AN UHF FREQUENCY-MODULATED CONTINUOUS WAVE WIND PROFILER - RECEIVER AND AUDIO MODULE DEVELOPMENT

A Thesis Presented

by

DAVID GARRIDO LOPEZ

Submitted to Universitat Politecnica de Catalunya

Escola Tecnica Superior d'Enginyeria de Telecomunicacio de Barcelona

in fulfillment of the requirements for the degree of

MASTER EN ENGINYERIA DE ELECTRONICA

December 2009

Electrical and Computer Engineering

AN UHF FREQUENCY-MODULATED CONTINUOUS WAVE WIND PROFILER - RECEIVER AND AUDIO MODULE DEVELOPMENT

A Thesis Presented

by

DAVID GARRIDO LOPEZ

Approved as to style and content by:

Stephen J. Frasier, Professor
Electrical and Computer Engineering

To my sister. To the one I adore and I tell my most inner thoughts.
I thank you for always being there when I needed.

ACKNOWLEDGMENTS

First of all I would like to thank Dr. Stephen Frasier for giving me the chance of being here at MIRSL, and for all the help he provided me. Being here at MIRSL it is been an excellent experience, that not only helped me to improve my knowledge in some areas such as remote sensing, antennas, signal processing or electronics. It also made me grew up as an engineer, and experience another ways of working and researching that I am sure will be useful in my future career.

Also I have to thank the Latino corner, Jorge Salazar, Jorge Trabal and Rafael Medina, the support and help they gave me. They were always willing to answer my questions, and that was always very helpful. Besides, Ivona Kostadinova deserves a special consideration, because she introduced me to the radar and did the previous work with it, which helped me get to the point where I am now.

To end, I want to thank my family their support in the distance, even with a ocean in between I did not feel alone.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The measurement of winds and processes taking place in the atmosphere is a fundamental requirement in both research and operational meteorology. This project is focused on the processes taking place in the lower troposphere called the atmospheric boundary layer (ABL). The ABL is important meteorologically in terms of assessing of convective instability. The entrainment zone at the top of the ABL acts as a lid on rising (and cooling) air parcels due to the temperature inversion. An external mechanism such as geographically forced uplift, vigorous surface heating or drylines, can break the entrainment layer, allowing the capped air parcels to rise freely. As a result, vigorous convection will begin producing severe thunderstorms.

ABL research and studies help (i) develop and improve existing numerical weather prediction models, (ii) understand the transfer of hear, water vapor and momentum between the Earth and the atmosphere, (iii) refine the analytical description of turbulent processes and (iv) quantify the absorption and emission in the troposphere, which is a major factor in shaping climate on Earth. The effect of the troposphere on wave propagation has also been studied extensively for the purposes of improving radio communications.

The main reason for the radar development is the need of continous monitoring of the winds and fields in the atmosphere, improving the in-situ measurements. Conventional radar profiler technologies usually are able to make atmospheric measurements of the boundary layer, but precluding the lower part of the ABL, (around 150 meters). The Frequency Modulated Continuous Wave, Spaced Antenna (FMCW-SA) Radar, that is being developed in University of Massachusetts - Amherst, at the Microwave Sensing Laboratory (MIRSL), will allow measurements of the lower part of the ABL.

The use of FMCW radars is introduced in order to improved the limitations of pulsed radars. Pulsed radars are limited by the pulse-width and switching speed of the transmit-receive switches because of the use of a common antennas for both functions. The pulsed nature of the radar dictates a high transmitter power, and consequently the need for switches that are both faster and high powered. FMCW radar alleviate this problem by using separate antennas for transmit and receive, also being able to be used at short ranges. The problem in dual-antenna systems is parallax at low altitudes due to the spatially separated antenna apertures, and some uncertainty in the actual sampling volume.

The primary objective of this thesis, is to explain the previous results and the problems encountered on the FMCW Wind Profiler Radar, and to provide a detailed account of the work one in order to fix remaining issues. Several problems were encountered on the radar's receiver. Noise and leakage were still not allowing the radar to achieve sensitivity enough to work properly. To solve it, the receiver was modified. This thesis, provides a detailed account of modifications including a new audio-module, modified FPGA and tests to report conclusions.

1.2 Summary of Chapters

Chapter 2 gives a introduction to the atmospheric boundary layer, explains its structure and characteristics followed by clear-air backscatter theory. After that the principles of FMCW radars are explained. An overview of the FMCW Wind Profiler is presented, and the initial configuration and previous results of the radar are explained and commented.

Chapter 3 explains the design of the receiver, explaining all the changes to the radar's receiver design. After that, an introduction to active filter theory is presented, and the design of the audio module is depicted using the theory presented before. Through all the design stages, simulations, relationships and explanations are provided .

In chapter 4, the audio module is tested, and the results are presented. A explanation of the DDS configuration process is also given, and then a modification of the FPGA is detailed. Finally, chapter 5 contains a brief summary of the research work presented here, and conclusions and recommendations for future work are drawn.

CHAPTER 2

FMCW WIND PROFILER PRINCIPLES AND OVERVIEW

2.1 FMCW Radar Principles

2.1.1 Atmospheric Boundary Layer

The boundary layer is the lowest 1-2 km of the atmosphere, the region most directly influenced by the exchange of momentum, heat, and water vapor at the earth's surface. Turbulent motions on time scales of an hour or less dominate the flow in this region, transporting atmospheric properties both horizontally and vertically through its depth. The mean properties of the flow in this layer, the wind speed, temperature, and humidity experience their sharpest gradients in the first 50-100 m, appropriately called the surface layer. Turbulent exchange in this shallow layer controls the exchange of heat, mass, and momentum at the surface and thereby the state of the whole boundary layer. It is hardly surprising we should have a lively curiosity about this region.

There are two main types of boundary layer. The convective boundary layer where heat from the Earth's surface creates possitive buoyancy flux and instabilities that lead to turbulences, and the stably stratified nocturnal boundary layer, where negative buoyancy flux decreases the turbulences and stable stratified conditions prevail. The ABL can reach over 3 km during daytime while the usual height at night is between 50 m. to 300 m. Typical boundary layer structure is depicted in Figure[2.1].

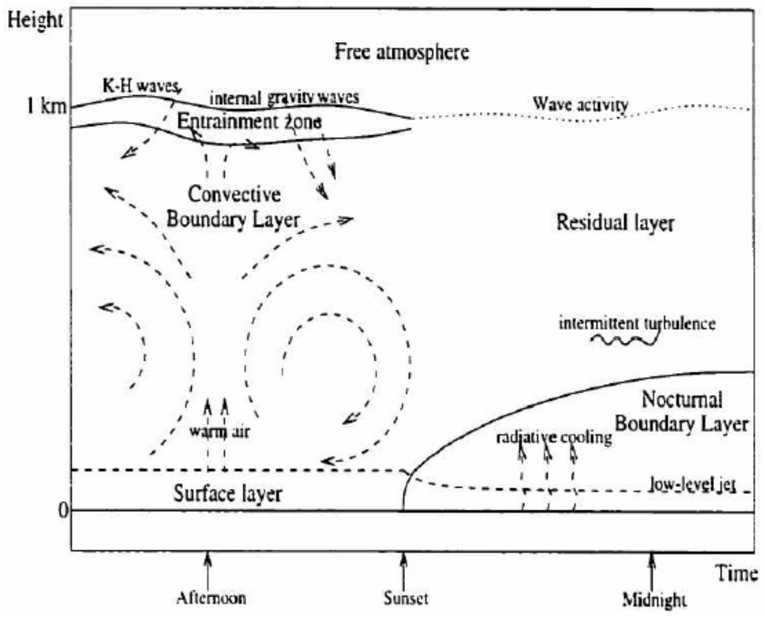


Figure 2.1: Boundary layer structure (from [5])

2.1.2 Clear Air Backscatter Theory

The backscattering from refractive index irregularities in clear air and has its relationship on the atmospheric structure and turbulence, as determined from experimental studies of angel echoes starting with Plank [1956] and Atlas [1959].

Radar backscattering from refractive index variation is able to provide helpful information about the atmospheric structure. First radar outlines regions of increased refractive index variability because of the enhanced backscattering. Second, the radar backscatter contains quantitative information about the variability in the refractive index field.

The radar backscattering from the clear air atmosphere, is caused by irregular small-scale fluctuations in the radio refractive index produced by turbulent mixing and dependent on the atmospheric water vapor, temperature, and pressure. The intensity of these fluctuations can be described by the structure constant C_n^2. In 1969 Ottersen [10] was the first to derive the relationship between the radar volume reflectivity and refractive index structure as

$$\eta(\lambda) \approx 0.38 C_n^2 \lambda^{-\frac{1}{3}} \tag{2.1}$$

For a given radar λ, the radar reflectivity η of a region of refractive index fluctuations is directly proportional to C_n^2 when the length scale of one-half of the radar wavelength falls within the inertial subrange. The more violent the turbulent mixing the larger the displacements, and the stronger the inhomogeneities will be. So, strong turbulence and sharp mean gradients contribute to high C_n^2 values.

The radar backscatter can be originated by other sources such as Rayleigh scattering for example from birds and insects the size of which should be much smaller than the radar wavelength and is proportional to λ^{-4}. In our application this kind of scattering will be considered as undesired noise.

2.1.3 FMCW Radar Basics

FMCW is common technique used on radars, that avoids the limitations of pulsed radars. It consists on the transmission of a sinusoid whose frequency changes over time, in our case linearly. Idealy the instantaneous frequency should augment indefinitely with time, but in order to have a realistic system the frequency will increase until a maximum value, and will start from the initial frequency once reached that point. So, the instantaneous frequency will have a saw tooth shape as observed in Figure[2.2].

The backscattered signal will be delayed by $t_{delay} = \frac{2R}{c}$, where c is the speed of light and R is the range to target as shown in Figure[2.3]. It will be also attenuated and possibly Doppler-shifted in case the target was not stationary. The received signal will be then mixed with a replica of the transmitted signal, a sinusoidal signal for every target will be produced. The frequency of which is called beat frequency

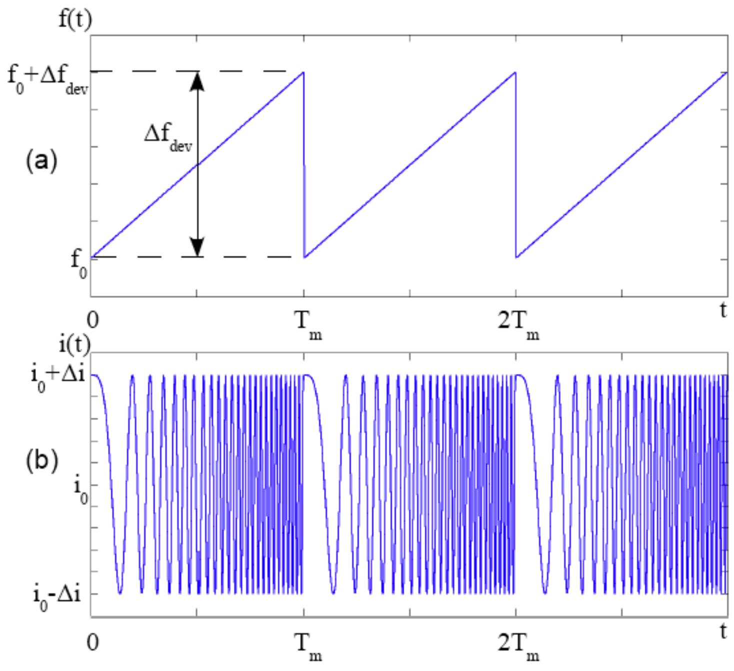


Figure 2.2: Time versus frequency diagram(a); FM signal (b).

and depends on the range to target and the radar parameters. For stationary targets, the relation is as follows

$$R = \frac{cT_p}{2B} f_b \tag{2.2}$$

where R is the range to target, c is the speed of light, B is the chirp bandwidth and T_p is the sweep time. Fast Fourier transform analysis is performed then in order to convert frequency information to range. The range resolution of the system is

$$\Delta R = \frac{c}{2B} \tag{2.3}$$

Parameters such as B and T_p are configurable in an FMCW radar thus providing flexibility to adapt to the most suitable mode, without change in the peak transmit power for a given sensitivity.

To retrieve velocities, it is possible to determine Doppler velocities comparing the changes of phase of two consecutive pulses received. The maximum unambiguous

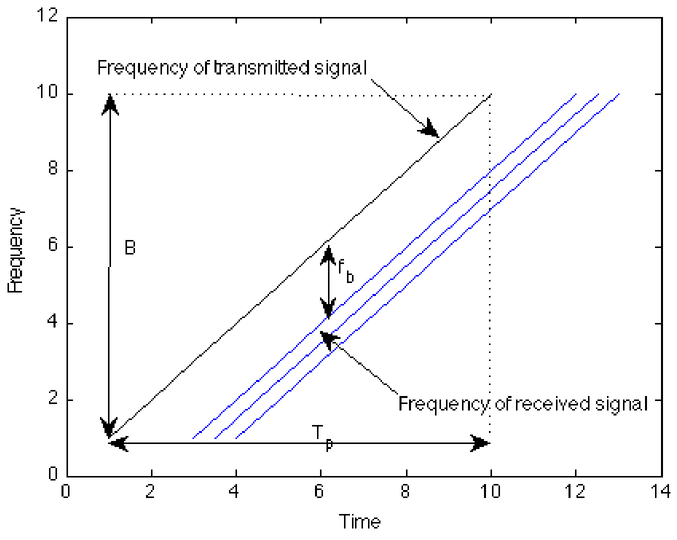


Figure 2.3: Difference on instantaneous frequencies on FMCW

velocity that can be measured is determined by the radar operating frequency and the pulse repetition frequency as

$$v_{rmax} = \frac{\lambda}{4} f_p \tag{2.4}$$

To maximize range resolution and sensitivity, both B and T_p should be as large as possible. The value of T_p, however is constrained by the coherence time of the atmospheric echo because the presented theory is based on the assumption that the target produces constant-frequency sinusoidal echo during the sweep [12]. The maximum range for FMCW radars is determined by the sweep time T_p and the sampling frequency used in the A/D conversion. The latter gives us the maximum beat frequency that can be detected without aliasing.

$$R_{max} = \frac{cT_p}{2B} F_{bmax} = \frac{cT_p}{2B} \frac{f_s}{2} \tag{2.5}$$

2.2 System Overview

The current FMCW radar project started as an analog of the S-Band FMCW boundary layer profiler developed in 2003 at University of Massachusetts at Amherst [12]. The change from S-Band to UHF was proposed in order to reduce the Rayleigh scattering from insects and birds, that appeared to dominate the observed vertical profile of the mean reflectivity at S-Band.

The current FMCW - Wind Profiler was started in the summer 2006 by the former students Albert Genis during 2008 and Iva Kostadinova on 2008-2009. Many changes have been introduced to the radar, after upgrading and overcoming the different problems that were affecting the radar through several years.

The purpose of the next sections is to explain the latter hardware configuration of the radar, and analyze the problems and changes that either were introduced or are being introduced to solve them.

2.2.1 Initial Hardware Configuration

The FMCW wind profiler operates in the 900MHz ISM frequency band (902 - 928 MHz), with a center frequency of 915MHz. The chirp is generated by Direct Digital Synthesizer (AD9858) with a bandwidth up to 25MHz and linear FM modulation. So achieving a maximum range resolution of 6m.

The default mode of operation had a PRF of 100Hz, and a duty cycle of 83.3%, that meaning that $T_p = 8.33ms$. These parameters allow to detect vertical unambiguous velocity up to ±8.25m/s, which exceeds the range of usual turbulent velocities on the boundary layer (3-5m/s).

All the signals needed to start the radar, such as start and set up the DDS, and synchronize and clock the sampling are generated by an FPGA Cyclone II from Altera, designed by Iva Kostadinova and David Garrido Lopez. The parameters of the radar

Parameter	**Value**
Transmitter	
Center frequency	915MHz
Peak Transmit power	30W
Transmitter type	Solid State RF (SSRF)
Sweep bandwidth	$\leq$25MHz
Sweep time	8.333ms
PRF	100Hz
Antennas	
Type	Four Parabolic dish
Gain	18dB
Polarization	Linear
Front to Back Ratio	22dB

Table 2.1: Initial System specifications

such as bandwidth, PRF, T_p are configurable. Allowing to adapt to the resolution and scenario you need, taking into account (2.3), (2.4) and (2.5) it is possible to calculate the optimal configuration.

The transmit amplifier is a compact solid state RF power amplifier providing a 30W output. The antennas are 4' diameter antenna parabolas, with dipole antenna feeds, 19 degree beamwidth and 18dB of gain. The antennas have a broad beam, in order to employ a spaced antenna technique for estimating horizontal winds. That broad antenna beamwidth has its problems. The main problem associated with that is the inadequate isolation between transmitter and receiver. The transmitter leakage received is powerful enough to saturate either the front-end of the receiver or the data acquisition board. To solve that problem a active cancellation loop was introduced, using a vector modulator (AD8340) a replica of the of the leaked signal with opposite phase is coupled to the receiver in order to cancel the leakage.

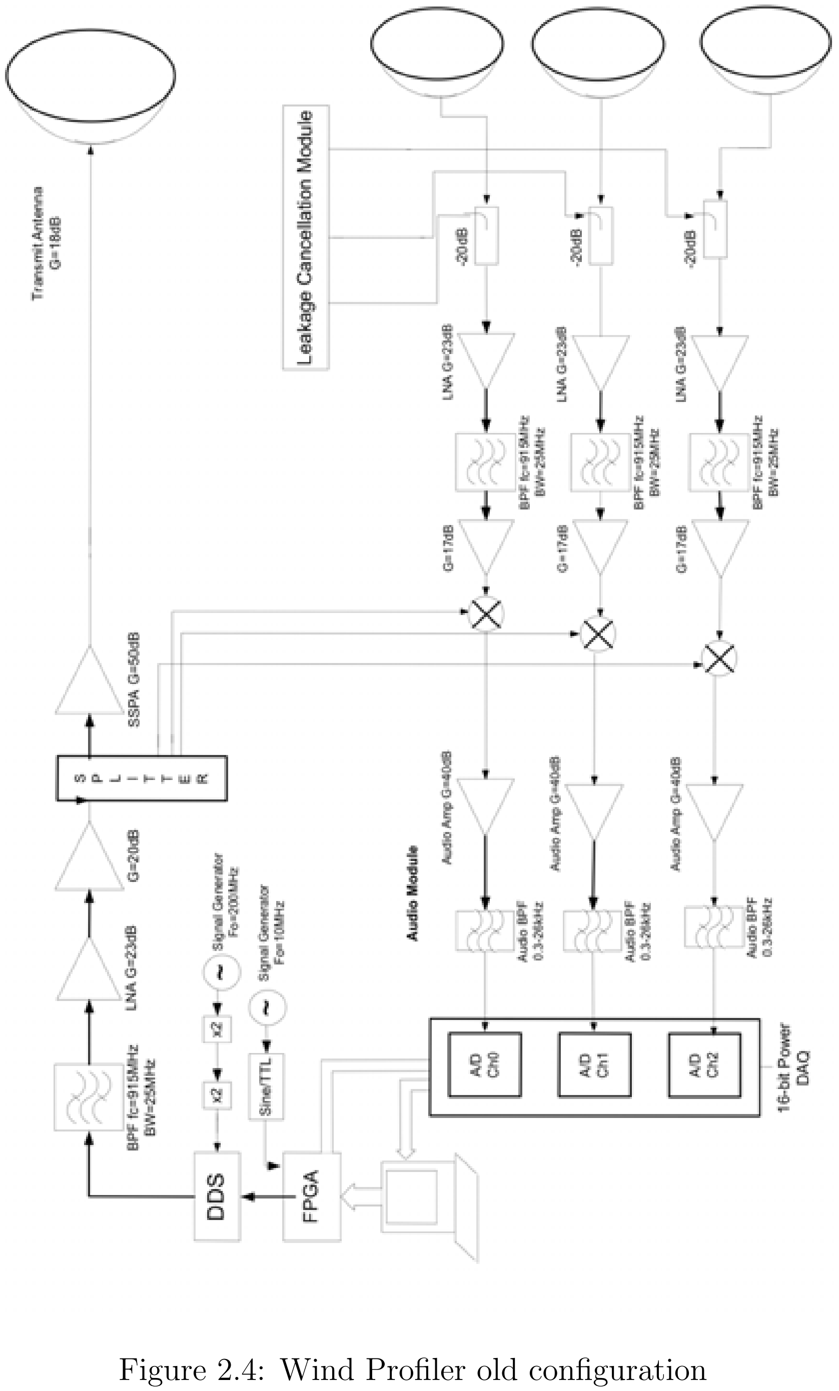


Figure 2.4: Wind Profiler old configuration

2.2.2 FMCW Radar Previous Results

After testing and deploy the FMCW radar, it turned out that there were some noise and sensivity problems in the previous design that needed to be fixed. In this section it will be explained in a consistent way the main issues that impede to detect and obtain correct data from the FMCW radar, and the possible solution to them.

