

HARDWARE IMPLEMENTATION OF  
A FULLY TUNABLE HETERODYNE  
NOTCH FILTER IN FPGA

By

DHINESH SASIDARAN

Bachelor of Science

Oklahoma State University

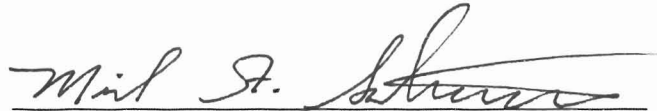
Stillwater, Oklahoma

1999

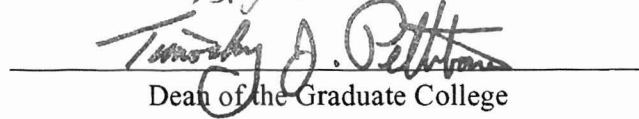
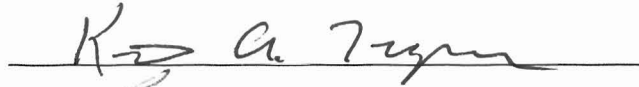
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
May, 2002.

HARDWARE IMPLEMENTATION OF  
A FULLY TUNABLE HETERODYNE  
NOTCH FILTER IN FPGA

Thesis Approved:



Thesis Adviser



Dean of the Graduate College

# Preface

Wireless and satellite communication systems often use spread spectrum techniques to modulate the narrowband information-bearing signal before transmission. This technique masks the transmitted signal as noise by spreading the narrowband signal, thereby preventing the effective use of jamming and interception techniques in interfering with the transmission. However, during such a transmission, a narrowband interference signal from nearby transmitters could find its way onto the channel of the desired signal. The existence of this interference signal on the channel of the desired signal could ultimately cause detection errors and anomalies at the receiver end. These detection errors and anomalies appear when the receiver latches onto the undesired signal instead of the desired signal.

In this thesis, a novel approach is taken to remove the unwanted interference frequency from the channel. A tunable filter system is used to remove the undesired signal at the receiver end of the transmission before the desired signal is extracted and used for any given application.

The entire tunable system actually employs the use of fixed filters in its lower-level structure to remove the narrowband interference signal, but the system itself is completely tunable. This continuously tunable property allows any frequency from 0 to DC to be attenuated by tuning a single parameter, which is the heterodyning frequency.

Since the underlying basis for this tunable filter is the use of the heterodyning concept, it can be shown also that all images created by the heterodyne (mixing) process are eventually cancelled without the use of any additional filters.

This thesis concentrates on the digital hardware implementation of the proposed filter structure. Since a digital version of the proposed filter has never been implemented, this thesis will put forth the idea of achieving a working-structure with minimal hardware of the filter to remove narrowband interferences from a channel. The platform for this implementation of the proposed structure is Field Programmable Gate Arrays (FPGAs) which provides a suitable hardware structure for rapid prototyping.

Although the tunable filter system presented in this thesis is implemented as a manually tunable structure, an adaptive version could easily be built by employing standard adaptive algorithms at the cost of incurring some additional hardware.



## **Dedication**

To My Parents, Dr Michael Soderstrand, Dr Louis Johnson  
and Regina Henry for all their support, encouragement and  
friendship.

## ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my research advisor, Dr. Michael A. Soderstrand for his intelligent supervision, constructive guidance, insight and friendship. His incomparable patience and confidence has guided me through my graduate study and provided me with a valuable experience and knowledge. I also wish to extend my most sincere appreciation and gratitude to my committee member Dr. Louis G. Johnson whose guidance, assistance, encouragement, and friendship are also invaluable. I would also like to thank committee member Dr. Keith Teague for carefully reviewing my thesis and giving me his precious advice.

I am thankful to all my research colleague in the Digital Signal Processing and Communication (DSP&C) Lab for their friendship and time and not to mention assisting and helping me on numerous occasions.

I cannot help but extend my eternal gratitude to Ms. Regina Henry at the International Students and Scholars office for being an invaluable friend, mother and advisor. I would like to thank all my close friends for providing me with love, encouragement and support during the period of my studies.

As a final note, I would like to acknowledge my family for all their patience and support. I would like to thank my parents who have provided me with constant support, love and encouragement through their prayers and love.

# Table of Contents

<b>CHAPTER 1 Introduction</b> .....	1
1.1 Broadband communications .....	1
1.2 Problem Statement .....	2
1.2.1 Narrowband Interferences (NBI) .....	2
1.3 Scope of Thesis .....	4
1.4 Thesis Outline .....	5
<b>CHAPTER 2 Background</b> .....	7
2.1 Spread Spectrum .....	7
2.1.1 Interference Suppression in spread Spectrum communications .....	7
2.1.1.1 Direct Sequence .....	8
2.1.1.2 Frequency Hopping .....	10
2.1.1.3 Signal Recovery .....	10
<b>CHAPTER 3 The Heterodyne Process</b> .....	11
3.1 Frequency translation .....	11
2.1.1 The heterodyne concept .....	11
3.1.2 Superheterodyne receiver .....	13
<b>CHAPTER 4 Previous Work</b> .....	15
4.1 Tunable Bandpass Filter .....	15
4.1.1 Theory of operation .....	16
4.2 Final Analysis .....	19
4.2.1 Advantages .....	19
4.2.2 Disadvantages .....	19
<b>CHAPTER 5 FTHN Filter System</b> .....	22
5.1 Splitter .....	22
5.2 Fixed-coefficient prototype filter .....	24
5.2.1 Why implement a notch filter? .....	24
5.2.2 Filter selection .....	25
5.3 Combiner .....	28
5.4 The Complete FTHN Filter System .....	29
5.4.1 Theory of Operation .....	30
<b>CHAPTER 6 MatLab Simulations</b> .....	37
6.1 Filter translation .....	47
<b>CHAPTER 7 Hardware Choice</b> .....	53
7.1 Field Programmable Gate Arrays (FPGA) .....	53

7.2	Xilinx XCV800 FPGA Structure	54
7.3	Configurable Logic Blocks (CLBs)	55
7.3.1	Function Generator/Look-Up table (LUT)	55
7.3.2	The Arithmetic Logic	55
7.3.3	Storage Elements	56
7.4	Input/Output Blocks	57
7.5	Block SelectRAM	57
7.6	CLB Configuration	58
<b>CHAPTER 8 Hardware Implementation</b>		<b>59</b>
8.1	Sine/Cosine Genrator	60
8.2	8-bit 2's Complement Multiplier	62
8.3	Splitter	64
8.4	Fixed-coefficient Prototype Filter	66
8.5	Combiner	69
8.6	Final Structure	70
8.7	Resource Usage	71
<b>CHAPTER 9 Final Results</b>		<b>72</b>
9.1	Experimental Results	72
9.1.1	Experimental Setup	72
<b>CHAPTER 10 Future Work</b>		<b>91</b>
10.1	Adaptive Heterodyne Notch Filter	91
10.2	Cordic Algorithm	93
10.3	Multiple-source Interferences	94
<b>Bibliography</b>		<b>95</b>
<b>Appendix A</b>		<b>98</b>

# List of Figures

<b>CHAPTER 1 Introduction .....</b>	<b>1</b>
<b>CHAPTER 2 Background.....</b>	<b>7</b>
<b>CHAPTER 3 The Heterodyne Process .....</b>	<b>11</b>
Figure 3-1. Mixer block diagram .....	12
Figure 3-2. Superheterodyne receiver block diagram .....	13
<b>CHAPTER 4 Previous Work.....</b>	<b>15</b>
Figure 4-1. Tunable Heterodyne Bandpass Filter Unit. ....	15
Figure 4-1. Tunable Bandpass Filter operation.....	16
<b>CHAPTER 5 FTHNF Filter System.....</b>	<b>23</b>
Figure 5-1. Splitter circuit .....	24
Figure 5-2. Virtual translation in frequency of a high-pass filter. ....	26
Figure 5-3. Direct Form II implementation of a first-order high-pass filter .....	27
Figure 5-4. Combiner circuit.....	28
Figure 5-5. Fully Tunable Heterodyne Notch Filter .....	29
Figure 5-6. Fully Tunable Heterodyne Notch Filter operation (part I).....	30
Figure 5-7. Fully Tunable Heterodyne Notch Filter operation (part II).....	31
Figure 5-8. Fully Tunable Heterodyne Notch Filter operation (part III) .....	32
<b>CHAPTER 6 MatLab Simulations.....</b>	<b>37</b>
Figure 6-1. Stages of the FTHN Filter structure .....	37
Figure 6-2. Block diagram of Splitter stage 1 .....	38
Figure 6-3. Time and frequency domain representation of the input signal .....	38
Figure 6-4. Normalized Power spectrum plot of cosout1 and sinout1.....	39
Figure 6-5. Block diagram of Filter Stage 1 .....	40
Figure 6-6. Magnitude response of IIR high-pass filter with 3dB cutoff frequency of 500Hz .....	40
Figure 6-7. Normalized power spectrum plot of signals cosout2 and sinout2.....	41
Figure 6-8. Block diagram of Splitter Stage 2 .....	41
Figure 6-9. Normalized power spectrum plot of signals cossin1 and coscos1 .....	42
Figure 6-10. Normalized power spectrum plot of signals sinsin1 and sincos1.....	42
Figure 6-11. Block diagram of Intermediate Stage.....	43
Figure 6-12. Normalized power spectrum plot of signals sum1(=sum4) and sum2 (=sum3) .....	44
Figure 6-13. Block diagram of Combiner stage 1.....	44
Figure 6-14. Normalized Power spectrum plot of signals sumreal and sumimaj .....	45
Figure 6-15. Block diagram of Filter stage 2 .....	45
Figure 6-16. Normalized power spectrum plot of signals in1 and in2.....	46
Figure 6-17. Block diagram of Combiner stage 2.....	46
Figure 6-18. Normalized power spectrum plot of the output signal .....	47
Figure 6-19. Frequency response of 1st order high-pass filter with 3dB cutoff at 500Hz.....	48
Figure 6-20. Pole-zero plot of original IIR high-pass filter prior to translation .....	48
Figure 6-21. Filter up-translated by 3kHz.....	49

