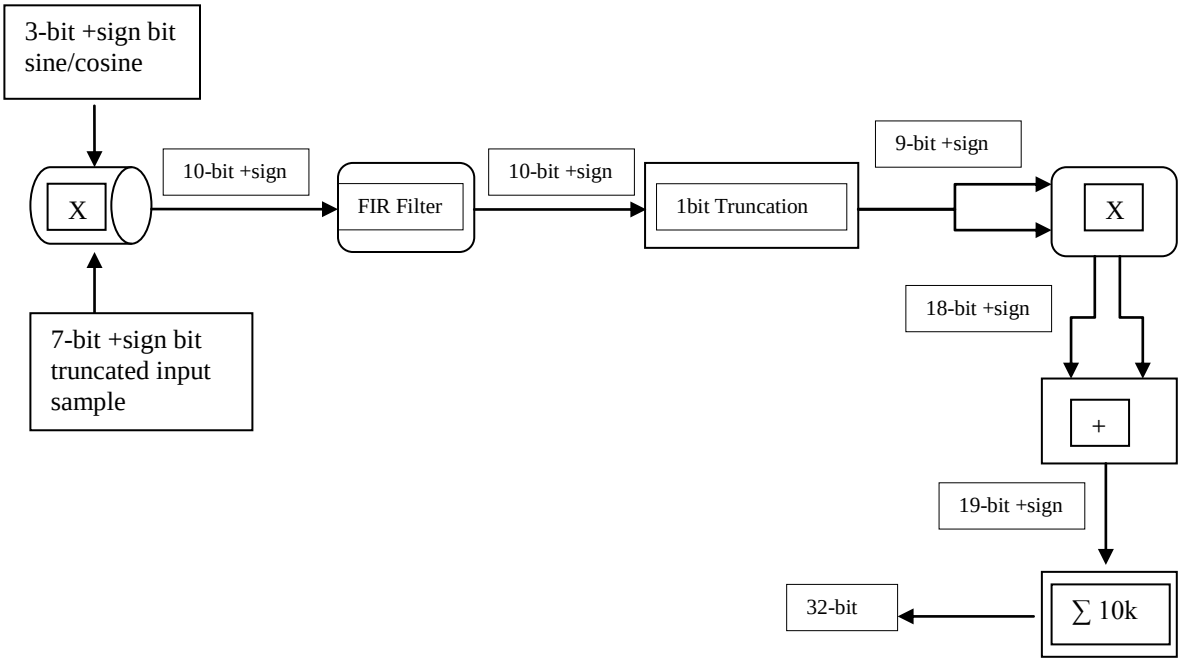

Figure 7-8 Binary data width management in FPGA programming

7.4 Appendix IV Vector Analysis to Construct Wind Vector From Three Line of Sight Measurements

In this section, the vector analysis to construct a 3D wind vector from the three line of sight measurements is presented taking into consideration the configuration set up in our research van.

The normal vector to the lens can be represented as:

$$\hat{n} = \hat{x}/\sqrt{2} + \hat{z}/\sqrt{2}$$

After a rotation of θ (ccw), the normal is

$$\hat{n} = \hat{x}Cos(\theta)/\sqrt{2} + \hat{y}Sin(\theta)/\sqrt{2} + \hat{z}/\sqrt{2}$$

Now we do a coordinate change to x', y', z' with

$$\widehat{x'} = \hat{n} = \hat{x}Cos(\theta)/\sqrt{2} + \hat{y}Sin(\theta)/\sqrt{2} + \hat{z}/\sqrt{2}$$

The output beam, by Snell's law, will be along the unit vector

$$\hat{o} = \widehat{x'}Cos(\theta)/\sqrt{2} + \widehat{y'}Sin(\theta)/\sqrt{2} + \widehat{z'}/\sqrt{2}$$

We need therefore to find the two unit vectors $\widehat{y'}, \widehat{z'}$ to form the right handed coordinate system with $\widehat{x'}$

$$\widehat{z'} = -\hat{x}Cos(\theta)/\sqrt{2} - \hat{y}Sin(\theta)/\sqrt{2} + \hat{z}/\sqrt{2}$$