Audio amplifiers

The main problem with the audio amplifier used in the old design is due to the noise it generates itself. Looking through the specifications it was found that there were about $50\mu Vrms$ of noise introduced at the input of the amplifier such as shown in Figure[2.5].

Due to the gain of the receiver it is necessary to make an audio filter/amplifier with low noise components. That do not increase the noise figure of our receiver that much.

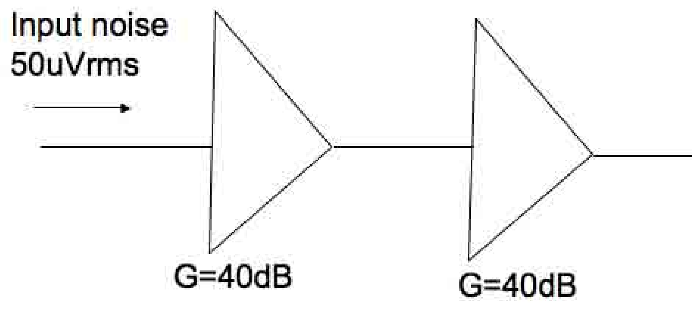


Figure 2.5: audio amplifier issue

Antennas

The isolation of our antennas is not good enough, so it was needed to attenuate 10dB in the front end of the receiver in order to not saturate the mixer. That attenuation has bad effects in our SNR because we attenuate signal but not noise, because the main noise comes from the audio amplifiers. So 10dB in SNR were lost there.

To solve this problem a new redesign of the antennas can be done. Also a shroud fence 20cm higher than the antennas was built around them but it did not improve the isolation significantly, a higher fence can be built so isolation will improve.

Analog Output Card

The AO card has noise of more or less the two LSB, which was being injected to our Vector modulator. Improving the design in a way that we avoid to use the vector modulator will improve the noise characteristics of our receiver.

Proposed Solution

The low transmit power and the low antenna gain require a receiver with low noise and very high gain. To achieve that the receiver will be redesigned avoiding the use of the active cancellation loop. A new audio module will be created with low noise components such as the Linear Technology, Ultralow Noise, Low Distortion, Audio Op Amp, LT1115 [7]. The design will be totally customized for the FMCW Wind profiler needs. The front-end of the receiver will be customized too. Moreover, as it will be seen on the next chapters, a new data acquisition board of 24 bits instead of 16 bits will be required, and a new modification of the FPGA design will be needed. In Figure(2.6) the new FMCW-Wind profiler block diagram is showed. Parts in color are modified from the previous design.

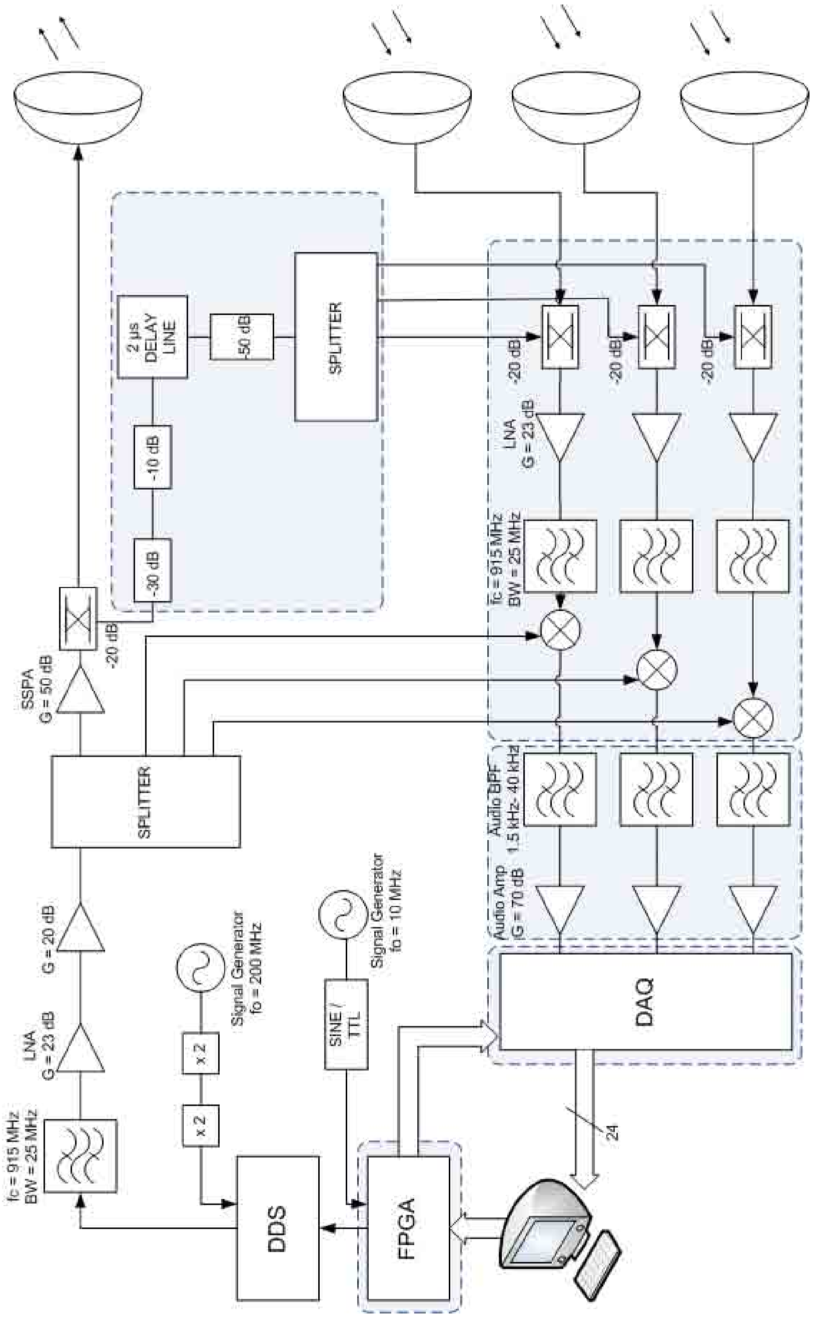


Figure 2.6: New FMCW Wind Profiler block diagram

CHAPTER 3

RECEIVER DESIGN

3.1 Parameter's Calculation

3.1.1 Front end

The transmitter leakage received in our receiver is between -20dBm and -30dBm, depending on the deployment, (-27dBm measured in Tilson farm). It is needed to amplify the received signal, but being careful to not saturate the mixer (7dBm aprox.).

As shown in Figure[2.4] it can be seen that the receiver's front end has $17dB + 23dB = 40dB$ gain. In our case such amplification will saturate the mixer, $-27dBm + 40dB = 13dBm$, so the new design will have only the 23dB amplification of the LNA. That means that more gain must be added to the audio module that it is needed to design.

3.1.2 Audio Module

The audio module needs to have sufficient gain to allow the Acquisition Board to detect the signal, at the same time needs to attenuate the leakage to not saturate it, do anti-aliasing filtering to prevent aliasing in the Acquisition sampling, while being careful with the receiver noise which must lie inside the Acquisition Board range. So we will have to design the audio section having in mind three main parameters such as maximum gain, cutoff frequencies and attenuation per decade.

We have 4 currently modes of operation on the radar. The modes are:

Mode1	Mode2
T_p=6.25 ms PRF=100Hz f_s=60KHz	T_p=8.33 ms PRF=80Hz f_s=60KHz
Mode3	Mode4
T_p=9 ms PRF=70.86Hz f_s=60KHz	T_p=12.5 ms PRF=60Hz f_s=60KHz

With the possibility to choose different modes and modify the bandwidth is possible to adapt the radar to several possibilities or scenarios as follows:

1. Nocturnal boundary layer
   - Weak and sporadic.
   - Range from 30m to 300m
   - Fine resolution, 6m (25MHz bandwidth).
2. Convective boundary layer
   - Unstable, during daytime, strong echo
   - Range below 3Km, typically 2Km in New England.
   - Resolution of 10m (15MHz bandwidth).
3. High sensitivity
   - Range of 3Km or above.
   - Resolution of 60m (2.5MHz bandwidth).

These will be the possible three different scenarios. So will be very important to chose carefully the mode and the bandwidth depending on the scenario. Apart from that is very important to notice that the parameter C_n^2 will vary depending on the season and the weather. Going from 10^{-17} in a cold and dry day in winter, to 10^{-12} in a hot, and humid day. In the next plots a C_n^2 of 10^{-15} is chosen, which is a reasonable average value.

Once the characteristics needed for the radar and the theory of operation are known, is time to choose the different parameters required for the audio module. To do that an IDL program that simulates the ideal response of our receiver is created, this program is used later introducing the response of the designed filter and displaying a more accurate receiver response. As shown in Figure[3.1], the election of a total gain in our audio moule of 110dB is a good election, theoretically we should be able to detect until $C_n^2 = 10^{-16}$. The blue dashed lines are the Acquisition Board range, the decreasing line is a idealization of the scatter (decreases depending on the range). Solid lines represents input variables, doted lines signal and noise after the front-end, and finally dashed line represents signal after the audio module. During summer 2009 some audio module prototypes were built. That work showed that it was not possible to meet the specifications needed for the Wind profiler. The high gain and high order of the filter made it unstable and distorting. The maximum gain achieved was around 85dB, but the filter was still introducing so much distortion.

The solution was to improve the dynamic range of the receiver somehow, in order to need less amplification in the audio module. To achieve that, a new data acquisition board was bought. The board was a four-channel low-noise 24-bit delta-sigma PC104-Plus from General Standards. With that new board the dynamic range was improved from 16 bits to 24 bits, which means a theoretical improve on 24dB.

An audio module with relaxed parameters it is needed now. To picture the ideal response of the receiver a new simulation with IDL is done. As seen in figure[3.2], a gain of 70dB in the audio module should be enough to can capture the data. The

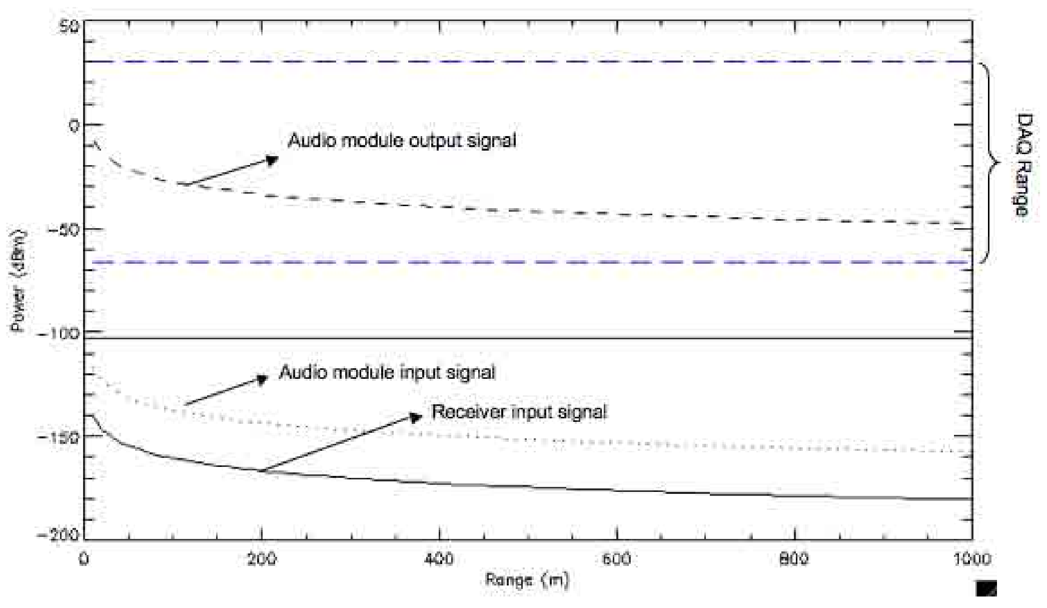


Figure 3.1: Theoretical response of our receiver with $G_{am} = 110dB$ and $C_n^2 = 10^{-15}$

red dashed lines are the old data acquisition board range, and the green dashed line is the new minimum range with the new board.

The new data acquisition board have and anti-aliasing filter incorporated, so the low-pass cut off frequency can be relaxed now, the new cut-off frequency will be 40KHz, so we filter the noise to improve the filter's performance, but do not care about the image frequency.

To choose the cut-off frequency it is necesary to review each of the three scenarios, their requirements depending the available modes we have and see which of of these requirements are the most constraining ones. Besides, in order to know the attenuation per decade and with that the order of the filters, it will be necesary to analyze the leakage and the aliasing throughfully.

Audio Module Requirements for Nocturnal Boundary Layer Scenario

That scenario needs a range of 30m to 300m approximately, and a very good resolution, 6m. To achieve that we will need theoreticaly a cut-off frequency of 800Hz,

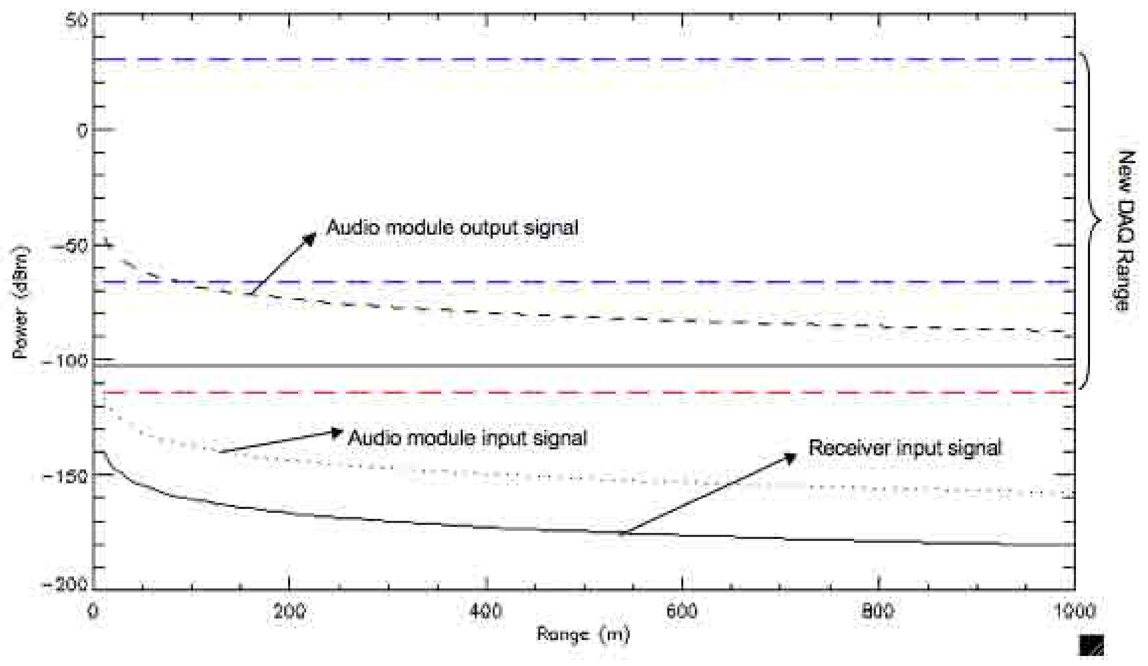


Figure 3.2: Theoretical response of our receiver with $G_{am} = 70dB$, $C_n^2 = 10^{-15}$ and the new Data Acquisition Card.

which is so close to the leakage frequency that is usually around 250Hz. The filter will increase in complexity so much, and have a very steep response in frequency, which can be fatal to the radar because that means that the filter will ring in time domain and that can cause the saturation of our data acquisition board, making us lose some information.

The solution can be done in other way, such as increase the cut-off frequency to 1.5KHz or even 2KHz and creating a new mode with a faster sweep time such as $T_p = 3ms$.

Audio Module Requirements for Convective Boundary Layer Scenario

The range needed in that scenario is below 3Km, typically 2Km in New England, with a resolution of about 10m. With a medium mode such as mode 2 ($T_p = 8.333ms$), we can achieve from 125m (at 1.5KHz) to 2Km (at 24KHz). With mode 4 ($T_p =$

12.5ms) we can reach 3Km at 24KHz, even though our minimum range detectable will be theoretically 187.5m. So, the cut-off frequency of 40kHz proposed for the audio module previously is perfectly suitable.

Leakage and Aliasing

The leakage must be attenuated in order to not saturate the data acquisition board, so the final power it must be below 30dBm. A good choice will be to attenuate the leakage to the center to the data acquisition board range, so we can eliminate it afterwards with the post-processing, and there is also some margin of error left.

$$P_{r_{leakage}} + G_{LNA} + G_{audiomod} - L_{audiomod_{leakage}} = \frac{DAQ_{max} + DAQ_{min}}{2} \tag{3.1}$$

$$-25dBm + 23dB + 70dB - L_{audiomod_{leakage}} = \frac{-110dBm + 30dBm}{2} \tag{3.2}$$

$$L_{audiomod_{leakage}} = 110dB \tag{3.3}$$

With the result in 3.3 we can conclude that it will be needed an attenuation per decade, assuming a low cut-off frequency of 1.5KHz and the frequency of our leakage in 300Hz, of $157\frac{dB}{decade}$. Which means that an order 8 high pass filter will be needed, but considering that the design will be done with order 2 modules, and that it is better to have some margin error in case more amplification is used in the front end in the future, the order of the high pass will be order 10. The anti-aliasing filter will need the same stages than the high pass, in order to filter noise and be able to amplify enough.

Once the general parameters are known, will be important to decide the architecture of the filter deciding in which order the different stages will be placed in the audio module, and what will the distribution of gain be between all the stages, in order to improve as much as possible the noise figure of out audio module, always taking into account the anti-leakage performance.

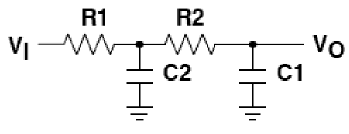
3.1.3 Design Architecture

The audio module must have the low-pass filter stages at the beginning in order to improve the noise performance. It will be needed too to have higher gains in the first stages, to have a smaller noise figure as possible. The gain in the first low pass stage should be higher as possible, but there is a problem with the leakage and is that we cannot amplify so much in the low-pass filter due to there is no attenuation of the leakage (300Hz) there is only amplification of it and there is the risk of saturating the operational amplifiers. Amplification in the high pass stage stage is not recommended due to the amplification of noise.

A new architecture should be used for the audio module due to the specific needs of the Wind Profiler audio module. To meet the specifications needed the architecture chosen is made of cells. Every cell will be composed by a low-pass filter followed by a high-pass filter, the amplification will be done on the low pass-filter. With that technique the cells with lowest Q can be put at the beginning of the chain. The maximum amplification on every cell should be chosen carefully in order to not saturate the operational amplifiers.