Figure 6-22. Pole-zero plot of high-pass filter up-translated in frequency .....	49
Figure 6-23. Frequency response of high-pass filter downtranslated by 3kHz .....	50
Figure 6-24. Pole-zero plot of filter down-translated by 3kHz.....	50
Figure 6-25. Final position of translated filter .....	51
Figure 6-26. Final pole and zero positions of translated filter .....	51
<b>CHAPTER 7 Hardware Choice.....</b>	<b>53</b>
Figure 7-1. Slice CLB .....	56
Figure 7-2. Detailed Virtex Slice .....	56
Figure 7-3. Virtex Input/Output (I/O) Block .....	57
<b>CHAPTER 8 Hardware Implementation .....</b>	<b>59</b>
Figure 8-1. Sine/Cosine generator .....	61
Figure 8-2. Synthesized sine/cosine generator circuit using Synplify Pro .....	62
Figure 8-3. 8-bit 2's complement multiplier block diagram .....	63
Figure 8-4. 8-bit, 2's complement multiplier implementation using Synplify Pro .....	64
Figure 8-5. Block diagram of the Splitter circuit .....	65
Figure 8-6. Synthesized Splitter circuit using Synplify Pro.....	66
Figure 8-7. Block diagram of 1st-order high-pass filter .....	68
Figure 8-8. Synthesized 1st order high-pass filter using Synplify Pro .....	69
Figure 8-9. Block diagram of Combiner .....	69
Figure 8-10. Synthesized structure of Combiner using Synplify Pro .....	70
Figure 8-11. Final synthesized structure of FTHN filter using Synplify Pro .....	70
<b>CHAPTER 9 Final Results.....</b>	<b>72</b>
Figure 9-1. Interface between codec and FPGA on the XSV-800 V1.1 board.....	73
Figure 9-2. 512 point, normalized Power spectrum plot of the input signal.....	74
Figure 9-3. 512 point, normalized Power spectrum plot of the heterodyning signal.....	75
Figure 9-4. 512 point, normalized Power spectrum plot of signal sinout1 .....	76
Figure 9-5. 512 point, normalized Power spectrum plot of signal cosout1 .....	77
Figure 9-6. 512 point, normalized Power spectrum plot of signal sinout2.....	78
Figure 9-7. 512 point, normalized Power spectrum plot of signal cosout2 .....	79
Figure 9-8. 512 point, normaized Power spectrum plot of signal cossin1 .....	80
Figure 9-9. 512 point, normaized Power spectrum plot of signal coscos1 .....	81
Figure 9-10. 512 point, normaized Power spectrum plot of signal sincos1 .....	82
Figure 9-11. 512 point, normaized Power spectrum plot of signal sinsin1 .....	83
Figure 9-12. 512 point, normaized Power spectrum plot of signal sum1 .....	84
Figure 9-13. 512 point, normaized Power spectrum plot of signal sum2 .....	85
Figure 9-14. 512 point, normaized Power spectrum plot of signal sumreal .....	86
Figure 9-15. 512 point, normaized Power spectrum plot of signal sumimaj .....	87
Figure 9-16. 512 point, normaized Power spectrum plot of signal in1 .....	88
Figure 9-17. 512 point, normaized Power spectrum plot of signal in2 .....	89
Figure 9-18. 512 point, normaized Power spectrum plot of signal output.....	90
<b>CHAPTER 10 Future Work .....</b>	<b>91</b>
Figure 10-1. Basic block diagram of second-order adaptive notch filter.....	92
Figure 10-2. Complete Adaptive Heterodyne Notch Filter.....	93

# List of Tables

<b>CHAPTER 8 Hardware Implementation .....</b>	<b>61</b>
Table 8-1. Resource usage comparison for sine/cosine generator and multiplier .....	73
Table 8-2. Resource usage comparison for FTHN filter components .....	73
Table 8-3. Resource usage comparison for entire structure .....	73

# 1 Introduction

## 1.1 Broadband communications

The importance of Digital Signal Processing (DSP) has been greatly exhibited over the years. The existence of DSP techniques and algorithms have opened up a plethora of inventions and advancements in the field of communications and in almost every aspect of engineering. In telecommunications, namely broadband communications, DSP algorithms have enhanced the performance factor, reliability and reputation of the broadband communications universe.

Broadband communications or sometimes known as wideband communications, can be described as the transmission of narrow band-limited information over a wide range of available frequencies in an assigned bandwidth. This form of telecommunication has a distinct advantage of allowing several different types of information to be multiplexed and sent to different frequencies within the wideband concurrently.

Previously, broadband communications were strictly confined in the analog domain where analog signals were transmitted and received from one end to the other. The transmission and reception of these signals were done using discrete analog devices which were considered primitive. However, modern developments in technology has allowed digital information to be transmitted within the confines of the analog environment. With the properties of data compression and manipulation available solely in the digital domain, the capability of



transmission has increased with the resultant increase in the number of channels available in a given frequency band. This rise in efficiency has given digital data transmission a slight edge over analog data transmission.

Even with these advancements, several issues still plague modern communication systems, primarily the existence of narrowband interferences in broadband Binary Phase Shift Keying (BPSK) and Quadri-Phase Shift Keying (QPSK) signals. These interferences will need to be addressed and eliminated from the desired signal.

## **1.2 The Problem Statement**

One of the most common problems in controls and communications applications is the detection of a sinusoid. The sinusoid signals may contain important information to be transmitted or may be of harmful nature whereby it is classified as a narrowband interference (NBI) signal. Also, these sinusoids are not constant with respect to time and therefore detection of these signals for extraction or attenuation can be a daunting task. In cases where the sinusoid is harmful and lying adjacent to a desired signal, we will need to attenuate these signals without affecting any nearby desired signals [1].

### **1.2.1 Narrowband Interferences (NBI)**

Narrowband interferences (NBIs) from external or internal noise sources are often strong enough to ‘drive’ down and not to mention corrupt the critical information carried by nearby adjacent signals. These interference signals, aptly called “adjacent channel” interferences, should be removed or greatly attenuated in order to preserve an intelligible amount of information that is being transmitted from one end to the other [3]. Examples of well

known external noise sources or Intersystem Interferences (ISI) are radio-frequency transmitters or certain weather conditions such as lightning; both of which produce radio-frequency interference (RFI) while resonance in a sensor system used in a control systems is a good example of an internal noise source [1][3]. These high power signals adjacent to the wanted signals can cause from slight to total masking of the desired signal [3]. With these effects on signals around it and on the receiver, narrowband interferences has been causing some major problems, especially in the mobile communications universe. These signals can appear anywhere in the entire bandwidth of the channel and can appear from one, upto any number of sources. It would seem impractical and not to mention impossible to guess where these interference frequencies would appear and to build a fixed filter based on that assumption. The only perceived logical method of removing these intruding signals is to have a flexible filter structure that can be tuned to locate the interferences in the channel and eliminate them where they appear.

The standard method of designing tunable filters would be to pre-calculate the filter coefficients that would translate a given filter in frequency to the desired location. However, the hardware implementation for this method seems like overkill when it comes to designing a tunable filter. This is because in order to tune the filter over a range of frequencies, the coefficients that make-up the filter will need to be changed for each distinct frequency to be tuned to [2]. Imagine the hardware required to store all these coefficients that would enable the filter to be tuned continuously over the entire band while maintaining the specified frequency response. For example, assuming our filter is of  $N$ th order with coefficients that are of 8-bit precision. In order to tune to  $M$  distinct frequencies from DC upto the Nyquist frequency,  $M$  would therefore represent the number of ROM tables that is required to store the

$N+1$  filter coefficients. Tuning this filter to  $M$  distinct frequencies would therefore be limited by the order of the filter,  $N$ , since an increase in order brings about an increase in the number coefficients which directly translates to an increase of the size of ROM tables required to store the filter coefficients, making the standard approach highly inefficient for higher-order filters. It is obvious that a more suitable method is required.

### **1.3 Scope of Thesis**

This thesis introduces a new filter scheme which allows a fixed-filter to be tuned continuously from DC to the Nyquist frequency using a single tuning parameter. For the purpose of this thesis, a high-pass filter is chosen to provide an important part of this single parameter tunable filter used for narrowband interference suppression.

This newly proposed tunable notch filter is the first known implementation of the heterodyne concept and the frequency translation process for active filters suggested by Soderstrand and Nelson [1][8]. This Fully Tunable Heterodyne Notch (FTHN) filter can be adjusted or ‘tuned’ to the appropriate frequency in order to attenuate any NBIs resulting from a single interference source and can also be proven to cancel out all images unavoidably created during the heterodyne process.

Although the proposed digital notch filter has been mathematically proven to perform as intended [1], the purpose of this thesis is to verify the hardware implementation of the FTHN filter using VHDL as the hardware description language along with the Xilinx XCV800 Series FPGA. As such, this thesis also plays the role of a feasibility study, since a digital version of this filter has yet to be implemented due to the complexity occurring from a finite spectrum and not to mention the high sampling rate required for an efficient

system. The choice was made for an FPGA implementation owing to the fact that it provides the flexibility as a test environment and also as a rapid prototyping solution. The successful implementation could eventually lead to the design being hardwired into an ASIC or VLSI process in the near future. Hardware description language was used to create a technology independent design which could be used when targeting different FPGA families or when converting to an ASIC process.

## **1.4 Thesis Outline**

In Chapter 2 of this thesis, a background review is presented to explore the classic techniques for interference suppression in spread spectrum communications. The two most popular methods of interference suppression is discussed in somewhat shallow detail here.

In Chapter 3, modulation techniques are discussed and the heterodyne concept is explored in some detail to present the basis of the tunable structure proposed in this thesis. This chapter goes through the foundation behind the heterodyne process employed throughout the entire filter structure.

In Chapter 4, a previous tunable heterodyne filter structure is discussed here. The previous version, a tunable bandpass filter, is made up of the basic building block of the FTHN filter presented in this thesis. Advantages and disadvantages of using the previous structure is discussed here leading to the proposal of a new and improved system.

In Chapter 5, the FTHNF system is introduced. Three main components that make up the entire filter are discussed. The chapter explains the workings of the three separate components that make-up the tunable filter system.

In Chapter 6, Matlab simulation data are provided as preliminary results for the proposed filter. The FTHN filter is divided into 7 stages and MatLab diagrams are provided at each stage of the structure to simulate the functionality of the circuit and to prove the theoretical equations [1] that brought forth the idea for the filter system.

In Chapter 7, the structure of the FPGA intended for this design is dissected and inspected to provide a clear understanding of the platform of implementation chosen for this thesis.