And since

$$\widehat{y'} = \widehat{z'} \times \widehat{x'} = -\hat{z}Cos(\theta)Sin(\theta)/2 + \hat{y}Cos(\theta)/2 + \hat{z}Cos(\theta)Sin(\theta)/2 - \hat{x}Sin(\theta)/2 + \hat{y}Cos(\theta)/2 - \hat{x}Sin(\theta)/2$$

$$\widehat{y'} = \hat{y}Cos(\theta) - \hat{x}Sin(\theta)$$

So going back to the xyz coordinates in the tuck frame, the output beam will then be along the unit vector:

$$\hat{o} = \hat{x}Cos^2(\theta)/2 + \hat{y}Cos(\theta)Sin(\theta)/2 + \hat{z}Cos(\theta)/2$$

$$-\hat{x}Sin^2(\theta)/\sqrt{2} + \hat{y}Cos(\theta)Sin(\theta)/\sqrt{2} + \hat{z}0$$

$$-\hat{x}Cos(\theta)/2 - \hat{y}Sin(\theta)/2 + \hat{z}/2$$

$$\hat{o} = \hat{x}(Cos^2(\theta)/2 - Cos(\theta)/2 - Sin^2(\theta)/\sqrt{2})$$

$$+\hat{y}(Cos(\theta)Sin(\theta)\left(\frac{1}{2} + \frac{1}{\sqrt{2}}\right) - Sin(\theta)/2)$$

$$+\hat{z}(\frac{1}{2} + Cos(\theta)/2)$$

After a rotation of - θ (cw), the second output beam will be in the direction of

$$\hat{q} = \hat{x}(Cos^2(\theta)/2 - Cos(\theta))/2 - Sin^2(\theta)/\sqrt{2})$$

$$-\hat{y}(Cos(\theta)Sin(\theta)\left(\frac{1}{2} + \frac{1}{\sqrt{2}}\right) - Sin(\theta)/2)$$

$$+\hat{z}(\frac{1}{2} + Cos(\theta)/2)$$

For an arbitrary wind vector in the frame of the truck

$$\vec{V} = A\hat{x} + B\hat{y} + C\hat{z}$$

We make measurements along directions $\hat{z}, \hat{o}, and\ \hat{q}$

$$M1 = \vec{V}.\hat{z}$$

$$M2 = \vec{V}.\hat{o}$$

$$M3 = \vec{V}.\hat{q}$$

So that

$$M1 = C$$

$$M2 = A(Cos^2(\theta)/2 - Cos(\theta)/2 - Sin^2(\theta)/\sqrt{2})$$

$$+B(Cos(\theta)Sin(\theta)\left(\frac{1}{2} + \frac{1}{\sqrt{2}}\right) - Sin(\theta)/2)$$

$$+C(\frac{1}{2} + Cos(\theta)/2)$$

$$M3 = A(Cos^2(\theta)/2 - Cos(\theta)/2 - Sin^2(\theta)/\sqrt{2})$$

$$-B(Cos(\theta)Sin(\theta)\left(\frac{1}{2} + \frac{1}{\sqrt{2}}\right) - Sin(\theta)/2)$$

$$C(\frac{1}{2} + Cos(\theta)/2)$$

The solution is then

$$C = M1$$

$$M2 - M3 = 2B(Cos(\theta)Sin(\theta)\left(\frac{1}{2} + \frac{1}{\sqrt{2}}\right) - Sin(\theta)/2)$$

$$B = (M2 - M3)/(Cos(\theta)Sin(\theta)\left(1 + \sqrt{2}\right) - Sin(\theta))$$

$$M2 + M3 = 2A(Cos^2(\theta)/2 - Cos(\theta)/2 - Sin^2(\theta)/\sqrt{2}) +$$

$$2\,C(\frac{1}{2} + Cos(\theta)/2)$$

$$M2 + M3 = A(Cos^2(\theta) - Cos(\theta) - \sqrt{2}Sin^2(\theta)) + M1(1 + Cos(\theta))$$

$$A = \left(M2 + M3 - M1\left(1 + Cos(\theta)\right)\right) \Big/ (Cos^2(\theta) - Cos(\theta) - \sqrt{2}Sin^2(\theta))$$

$$B = (M2 - M3)/(Cos(\theta)Sin(\theta)\left(1 + \sqrt{2}\right) - Sin(\theta))$$

$$C = M1$$

Now that we have the three components of the wind Vector in the Truck frame, we need a final rotation to the Earth frame so that the x axis of the truck frame rotates into the East Direction and the Y axis is then the North Direction; z direction is the zenith or updraft component.