3.2 Active Filter Theory

3.2.1 Introduction



$$\frac{Vo}{Vi} = \frac{1}{s^2(R1C2R2C1) + s(R1C2 + R2C1 + R1C1) + 1}$$

Figure 3.3: Basic Second Order Low-Pass Filter

Figure[3.3] shows a two-stage RC network that forms a second order low-pass filter. This filter is limited because its Q is always less than 1/2. With R1=R2 and

C1=C2 Q=1/3. Q approaches the maximum value of 1/2 when the impedance of the second RC stage is much larger than the first. Most filter require Qs larger than 1/2.

Larger Qs are attainable by using a positive feedback amplifier. If the positive feedback is controlled -localized to the cut-off frequency of the filter- almost any Q can be realized, limited mainly by the constraints of the power supply and component tolerances. Figure[3.4] shows a unity gain amplifier used in this manner. Capacitor C2, no longer connected to ground, provides a positive feedback path. In 1995, R.P. Sallen and E.L. Key described these filter circuits, and hence they are generally known as Sallen-Key filters.

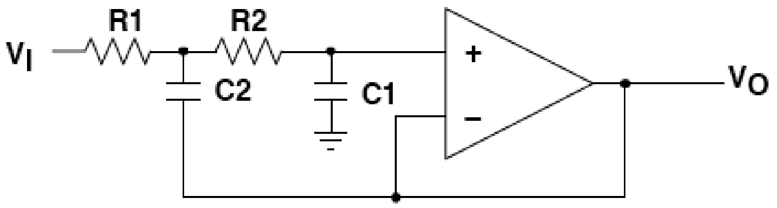


Figure 3.4: Unity Gain Sallen-Key Low-Pass Filter

The operation can be described qualitatively:

- At low frequencies, where C1 and C2 appear as open circuits, the signal is simply buffered to the output.
- At high frequencies, where C1 and C2 appear as short circuits, the signal is shunted to ground at the amplifier's input, the amplifier amplifies this input to its output, and the signal does not appear at V_o.
- Near the cut-off frequency, where the impedance of C1 and C2 is on the same order as R1 and R2, positive feedback through C2 provides Q enhacement of the signal

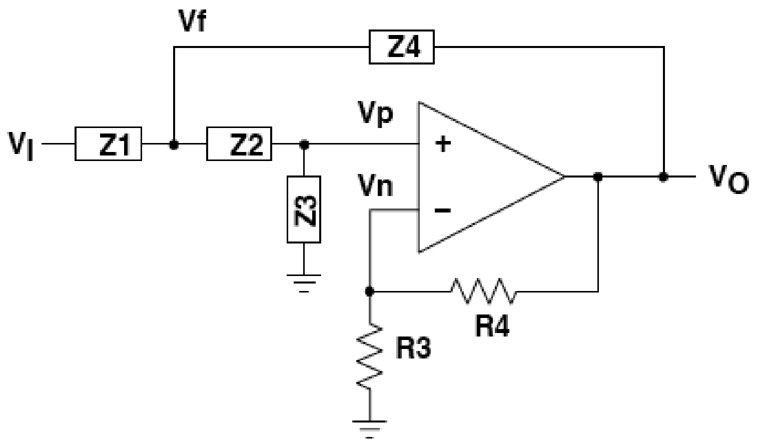


Figure 3.5: Generalized Sallen-Key Circuit

3.2.2 Generalized Circuit Analysis

The circuit shown in Figure[3.5] is a generalized form of the Sallen-Key circuit, where generalized impedance terms, Z, are used for the passive filter components, and R3 and R4 set the pass-band gain.

To find the circuit solution for this generalized circuit, find the mathematical relationships between V_i, V_o, V_p, and V_n, and construct a block diagram.

KCL at V_f:

$$V_f(\frac{1}{Z1}+\frac{1}{Z2}+\frac{1}{Z4}) = V_i(\frac{1}{Z1}) + V_p(\frac{1}{Z2}) + V_o(\frac{1}{Z4}) \tag{3.4}$$

KCL at V_p:

$$V_f(\frac{1}{Z2}+\frac{1}{Z3}+) = V_f(\frac{1}{Z2}) \Rightarrow V_f = V_p(1+\frac{Z2}{Z3}) \tag{3.5}$$

Substitute Equation[3.5] into Equation[3.4] and solve for V_p:

$$\begin{aligned} V_p = V_i(&\frac{Z2Z3Z4}{Z2Z3Z4+Z1Z2Z4+Z1Z2Z3+Z2Z2Z4+Z2Z2Z1}) + \\ &V_o(\frac{Z1Z2Z3}{Z2Z3Z4+Z1Z2Z4+Z1Z2Z3+Z2Z2Z4+Z2Z2Z1}) \end{aligned} \tag{3.6}$$

KCL at V_n:

$$V_n(\frac{1}{R3}+\frac{1}{R4}+) = V_o(\frac{1}{R4}) \Rightarrow V_n = V_o(\frac{R3}{R3+R4}) \tag{3.7}$$

3.2.2.1 Gain Block Diagram

By letting: a(f)= the open-loop gain of the amplifier, b=($\frac{R3}{R3+R4}$),
$c = \frac{Z2Z3Z4}{Z2Z3Z4+Z1Z2Z4+Z1Z2Z3+Z2Z2Z4+Z2Z2Z1}$, $d = \frac{Z1Z2Z3}{Z2Z3Z4+Z1Z2Z4+Z1Z2Z3+Z2Z2Z4+Z2Z2Z1}$, and $V_e = V_p - V_n$, the generalized Sallen-Key filter circuit is represented in gain-block form as shown in Figure[3.6].

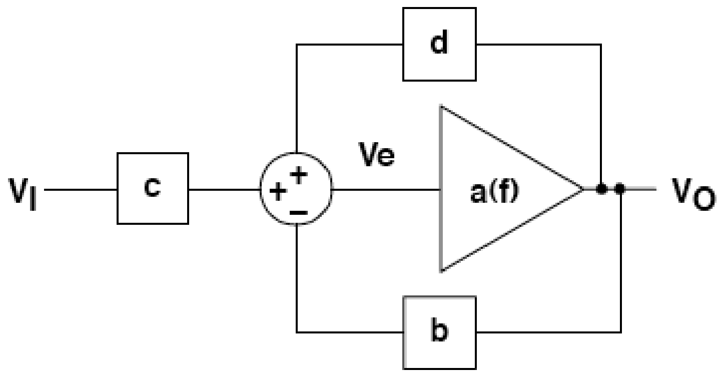


Figure 3.6: Gain-Block Diagram of the Generalized Sallen-Key Filter

From the gain-block diagram the transfer function can be solved easily by observing, $V_o = a(f)V_e$ and $V_e = cV_i + dV_o - bV_o$. Solving for the generalized transfer function from gain block analysis gives:

$$\frac{V_o}{V_i} = (\frac{c}{b})(\frac{1}{1+\frac{1}{a(f)b}-\frac{d}{b}}) \tag{3.8}$$

3.2.2.2 Ideal Transfer Function

Assuming a(f)b is very large over the frequency of operation, $\frac{1}{a(f)b} \approx 0$, the ideal transfer function from gain block analysis becomes:

$$\frac{V_o}{V_i} = (\frac{c}{b})(\frac{1}{1-\frac{d}{b}}) \tag{3.9}$$

By letting $\frac{1}{b} = K$, $c = \frac{N1}{D}$, and $d = \frac{N2}{D}$, where N1, N2, and D are the numerators and denominators shown above, the ideal equation can be rewritten as: $\frac{V_o}{V_i} = (\frac{K}{\frac{D}{N1} - \frac{KN2}{N1}})$. Plugging in the generalized impedance terms gives the ideal transfer function with impedance terms:

$$\frac{V_o}{V_i} = \frac{K}{\frac{Z1Z2}{Z3Z4} + \frac{Z1}{Z3} + \frac{Z2}{Z3} + \frac{Z1(1-K)}{Z4} + 1} \tag{3.10}$$

3.2.3 Low-Pass Circuit

The standard frequency domain equation for a second order low-pass filter is:

$$H_{LP} = \frac{K}{-\frac{f}{f_c}^2 + \frac{jf}{Qf_c} + 1} \tag{3.11}$$

Where f_c is the corner frequency (note that f_c is the breakpoint between the pass band and stop band, and is not necessarily the -3dB point) and Q is the quality factor. When $f \ll f_c$ Equation[3.11] reduces to K, and the circuit passes signals multiplied by a gain factor K. When $f = f_c$, Equation[3.11] reduces to -jKQ, and signals are enhanced by factor Q. When $f \gg f_c$, Equation[3.11] reduces to $-K(\frac{f_c}{f})^2$, and signals are attenuated by the square of the frequency ratio. With attenuation at higher frequencies increasing by a power of two, the formula describes a second order low-pass filter.

Figure[3.7] shows the Sallen-Key circuit configured for low-pass.

From Equation[3.10], the ideal low-pass Sallen-Key transfer function is:

$$\frac{V_o}{V_i}(l_p) = \frac{K}{s^2(R1R2C1C2) + s(R1C1 + R2C1 + R1C2(1-K)) + 1} \tag{3.12}$$

By letting:

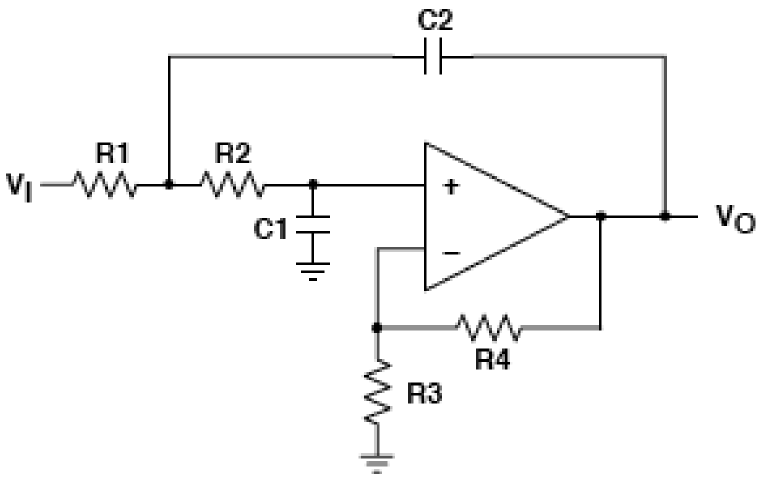


Figure 3.7: Low-Pass Sallen-Key Circuit

$s = j2\pi f$, $FSF * f_C = \frac{1}{2\pi\sqrt{R1R2C1C2}}$, and $Q = \frac{\sqrt{R1R2C1C2}}{R1C1+R2C1+R1C2(1-K)}$, Equation[3.12] follows the same form as Equation[3.11]. With some simplifications, these equations can be dealt with efficiently.

3.2.4 High-Pass Circuit

The standard equation (in frequency domain) for a second order high-pass is:

$$H_{HP} = \frac{-K\frac{f}{f_c}^2}{-\frac{f}{f_c}^2 + \frac{jf}{Qf_c} + 1} \tag{3.13}$$

When $f \ll f_c$, Equation[3.13] reduces to $-K(\frac{f}{f_c})^2$. Below f_c signals are attenuated by the square of the frequency ratio. When $f = f_c$, Equation[3.13] reduces to -jKQ, and signals are enhanced by the factor Q. When $f \gg f_c$, Equation[3.13] reduces to K, and the circuit passes signals multiplied by the gain factor K. With attenuation at lower frequencies increasing by a power of 2, Equation[3.13] describes a second order high-pass filter.

Figure[3.8] shows the Sallen-Key circuit configured for high-pass.

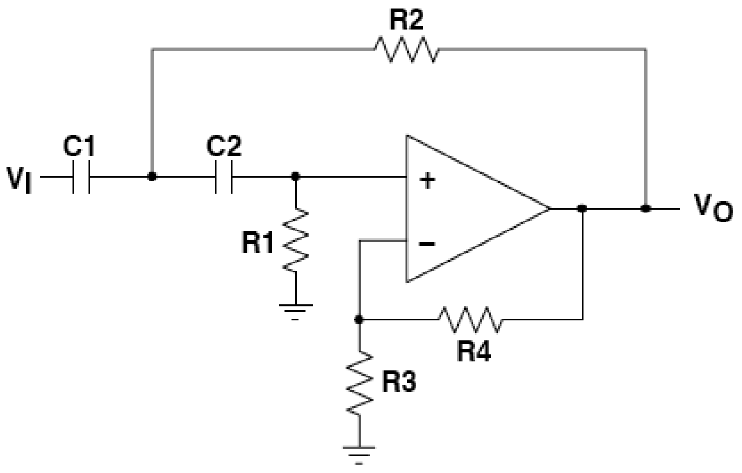


Figure 3.8: High-Pass Sallen-Key Circuit

From Equation[3.10], the ideal high-pass transfer function is:

$$\frac{V_o}{V_i}(hp) = \frac{K}{\frac{1}{s^2(R1R2C1C2)} + \frac{1}{s}(\frac{1}{R1C1} + \frac{1}{R1C2} + \frac{(1-K)}{R2C1}) + 1} \tag{3.14}$$

with some manipulation becomes

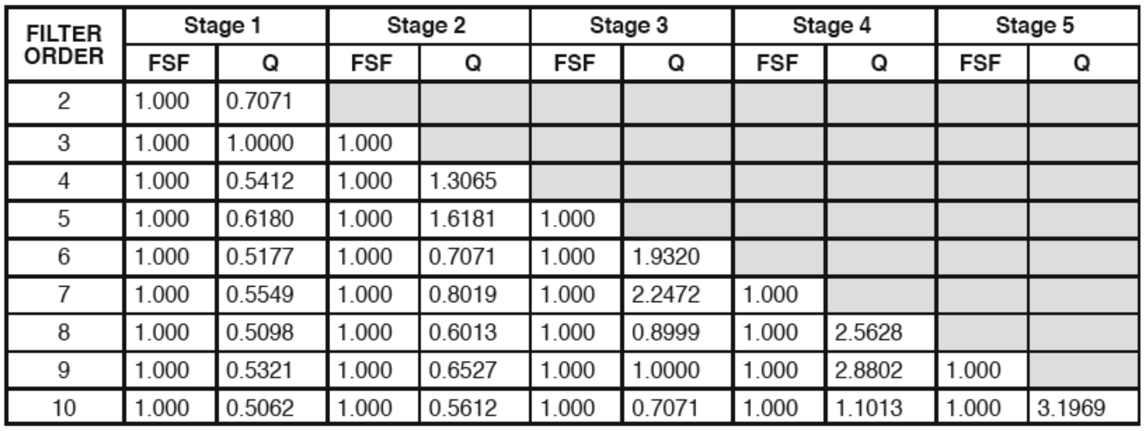
$$\frac{V_o}{V_i}(hp) = \frac{K(s^2(R1R2C1C2))}{s^2(R1R2C1C2) + s(R2C2 + R2C1 + R1C2(1-K)) + 1} \tag{3.15}$$

By letting $s = j2\pi f$, $FSFf_o = \frac{1}{2\pi sqrtR1R2C1C2}$, and $Q = \frac{\sqrt{R1R2C1C2}}{R2C2+R2C1+R1C2(1-K)}$, Equation[3.15] follows the same form as Equation[3.13]. As above, simplfications make these equations much easier to deal with.

3.2.5 Filter Tables

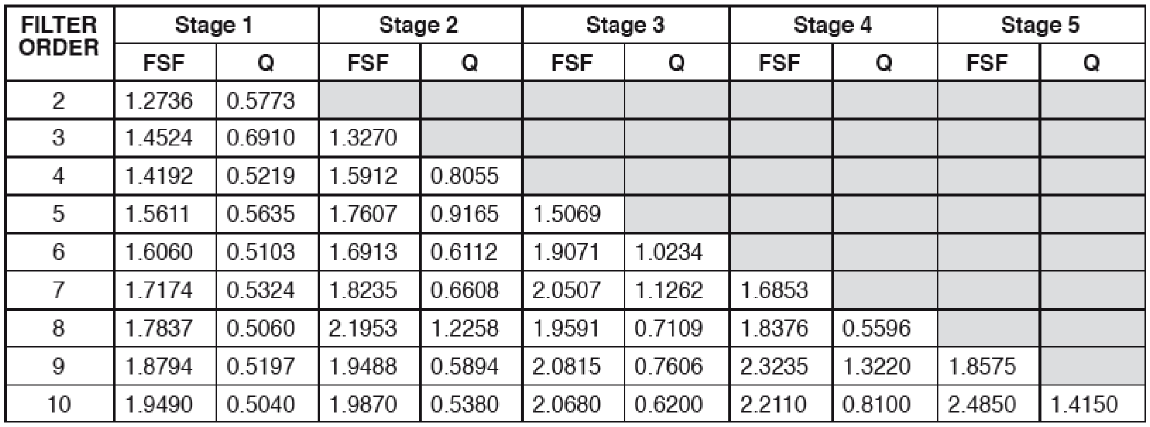
Typically, filter books list the zeroes or the coefficients of the particular polynomial being used to define the filter type. It takes a certain amount of mathematical manipulation to turn this information into a circuit realization. It is implicit that higher-order filters are constructed by cascading second order stages for even-order filters (one for each complex-zero pair). A first-order stage is then added if the filter order is odd. With the filter tables on Figures (3.9), (3.10) and (3.12), the preliminary work is done, and the proper circuit components can be calculated with just three formulas.

For a low-pass Sallen-Key filter with cut-off frequency f_c and pass band gain K, set $K = \frac{R3+R4}{R3}$, $FSF * f_c = \frac{1}{2\pi\sqrt{R1R2C1C2}}$, and $Q = \frac{\sqrt{R1R2C1C2}}{R1C1+R2C1+R1C2(1-K)}$ for each second order stage. If an odd order is required, set $FSF * f_c = \frac{1}{2\pi RC}$ for that stage. The tables are arranged so that increasing Q is associated with increasing stage order. High-order filters are normally arranged in this manner to help prevent clipping.