In Chapter 8, Actual hardware implementation data is presented here from experimental results. These are results provided from the 7 stages that the system was divided into. Results here are to confirm the initial simulations obtained from MatLab provided in Chapter 6.

In Chapter 9, the final analysis of the proposed system is presented.

In Chapter 10, ideas for future work are presented.

# 2 Background

Previous modulation techniques concentrated the power of an information-bearing signal in a narrowband as compared to the bandwidth of the signal before being transmitted. It was discovered later that this technique made it easy for the signal to be suppressed or corrupted by an adjacent NBI signal. A solution to the problem came in the form of spread spectrum modulation and has been employed ever since [21].

## 2.1 Spread Spectrum

Spread spectrum is a form of wireless communication modulation technique in which the frequency of the transmitted signal is deliberately varied, resulting in the bandwidth of the modulated signal being spread beyond the bandwidth of the modulating signal [6]. This enables the transmitted signal to occupy more than one possible spectral location and therefore make signal-interception or frequency-jamming techniques extremely difficult or highly impractical. The primary use of Spread Spectrum Communications (SSC) is that of interference suppression and is the lifeline of the mobile communications industry [6].

### 2.1.1 Interference suppression in spread spectrum communications

Generally, a conventional wireless signal has a frequency that is constant with a particular bandwidth associated with it. This property allows the signal to be easily located by the receiver to retrieve the information contained in it. Unfortunately, a signal that is constant

in frequency can be greatly disturbed when another signal is transmitted on to or close to it. To recover the desired signal and its information, frequency modulation techniques will have to be employed, such as the spread spectrum frequency modulation method. There are several types of spread spectrum modulation techniques that can be used to remove unwanted interference signals, but the two most popular techniques in practice are Direct Sequence (DS) and Frequency Hopping (FH) [6].

#### **2.1.1.1 Direct Sequence**

Direct Sequence Spread Spectrum (DSSS) which is also known as Direct Sequence Code Division Multiple Access (DS-CDMA) is a common modulation technique in a coherent communications system, the information that is to be transmitted is multiplied by a Pseudo-Noise (PN) Sequence, which is essentially a pseudo number generator taking on a random binary sequence (spreading code) [6]. Figure 2-1 shows the spreading/despreading procedure used in this method [15]. This multiplication effectively spreads the wanted signal across its assigned bandwidth or channel, masking the signal as noise. However, a narrow-band interference signal could still find its way onto this frequency spectrum of the spread signal. Hence, to eliminate the interference signal, the receiver ‘despreads’ the wanted signal and acquires it using the replica of the spreading code while this function also effectively spreads out the interference signal, as shown in Figure 2-2 [15]. This system however can result in long acquisition time due to long PN codes used and it also suffers from the well-known ‘near-to-far’ problem where transmitters which are closer to the receiving station can overpower distant transmitters and in effect, destroy their signals [7]. Although this method is somewhat effective, an NBI signal could still have a relatively high-power associated with it when it is spread-out at the receiver.

At the front end of the receiver, such high-powered signals could drive the operational amplifier into its saturation (nonlinear) region, preventing the desired signal to be effectively extracted.

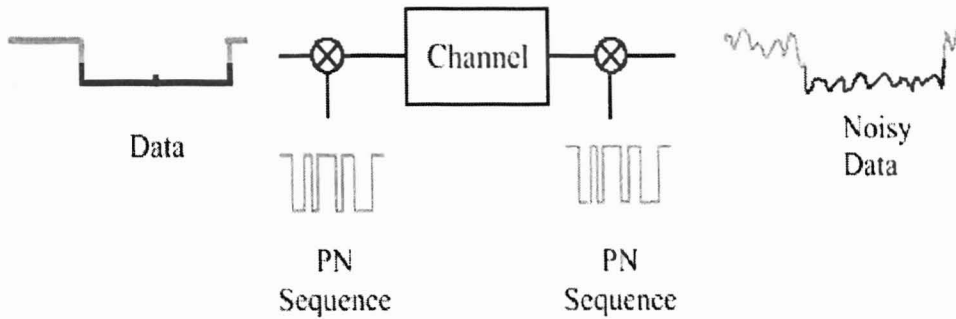


Figure 2-1. Spreading/despreading method.

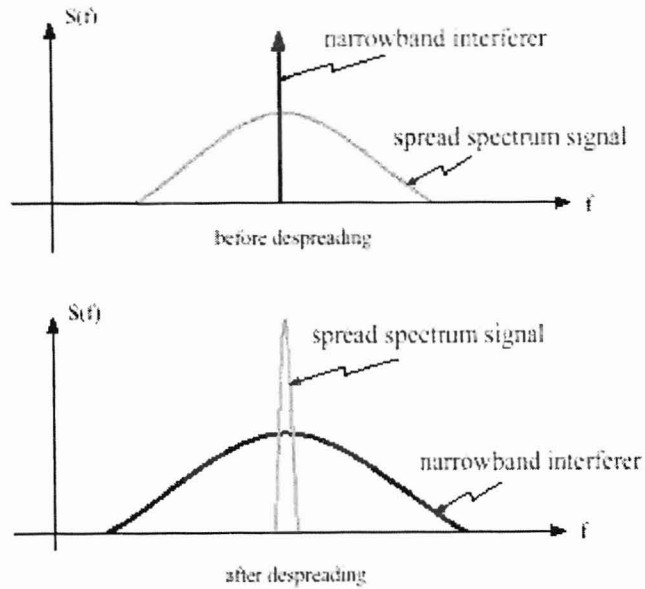


Figure 2-2. Spectral effects of spreading/despreading.



### **2.1.1.2 Frequency Hopping**

In Frequency Hop Spread Spectrum (FHSS), the frequency of the transmitted signal is randomly 'hopped' around onto a given set of frequencies in an assigned bandwidth or channel to avoid frequency jamming or interception [6]. The frequency hopping is generated by a Hopping Code Generator (HCG) which essentially shifts the signal onto a different frequency randomly. At the receiver end, the hopping code that was used to create the frequency shifts is replicated and synchronized with the hopping pattern to retrieve the original transmitted signal. This system requires a complex frequency synthesizer to generate the frequency hops from  $m$  binary digits which are then mapped to one of  $N=2^m$  frequencies in the channel [7].

### **2.1.1.3 Signal recovery**

Any spread spectrum receiver can only suppress a given amount of interference and the system will not function as desired if the power associated with the interference signal becomes too great. With the above mentioned techniques, the wanted signal at the receiver is separated from the unwanted signal with the use of digital signal processing techniques and algorithms, before further processing is done on the wanted signal. The proposed FTHN filter can be shown to provide the necessary means to suppress any NBI signal appearing at the receiver end before passing the wanted signal to the rest of the receiver, thereby greatly increasing its effectiveness and not to mention the quality of the received signal.

# 3 The Heterodyne Process

Information-bearing signals have to be modulated before being transmitted through free space or through a communications channel. The modulation process allows the information to be transmitted over long distances and typically translates the information-bearing signal to a new spectral location. It is a necessary process in communications theory when transmitting a message signal through free-space in order to have effective transmission using a reasonable-sized antenna. In a communications channel, the modulation process allows different signals to be transmitted to different spectral locations when more than one signal can occupy the channel [6]. There are several types of modulation techniques that can be employed based upon the type of application and the factors that can influence the transmission. For the purposes of this thesis, we will look at the method known as ‘Frequency Translation’.

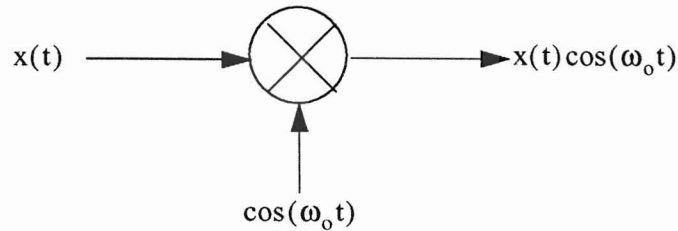
## 3.1 Frequency translation

Any signal can be translated or moved to a new spectral location. This translation process employs the most widely used modulation technique known as ‘heterodyning’.

### 3.1.1 The heterodyne concept

Heterodyning or ‘mixing’ is the process of translating a signal to a new spectral location by mixing two frequencies together to produce a different frequency output. In the continuous-

time domain, this mixing process can be described as a multiplication of two signals with different frequencies to produce a third signal.



**Figure 3-1.** Mixer block diagram.

Figure 3-1 shows the block diagram of the heterodyne concept in continuous time. Looking at the frequency domain, the results of this multiplication translates into a shift in frequency of the original signal. Taking the Fourier Transform of the output of the multiplication process produces the following result:

$$\mathfrak{F}\{x(t)\cos(\omega_0 t)\} \rightarrow \frac{1}{2}X(f - f_0) + \frac{1}{2}X(f + f_0) \quad (3.1)$$

As can be observed from the above equation, the multiplication in the time domain process translated the incoming signal to two different spectral locations containing the sum and difference frequency components of the two input signals. Notice also that the heterodyne process effectively doubles the bandwidth of the signal with the creation of the two frequency components, each with the original bandwidth. In the analog domain, bearing in mind that the spectra is infinite, this effect is acceptable since one of the images can be safely removed. The basis for the proposed implementation in this thesis has been derived from this frequency translation concept which is widely applied in radio communications.

One of the most popular application for this concept appears in the use of superheterodyne receivers.

### 3.1.2 Superheterodyne receiver

In the superheterodyne receiver, the incoming signal with frequency ( $f_c$ ) is 'mixed' with a signal containing the heterodyne frequency ( $f_{LO} = f_c - f_{if}$ ). This process serves to translate the input signal which is centered on the carrier frequency,  $f_c$  to the intermediate frequency,  $f_{if}$  where the signal is extracted using an appropriate filter before being processed for a given application. Employing the frequency translation theorem mentioned previously in section 3.1.1, the local oscillator (LO) is tuned to the frequency at which the input frequency ( $f_c$ ) is translated into the passband of the intermediate frequency filter located at  $f_{if}$ . Essentially, we are simply reducing the frequency of the incoming signal which is either Amplitude Modulated (AM) or Frequency Modulated (FM) by virtue of mixing the signal with another signal. Figure 3-3 shows the block diagram of the superheterodyne filter [6].