The angle of rotation φ is the clockwise rotation required to move the x axis in the truck to the frame of the earth.

$V_z = C$

$V_{east} = ACos(\varphi) - BSin(\varphi)$

$V_{north} = ASin(\varphi) + BCos(\varphi)$

No we have the wind vector in the frame of the Earth.

8 Bibliography

[1] A. V. J. a. J. A. L. T. R. M. Huffaker, "Laser-Doppler system for detection of aircraft trailing vortices," *IEEE,* vol. 58, p. 322–326, 1970.

[2] T. A. K. A. Y. H. a. S. W. S. Kameyama, "Compact all-fiber pulsed coherent Doppler lidar system for wind sensing," *Applied Optics,* vol. 46, pp. 1953-1962, 2007.

[3] F. Å. A. O. D. L. a. M. H. Christer J. Karlsson, "ll-Fiber Multifunction Continuous-Wave Coherent Laser Radar at 1.55 vm for Range, Speed, Vibration, and Wind Measurements," *Appl. Opt.,* vol. 39, pp. 3716-3726, 2000.

[4] P. M. S. C. P. H. S. M. H. J. R. M. D. L. B. a. E. H. Y. S. W. Henderson, "Coherent laser radar at 2 mm using solid state lasers," *IEEE,* vol. 31, no. Trans. Geosci. Remote Sens, p. 4–15, 1993.

[5] J. D. K. a. T. J. W. T. J. Kane, "Flight test of 2-mm diode pumped laser radar system," *in Air Traffic Control Technologies, R. G. Otto and J. Lenz, eds.SPIE,* vol. 2464, p. 103–108 , 1995.

[6] S. W. H. J. R. M. C. P. H. a. R. M. H. M. J. Kavaya, "Remote wind profiling with a solid-state Nd:YAG coherent lidar system," *Opt. Lett.,* vol. 14, p. 776–778 , 1989.

[7] B. A. M. V. Jean-Pierre C., "Laser Source requirements for coherent lidars based on fiber technology," *C. R. Physique,* vol. 7, p. 213–223, 2006.

[8] J. F. H. a. B. J. Rask, "Optimum optical local-oscillator power levels for coherent detection

with photodiodes," *Appl. Opt.,* vol. 34, pp. 927-933, 1995.

[9] D. P. U. S. a. M. K. Farzin Amzajerdian, "Optimum Integrated Heterodyne Photoreceiver for Coherent Lidar Applications," *MRS,* vol. doi:10.1557, pp. PROC-883-FF6.3, 2005.

[10] R. G. Frehlich, "Conditions for optimal performance of monostatic coherent laser radar," *Opt. Lett.,* vol. 15, pp. 643-645, 1990.

[11] T. A. K. A. Y. H. a. S. W. S. Kameyama, "Performance of discrete-Fourier-transform-based velocity estimators for a wind-sensing coherent Doppler lidar system in the Kolmogorov turbulence regime," *IEEE Trans. Geosci.Remote Sens.,* no. 47, p. 3560–3569 , 2009.

[12] T. M. A. D. P. B. J. K. H. J. N. Darcie, "Noise Reduction in Class-AB Microwave-Photonic Links," *Microwave Photonics,* vol. 12, no. 14, pp. 329- 332, Oct. 2005.

[13] R. G. F. a. M. J. Kavaya, "Coherent laser radar performance for general atmospheric refractive turbulence," *Appl. Opt.,* vol. 30, pp. 5325-5352, 1991.

[14] S. F. C. a. S. Wandzura, "Monostatic heterodyne lidar performance: the effect of the turbulent atmosphere," *Appl. Opt.,* vol. 20, pp. 514-516, 1981.

[15] S. M. Wandzura, "Meaning of quadratic structure functions," *J. Opt. Soc. Am.,* vol. 70, pp. 745-747, 1980.