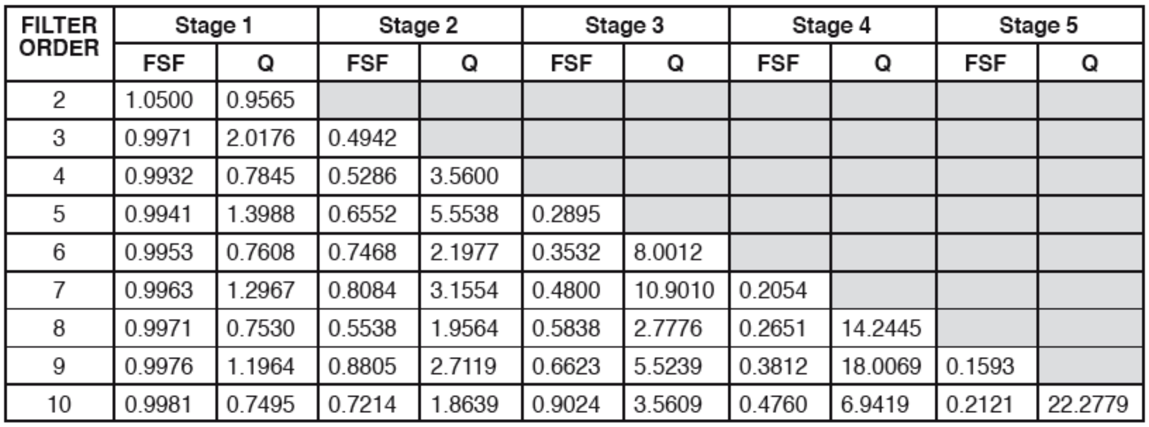
FILTER ORDER	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
	FSF	Q	FSF	Q	FSF	Q	FSF	Q	FSF	Q
2	1.000	0.7071								
3	1.000	1.0000	1.000							
4	1.000	0.5412	1.000	1.3065						
5	1.000	0.6180	1.000	1.6181	1.000					
6	1.000	0.5177	1.000	0.7071	1.000	1.9320				
7	1.000	0.5549	1.000	0.8019	1.000	2.2472	1.000			
8	1.000	0.5098	1.000	0.6013	1.000	0.8999	1.000	2.5628		
9	1.000	0.5321	1.000	0.6527	1.000	1.0000	1.000	2.8802	1.000	
10	1.000	0.5062	1.000	0.5612	1.000	0.7071	1.000	1.1013	1.000	3.1969

Figure 3.9: Butterworth Filter Table



FILTER ORDER	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
	FSF	Q	FSF	Q	FSF	Q	FSF	Q	FSF	Q
2	1.2736	0.5773								
3	1.4524	0.6910	1.3270							
4	1.4192	0.5219	1.5912	0.8055						
5	1.5611	0.5635	1.7607	0.9165	1.5069					
6	1.6060	0.5103	1.6913	0.6112	1.9071	1.0234				
7	1.7174	0.5324	1.8235	0.6608	2.0507	1.1262	1.6853			
8	1.7837	0.5060	2.1953	1.2258	1.9591	0.7109	1.8376	0.5596		
9	1.8794	0.5197	1.9488	0.5894	2.0815	0.7606	2.3235	1.3220	1.8575	
10	1.9490	0.5040	1.9870	0.5380	2.0680	0.6200	2.2110	0.8100	2.4850	1.4150

Figure 3.10: Bessel Filter Table



FILTER ORDER	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
	FSF	Q	FSF	Q	FSF	Q	FSF	Q	FSF	Q
2	1.0500	0.9565								
3	0.9971	2.0176	0.4942							
4	0.9932	0.7845	0.5286	3.5600						
5	0.9941	1.3988	0.6552	5.5538	0.2895					
6	0.9953	0.7608	0.7468	2.1977	0.3532	8.0012				
7	0.9963	1.2967	0.8084	3.1554	0.4800	10.9010	0.2054			
8	0.9971	0.7530	0.5538	1.9564	0.5838	2.7776	0.2651	14.2445		
9	0.9976	1.1964	0.8805	2.7119	0.6623	5.5239	0.3812	18.0069	0.1593	
10	0.9981	0.7495	0.7214	1.8639	0.9024	3.5609	0.4760	6.9419	0.2121	22.2779

Figure 3.11: 1-dB Chebyshev Filter Table

3.3 Active Filter Design

The first step is to deign the low-pass filter and the high-pass filter, using the Sallen-Key active filter theory, and the filter tables. With the specifications calculated before at the beginning of the receiver design chapter. Once the filters are designed, their stages will be cascaded creating low-pass with high-pass stages that will form the band-pass filter. Using Spice simulations the filter will be check in order to find out if it satisfies the requirements and to check its performance.

The filter type used on the design is Butterworth because it has the flatest possible pass-band magnitude response. Attenuation is -3dB at the design cut-off frequency.

Attenuation above the cutoff frequency is a moderately steep 20-dB per decade per pole. And it has moderate overshoot and ringing.

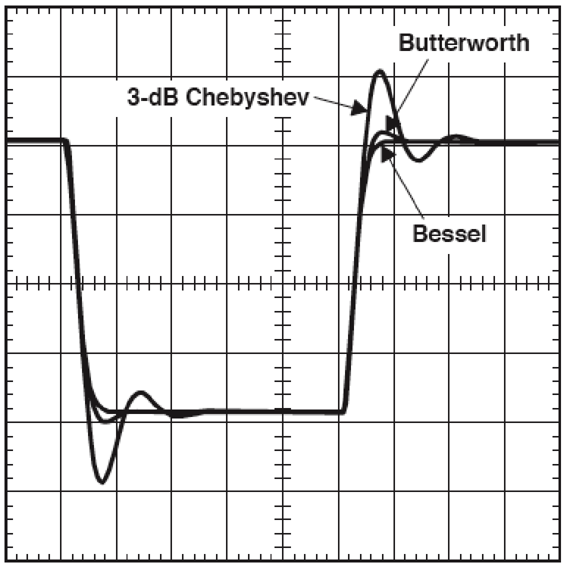


Figure 3.12: Transient Response of the Three Filters Type

As it will be seen in the next chapter ringing and overshooting is a very important issue in the audio module performance. So a Bessel type filter was built, due to its linear phase response, and minimal overshoot and ringing. But the improvement in performance was not noteworthy. Besides it takes a higher-order Bessel filter to give a magnitude response which approaches that of a given Butterworth filter. So in the next sections it will be explained the design of the Butterworth filter type, that is currently working on the radar.

3.3.1 Component selection

Theoretically, any value of R and C can be used, but practical considerations should take into account. Given a specific cut-off frequency R and C are inversely proportional. In the case of low-pass Sallen-Key filter, the larger the value of R the lower the transmission of signals at high frequency. Making R too large make C become so small that the parasitic capacitances cause errors.

For the high-pass filter, stray capacitance in the circuit, including the input capacitance of the amplifier, makes the choice of small capacitors and thus large resistors, undesirable. Also, being high-pass circuit the band-pass is potentially very large, and resistor noise associated with increased values can become an issue.

So, capacitors of less than 1nF and low quality dielectrics such as electrolytic capacitors will be avoided. Resistor values will be in the range of few hundred ohms to few thousand ohms whenever it is possible, metal film resistors will be used in order to achieve good temperature coefficients, and 1% tolerance E96 values will be used.

3.3.2 Low-Pass Filter

The low-pass filter needs a total of 70dB of amplification, a cut-off frequency of $f_c = 40KHz$ (the new data acquisition board has its own anti-aliasing filter), and order ten (same order as high pass so the band-pass filter can be implemented).

The first stage will be designed with 0dB gain in order to be sure the leakage will not saturate in any of the stages of the final filter. The next stages will start with decreasing gains, always taking into account the equations $K = \frac{R3+R4}{R3}$, $FSF * f_C = \frac{1}{2\pi\sqrt{R1R2C1C2}}$, and $Q = \frac{\sqrt{R1R2C1C2}}{R1C1+R2C1+R1C2(1-K)}$, to design the stages with the proper gain so the leakage will not saturate the next stages. In addition appropriate commercial values of R and C are chosen. The gains chosen for each stage are $G_1 = 0dB$, $G_2 = 20dB$, $G_3 = 20dB$, $G_4 = 17dB$ and $G_5 = 13dB$.

With all the parameters calculated before, the gain for every stage, the previous equations and the filter tables. It is proceed to design the low-pass filter. The design of that filter had a problem due to the due to the high quality factor required on the last stage $Q = 3.1969$ the cut-off frequency of $f_c = 40KHz$, and the amplification of 13dB. Taking into account the parameters required for the last stage, an operational amplifier of a Gain-Bandwidth product exceeding 40MHz would be needed. As seen in [7] the LT1115 operational amplifier used in the design has a GBP of 40MHz. So the last stage is modified in order to can achieve a better performance. The cut-off frequency of the last stage is reduced to $f_{c5} = 28KHz$ and its quality factor is reduced too to $Q = 1.2962$. The resultant low-pass filter is showed in Figure(3.13).

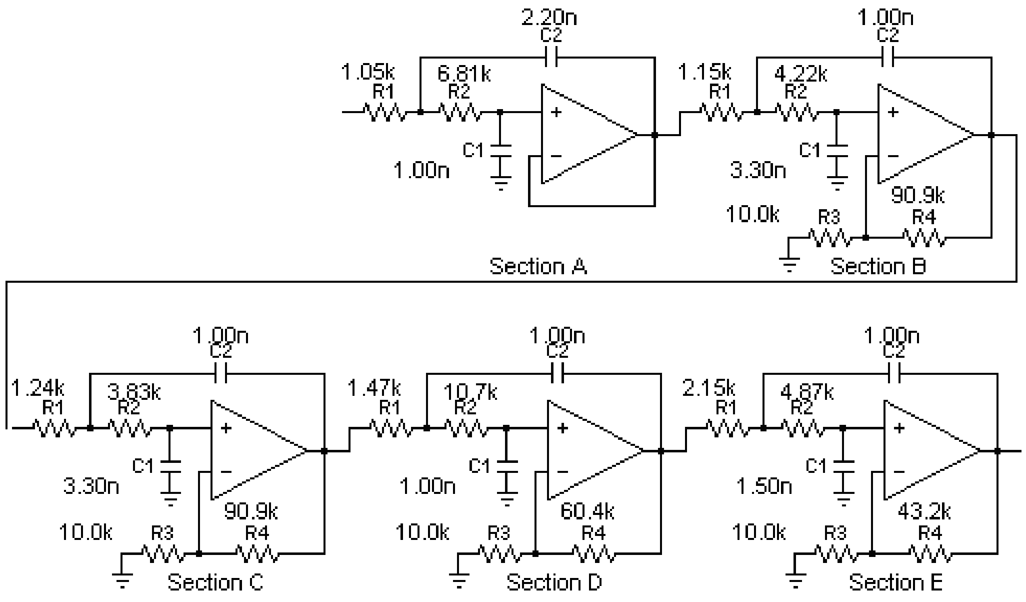


Figure 3.13: Low-Pass Filter implementation

The frequency response of the low-pass filter is depicted in Figure(3.14). It can be observed that the resultant cut-off frequency is not 40KHz, it is 28KHz, the response is flat in the pass-band, and besides the frequencies of interest are all below $\frac{f_s}{2} = 30KHz$, so the filter is suitable.

3.3.3 High-Pass Filter

The high-pass filter has a 0dB amplification in order to have a better noise performance, a cut-off frequency of $f_c = 2.2KHz$ (the previously calculated $f_c = 1.5KHz$ turned out to be too low), and order ten in order to attenuate the leakage enough.

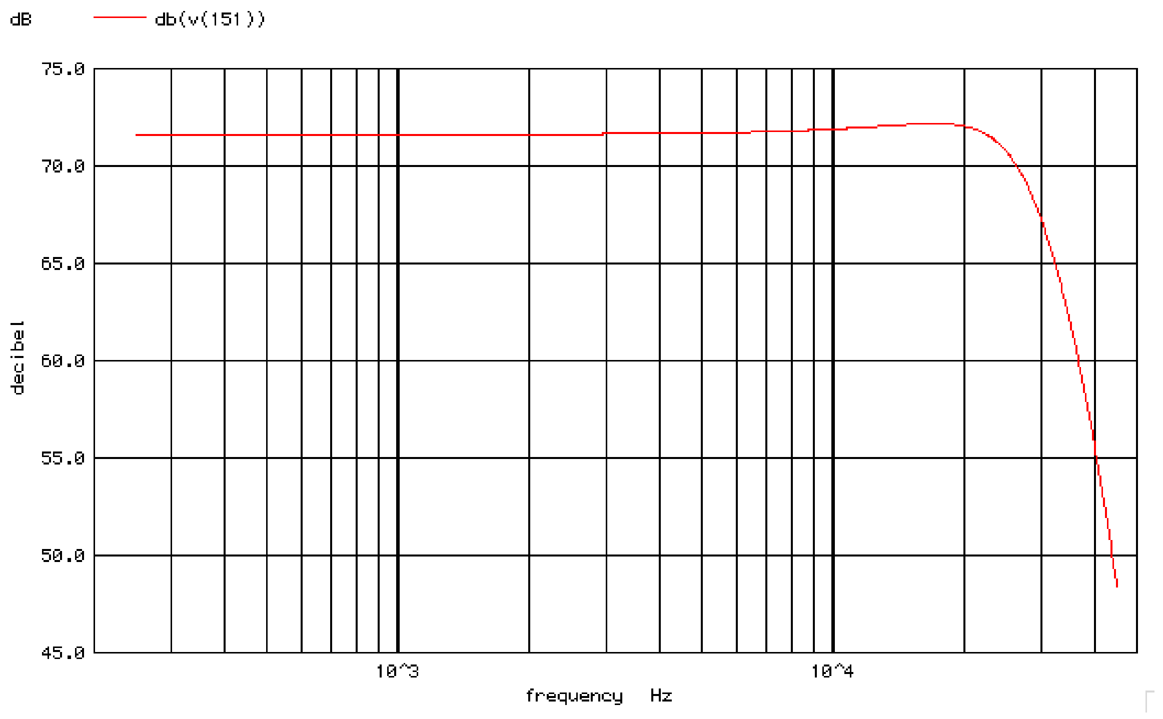


Figure 3.14: Low-Pass Filter response

The stages are designed following the filter tables and the following equations $K = \frac{R3+R4}{R3}$, $FSFf_o = \frac{1}{2\pi sqrtR1R2C1C2}$, and $Q = \frac{\sqrt{R1R2C1C2}}{R2C2+R2C1+R1C2(1-K)}$. With all the parameters calculated it is proceed to design the high-pass filter with the following results, Figure(3.15).

The frequency response of the high-pass filter is depicted in Figure(3.16).

3.3.4 Band-Pass Filter

The band-pass filter is created by cascading the low-pass filter and the high-pass filter designed before. So the first stage of the band-pass will be the first of the low-pass, the second will be the first of the high-pass and so on. Before implementation, a spice simulation is done, adding the resistor of 50Ω of the input impedance and the $R_{DAQ} = 1M\Omega$ with the following results, Figure(3.17).

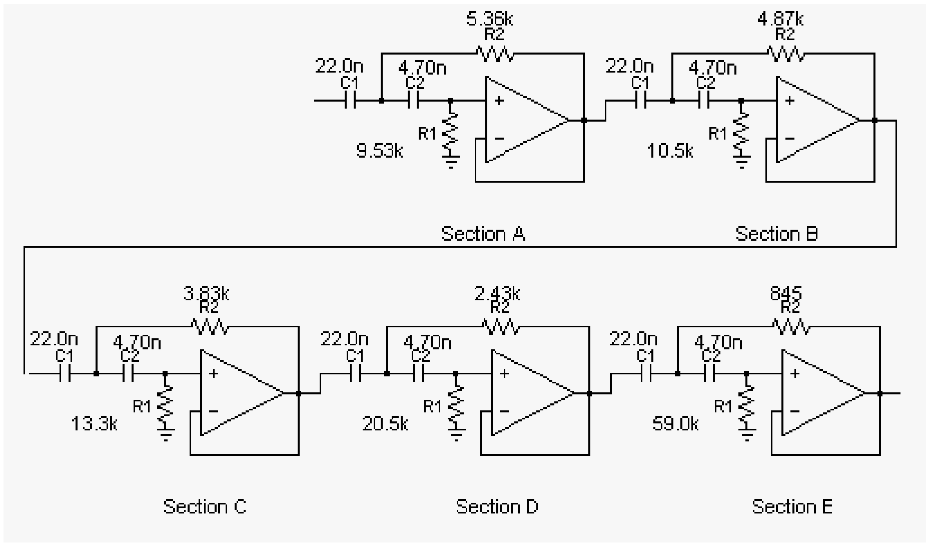


Figure 3.15: High-Pass Filter implementation

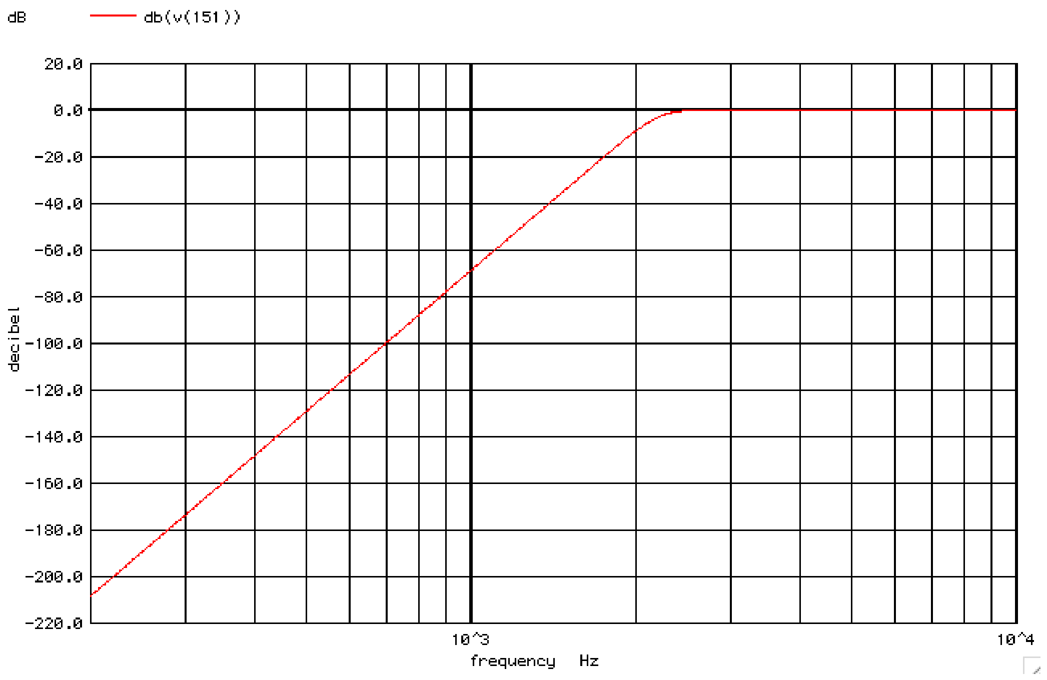


Figure 3.16: High-Pass Filter response

The gain in the pass-band is about 71.3dB, which is close to the 70dB goal it was proposed. The final cut-off frequencies are $f_{clow} = 2.17KHz$ and $f_{chigh} = 29.3KHz$. The gain at $f_{leakage} = 300Hz$ is -101dB.

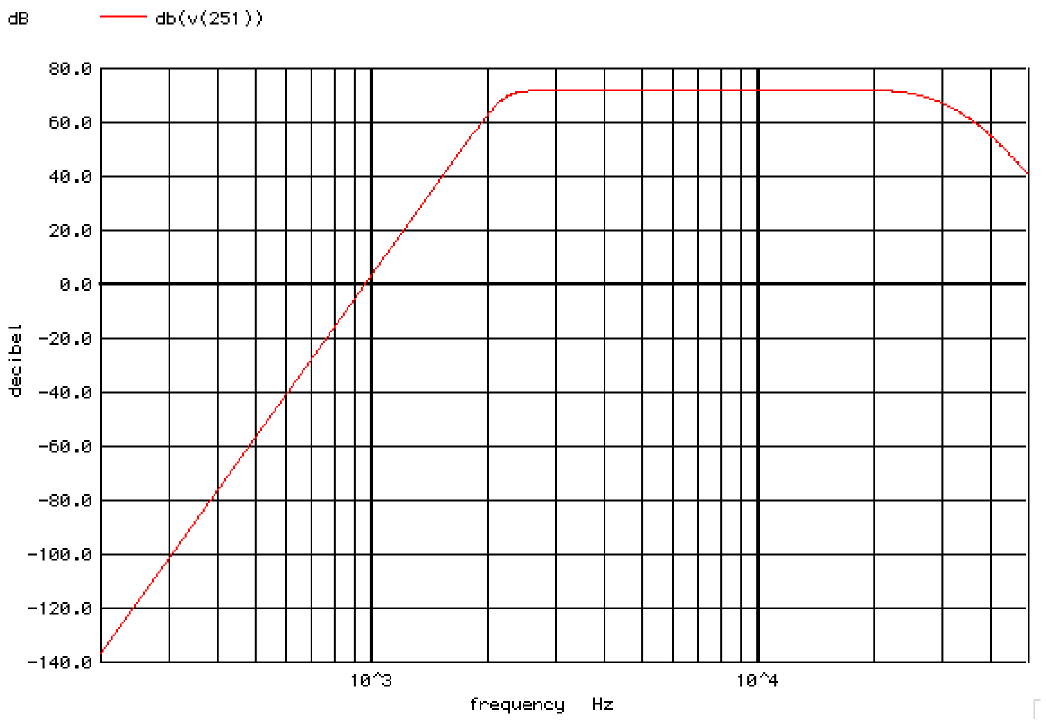


Figure 3.17: Band-Pass Filter response

With that results it can be concluded that the design satisfied the requirements.