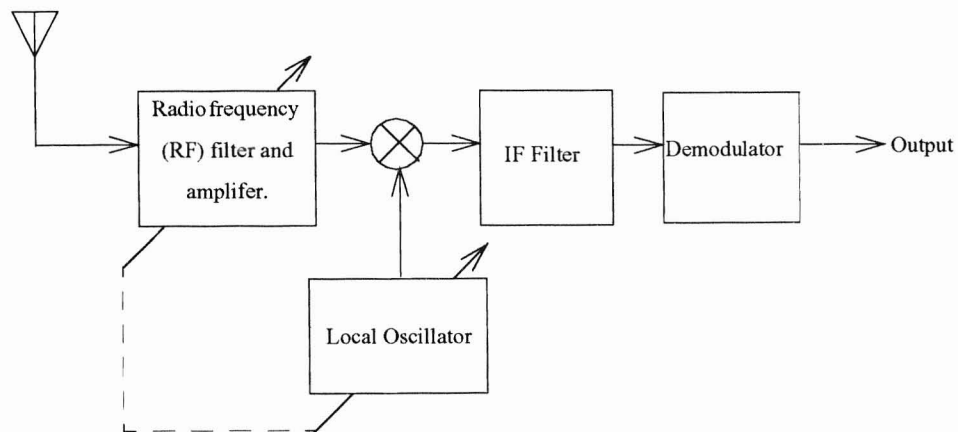


Figure 3-2. Superheterodyne receiver block diagram.

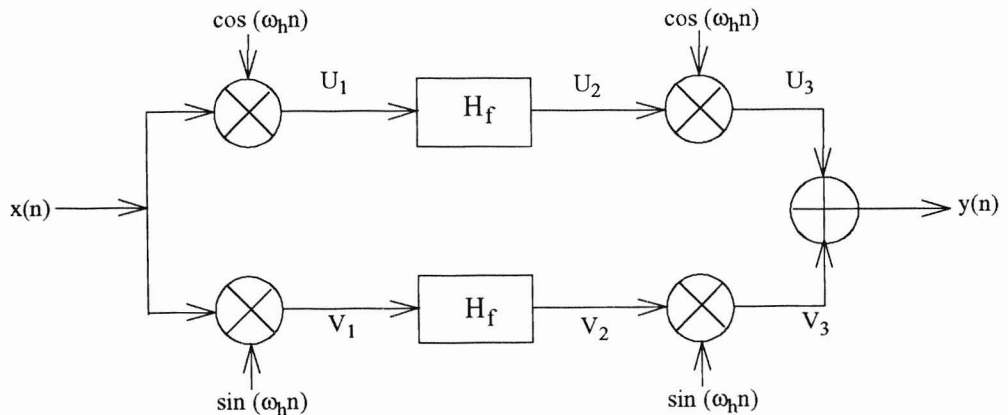
# 4 Previous Work

A previous implementation of a tunable heterodyne filter was explored and implemented on a Virtex XCV800 FPGA in a Thesis work by Azam [2]. The Tunable Heterodyne Band-Pass Filter implementation by Azam makes use of a very narrow bandpass filter to isolate the NBI signal. The range of tunability in this filter is from 0 to  $\pi/4$  and  $\pi/4$  to  $\pi/2$ .

## 4.1 Tunable Bandpass Filter

Various tunable heterodyne bandpass filters implementation using the basic building block in Figure 4-1, was discussed by Azam in his thesis work [2]. These tunable bandpass structures can be characterized by the following transfer function [1]:

$$H(z, \omega_h) = \frac{1}{2} [H_f(z e^{j\omega_h}) + H_f(z e^{-j\omega_h})] \quad (4.1)$$



**Figure 4-1.** Tunable Heterodyne Bandpass Filter basic building block.

### 4.1.1 Theory of operation

The operation behind the tunable bandpass structure shown in Figure 4-1 can be explained using Figure 4-2. We assume that the input signal,  $x(n)$  contains an undesired sinusoid at 3kHz and desired sinusoid at 5kHz and the fixed filter has a narrow passband centered at 10kHz. The heterodyning frequency,  $f_h$  ( $\omega_h/2\pi$ ) required in this case would be 7kHz (10kHz-3kHz), which would translate the undesired signal into the filter's passband region. Frequency components on the negative axis are ignored in the following demonstration for sake of simplicity.

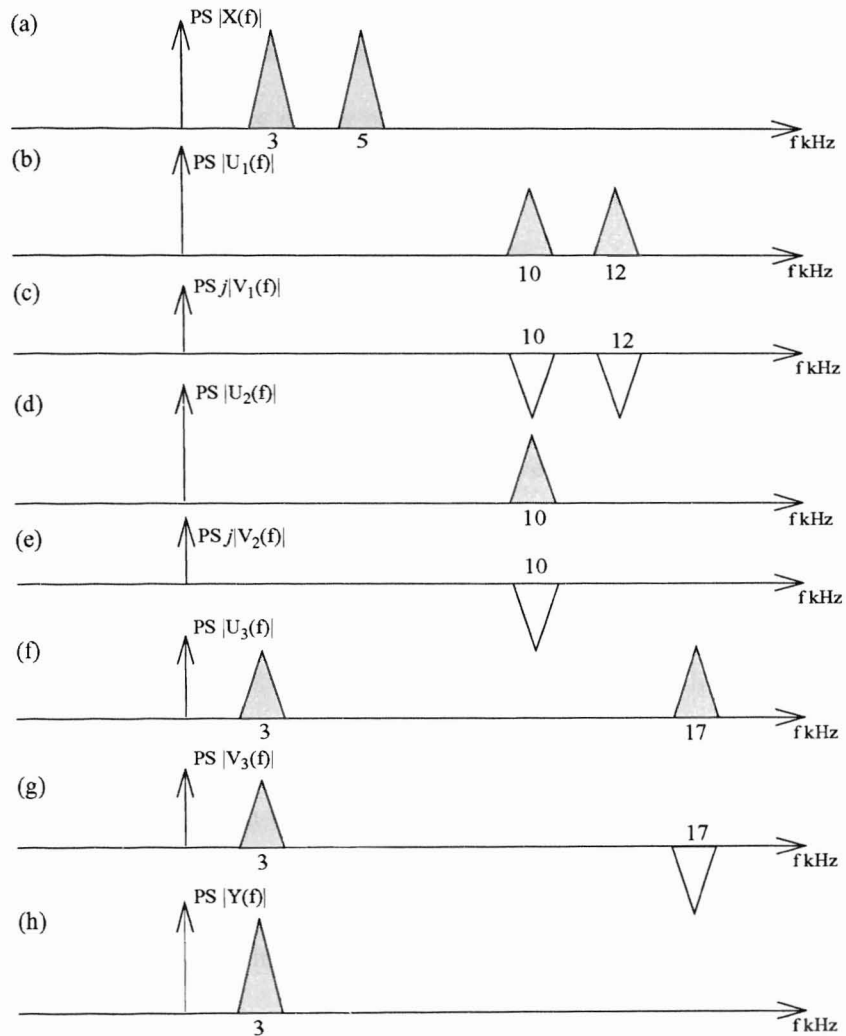


Figure 4-2. Tunable Bandpass Filter operation.

With the Fourier transform properties of cosine  $(\frac{1}{2}e^{j2\pi f_h t} + \frac{1}{2}e^{-j2\pi f_h t})$  and sine  $(\frac{1}{2j}e^{j2\pi f_h t} - \frac{1}{2j}e^{-j2\pi f_h t})$ , Figure 4-2 follows the signal through the block diagram of Figure 4-1 showing the power spectrum plots of the in-phase and quadrature components of the input signal traversing the structure. In-phase components are plotted against the real spectrum axis, while the quadrature components are plotted against the imaginary spectrum axis.

In (a) of Figure 4-2, the power spectrum of the input signal,  $x(n)$  containing frequency components centered at 3kHz and 5kHz is shown. In (b), we plot  $U_1$  formed when we heterodyne the input signal  $x(n)$  with a cosine wave of frequency  $f_h$  Hz. The positive phasor portion of the cosine signal  $(\frac{1}{2}e^{j2\pi f_h t})$  will translate the spectral components of the input signal to their new locations shown at 10kHz and 12kHz. In (c), we plot  $V_1$  formed when we heterodyne the input signal with a sine wave of frequency  $f_h$  Hz, the frequency components of the input signal are translated to 10kHz and 12kHz, except that the translated spectra lies below the frequency axis due to the negative sign of the positive frequency  $(-\frac{1}{2j}e^{j2\pi f_h t})$ . In (d) and (e), we see that the spectrum of  $U_2$  and  $V_2$  after the filter  $H_f$ , hence the spectral component at 10kHz remains after being filtered with the bandpass filter  $H(f)$ . Notice that the resulting spectra are out of phase with each other. In (f), the heterodyne process on  $U_2$  translates the filtered signal  $U_3$  on the in-phase branch to its new spectral locations at 3kHz and 17kHz due to the positive and negative phasor portions of the cosine signal  $(\frac{1}{2}e^{j2\pi f_h t} + \frac{1}{2}e^{-j2\pi f_h t})$ . In (g), the signal  $V_2$  is translated to  $V_3$ , with its new spectral locations at 3kHz and 17kHz. With the second multiplication in the time domain, the spectrum is now plotted on the real axis. The negative amplitude portion of the spectrum at 10kHz will be translated upwards to 17kHz and downwards to 3kHz on the real axis. Due

to the positive sign of the negative frequency, and the fact that the spectrum in (e) already contains the imaginary component,  $j$ , the translated signal at 3kHz lies above the frequency axis, while the translated signal at 17kHz lies below the frequency axis. Finally in (h),  $U_3$  and  $V_3$  which are both plotted on the real axis are added forming the output  $y(n)$ . Since the frequency components at 3kHz are in-phase and the frequency components at 17kHz are out of phase, the summation of these two branches yields only the in-phase spectra at 3kHz. Therefore, the NBI signal was effectively isolated from the desired signal [2].

The circuit of Figure 4-1 allows a single parameter to be tuned to translate the incoming interference signal into the center of the bandpass filter via the heterodyne process. The bandpass filter isolates the unwanted signal before it is translated back to its baseband by virtue of another heterodyne process. As one might have already guessed, this system however is not complete, it does not produce the desired signal at the output but serves to 'capture' the interference signal instead. The structure can further be used to actually remove the interference frequency and produce the desired signal at the output at the cost of incurring some additional hardware. However, the effectiveness of the digital heterodyne process is clearly visible.