[16] R. T. M. D. A. H. U. P. O. a. P. H. F. M. J. Kavaya, "Target reflectance measurements for calibration of lidar atmospheric backscatter," *Appl. Opt.,* vol. 22, p. 2619–2628, 1983.

[17] M. J. P. a. R. M. H. Y. Z. Zhao, "Receiving efficiency of monostatic pulsed coherent lidars. 2: Applications," *Applied Optics,* vol. 29, pp. 4120-4132, 1990.

[18] D. L. Fried, "Optical heterodyne detection of an atmospherically distorted signal wave front," *IEEE,* vol. 55, pp. 57-66, 1967.

[19] R. M. Hardesty, "Performance of a spectral peak frequency estimator for Doppler wind velocity measurement," *IEEE Trans. Geosci. Remote Sens.,* Vols. GRS-24, no. 5, p. 777–783, Sep. 1986.

[20] F. Amzajerdian, "Analysis of Optimum Heterodyne Receivers for Coherent Lidar Applications," in *21 st International Laser Radar Conference ILRC*, Quebec Canada, July 2002.

[21] K. O. S. C. W. a. P. H. F. J. M. Vaughan, "Coherent laser radar in Europe," *IEEE,* vol. 84, p. 205–226, 1996.

[22] R. M. H. a. R. M. Hardesty, "Remote sensing of atmospheric wind velocities using solid-state and CO2 coherent laser systems," *IEEE,* vol. 84, p. 181–204, 1996.

[23] J. D. A. F. P. G. B. A. O. Zarader, "Adaptive parametric algorithms for processing coherent Doppler-lidar signal," *Geoscience and Remote Sensing, IEEE Transactions,* vol. 36, no. 6, pp. 2678-2691, Nov 1999.

[24] R. M. Hardesty, "Performance of a Discrete Spectral Peak Frequency Estimator for Doppler

Wind Velocity Measurements," *IEEE Trans. Geosci. Remote Sens.,* Vols. GE-24, no. 6, Sep. 1986.

[25] R. J. D. a. D. S. Zrnic, Doppler Radar and Weather Observations, Orlando, FL: Academic, 1984.

[26] D. S. a. B. Bumgarner, "Numerical comparison of five mean frequency estimators," *J. Appi. Meteorol.,* vol. 14, pp. 991-1003, 1975.

[27] B. J. R. a. R. M. Hardesty, "Discrete spectral peak estimation in Doppler lidar. I: Incoherent accumulation and the Cramer-Rao lower bound," *Geoscience and Remote Sensing, IEEE Transactions,* vol. 31, no. 1, pp. 16-27, Jan 1993.

[28] B. J. R. a. R. M. Hardesty, "Discrete spectral peak estimation in incoherent backscatter heterodyne lidar. II: Correlogram accumulation," *IEEE Trans. Geosci. Remote. Sens.,* vol. 31, no. 1, p. 28–35, Jan. 1993.

[29] R. G. F. a. M. J. Yadlowsky, "Performance of mean-frequency estimators for Doppler radar and lidar," *J. Atmos. Ocean. Technol.,* vol. 11, no. 5, p. 1217–1230, Oct. 1994.

[30] R. Frehlich, "Simulation of coherent Doppler lidar performance in the weak-signal regime," *J. Atmos. Ocean. Technol.,* vol. 13, no. 3, p. 646–658, Jun. 1996.

[31] C. J. R. M. B. J. L. G. J. N. H. M. J. P. R. A. R. A. M. W. Grund, "High-resolution pertur lidar for boundary layer and cloud research," *J. Atmos. Oceanic Technol.,* vol. 18, p. 376–393, 2001.

[32] I. Smalikho, "Techniques of wind vector estimation from data measured with a scanning coherent pertur lidar," *J. Atmos. Oceanic Technol.,* vol. 20, p. 276–291, 2003.

[33] S. Abdelazim, "Coherent Doppler Lidar for wind sensing," in *CLRC*, Long Beach, CA, 2010.

[34] J. D. K. a. T. J. W. T. J. Kane, "Flight test of 2-mm diode pumped laser radar system," in *SPIE*, 1995.