Now the response of the filter is introduced in the IDL program to see the theoretical response of the receiver adding the filter response.

The Figure 3.18 shows the response of the receiver with a $C_n^2 = 10^{-15}$. The signal is put into the DAQ range, as desire, with a minimum detectable frequency around 1KHz which is better than the 1.5KHz expected, but with lower C_n^2 this minimum value will increase.

3.3.4.1 Noise analysis

A noise analysis is performed with Spice for the audio module, with the result shown in Figure[3.19]. The result is given in μV^2, with a maximum of $982\mu V^2$ which is around -53dBm (at 50Ω), still in the DAQ range but as we can see. The noise results are not as good as expected, and it is so because the design of the filter does

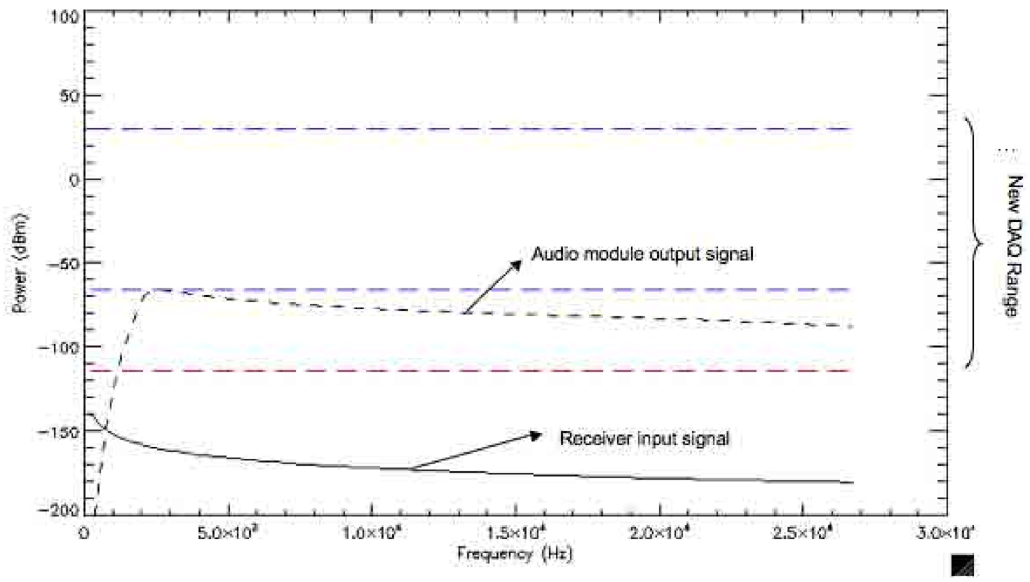


Figure 3.18: Receiver theoretical response with designed filter.

not follow the conventional design rules. The filter is designed in a way that it can satisfy the FMCW-Wind Profiler specifications and for doing so a special architecture was needed as it was explained, not allowing the audio module designed to achieve an optimal noise performance. The results should be sufficient, due to the SNR is highly improved after processing.

3.3.4.2 Analysis of component tolerances effects

Before building and testing the filter in the lab, a final analysis is performed in order to know the effect of the tolerances of the components that will be used later. The resistors are assumed to have a tolerance of ±1% while the capacitors will be suppose to have a ±15% tolerance even though we will try to mount the final filter with ±10% capacitors, but with that choice we can try to have a more realistic analysis because when mounting the filter another problems will come up, such as stray capacitances.

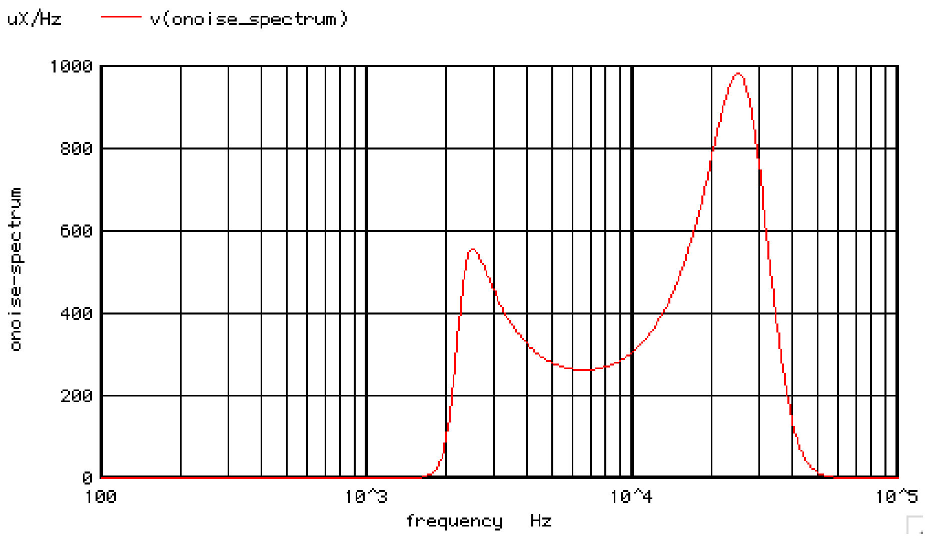


Figure 3.19: Output noise spectrum of the Band-pass filter

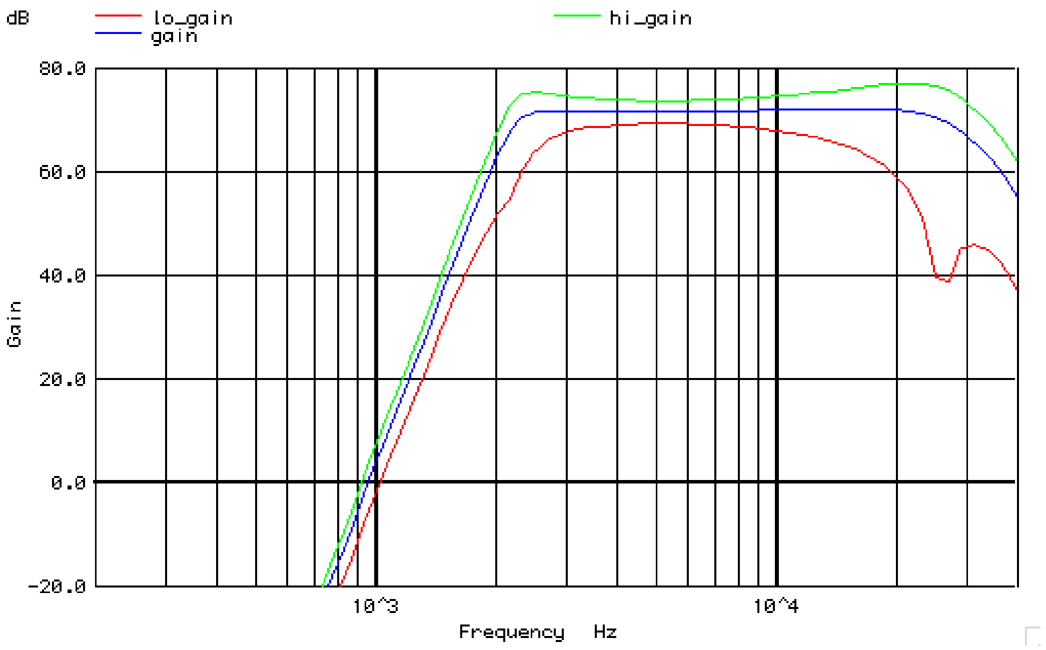


Figure 3.20: Analysis of the effect of the component tolerances

The results are shown in Firgure[3.20], the response of the filter will be with a 90% of probability inside the limits higain and logain. These limits are: $G_{max} \in [69.1, 77]dB$, which are suitable for the design. The next step will be build and test the audio module in the lab, and when positive results are obtained apply the audio module to the radar and make the necesary modifications in order to test the radar.

CHAPTER 4

AUDIO MODULE IMPLEMENTATION AND TESTS

After designing and simulating the audio module, a prototype was built. First, the frequency response was tested and was introduced into the IDL program in order to see the theoretical performance with the prototype. After that the audio module was tested with real radar signal, simulating the leakage and calibration signal, and observing the output.

4.1 Laboratory Tests and Results

The audio module prototype was built in a breadboard Figure(4.1), and then its frequency response was tested. The results obtained are showed in Figure(4.2), where it is observed the response is quite similar to the previously spice simulated response.

Once the frequency response is obtained, a simulation of the receiver response with IDL is done including it. The results can be seen in Figure(4.3). All the frequencies of interest are within the data acquisition card range, which is the main goal. Also it can be seen that very low frequency are theoretically inside the range, but that is not a big problem as far as it does not saturate either the audio module or the data acquisition card, because with signal processing the non-desired low-frequencies or leakage can be erased.

At this point it is time to utilize real radar signal to test the audio module. To do so, the radar is set up in the lab, and a leakage signal is simulated by introducing

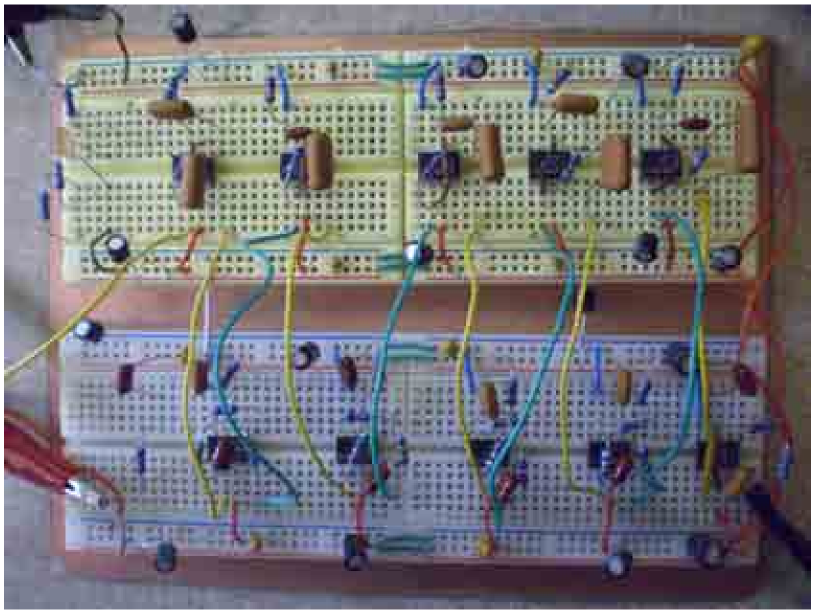


Figure 4.1: Audio Module Prototype Implementation

attenuation and delay of the same magnitude range of the real leakage received on the antennas when the radar is deployed on the field. The results obtained are showed in Figure(4.4). It can be seen that the filter eliminates the leakage, because no lower frequencies are contained on the signal. Only frequencies of 5KHz and 15KHz are detected, that is so because of the calibration signal (delay line) that it is added to the received signal, see block diagram, figure(2.6). On the other hand, it is observed that there is quite a long transient, that can even saturate the data acquisition card at the beginning of the chirp. To improve that, what it can be done is to increase the duty cycle of the chirp, to nearly 100%, so the transient will be lower due to the elimination of the step response of the original radar signal. To achieve that, a modification of the current FPGA design was done, that modification is explained in the next section.

With the transmitting chirp of 100% duty cycle, the response improved as seen in Figure(4.5). Now the transient is shorter and it does not saturate that much at the beginning of the chirp. The calibration signal can be observed in Figure(4.6) where

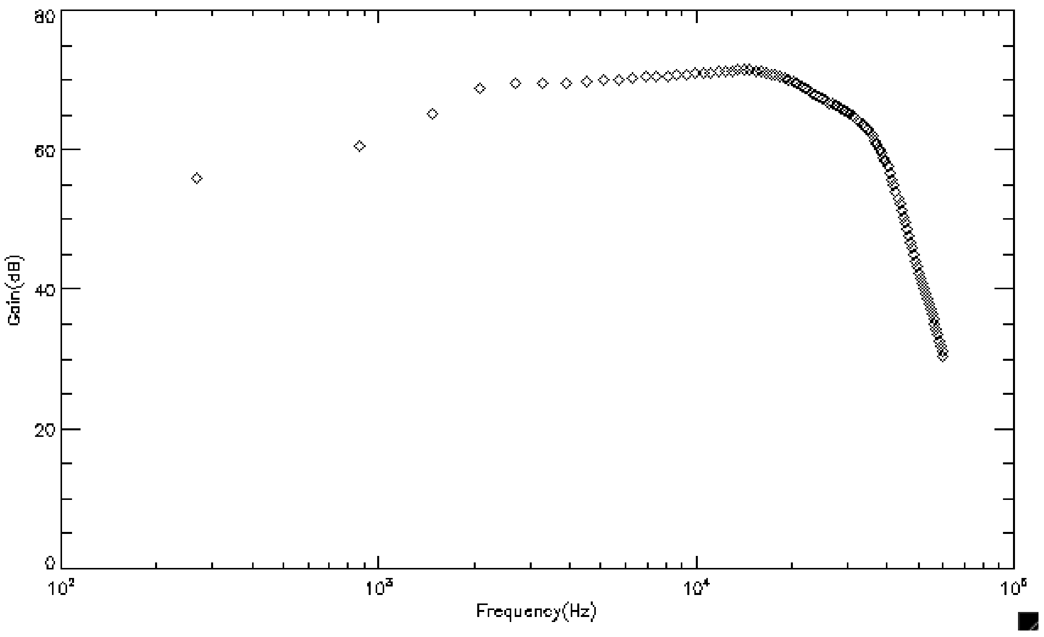


Figure 4.2: Audio Module Prototype frequency response

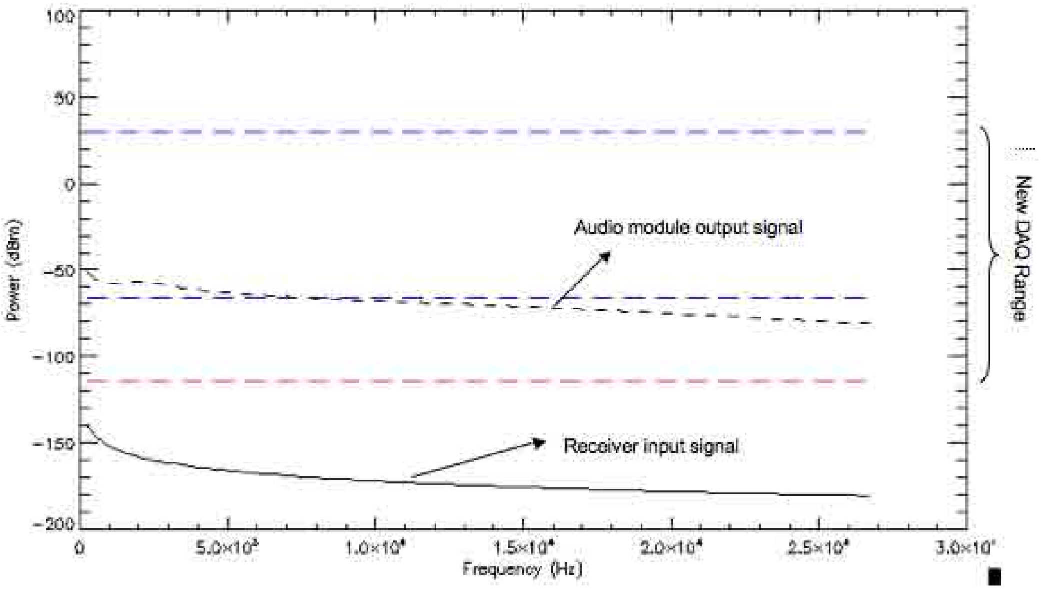


Figure 4.3: Theoretical Receiver Response with Audio Module Prototype.

a zoomed in picture of the filtered signal is showed, the FFT included on that plot also shows the 5KHz and 15KHz frequency components of the calibration signal.

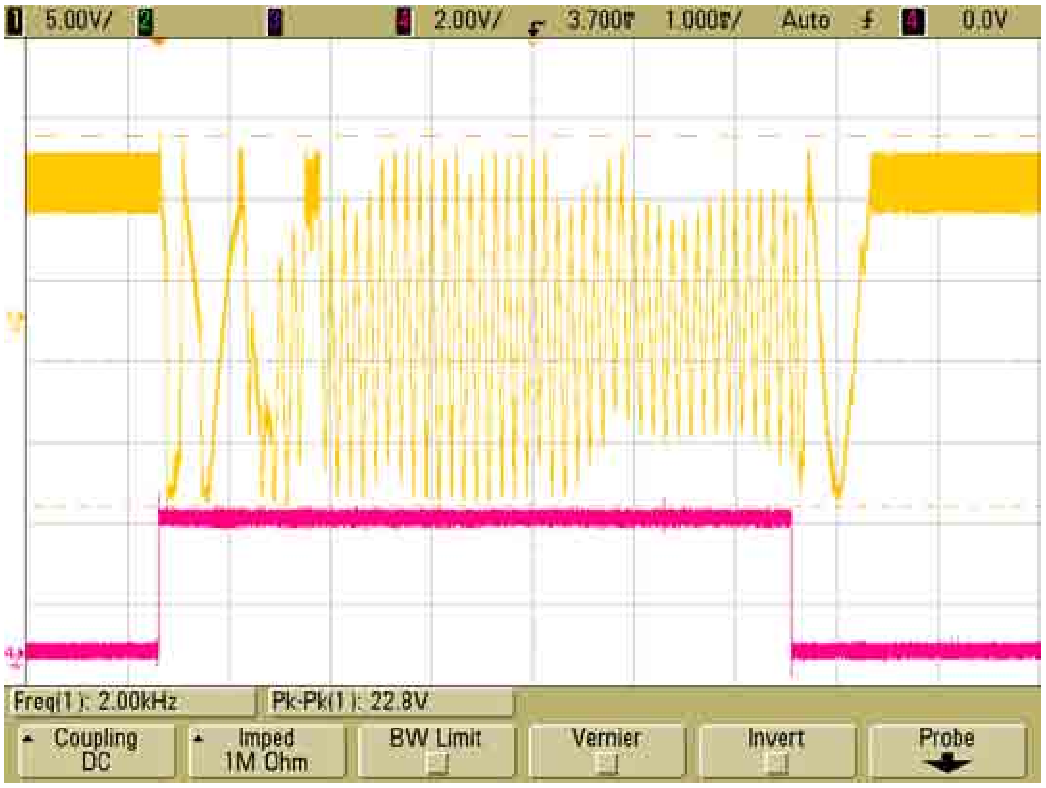


Figure 4.4: Audio module output of an entire chirp

4.2 FPGA Modification

4.2.1 DDS Description

The DDS AD9858 from Analog Devices provides an automated frequency sweeping capability. The frequency sweep is implemented through the use of a frequency accumulator, where a fixed incremental quantity is added repeatedly over time, creating new frequency tuning word, and so generating new frequencies. The first frequency is loaded in FTW register, the frequency increment, or step size, is loaded into the delta frequency tuning word (DTFW) register. The rate at which the frequency is incremented is set by the delta frequency ramp rate word (DFRRW) register. That registers enable the AD9858 to sweep from a beginning frequency set by FTW, upwards or downwards, at a desired rate and frequency step.