The final transfer function implies that the filters were translated up and translated down and then added together to produce the end result. Since the filters used were fixed filters, clearly the up-translation and down-translation discussion here is merely a virtual phenomena. Realistically speaking, the translation process only applies to the incoming signal and not the filters themselves.

The tunable bandpass filter of Figure 4-1 can be constructed using high-pass filters, low-pass filters and band-pass filters. The difference between these three filters comes in the



range of tunability across the frequency spectrum. High-pass and low-pass filters used in the design allows the filter to be completely tuned from DC to Nyquist. The bandpass filter's tuning capability however is divided into two regions: from DC to  $\pi/4$  and from  $\pi/4$  to Nyquist. The summation of the up-translated and down-translated poles and zeros of the prototype bandpass filter virtually creates 2 bandpass filters in the regions mentioned above. A similar situation presents itself when using a notch filter as the prototype filter in the structure.

## **4.2 Final Analysis**

The previous filter structure has several advantages and disadvantages when implemented as a tunable structure for narrowband interference rejection.

### **4.2.1 Advantages**

From the proposed structure in Figure 4-1 it can be seen that several types of filters can be used in the design to implement the NBI extraction from the transmitted signal. Lowpass, high-pass and bandpass filters can be interchanged depending on the type of application.

### **4.2.2 Disadvantages**

Low-order filters proved to be unsuitable for the application as it failed to completely isolate the interference signal from the desired signal. The solution to the problem of isolating the interference frequency came in the form of implementing higher-order filters to replace the ineffective lower-order filters [2]. It is well documented that higher-order filters require extensively more hardware and are more difficult to implement. This is because the number of coefficients that make-up the filter increases linearly with the increase in order of the fil-

ter. Since each of the coefficients are implemented as multipliers, considerable hardware will need to be expended in order to achieve an acceptable frequency response at the output of the system.

Having a bandpass filter with a center frequency other than 0 (low-pass) or  $f_s/2$  (high-pass) can prove to be problematic when dealing with zeros in some cases where the bandpass is not one that is extremely narrow. Assuming that  $H(z)$  has both poles and zeros,  $H(Z) = B(z)/A(z)$ , then the problem can be explained using the following equations [1]:

$$\begin{aligned}
 H(z, \omega_h) &= \frac{1}{2} [H_{f1}(ze^{j\omega_h}) + H_{f2}(ze^{-j\omega_h})] \\
 Y(z, \omega_h) &= \frac{1}{2} \left\{ H_{f1}(ze^{j\omega_h}) + H_{f2}(ze^{-j\omega_h}) \right\} X(z, \omega_h) \\
 &= \frac{1}{2} \left\{ \frac{B_1(ze^{j\omega_h})}{A_1(ze^{j\omega_h})} + \frac{B_2(ze^{-j\omega_h})}{A_2(ze^{-j\omega_h})} \right\} X(z, \omega_h) \\
 &= \frac{1}{2} \left\{ \frac{B_1(ze^{j\omega_h})A_2(ze^{-j\omega_h}) + B_2(ze^{-j\omega_h})A_1(ze^{j\omega_h})}{A_{1(2)}(ze^{-j\omega_h})A_2(ze^{j\omega_h})} \right\} X(z, \omega_h) \\
 H(z, \omega_h) &= \frac{1}{2} \left\{ \frac{B_1(ze^{j\omega_h})A_2(ze^{-j\omega_h}) + B_2(ze^{-j\omega_h})A_1(ze^{j\omega_h})}{A_1(ze^{-j\omega_h})A_2(ze^{j\omega_h})} \right\} \quad (4.2)
 \end{aligned}$$

As observed, in equation 4.2, the poles of the filter gets multiplied with each other in the denominator and translates linearly with respect to the heterodyning frequency,  $f_h$ , as expected. However, we have lost control over the zeros of the final transfer function as they are mixed with the poles and zeros in the numerator. With this difficulty of predicting and controlling where the zeros of the transfer function will end-up, the design of the filter will

need to take on a different approach. The transfer function of the separate filter will need to be somewhat pre-distorted to end up with the desired overall response when the addition of the separate transfer functions take place [1][2].

# 5 FTHN Filter System

The FTHN filter system is composed of three (3) main components:

1. Splitter
2. Fixed-coefficient prototype filter
3. Combiner

## 5.1 Splitter

The Splitter component is the first heterodyne unit in the system. The structure is divided into 2 branches, called the “in-phase” and “quadrature” branches respectively. The incoming signal is split into these branches and multiplied with a cosine wave on the in-phase branch and a sine wave on the quadrature branch; both of which are carrier waves at the heterodyning frequency. Using the trigonometric identities given by equations 5.1 and 5.2 below and the frequency translation theorem given by equation 5.3, it can be shown that this multiplication process in the time domain effectively translates the incoming signal to new spectral locations.

$$\cos(\omega_0 n) = \frac{1}{2} \left( e^{+j(\omega_0 n)} + e^{-j(\omega_0 n)} \right) \quad (5.1)$$

$$\sin(\omega_0 n) = \frac{1}{2j} \left( e^{+j(\omega_0 n)} - e^{-j(\omega_0 n)} \right) \quad (5.2)$$

$$x(n)e^{j(\omega_0 n)} \rightarrow X(f-f_0), \quad \omega_0 = 2\pi f_0 \quad (5.3)$$

Figure 5-1 shows the Splitter implementation. The frequency domain function of the Splitter can be explained using equations 5.4 and 5.5.

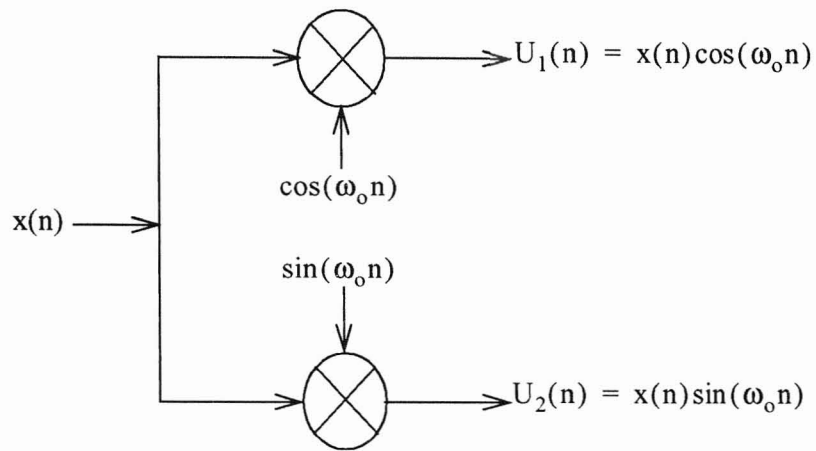


Figure 5-1. Splitter circuit

$$U_1 = \frac{1}{2}X(z e^{j\omega_0 n}) + \frac{1}{2}X(z e^{-j\omega_0 n}) \quad \text{where } \omega_0 = 2\pi f_0 \quad (5.4)$$

$$U_2 = \frac{1}{2j}X(z e^{j\omega_0 n}) - \frac{1}{2j}X(z e^{-j\omega_0 n}) \quad (5.5)$$

As observed in equations 5.4 and 5.5, the input signal is translated to a pair of new spectral locations after being passed through the splitter. The locations of these spectra are determined by the heterodyning frequency,  $f_0$ .

## **5.2 Fixed-coefficient prototype filter**

For the initial implementation, a first-order fixed-coefficient IIR filter will be used to study the feasibility of using a low-order filter to provide the necessary filtering needed for the system to function as required. The choice for an IIR filter instead of the more stable FIR was made to take advantage of the sharp transition band property associated with IIR filters. We can effectively design a low order IIR filter with sharp transition band rolloffs as compared to an FIR filter with the same characteristics which would greatly increase the order and the hardware required. This sharp transition band property of an IIR filter enables the system to effectively isolate and simultaneously remove the interfering signal while trying not to compromise any part of the wanted signal components. Logically, a notch filter seems to be the appropriate choice for this implementation.

### **5.2.1 Why implement a notch filter?**

NBI signals are signals which often times have most, if not all their power concentrated in a very narrow band. These signals occupy a very small portion in the frequency domain but have strong enough power contained within itself to corrupt nearby signals. The removal of these signals therefore becomes tricky situation when they appear on or nearby the desired signal. This is because it leaves us with the difficulty of designing a filter that is broad enough to isolate the entire interfering signal yet narrow enough as to not remove any other wanted components around or under it.

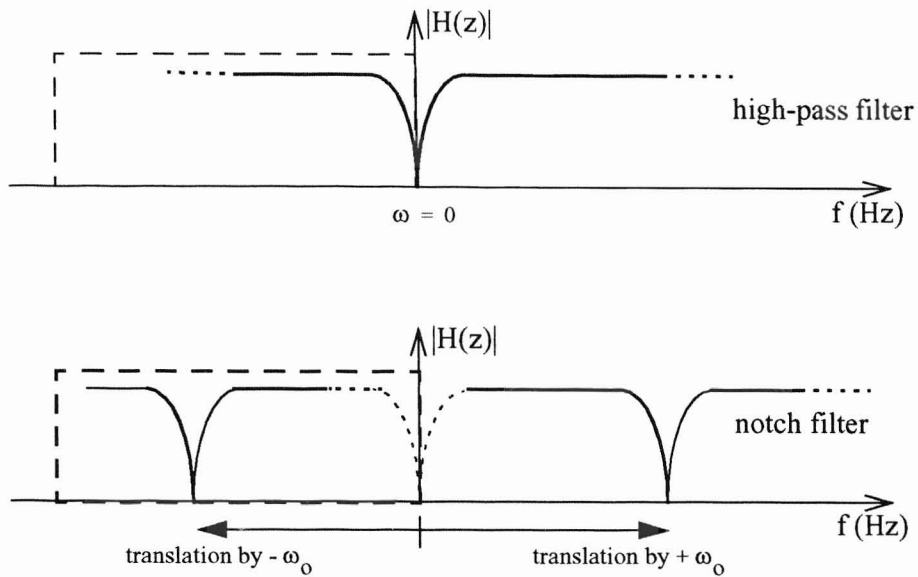
The notch filter implementation with a very narrow stopband was chosen to provide the most suitable method of removing these NBI signals without removing the original infor-

mation that was being transmitted. However, to obtain a notch filter, a high-pass filter was used in the structure. The question then arises as to why implement a high-pass filter in the structure instead of just a plain notch filter? It can be shown that by using a fixed high-pass filter in the structure, we can successfully create via frequency translation, a notch filter that does exactly what we intended to do, that is to remove the unwanted signal.