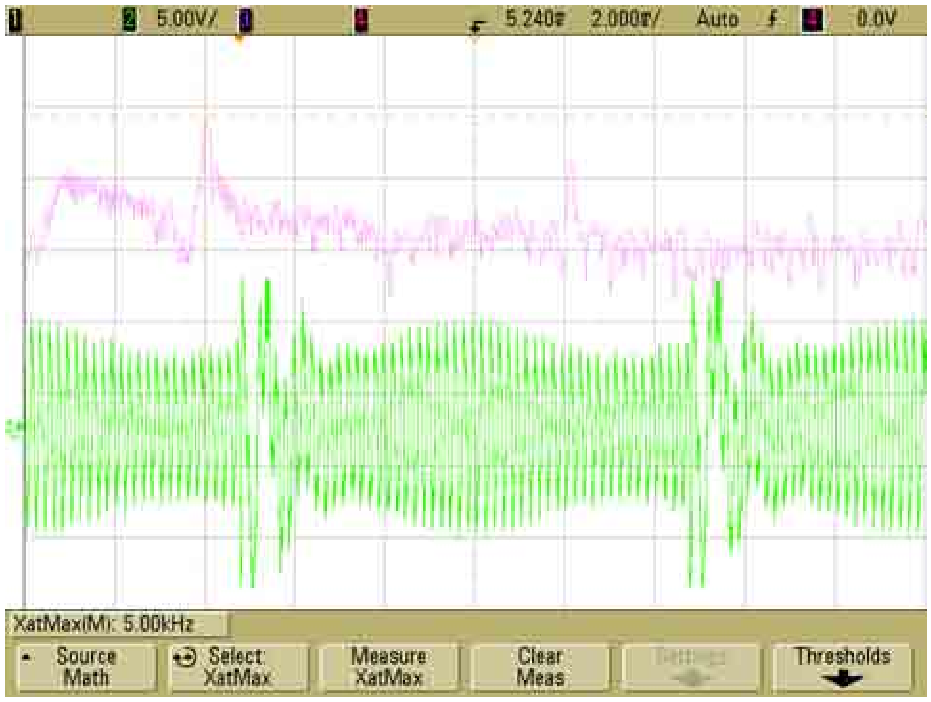


Figure 4.5: Audio module output of a entire chirp and FFT with 100% duty cycle chirp

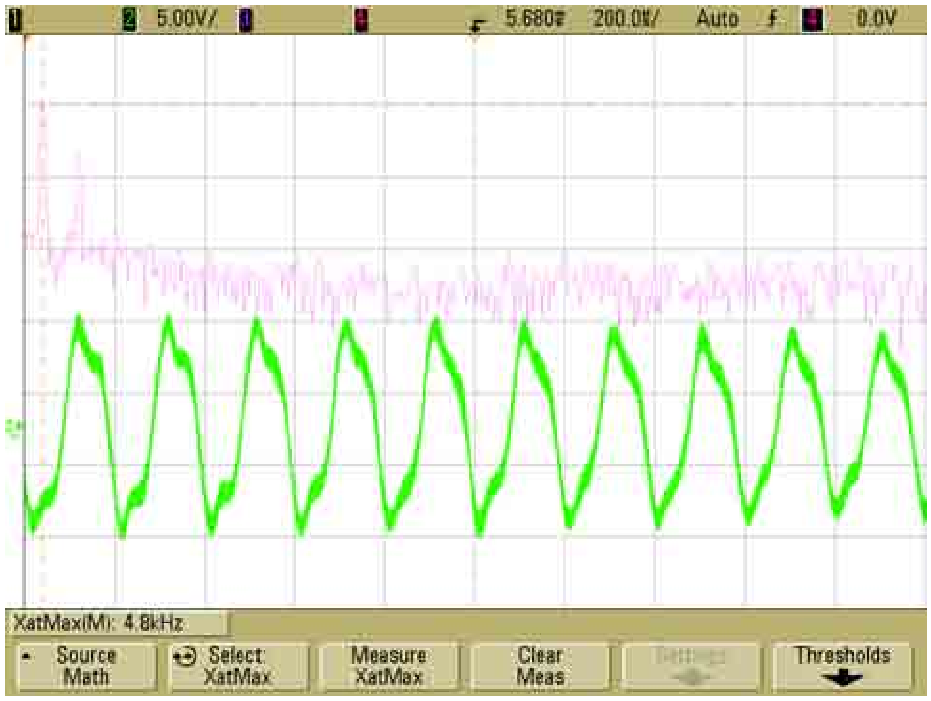
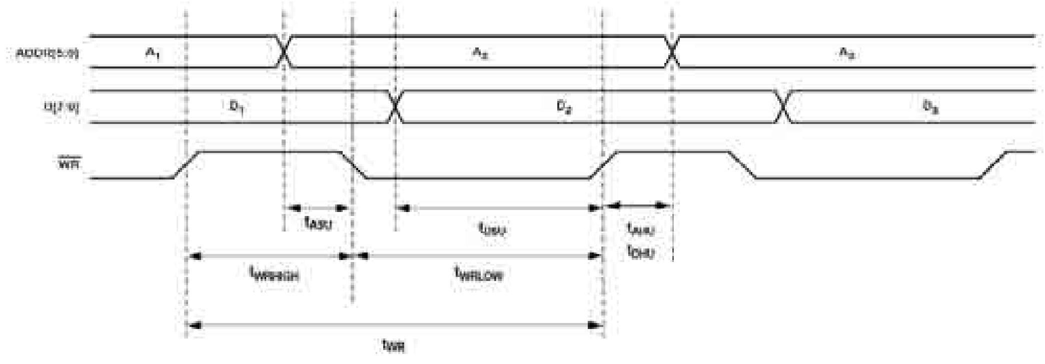


Figure 4.6: Audio module output and FFT with 100% duty cycle

The original frequency tuning word (FTW), written into the register, does not change over time. So it is possible to return to the initial frequency any time during a sweep. Clearing the frequency accumulator to 0, the DDS instantly returns to the frequency stored as FTW.

4.2.2 DDS Configuration Process

The communication with the DDS is a 2-step process in which data written to the I/O buffer and after it it is latched to the memory registers. To do so it is needed to toggle the FUD pins in order to update all the elements of the I/O buffer memory to the DDS's core memory registers. For the FMCW Wind Profiler parallel programming is used. In this mode the I/O port uses eight bidirectional data pins (D0 to D7), six address input pins (A0 to A5), a read input pin (RD*), and a write input pin (WR*). The write cycle is illustrated in Figure (4.7).



SPECIFICATION	VALUE	DESCRIPTION
t_{ASU}	8ns	ADDRESS SETUP TIME TO $\overline{WR}$ SIGNAL ACTIVE
t_{DSU}	3.5ns	DATA SETUP TIME TO $\overline{WR}$ SIGNAL INACTIVE
t_{AHU}	0ns	ADDRESS HOLD TIME TO $\overline{WR}$ SIGNAL INACTIVE
t_{DHU}	0ns	DATA HOLD TIME TO $\overline{WR}$ SIGNAL INACTIVE
t_{WRLOW}	3ns	$\overline{WR}$ SIGNAL MINIMUM LOW TIME
t_{WRHIGH}	6ns	$\overline{WR}$ SIGNAL MINIMUM HIGH TIME
t_{WR}	5ns	$\overline{WR}$ SIGNAL MINIMUM PERIOD

Figure 4.7: DDS write cycle timimg [from AD9858 manual]

4.2.3 FPGA Design

To be able to generate a 100% duty cycle, the FPGA needed to be modified. In the prior design, for every chirp the frequency accumulator of the DDS was cleared while the radar was not transmitting. Now to achieve the 100% duty cycle, the procedure used is different. Because now the "auto-clear frequency accumulator" and "auto-clear phase accumulator" will be used, see Table(4.1). Activating that bits, the frequency and phase accumulator will clear synchronously the accumulators for one cycle upon reception of the FUD sequence indicator, refer to [1] for more information. The phase accumulator is cleared too, in order to have a phase coherent signal, needed in order to can detect stationary targets. The DDS changes of phase are always continuous, but in order to have a coherent signal the phase accumulator is cleared too, Figure(4.8).

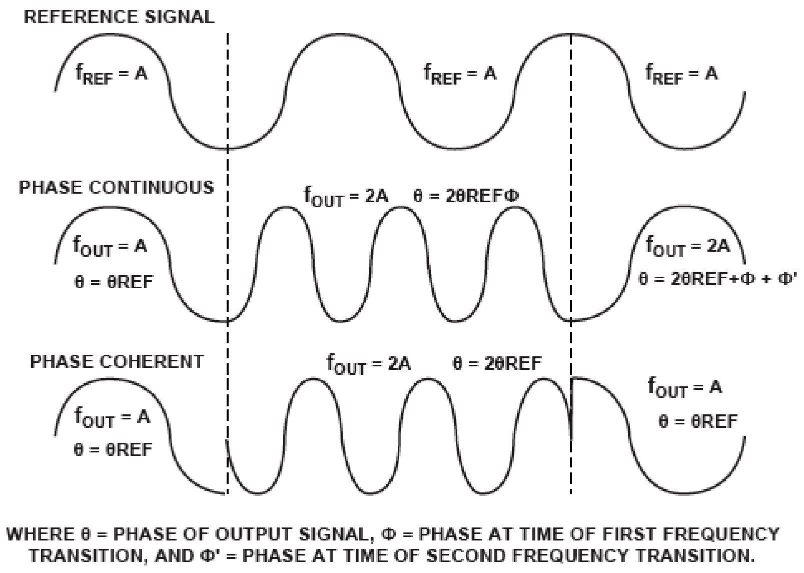


Figure 4.8: Difference between a Phase Continuous Frequency Change and a Phase Coherent Change

In addition the trigger signals were modified too, the DDS trigger needed to change drastically in order to can achieve the 100% goal. The DAQ trigger was adapted to the new board, having now the PRF signal as a trigger, because due to the 100% duty cycle the PRF signal is totally synchronized with the chirp now.

Finally after all that modifications on the FPGA, the c program that sends via RS-232 transceiver all the data needed for the FPGA in order to run the radar. For further information see appendix B.

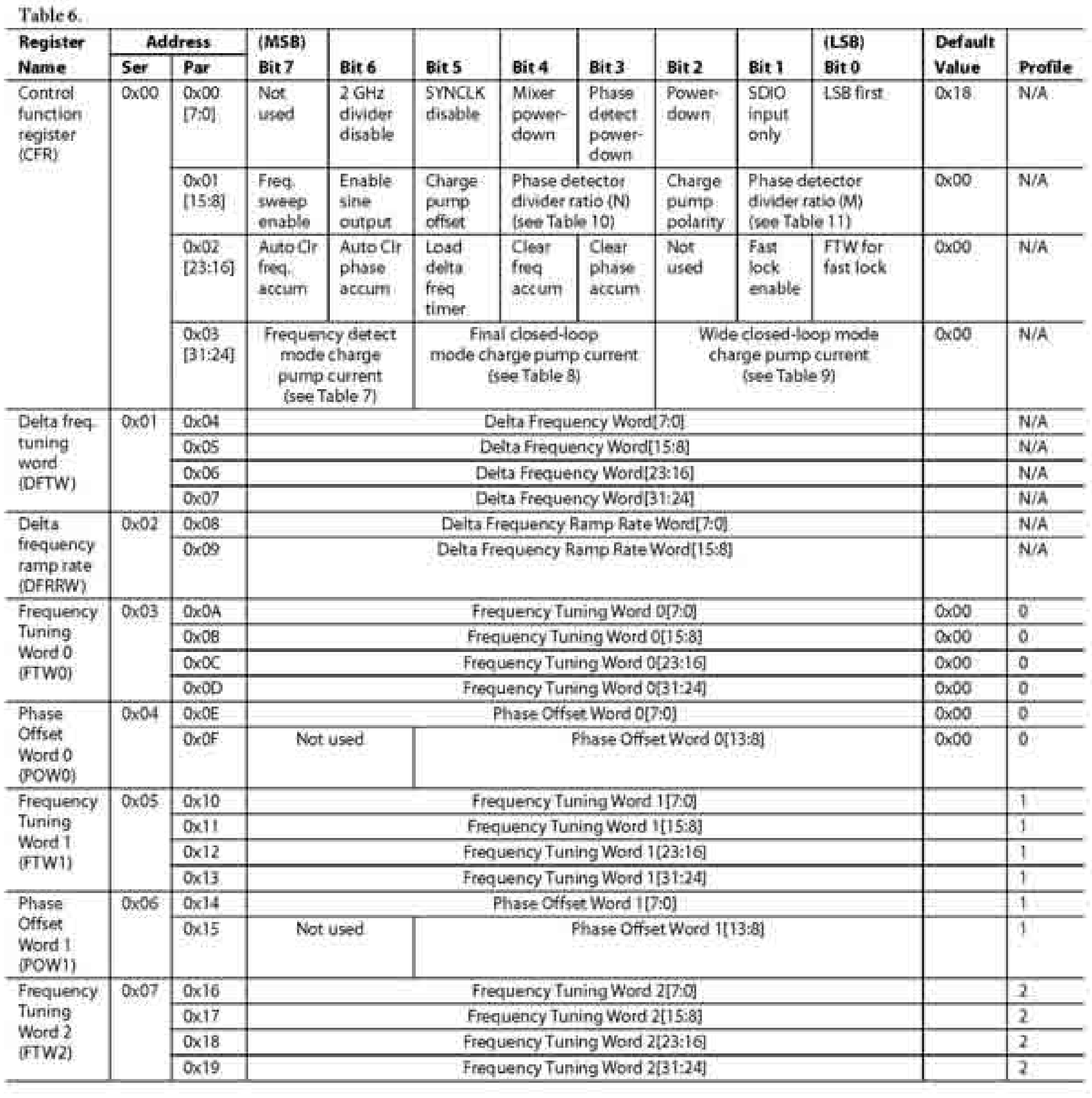


Table 6.

Register	Address		(MSB)							(LSB)	Default	
Name	**Ser**	**Par**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Value**	**Profile**
Control function register (CFR)	0x00	0x00 [7:0]	Not used	2 GHz divider disable	SYNCLK disable	Mixer power-down	Phase detect power-down	Power-down	SDIO input only	LSB first	0x18	N/A
		0x01 [15:8]	Freq. sweep enable	Enable sine output	Charge pump offset	Phase detector divider ratio (N) (see Table 10)		Charge pump polarity	Phase detector divider ratio (M) (see Table 11)		0x00	N/A
		0x02 [23:16]	Auto Clr freq. accum	Auto Clr phase accum	Load delta freq timer	Clear freq accum	Clear phase accum	Not used	Fast lock enable	FTW for fast lock	0x00	N/A
		0x03 [31:24]	Frequency detect mode charge pump current (see Table 7)		Final closed-loop mode charge pump current (see Table 8)			Wide closed-loop mode charge pump current (see Table 9)			0x00	N/A
Delta freq. tuning word (DFTW)	0x01	0x04	Delta Frequency Word[7:0]									N/A
		0x05	Delta Frequency Word[15:8]									N/A
		0x06	Delta Frequency Word[23:16]									N/A
		0x07	Delta Frequency Word[31:24]									N/A
Delta frequency ramp rate (DFRRW)	0x02	0x08	Delta Frequency Ramp Rate Word[7:0]									N/A
		0x09	Delta Frequency Ramp Rate Word[15:8]									N/A
Frequency Tuning Word 0 (FTW0)	0x03	0x0A	Frequency Tuning Word 0[7:0]								0x00	0
		0x0B	Frequency Tuning Word 0[15:8]								0x00	0
		0x0C	Frequency Tuning Word 0[23:16]								0x00	0
		0x0D	Frequency Tuning Word 0[31:24]								0x00	0
Phase Offset Word 0 (POW0)	0x04	0x0E	Phase Offset Word 0[7:0]								0x00	0
		0x0F	Not used		Phase Offset Word 0[13:8]						0x00	0
Frequency Tuning Word 1 (FTW1)	0x05	0x10	Frequency Tuning Word 1[7:0]									1
		0x11	Frequency Tuning Word 1[15:8]									1
		0x12	Frequency Tuning Word 1[23:16]									1
		0x13	Frequency Tuning Word 1[31:24]									1
Phase Offset Word 1 (POW1)	0x06	0x14	Phase Offset Word 1[7:0]									1
		0x15	Not used		Phase Offset Word 1[13:8]							1
Frequency Tuning Word 2 (FTW2)	0x07	0x16	Frequency Tuning Word 2[7:0]									2
		0x17	Frequency Tuning Word 2[15:8]									2
		0x18	Frequency Tuning Word 2[23:16]									2
		0x19	Frequency Tuning Word 2[31:24]									2

Table 4.1: DDS AD9858 register map (from AD9858 manual [1])

The results of the new FPGA can be observed in Figures(4.9)(4.10), where the spectrum of a chirp is depicted, and a received signal before the audio module is showed in the latter.

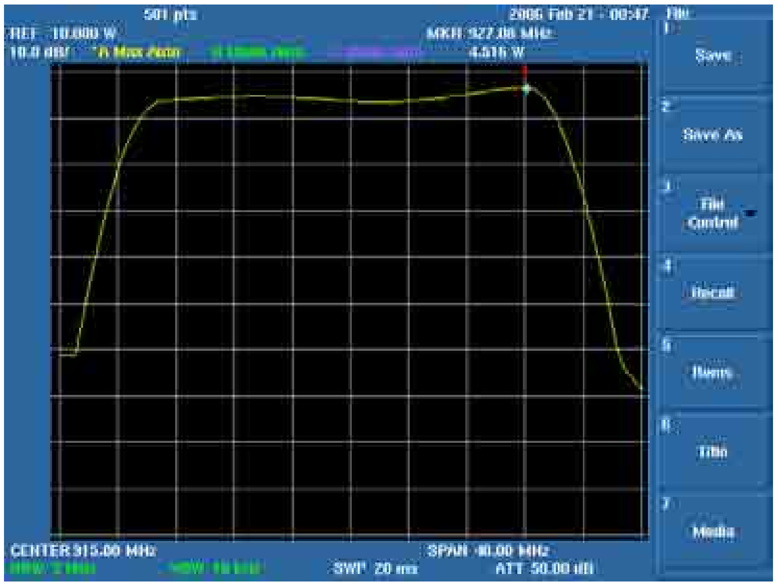


Figure 4.9: Spectrum of RF signal

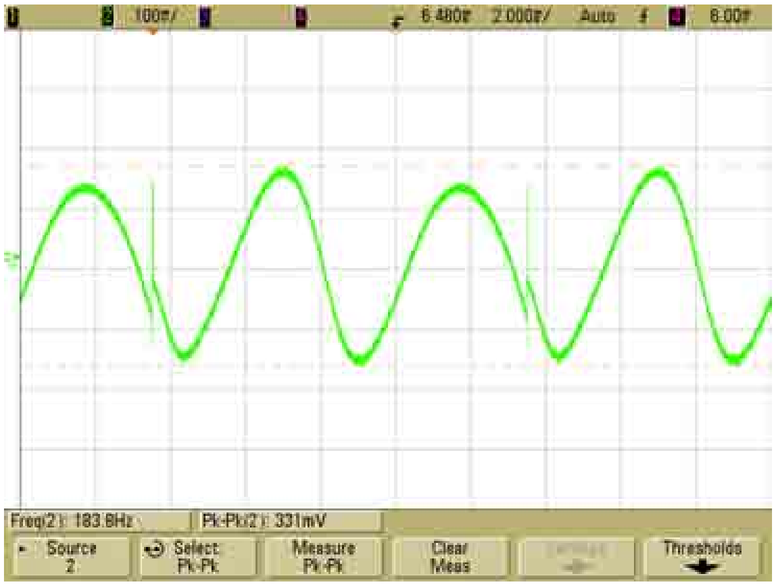


Figure 4.10: IF signal before audio module

CHAPTER 5

CONCLUSIONS

5.1 Summary

This thesis has described the FMCW Wind Profiler state as of Fall 2009. The previous configurations and problems have been explained and then solutions provided. Thanks to the previous work done with the radar, the sources of error can be identified and the radar receiver was upgraded in order to improve its performance.

The receiver front end was modified, and the leakage cancellation loop used before is now avoided. A new audio module was specifically designed. After decreasing the gain in the front end, and avoiding the use of the leakage cancellation loop, a high gain in the IF stage is needed. Simulations were performed and a prototype was built and tested on the radar.

Later on, the FPGA and .c programs were modified in order to achieve a 100% duty cycle chirp, which improves the performance of the receiver, due to the reduction of the transient response on the audio module or IF stage.

5.2 Future Work

The radar still needs some work in order to can be tested on field deployments.

The new data acquisition card 24DSI6LN from General Standards is being tested. A data acquisition program using that board needs to be done. That program have to acquire data synchronized with the transmitted signal, so Doppler velocities can be correctly detected.

Also a new data processing program needs to be implemented, adapted to new data obtained from the new data acquisition card. With all that radar tests and maybe upgrades to the radar can be done before field deployment tests.

After having checked the correct performance on the lab, real data should be acquired from field deployments. Processing the data collected the performance of the radar can be checked, and modifications can be added in case the data collected is not satisfactory.

In a second phase, it is planned to use a space antenna technique [9] to be able to analyze horizontal winds too. These techniques can obtain more rapid wind estimates compared to Doppler beam swinging systems.

APPENDIX A

BAND-PASS FILTER SPICE

```
    *Band_pass_filter_module.mod

    *********************************************

BAND_PASS_FILTER_MODULE.MOD

*********************************************

* Vcc 10 and 11
* Gnd 0
* Input 100
* Output 251

***Stage 1

Rg 100 101 50
RL11 101 102 1050
RL21 102 103 6810
CL11 103 0 1n
CL21 102 111 2.2n
```

```
RL41 111 104 0.1

XL21 103 104 10 11 111 LT1115


Vj1 111 201 0


CH11 201 202 22e-9

CH21 202 203 4.7e-9

RH11 203 0 9530

RH21 202 211 5360

RH41 211 204 0.1

XH21 203 204 10 11 211 LT1115


Vj2 211 311 0


*** Stage 2


RL12 311 312 1150

RL22 312 313 4220

CL12 313 0 3.3n

CL22 312 321 1n

RL42 321 314 90.9e3

RL32 314 0 10e3

XL22 313 314 10 11 321 LT1115


Vj3 321 411 0


CH12 411 412 22e-9
```

```
CH22 412 413 4.7e-9
RH12 413 0 10500
RH22 412 421 4870
RH42 421 414 0.1
XH22 413 414 10 11 421 LT1115

Vj4 421 121 0

***Stage 3

RL13 121 122 1240
RL23 122 123 3830
CL13 123 0 3.3n
CL23 122 131 1n
RL43 131 124 90.9e3
RL33 124 0 10e3
XL23 123 124 10 11 131 LT1115

Vj5 131 221 0

CH13 221 222 22e-9
CH23 222 223 4.7e-9
RH13 223 0 13300
RH23 222 231 3830
RH43 231 224 0.1
XH23 223 224 10 11 231 LT1115
```

```
Vj6 231 331 0

***Stage 4

RL14 331 332 1470
RL24 332 333 10700
CL14 333 0 1n
CL24 332 341 1n
RL44 341 334 60.4e3
RL34 334 0 10e3
XL24 333 334 10 11 341 LT1115

Vj7 431 341 0

CH14 431 432 22e-9
CH24 432 433 4.7e-9
RH14 433 0 20500
RH24 432 441 2430
RH44 441 434 0.1
XH24 433 434 10 11 441 LT1115

Vj8 441 141 0

*** Stage 5

RL15 141 142 2670
RL25 142 143 8060
```

```
CL15 143 0 1.5n

CL25 142 151 1n

RL45 151 144 43.2e3

RL35 144 0 10e3

XL25 143 144 10 11 151 LT1115


Vj9 151 241 0


CH15 241 242 22e-9

CH25 242 243 4.7e-9

RH15 243 0 59000

RH25 242 251 845

RH45 251 244 0.1

XH25 243 244 10 11 251 LT1115


RDAQ 251 0 1e6


vin 100 0 DC 0 AC 1 SIN(0,1,50e3)


*Decoupling capacitors.

CC1 10 0 1E-6

CC2 11 0 1E-6
```

```
    *Band_pass_filter_module.cir
****************************************************
BAND_PASS_FILTER_MODULE.CIR
****************************************************

*.options abstol=1n vntol=1m reltol=0.01
.include LT1115.lib
.include Band_pass_filter_module.mod

* Power sources

vcc1 10 0 15
vcc2 11 0 -15

*Simulation
*.AC LIN 100 266.667 26666.7
*.ac dec 300 200 50k
*.control
*run
*set
*let H=db(v(251))
*plot db(v(251))


************************ MC analysis, 90%************************

.control
```

```
* Illustrative example by CDHW of using SENS instead of monte carlo to
* investigate the effect of component tolerances.
*
ac dec 30 200 40e3
sens v(251) ac dec 30 200 40e3
* set the tolerances here:
let restol = 1e-2
let captol =15e-2


* set the number of standard deviations here
*
* 1.645 gives a 90% confidence band between lo_gain and hi_gain
*
let stdevs = 1.645
let sumsq = 0*v(ac.251)
set res= ( RL11 RL21 RL41 RH11 RH21 RH41 RL12 RL22 RL42 RL32 RH12 RH22
RH42 RL13 RL23 RL43 RL33 RH13 RH23 RH43 RL14 RL24 RL44 RL34 RH14
RH24 RH44 RL15 RL25 RL45 RL35 RH15 RH25 RH45)
set cap = ( CL11 CL21 CH11 CH21 CL12 CL22 CH12 CH22 CL13 CL23 CH13 CH23
CL14 CL24 CH14 CH24 CL15 CL25 CH15 CH25 )
foreach device $res
let thisDev = restol * sqrt(1/3) * abs($device) * @{$device}[resistance]
let sumsq = sumsq + thisDev*thisDev
end
foreach device $cap
let thisDev = captol * sqrt(1/3) * abs($device) * @{$device}[capacitance]
let sumsq = sumsq + thisDev*thisDev
```

```
end
let gain = db(v(ac.251)) - db(v(ac.100))
let lo_gain = db(abs(v(ac.251)) - stdevs * sqrt(sumsq) ) - db(v(ac.100))
let hi_gain = db(abs(v(ac.251)) + stdevs * sqrt(sumsq) ) - db(v(ac.100))
plot lo_gain hi_gain gain xlabel "Frequency" ylabel "Gain" ylimit -20 80

************************************************* Noise simulation!!!!!!!!!!!!!!

*.control
*run
* echo "At the start of this script, the current plot is '$curplot'."
* echo "Perform an 'op' analysis..."
*op
* echo " ... done. It created plot '$curplot' ($curplotname)."
* echo "Perform a noise analysis..."
*noise v(251) vin dec 100 100 100k 1
* echo " ... done. It created two plots and has set the current plot"
* echo " to the second of these '$curplot' ($curplotname)."
* echo "Use 'print all' to display the contents of the current plot:"
*print all
* echo "Next, use the 'destroy' command to remove the current plot ($curplot) ..."
*destroy
* echo " ... done. Now the current plot is '$curplot' ($curplotname),"
* echo " this is the first of the two plots created by the above 'noise' command."
* echo "Create a graph of v(onoise_spectrum) which is in '$curplot'..."
*plot v(onoise_spectrum)
* echo " ... done."
```

```
.endc

.END
```

APPENDIX B

FPGA MODIFIED MODULES CODE FILES

B.1 dds_run.tdf

TITLE "Init and run DDS in three modes";

–% *********************%

–% Wind Profiler Radar %

–% ********************* %

–% April 2008 written by D. Perkovic, perfected by David Garrido%

–% Filename: dds_run.tdf %

–% Description: This module inits and runs the DDS system: AD9858 chirp generator %

–% 1) Once the module is enabled, it sends the different parameters to the dds chip %

–% 2) After this initization, this module takes care to trigger the DDS system

– at the rate given by the incoming trigger (PRF) %

–

–% The outputs of this module match with the parallel interface of the AD9858

– evaluation board %

–

–% Parameters: %

–% INPUTS: %

```
–% clk: Clock frequency - active by rising edges (40 MHz)%
–% /enable: validates the data at the input and begins the DDS inialization process
%
–% if /enable =0 then the module is enabled otherwise it is OFF %
–% trigger: PRF signal %
–% DFTW[31..0] : DDS delta frequency register %
–% DFRRW[15..0] : DDS ramp rate register%
–% FTW0[31..0]: DDS starting frequency register%
–
–% OUTPUTS: %
–% address[5..0] - see AD9858 specs
– data[7..0] - see AD9858 specs
– /wr - see AD9858 specs active low write signal
– fud - see AD9858 specs %

SUBDESIGN dds_run
(
clk, enable :INPUT;
trigger :INPUT;
trigger_pulse :INPUT;
reset :INPUT;
DFTW[31..0] :INPUT;
DFRRW[15..0] :INPUT;
FTW0[31..0] :INPUT;
address[5..0] :OUTPUT;
data[7..0] :OUTPUT;
wr :OUTPUT;
```

```
fud :OUTPUT;
)

VARIABLE
state: MACHINE OF BITS (s[5..0])
WITH STATES (
xIDLE = B"000000",
xDFTW1 = B"000001",
xWR1 = B"000010",
xDFTW2 = B"000011",
xWR2 = B"000100",
xDFTW3 = B"000101",
xWR3 = B"000110",
xDFTW4 = B"000111",
xWR4 = B"001000",
xDFRRW1 = B"001001",
xWR5 = B"001010",
xDFRRW2 = B"001011",
xWR6 = B"001100",
xFTW01 = B"001101",
xWR7 = B"001110",
xFTW02 = B"001111",
xWR8 = B"010000",
xFTW03 = B"010001",
xWR9 = B"010010",
xFTW04 = B"010011",
xWR10 = B"010100",
xCFR0 = B"010101",
xWR11 = B"010110",
xCFR1 = B"010111",
xWR12 = B"011000",
xCFR2 = B"011001",
xWR13 = B"011010",
xCFR3 = B"011011",
xWR14 = B"011100",
xWAITTRIGUP = B"011101",
xRUN1 = B"011110",
xRUN = B"011111",
xWAITTRIGDOWN = B"100000",
xTRIGDOWN = B"100001");



regadd[5..0] :DFF;
regdata[7..0] :DFF;
wr_reg :DFF;
fud_reg :DFF;

–%DFTW[31..0] :NODE;
–DFRRW[15..0] :NODE;
–FTW0[31..0] :NODE;
–FTW1[31..0] :NODE;%
```

```
BEGIN

regadd[].(clk,clrn)=(clk,!enable);
regdata[].(clk,clrn)=(clk,!enable);
wr_reg.(clk,clrn)=(clk,!enable);
fud_reg.(clk,clrn)=(clk,!enable);

address[]=regadd[].q;
data[]=regdata[].q;
wr=!wr_reg.q or clk;
fud=fud_reg.q and clk;

state.(clk,reset)=(clk,enable or !reset);

CASE(state)IS

WHEN xIDLE =>
regadd[].d = H"4";
regdata[].d = DFTW[7..0];
wr_reg.d = gnd;
fud_reg.d = gnd;
if(enable or !reset) then
state = xIDLE;
else
state = xDFTW1;
end if;
```

```
WHEN xDFTW1 =>
regadd[].d = H"4";
regdata[].d = DFTW[7..0];
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR1;

WHEN xWR1 =>
regadd[].d = H"5";
regdata[].d = DFTW[15..8];
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xDFTW2;

WHEN xDFTW2 =>
regadd[].d = H"5";
regdata[].d = DFTW[15..8];
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR2;

WHEN xWR2 =>
regadd[].d = H"6";
regdata[].d = DFTW[23..16];
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xDFTW3;
```

```
WHEN xDFTW3 =>
regadd[].d = H"6";
regdata[].d = DFTW[23..16];
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR3;

WHEN xWR3 =>
regadd[].d = H"7";
regdata[].d = DFTW[31..24];
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xDFTW4;

WHEN xDFTW4 =>
regadd[].d = H"7";
regdata[].d = DFTW[31..24];
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR4;

WHEN xWR4 =>
regadd[].d = H"8";
regdata[].d = DFRRW[7..0];
wr_reg.d = gnd;
fud_reg.d = vcc;
```

```
state = xDFRRW1;

WHEN xDFRRW1 =>
regadd[].d = H"8";
regdata[].d = DFRRW[7..0];
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR5;

WHEN xWR5 =>
regadd[].d = H"9";
regdata[].d = DFRRW[15..8];
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xDFRRW2;

WHEN xDFRRW2 =>
regadd[].d = H"9";
regdata[].d = DFRRW[15..8];
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR6;

WHEN xWR6 =>
regadd[].d = H"A";
regdata[].d = FTW0[7..0];
wr_reg.d = gnd;
```

```
fud_reg.d = vcc;

state = xFTW01;


WHEN xFTW01 =>

regadd[].d = H"A";

regdata[].d = FTW0[7..0];

wr_reg.d = vcc;

fud_reg.d = gnd;

state = xWR7;


WHEN xWR7 =>

regadd[].d = H"B";

regdata[].d = FTW0[15..8];

wr_reg.d = gnd;

fud_reg.d = vcc;

state = xFTW02;


WHEN xFTW02 =>

regadd[].d = H"B";

regdata[].d = FTW0[15..8];

wr_reg.d = vcc;

fud_reg.d = gnd;

state = xWR8;


WHEN xWR8 =>

regadd[].d = H"C";

regdata[].d = FTW0[23..16];
```

```
wr_reg.d = gnd;

fud_reg.d = vcc;

state = xFTW03;


WHEN xFTW03 =>

regadd[].d = H"C";

regdata[].d = FTW0[23..16];

wr_reg.d = vcc;

fud_reg.d = gnd;

state = xWR9;


WHEN xWR9 =>

regadd[].d = H"D";

regdata[].d = FTW0[31..24];

wr_reg.d = gnd;

fud_reg.d = vcc;

state = xFTW04;


WHEN xFTW04 =>

regadd[].d = H"D";

regdata[].d = FTW0[31..24];

wr_reg.d = vcc;

fud_reg.d = gnd;

state = xWR10;


WHEN xWR10 =>

regadd[].d = H"0";
```

```
regdata[].d = H"78";
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xCFR0;

%End adding FTW0%

WHEN xCFR0 =>
regadd[].d = H"0";
regdata[].d = H"78";
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR11;

WHEN xWR11 =>
regadd[].d = H"1";
regdata[].d = H"80"; --H"80";
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xCFR1; ---state = xCFR0;

WHEN xCFR1 =>
regadd[].d = H"1";
regdata[].d = H"80";
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR12;
```

```
WHEN xWR12 =>
regadd[].d = H"2";
regdata[].d = H"0"; – H"80";
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xCFR2;

WHEN xCFR2 =>
regadd[].d = H"2";
regdata[].d = H"0";
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWR13;

WHEN xWR13 =>
regadd[].d = H"3";
regdata[].d = H"0";
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xCFR3;

WHEN xCFR3 =>
regadd[].d = H"3";
regdata[].d = H"0";
wr_reg.d = vcc;
fud_reg.d = gnd;
```

```
state = xWR14;

WHEN xWR14 =>
regadd[].d = H"2";
regdata[].d = H"C0"; % H"80"; Now clearing phase accum too
wr_reg.d = gnd;
fud_reg.d = vcc;
state = xWAITTRIGUP;

WHEN xWAITTRIGUP =>
wr_reg.d = gnd;
fud_reg.d = vcc;
regadd[].d = H"2";
regdata[].d = H"C0";
if(!trigger) then
state = xWAITTRIGUP;
else

regadd[].d = H"2";
regdata[].d = H"C0";
wr_reg.d = vcc;
fud_reg.d = gnd;
state = xWAITTRIGDOWN;
end if;
WHEN xWAITTRIGDOWN =>

regadd[].d = H"2";
```

```
regdata[].d = H"0";
wr_reg.d = gnd;
fud_reg.d = gnd;
if(trigger) then
state = xWAITTRIGDOWN;
else

regadd[].d = H"2";
regdata[].d = H"0";
wr_reg.d = gnd;
fud_reg.d = gnd;
state = xWAITTRIGUP;
end if;

end case;

end;
```