### **5.2.2 Filter selection**

The high-pass filter was selected as the fixed-coefficient prototype filter for the structure since its virtual translation in the frequency domain will produce a notch filter centered at the desired frequency. An example of this translation effect is illustrated in Figure 5-2. This effect can also be described by equations 5.6 and 5.7 as the translation on the z-plane of the transfer function  $H(z)$ .

As mentioned before, the filters used in this structure are fixed-filters and not adaptive ones. The intended purpose of using fixed-filters is to show that instead of moving the entire filter system to the signal as one would expect, here, the signal is moved/translated to the fixed-filters to perform the filtering processes. However, looking at the final transfer function of the system, one could argue that the filter itself is being translated, which is not true. Finally, after filtering, the intended signal is then moved back to its baseband.



**Figure 5-2.** Virtual translation in frequency of a high-pass filter.

$$H(z, \omega_0) = H\left(z e^{j(\omega + \omega_0)}\right) \quad (5.6)$$

Equation 5.6 shows an ‘up’ translation, while Equation 5.7 shows the ‘down’ translation of the transfer function. This ‘up’ or ‘down’ translation depends on whether we are moving anti-clockwise or clockwise on the unit circle. The explanation for this phenomena comes from the fact that when sampling a real signal, the frequency response of the filter repeats every sampling frequency (periodic). Therefore, a low-pass or high-pass filter can be converted into a notch filter simply by translating the filter transfer function as shown in equation 5.6 and 5.7 on the unit-circle.

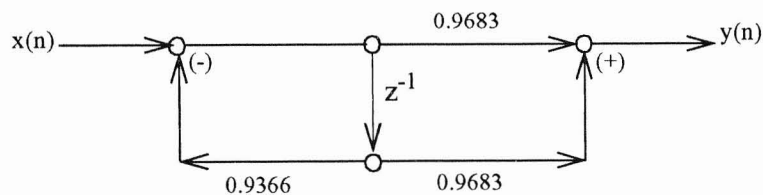


$$H(z, \omega_o) = H\left(ze^{j(\omega - \omega_o)}\right) \quad (5.7)$$

Equation 5.8 below shows the first-order filter equation designed for this implementation. The design was based on the fact that the 3db bandwidth of the stop-band was to be as narrow as possible. We will assume for the purposed of this thesis that a 500Hz stopband region for the high-pass filter will be sufficient to remove a given narrowband signal, since the stopband doubles to 1000Hz when frequency translation occurs. The following equation produces a first-order high-pass filter with a flat passband and a 3db stopband width of 500Hz.

$$H(z) = \frac{k(z^{-1} + 1)}{1 - rz^{-1}}, \text{ where } k = 0.9683 \text{ and } r = 0.9366 \quad (5.8)$$

Using Direct Form II implementation, the filter structure is obtained and shown in Figure 5-3.

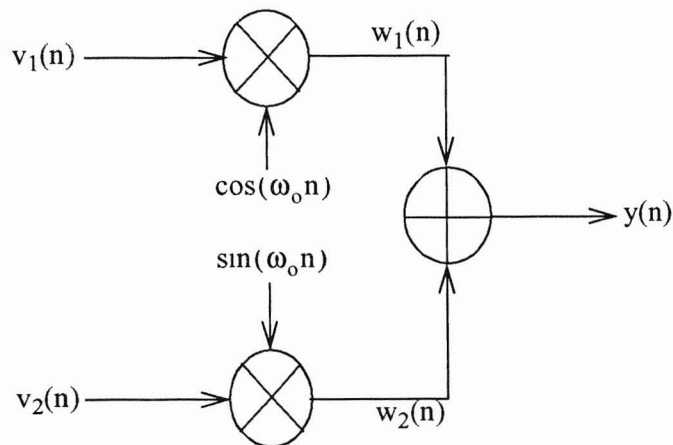


**Figure 5-3.** Direct Form II implementation of a first-order high-pass filter

## 5.3 Combiner

The Combiner is the final component in the FTHN filter structure. The combiner multiplies the incoming signal by a cosine (in-phase branch) and sine (quadrature branch) and the resultant outputs of the multiplication process are then added together.

Signals passing through the system will create images when multiplied with a sine or cosine signal. Each multiplication with a cosine translates the signal to two different frequencies but the signal components still hold on to their original phase relationship. Each multiplication of the incoming signal with a sine wave translates the signal to two different frequencies but inverts the phase of one of the signal components. This phenomena is taken advantage of in this structure and is critical in removal of images created via the heterodyning process throughout the filter. Figure 5-4 shows the structure of the combiner circuit.



**Figure 5-4.** Combiner circuit.

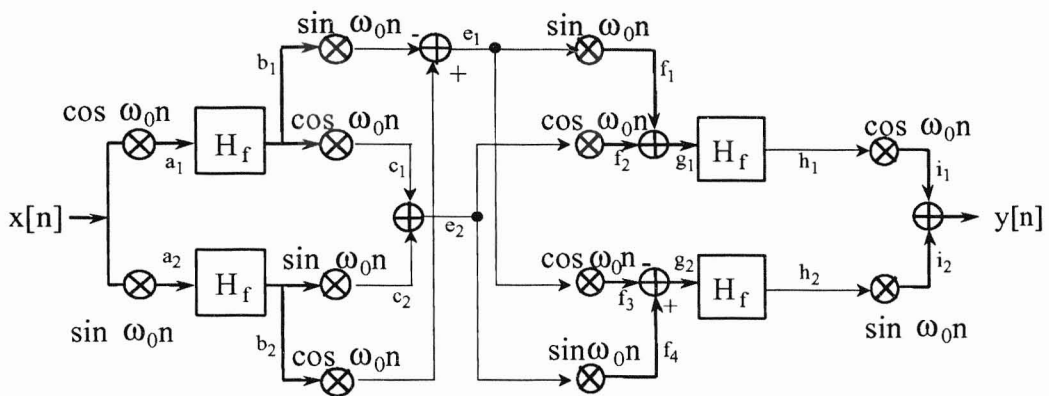
The combiner acts the same way a splitter circuit functions via frequency translation. However, the combiner circuit serves to bring its input signal back to baseband after multiplica-

tion with cosine and sine. The output of the combiner is simply the addition of its in-phase and quadrature branches as shown in equation 5.9.

$$Y(z) = W_1(z) + W_2(z) \quad (5.9)$$

## 5.4 The Complete FTHN Filter System

The complete system for the Fully Tunable Heterodyne Notch (FTHN) Filter is given in Figure 5-5.



**Figure 5-5.** Fully Tunable Heterodyne Notch Filter

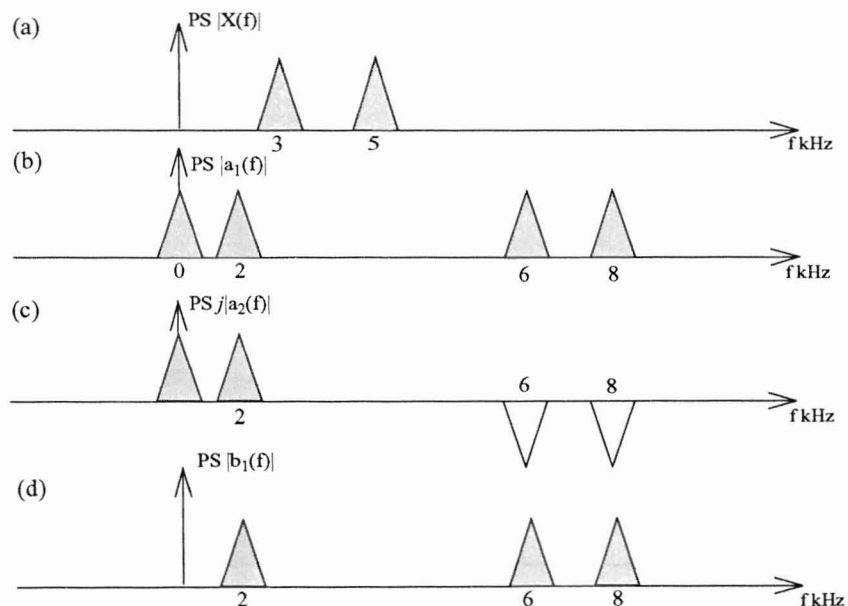
There is a noticeable double-edge effect of using this structure as shown in equation 5.10 and proven in Appendix A which shows that although the signal is being moved/translated to the filter, the final equation gives the impression that the ‘fixed’ filter itself is actually being translated in frequency. This is the main advantage of using this structure.

$$H_{\text{final}}(z, \omega_o) = \left[ H\left(ze^{-j\omega_o}\right)H\left(ze^{j\omega_o}\right) \right] \quad (5.10)$$

As the final transfer function of the system, equation 5.10 also shows that although the system is made up of nonlinear components (variable multipliers), the output of the filter is completely linear to the single tuning parameter,  $f_o$ .

### 5.4.1 Theory of Operation

The operation behind the fully tunable heterodyne notch filter structure shown in Figure 5-5 can be explained using Figures 5-6, 5-7 and 5-8. We assume that the input signal,  $x(n)$  contains an undesired sinusoid at 3kHz and desired sinusoid at 5kHz and the fixed high-pass filter has a narrow stopband of 500Hz. The heterodyning frequency,  $f_h$  ( $\omega_h/2\pi$ ) required in this case would be 3kHz, which would translate the undesired signal into the filter's stopband region. Frequency components on the negative axis and amplitude changes are ignored in the following demonstration for sake of simplicity.