B.2 profiler_control2.tdf

```
TITLE "Init and run the Wind Profiler";

–% ********************** %
–% Wind Profiler radar %
–% ********************** %
–% March 2008 written by Dragana Perkovic, perfected by Joe McManus%
–% Modified by David - Mar 2009 %
–% Filename: profiler_control.tdf %
–% Description: This module inits and runs the radar %
–% 1) Once the module is enabled, it starts looping through the states to produce
the various signals %
–
–% Parameters: %
–% INPUTS: %
–% clk: Clock frequency - active by rising edges 10 MHz for now%
–% /enable: validates the data at the input and begins operation %
–% if /enable =0 then the module is enabled otherwise it is OFF %
–% WAIT0[15..0] - wait state 0 in no. of clk cycle of the FPGA clk%
–% WAIT1[15..0] - wait state 1 in no. of clk cycle of the FPGA clk%
–% WAIT2[15..0] - wait state 2 in no. of clk cycle of the FPGA clk%
–% WAIT3[15..0] - wait state 3 in no. of clk cycle of the FPGA clk%
–% WAIT4[15..0] - wait state 4 in no. of clk cycle of the FPGA clk%
–
–
–% OUTPUTS: %
–% switches[3..0] - PRF, DAQ trigger, AO trigger, DDS trigger%
–% state[3..0] - outputs the current state of this state machine for testing purposes.%

SUBDESIGN profiler_control2
(
clk :INPUT;
enable :INPUT;
WAIT0[15..0] :INPUT;
WAIT1[15..0] :INPUT;
```

```
WAIT2[15..0] :INPUT;
WAIT3[15..0] :INPUT;
WAIT4[15..0] :INPUT;
reset :INPUT;
DAQ_FREQ[6..0] :INPUT;
AO_FREQ[6..0] :INPUT;
state_out[2..0] :OUTPUT;
counter_value[15..0] :OUTPUT;
switches[3..0] :OUTPUT;
DAQ_clock :OUTPUT;
AO_clock :OUTPUT;
wait_out_AO[6..0] :OUTPUT;
)

VARIABLE
state: MACHINE OF BITS (s[2..0])
WITH STATES (
IDLE = B"000",
STATE_1 = B"001",
STATE_2 = B"010",
STATE_3 = B"011",
STATE_4 = B"100",
STATE_5 = B"101",
STATE_6 = B"110");

AO_state: MACHINE OF BITS (s1[1..0])
WITH STATES (
IDLE_AO = B”00”,
STATE_1_AO = B”01”,
STATE_2_AO = B”10”);

DAQ_state: MACHINE OF BITS (s2[1..0])
WITH STATES (
IDLE_DAQ = B”00”,
STATE_1_DAQ = B”01”,
STATE_2_DAQ = B”10”);

reg_switches[3..0] :DFF;
wait_count[15..0] :DFF;

reg_switches_DAQ :DFF;
wait_count_DAQ[6..0] :DFF;
reg_switches_AO :DFF;
wait_count_AO[6..0] :DFF;

BEGIN
reg_switches[].(clk,clrn)=(clk,!enable);
wait_count[].(clk,clrn)=(clk,!enable);
state.(clk,reset)=(clk,enable or !reset);
switches[]=reg_switches[].q;
state_out[]=s[];
counter_value[]=wait_count[].q;

reg_switches_DAQ.(clk,clrn)=(clk,!enable);
```

```
wait_count_DAQ[].(clk,clrn)=(clk,!enable);
DAQ_state.(clk,reset)=(clk,enable or !reset);
DAQ_clock=reg_switches_DAQ.q;

reg_switches_AO.(clk,clrn)=(clk,!enable);
wait_count_AO[].(clk,clrn)=(clk,!enable);
AO_state.(clk,reset)=(clk,enable or !reset);
AO_clock=reg_switches_AO.q;

wait_out_AO[]=wait_count_AO[].q;

CASE(state)IS
WHEN IDLE =>
reg_switches[].d = H"3";
wait_count[15..0].d = WAIT0[15..0];
if (enable or !reset) then state = IDLE;
else
state = STATE_1;
end if;

WHEN STATE_1 =>
reg_switches[].d = H"3";
if (wait_count[15..0].q == H"1") then
state = STATE_2;
wait_count[15..0].d = WAIT1[15..0];
else
wait_count[15..0].d = wait_count[15..0].q-1;
```

```
state = STATE_1;
end if;

WHEN STATE_2 =>
reg_switches[].d = H"B";
if (wait_count[15..0].q == H"1") then
state=STATE_3;
wait_count[15..0].d = WAIT2[15..0];
else
wait_count[15..0].d = wait_count[15..0].q-1;
state = STATE_2;
end if;

WHEN STATE_3 =>
reg_switches[].d = H"D";
if (wait_count[15..0].q == H"1") then
state=STATE_4;
wait_count[15..0].d = WAIT0[15..0];
else
wait_count[15..0].d = wait_count[15..0].q-1;
state = STATE_3;
end if;

WHEN STATE_4 =>
reg_switches[].d = H"A"; --H"2";
if (wait_count[15..0].q == H"1") then
state=STATE_5;
```

```
wait_count[15..0].d = WAIT1[15..0];
else
wait_count[15..0].d = wait_count[15..0].q-1;
state = STATE_4;
end if;

WHEN STATE_5 =>
reg_switches[].d = H"A";
if (wait_count[15..0].q == H"1") then
state = STATE_6;
wait_count[15..0].d = WAIT2[15..0];
else
wait_count[15..0].d = wait_count[15..0].q-1;
state = STATE_5;
end if;

WHEN STATE_6 =>
reg_switches[].d = H"C";
if (wait_count[15..0].q == H"1") then
state = STATE_1;
wait_count[15..0].d = WAIT0[15..0];
else
wait_count[15..0].d = wait_count[15..0].q-1;
state = STATE_6;
end if;
end case;
```

```
CASE(DAQ_state)IS
WHEN IDLE_DAQ =>
reg_switches_DAQ.d = B"0";
wait_count_DAQ[6..0].d = DAQ_FREQ[6..0];
if (enable or !reset) then DAQ_state = IDLE_DAQ;
else
DAQ_state = STATE_1_DAQ;
end if;

WHEN STATE_1_DAQ =>
reg_switches_DAQ.d = B"1";
if (wait_count_DAQ[6..0].q == H"1") then
DAQ_state = STATE_2_DAQ;
wait_count_DAQ[6..0].d = DAQ_FREQ[6..0];
else
wait_count_DAQ[6..0].d = wait_count_DAQ[6..0].q-1;
DAQ_state = STATE_1_DAQ;
end if;

WHEN STATE_2_DAQ =>
reg_switches_DAQ.d = B"0";
if (wait_count_DAQ[6..0].q == H"1") then
DAQ_state = STATE_1_DAQ;
wait_count_DAQ[6..0].d = DAQ_FREQ[6..0];
else
wait_count_DAQ[6..0].d = wait_count_DAQ[6..0].q-1;
DAQ_state = STATE_2_DAQ;
```

```
end if;

end case;

CASE(AO_state)IS
WHEN IDLE_AO =>
reg_switches_AO.d = B"0";
wait_count_AO[6..0].d = AO_FREQ[6..0];
if (enable or !reset) then AO_state = IDLE_AO;
else
AO_state = STATE_1_AO;
end if;

WHEN STATE_1_AO =>
reg_switches_AO.d = B"1";
if (wait_count_AO[6..0].q == H"1") then
AO_state = STATE_2_AO;
wait_count_AO[6..0].d = AO_FREQ[6..0];
else
wait_count_AO[6..0].d = wait_count_AO[6..0].q-1;
AO_state = STATE_1_AO;
end if;

WHEN STATE_2_AO =>
reg_switches_AO.d = B"0";
if (wait_count_AO[6..0].q == H"1") then
AO_state = STATE_1_AO;
```

```
wait_count_AO[6..0].d = AO_FREQ[6..0];
else
wait_count_AO[6..0].d = wait_count_AO[6..0].q-1;
AO_state = STATE_2_AO;
end if;

end case;

end;
```

APPENDIX C

CONTROL SUBSYSTEM .C PROGRAM

```
/* ————————————————————————————————
* C program for WIND PROFILER
* Read data from a parameter file and send it to serial port
* D. Garrido – Sep 2009
* (using "sonic" - S.J. Frasier - Sept 2000 & "trial_c_code" from Marine Radar)
*
* usage: windprofiler <file_name>
* ————————————————————————————————
*/

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <termios.h>
#include <sys/stat.h>
#include <stdint.h>
#include <stdlib.h>
#include <signal.h>
#include <math.h>
```

```
#include <errno.h>
#include <ctype.h>
#include <stddef.h>

#define FALSE 0
#define TRUE 1
#define COMM "/dev/ttyS0" /* default serial port */
#define NUM_PARAM 10

int main (int argc, char *argv[])
{
/* variable definition */
float num;
char datafile[40], commport[40], line[256], day[4], month[4], date[3], time[9], year[5],
var[5],c;  FILE *fp;

float params[NUM_PARAM];
char *sign_ptr;
int port,cntr_line,i;
struct termios termio;
size_t param_check;

double fpga_clk_freq,daq_freq, ao_freq, dds_cntr_freq,chirp_bw, dds_clk;
float txPulse, txPulse_numb,extra_pulse_width,prf, prf_numb, ddsStart, delay;

unsigned int DFTW, DFRRW,FTWO;
unsigned int WAIT0,WAIT1,WAIT2,WAIT3,WAIT4,WAIT5, WAIT6, WAIT7;
```

```
u_int16_t txPulse_reg, daq_freq_reg, numb_sampls_reg, ao_freq_reg ;
u_int8_t DFTW0, DFTW1, DFTW2, DFTW3, DFRRW0, DFRRW1, FTWO0, FTWO1,
FTWO2, FTWO3;
u_int8_t WAIT00_reg,WAIT01_reg,WAIT10_reg,WAIT11_reg,WAIT20_reg;
u_int8_t WAIT21_reg,WAIT30_reg,WAIT31_reg,WAIT40_reg,WAIT41_reg;
u_int8_t WAIT50_reg,WAIT51_reg, WAIT60_reg, WAIT61_reg, WAIT70_reg,WAIT71_reg;
u_int8_t txPulse_reg0, txPulse_reg1, daq_freq_reg0, daq_freq_reg1, ao_freq_reg0, ao_freq_reg1,
delay_reg;


/* use "wind_params.dh" as a default file */
if (argc == 1)
strcpy(datafile, "wind_params.dh");

/* use first argument in command line as parameter file */
if (argc == 2)
strcpy(datafile, argv[1]);

/* check for too many arguments */
if (argc > 2)
{
printf("Wrong number of arguments. n");
printf("Calling syntax: windprofiler <file_name> n");
return(FALSE);
}
```

```
/* open parameter file for reading only*/
if ((fp = fopen(datafile, ”r”)) == NULL)
{
printf(”Error opening %s n”, datafile);
fclose(fp);
close(port);
return(FALSE);
}

/* read from parameter file to obtain the Wind Profiler parameters */
for (cntr_line = 0; cntr_line < NUM_PARAM; cntr_line++)
{
/* lets get the lines */
if (fgets(line,sizeof line,fp) == NULL)
{
printf(”Error: File line cannot be read”);
fclose(fp);
return 0;
}

/* Searching for ’= sign in line */
if ((sign_ptr = strchr(line,’:’)) == NULL)
{
printf(”Error: Line improperly written. It is missing a ’: sign. n”);
fclose(fp);
return 0;
}

/* Reading in Parameters */
/*if(cntr_line == 0)
{
param_check = sscanf(sign_ptr+sizeof(char),"%s %s %s %s %s", day,month,date,time,year);
}
else
{*/
param_check = sscanf(sign_ptr+sizeof(char),"%s",var);
//}

if (param_check <= 0 )
{
printf("Error: Wind Profiler Radar parameter incorrectly set in %s:%d n",datafile,cntr_line);
fclose(fp);
return 0;
}

/*if(cntr_line > 0)
{*/
params[cntr_line]=atof(var);
//}
}

/* close parameter file */
fclose(fp);
```

```
printf("*************Entering FPGA programming routine*********** n");
/* copy parameter values to variables */
fpga_clk_freq = 10*1e6;
dds_cntr_freq = params[0]*1e6;
chirp_bw = params[1]*1e6;
txPulse = params[2]*1e-3;
prf = params[3];
ao_freq=params[5]*1e3;
daq_freq = params[4]*1e3;
delay=6*1e-6;


/* print out parameter values */
//printf("Day Month Date Time and Year: %s %s %s %s %s n",day,month,date,time,year);
printf("dds_cntr_freq: %f MHz n", dds_cntr_freq*1e-6);
printf("chirp_bw: %f MHz n", chirp_bw*1e-6);
printf("txPulse: %f ms n", txPulse*1e3);
printf("PRF: %f Hz n",prf);
printf("fpga_clk_freq: %f MHz n",fpga_clk_freq*1e-6);
printf("daq_freq: %f KHz n", daq_freq*1e-3);


/* calculate values to send */

ddsStart = dds_cntr_freq - (chirp_bw/2.0);
```

```
dds_clk = 8e8;

/* DFRRW Calculation */
DFRRW=2;// we are setting it to 1 for fastest update. Set to 0 if you want single
tone at starting f.

/* DFTW Calculation */
DFTW = (int)ceil((ldexp(chirp_bw*DFRRW*prf/(dds_clk*dds_clk),35)));
printf(" n DFTW calculated value: %d",DFTW);

/* FTWO Calculation */
FTWO = (int)ceil(ldexp(ddsStart/dds_clk,32));
printf(" n FTWO calculated value: %d",FTWO);



/* DFTW */
DFTW0 = DFTW & 0x00ff;
DFTW1 = (DFTW >> 8) & 0x00ff;
DFTW2 = (DFTW >> 16) & 0x00ff;
DFTW3 = (DFTW >> 24) & 0x00ff;
printf(" n DFTW0 calculated value: %d",DFTW & 0xff);
printf(" n DFTW1 calculated value: %d",(DFTW >> 8) & 0xff);
printf(" n DFTW2 calculated value: %d",(DFTW >> 16) & 0xff);
printf(" n DFTW3 calculated value: %d",(DFTW >> 24) & 0xff);

/* DFRRW */
DFRRW0 = (DFRRW & 0x00ff);
```

```
DFRRW1 = ((DFRRW >> 8) & 0x00ff);

/* FTWO */
FTWO0 = FTWO & 0x00ff;
FTWO1 = (FTWO >> 8) & 0x00ff;
FTWO2 = (FTWO >> 16) & 0x00ff;
FTWO3 = (FTWO >> 24) & 0x00ff;
printf(” n FTWO0 calculated value: %d”,FTWO & 0xff);
printf(” n FTWO1 calculated value: %d”,(FTWO >> 8) & 0xff);
printf(” n FTWO2 calculated value: %d”,(FTWO >> 16) & 0xff);
printf(” n FTWO3 calculated value: %d”,(FTWO >> 24) & 0xff);

/* DAQ Sampling Frequency (16 bits) */
daq_freq_reg = ceil(fpga_clk_freq/(daq_freq*2));//If we use David’s blocks to generate
the clocks
printf(” daq_freq_reg = %d n”,daq_freq_reg);
daq_freq_reg0 = daq_freq_reg & 0x00ff;
daq_freq_reg1 = (daq_freq_reg >> 8) & 0x00ff;

/* AO Sample frequency (16 bits) */
ao_freq_reg = ceil(fpga_clk_freq/(ao_freq*2)); //If we use David’s blocks to generate
the clocks
printf(” ao_freq_reg = %d n”,ao_freq_reg);
ao_freq_reg0 = ao_freq_reg & 0x00ff;
ao_freq_reg1 = (ao_freq_reg >> 8) & 0x00ff;

/* Calculate the delay between PRF and the trigger for the DDS*/
```

```
delay_reg = (unsigned int)round(delay*fpga_clk_freq) & 0xffff;
printf("The delay between PRF and the transmit pulse is: %d n",delay_reg);



/* Calculate the number of clock cycles in the PRF period*/


prf_numb = ceil(fpga_clk_freq/prf);
printf("The number of clock cycles in a PRF is: %f n", prf_numb);


/*Calculate the number of clock cycles in the txPulse width*/


txPulse_numb = ceil(fpga_clk_freq / prf);
printf("The number of cycles in the transmit pulse is: %f n",txPulse_numb);


/* Calculate the wait for each of the states and the DDS parameters*/


WAIT1=(unsigned int) round(prf_numb/4)-delay_reg;
WAIT00_reg = delay_reg & 0x00ff;//LSB
WAIT01_reg = (delay_reg >> 8) & 0x00ff;//MSB
printf("WAIT0 is: %d n",prf_numb/2);
//Define the first delay (between PRF and DDS Trigger) - WAIT1
printf("WAIT1 is: %d n",WAIT1);
WAIT10_reg = WAIT1 & 0x00ff; // LSB
WAIT11_reg = (WAIT1 >> 8) & 0x00ff; //MSB


//Define the 2nd delay
WAIT2= (unsigned int) round(prf_numb/4);
```

```
printf(”WAIT2 is: %d n”,WAIT2);
WAIT20_reg = WAIT2 & 0x00ff;//LSB
WAIT21_reg = (WAIT2 >> 8) & 0x00ff;//MSB

//Define the third delay
WAIT3 = WAIT2;
printf(”WAIT3 is: %d n”,WAIT3);
WAIT30_reg = WAIT3 & 0x00ff; //LSB
WAIT31_reg = (WAIT3 >> 8) & 0x00ff; //MSB

//Define the forth delay
WAIT4 = WAIT3;
printf(”WAIT4 is: %d n”,WAIT4);
WAIT40_reg = WAIT4 & 0x00ff; //LSB
WAIT41_reg = (WAIT4 >> 8) & 0x00ff; //MSB

//Define the fifth delay
WAIT5 = (unsigned int) round((prf_numb - (txPulse_numb + delay_reg))/2);
printf(”WAIT5 is: %d n”,WAIT5);
WAIT50_reg = WAIT5 & 0x00ff; //LSB
WAIT51_reg = (WAIT5 >> 8) & 0x00ff; //MSB

//Define the sixth delay -
WAIT60_reg = WAIT5 & 0x00ff;
WAIT61_reg = (WAIT5 >> 8) & 0x00ff;
printf(”WAIT6 is: %d n”,WAIT5);
//Define the seventh delay - it’s unused now
```

```
printf("WAIT7 is: %d n",WAIT3);

WAIT70_reg = WAIT3 & 0x00ff;

WAIT71_reg = (WAIT3 >> 8) & 0x00ff;


/* open serial port */

strcpy(commport, COMM);

//printf("Opening serial port %s n", commport);

if ((port = open(commport, O_WRONLY — O_SYNC)) < 1)

{

printf("Error opening %s n", commport);

return(0);

}


    /* set I/O baud rate to 9600 */
tcflush(port, TCIOFLUSH);
tcgetattr(port, &termio);
cfsetispeed(&termio, B9600);
cfsetospeed(&termio, B9600);
/* set serial port configuration */

termio.c_cflag —=(CLOCAL—CREAD);
termio.c_cflag &=  PARENB;
termio.c_cflag &=  CSTOPB;
termio.c_cflag &=  CSIZE;
termio.c_cflag —= CS8;
termio.c_lflag &=  (ICANON—ECHO—ECHOE—ISIG);
termio.c_iflag &=  INPCK;
termio.c_iflag —= IGNPAR;
termio.c_iflag &=  ISTRIP;
termio.c_iflag &=  (IXON—IXOFF—IXANY);
termio.c_oflag &=  OPOST;

if (tcsetattr(port, TCSANOW, &termio) == -1)
{
printf("Error setting port speed n");
close(port);
return(0);
}

/* 3 starting bytes necessary for the FPGA */
c=0x00;
write(port, &c, 1);
usleep(1000);
write(port, &c, 1);
usleep(1000);
write(port, &c, 1);
usleep(1000);

/* Parameters */
c=0x01;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT00_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT01_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x02;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT10_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT11_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x03;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT20_reg;
write(port, &c, sizeof(c));
usleep(1000);
```

```
c=(char) WAIT21_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x04;

write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT30_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT31_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x05;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT40_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT41_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x06;
write(port, &c, sizeof(c));
c=(char) WAIT50_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT51_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);
```

```
c=0x07;
write(port, &c, sizeof(c));
c=(char) WAIT60_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT61_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x08;
write(port, &c, sizeof(c));
c=(char) WAIT70_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) WAIT71_reg;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x09;
//c=0x09;
write(port, &c, sizeof(c));
c=(char) DFTW0;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) DFTW1;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x0a;
//c=0x0a;
write(port, &c, sizeof(c));
c=(char) DFTW2;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) DFTW3;
```

```
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x0b;
//c=0x0b;
write(port, &c, sizeof(c));
c=(char) DFRRW0;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) DFRRW1;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x0c;
//c=0x0c;
write(port, &c, sizeof(c));
c=(char) FTWO0;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) FTWO1;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x0d;
//c=0x0d;
write(port, &c, sizeof(c));
c=(char) FTWO2;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) FTWO3;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);
```

```
c=0x0e;
write(port, &c, sizeof(c));
c=(char) daq_freq_reg0;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) daq_freq_reg1;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

c=0x0f;
write(port, &c, sizeof(c));
c=(char) ao_freq_reg0;
write(port, &c, sizeof(c));
usleep(1000);
c=(char) ao_freq_reg1;
write(port, &c, sizeof(c));
usleep(1000);
c=0x00;
write(port, &c, sizeof(c));
usleep(1000);

/* control byte meaning end of parameters this will start indicate the end of writing
to memory from serial
port and the availability of data for reading */
c=0xfe;
//write(port, &c, sizeof(c));
write(port, &c, sizeof(c));
usleep(1000);
/* close serial port */
close(port);

return(TRUE);
}
```

BIBLIOGRAPHY

[1] Analog Devices“1 GSPS Direct Digital Synthesizer, AD9858”. Datasheet.

[2] David Atlas.“Radar in Metrology”. American Meteorological Society, Boston, 1990.

[3] T.Deliyannis.“Continous time active filter design” CRC Press, 1999.

[4] Earl E. Gossard.“Refractive index variance and its height distribution in different air masses”. Radio Science 12, 1977.

[5] Iva Kostadinova.“An UHF Frequency-Modulated Continous Wave Wind Profiler - Developement and Initial Results”. Master’s Thesis, University of Massachusetts at Amherst.

[6] Liam E. Gumley.“practical IDL programming”. Morgan Kaufmann, San Diego, 2002.

[7] Linear Technology“Ultralow Noise, Low Distortion, Audio Op Amp, LT1115”. Datasheet.

[8] R. Mancini. “Op Amps for Everyone”. Texas Instruments, Houston. TX, 2002.

[9] Muschinski, A., S.J. Frasier, “Supplement to Investigation of Turbulence and Intermittency in the Convective Boundary Layer Using a 915MHz Volume Wind Profiler’: Profiler Fabrication”, 2006

[10] Ottersen H.“Atmospheric structure and radar backscattering in clear air”. Radio Science 4, 12 (1969), 1179-1193.

[11] David M. Pozar.“Microwave and RF Design of Wireless Systems”. John Wiley & Sons, 2001.

[12] Turker, Frasier, S., Muschinski, A., and Pazmany, A. An s-band fmcw boundary layer profiler: Description and initial results. “Radio Science 38”, 4 (2003).

[13] Andrei Vladimirescu“The spice book”. John Wiley & Sons, Inc., 1994