**Figure 5-6.** Fully Tunable Heterodyne Notch Filter operation (part I).

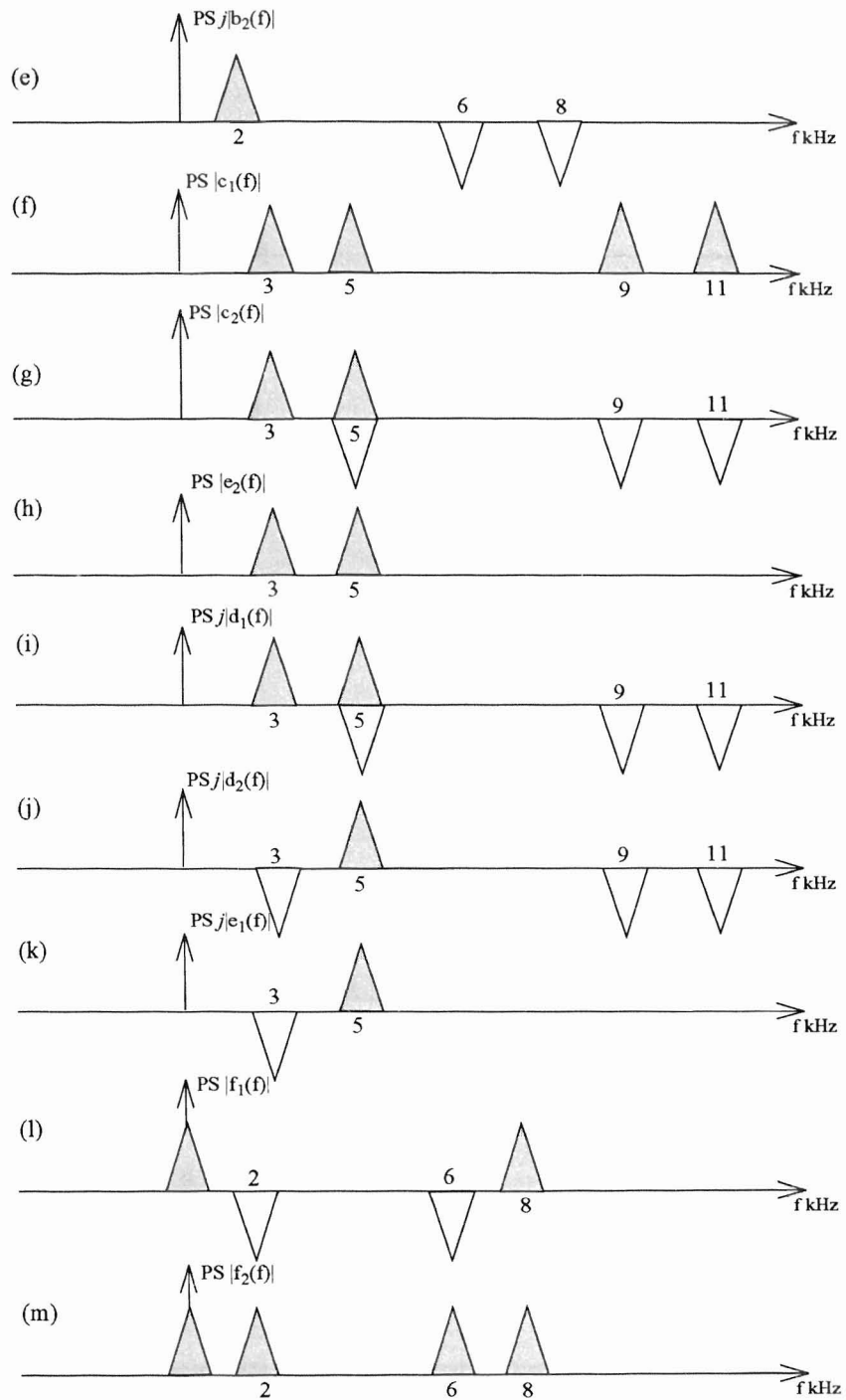


Figure 5-7. Fully Tunable Heterodyne Notch Filter operation (part II).

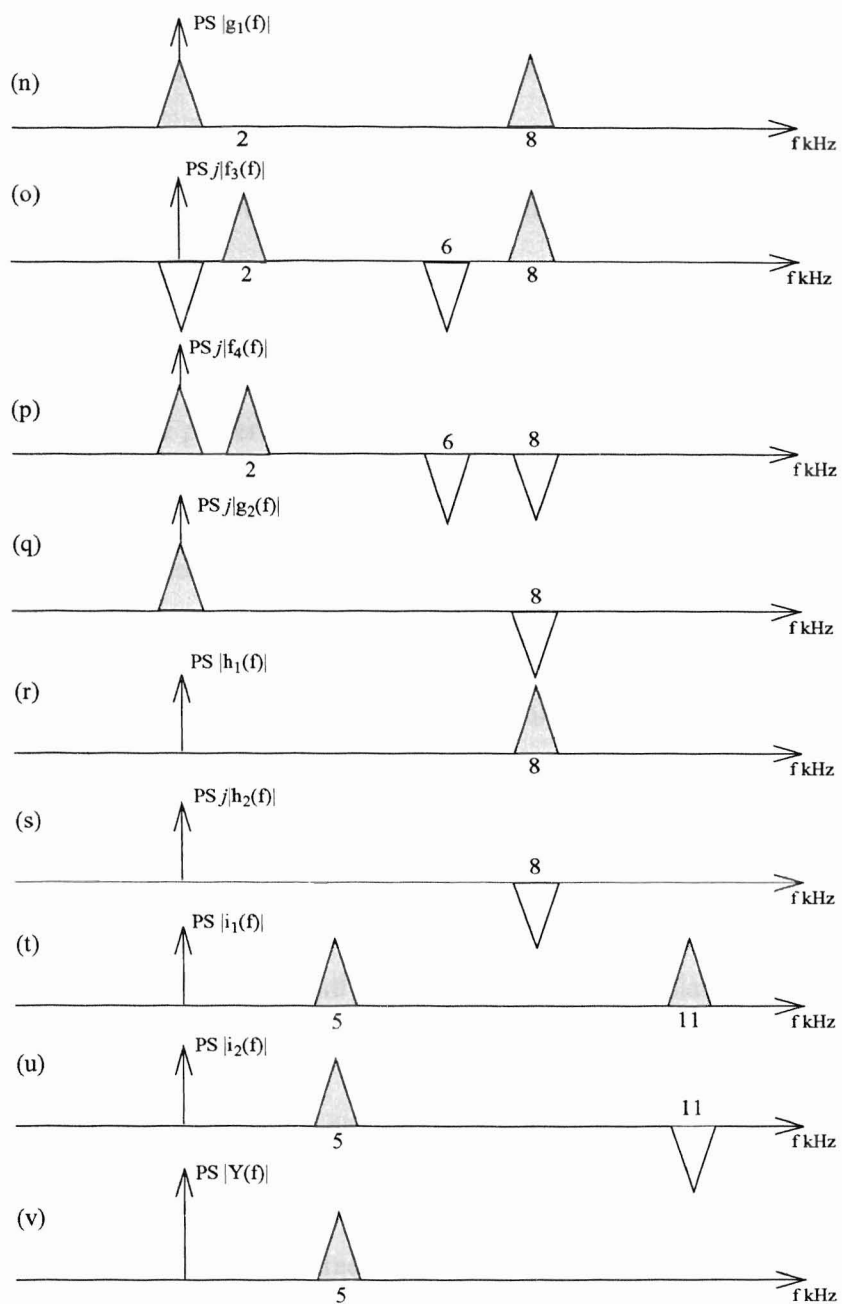


Figure 5-8. Fully Tunable Heterodyne Notch Filter operation (part III).

With the Fourier transform properties of cosine ( $\frac{1}{2}e^{j2\pi f_h t} + \frac{1}{2}e^{-j2\pi f_h t}$ ) and sine

$(-\frac{1}{2}je^{j2\pi f_h t} + \frac{1}{2}je^{-j2\pi f_h t})$ , Figure 5-6, 5-7 and 5-8 follows the signal through the block diagram of Figure 5-5 showing the power spectrum plots of the in-phase and quadrature components of the input signal traversing the structure. The in-phase components are plotted against the real spectrum axis while quadrature components are plotted against the imaginary spectrum axis.

In (a) of Figure 5-6, the power spectrum of the input signal,  $x(n)$ , containing frequency components at 3kHz and 5kHz is shown. In (b), we plot  $a_1$  formed when we heterodyne the input signal  $x(n)$  with a cosine wave of frequency  $f_h$  Hz. The positive phasor portion of the cosine signal  $(\frac{1}{2}e^{j2\pi f_h t})$  will translate the spectral components of the input signal to their new locations shown at 6kHz and 8kHz while the negative phasor portion of the cosine signal  $(\frac{1}{2}e^{-j2\pi f_h t})$  will translate the spectral components of the input signal to their new locations shown as 0 kHz (DC) and 2kHz. In (c) we plot  $a_2$  formed when we heterodyne the input signal with a sine wave of frequency  $f_h$  Hz. The frequency components of the input signal are translated to 6kHz and 8kHz, except that the translated spectra lies below the frequency axis due to the negative sign of the positive phasor portion of the sine wave  $(-\frac{1}{2}je^{j2\pi f_h t})$ . The positive sign of the negative phasor portion of the sine wave  $(\frac{1}{2}je^{-j2\pi f_h t})$  however translates the input signal components to 0kHz (DC) and 2kHz and causes these signal components to lie above the frequency axis when the heterodyne operation is performed.

In (d), we plot  $b_1$  formed when the signal components in (b) are passed through the fixed high-pass filter with a 3dB stopband width of 500 Hz. The signal component at DC has been removed by the stopband of the filter while the other components remain since they

are clearly within the passband region. In (e), we plot  $b_2$  formed when the signal components in (c) are passed through the fixed high-pass filter with a 3db stopband width of 500Hz. Again, the signal component at DC has been removed while the other components of the signal remain intact.

In (f), we plot  $c_1$  formed when we heterodyne the signal components in (d) with a cosine wave of frequency  $f_h$  Hz. Again, the positive phasor portion of the cosine signal ( $\frac{1}{2}e^{j2\pi f_h t}$ ) translates the signal components in (d) to 5kHz, 9kHz and 11kHz, while the negative phasor portion of the cosine signal ( $\frac{1}{2}e^{-j2\pi f_h t}$ ) translates the signal components in (d) to 3kHz and 5kHz. In (g), we plot  $c_2$  formed when we heterodyne the signal components in (e) with a sine wave of frequency  $f_h$  Hz. With the second sine multiplication in the time domain, the resulting spectrum is now plotted on the real axis. Signal components in (e) are translated to 5kHz, 9kHz and 11kHz and lie below the frequency axis due multiplication with the positive phasor portion of the sine signal ( $-\frac{1}{2}je^{j2\pi f_h t}$ ). Signal components in (e) are also translated to 3kHz and 5kHz due to multiplication with the negative phasor portion of the sine signal ( $\frac{1}{2}je^{-j2\pi f_h t}$ ). As observable in (g), the signal component at 5kHz subsequently will be cancelled after translation due to heterodyning.

In (h), we plot  $e_2$  formed when signal components from (f) and (g) are added together. This addition process results in the elimination of signal components at 9kHz and 11kHz, leaving behind signal components at 3kHz and 5kHz. In (i), we plot  $d_1$  formed when we heterodyne signal components in (d) with a sine wave of frequency,  $f_h$  Hz. Signal components in (d) are translated to 5kHz, 9kHz and 11kHz due to the multiplication with the positive phasor portion of the sine wave ( $-\frac{1}{2}je^{j2\pi f_h t}$ ), while signal components at 3kHz and 5kHz are the results of the multiplication in the time domain with the negative phasor portion of



the sine wave  $(\frac{1}{2}e^{-j2\pi f_h t})$ . As observable in (i), the signal component at 5kHz subsequently will be cancelled after translation due to heterodyning.

In (j), we plot  $d_2$  formed when we heterodyne signal components in (e) with a cosine wave of frequency  $f_h$  Hz. The positive phasor portion of the cosine wave  $(\frac{1}{2}e^{j2\pi f_h t})$  will translate the signal components at 2kHz to 5kHz, 6kHz to 9kHz and 8kHz to 11kHz respectively. The negative phasor portion of the cosine wave  $(\frac{1}{2}e^{-j2\pi f_h t})$  will translate the signal components at 6kHz to 3kHz and 8kHz to 5kHz respectively. In (k), we plot the summation of (i) and (j) which leaves us with frequencies at 3kHz and 5kHz.

In (l), we plot  $f_1$  formed when we heterodyne signal components in (k) with a sine wave of frequency  $f_h$  Hz and in (m), we plot  $f_2$  formed when we heterodyne signal components in (h) with a cosine wave of frequency  $f_h$  Hz. In (n), we plot the addition of signal spectra from (l) and (m). In (o), we plot  $f_3$  formed when we heterodyne signal components in (k) with a cosine of frequency  $f_h$  Hz and in (p) we plot  $f_4$  formed when we heterodyne signal components in (h) with a sine of frequency  $f_h$  Hz. In (q), we plot the subtraction of the spectra in (o) from the spectra in (p) resulting in frequencies at DC, 2kHz and 8kHz.

In (r), we plot the resulting spectra when signal components in (n) are passed through the fixed high-pass filter with a -3dB stopband of 500Hz and in (s), we plot the resulting spectra when signal components in (q) are passed through the fixed high-pass filter.

In (t), we plot  $i_1$  formed when we heterodyne signal components in (r) with a cosine of frequency  $f_h$  Hz and in (u), we plot  $i_2$  formed when we heterodyne signal components in (s) with a sine of frequency  $f_h$  Hz. Finally, in (v), we plot the resulting spectra when signal

components in (t) are added to signal components in (u). As observed, the interference frequency designated at 3kHz have been eliminated from the system along with any images created through the heterodyne process.

# 6 MatLab Simulations

In this chapter, we will go through the simulation of the proposed system using MatLab to verify that the FTHN filter functions as predicted at each stage of the system.

A combination of two input signals of frequency 3kHz and 5kHz, sampled at 48kHz are used to immitate an NBI signal and a desired signal respectively. The goal here is to remove that interference signal at 3kHz without affecting the desired signal component at 5kHz. For purposes of clarity, we will break up the FTHN Filter structure into stages and use the MatLab plots to describe the functionality of the system. Figure 6-1 shows the stages in the FTHN Filter structure.

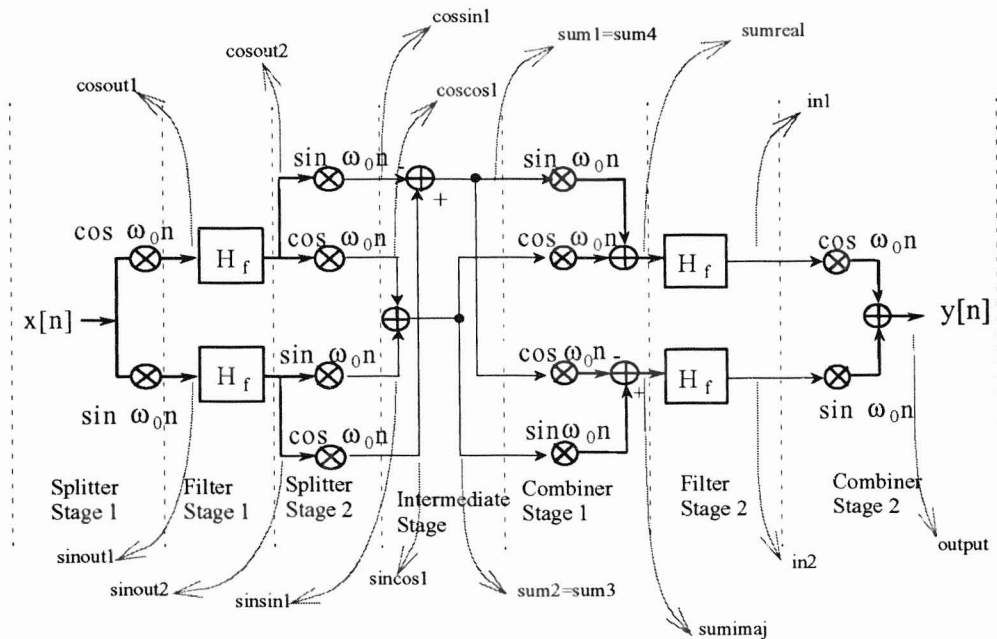
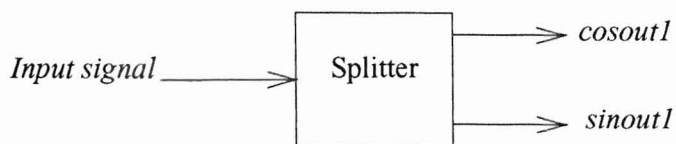
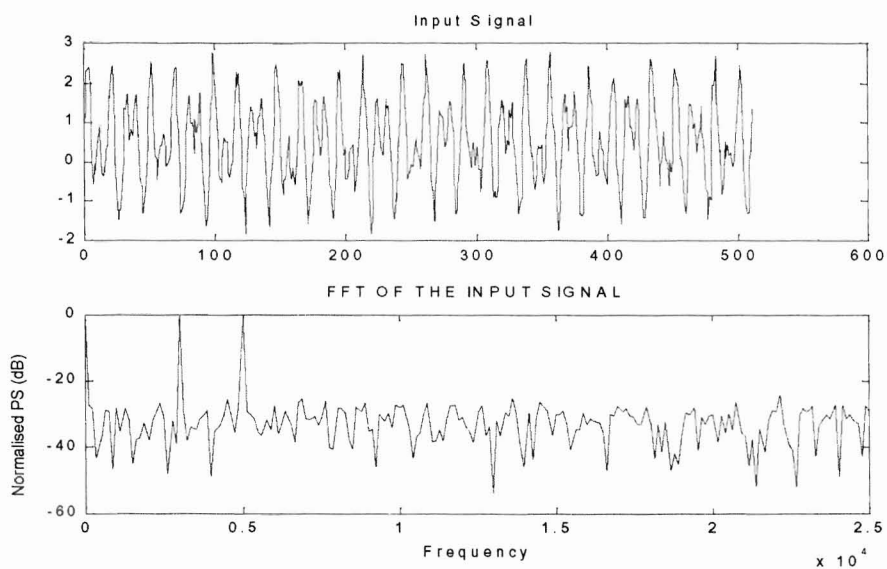


Figure 6-1. Stages of the FTHN Filter structure

Figure 6-2 shows the block diagram of Splitter Stage 1. Figure 6-3 shows the time domain plot of the input signal added with a random noise component and its corresponding 512-point normalized power spectrum plot.

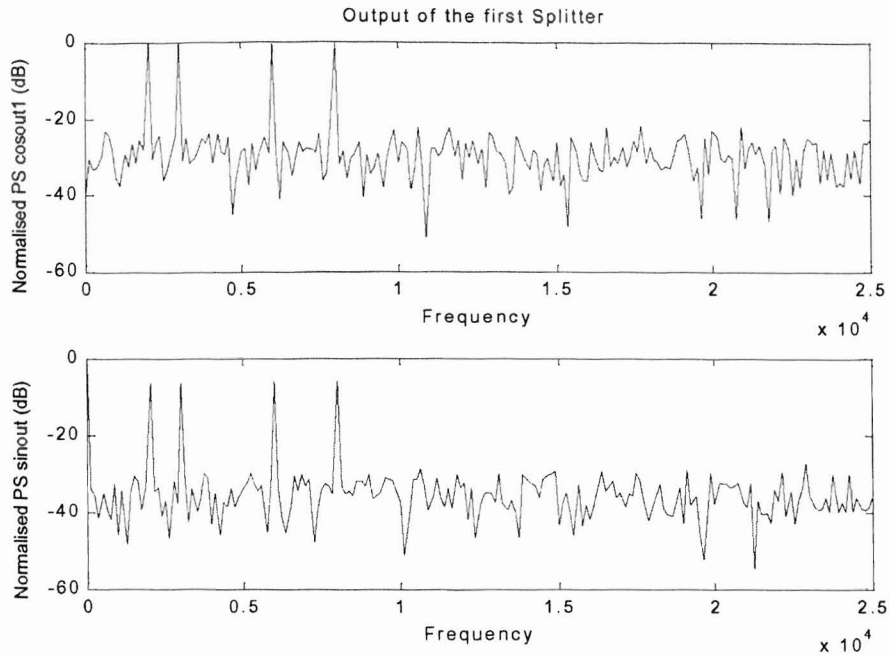


**Figure 6-2.** Block diagram of Splitter Stage 1.



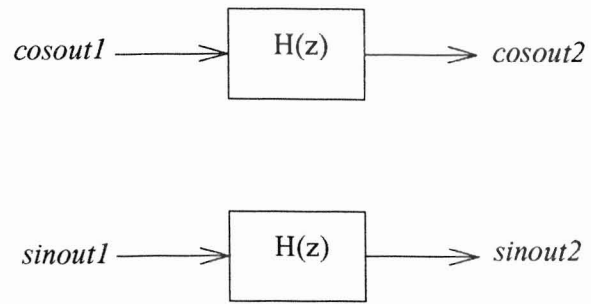
**Figure 6-3.** Time and frequency domain representation of the input signal.

Figure 6-4 shows the normalized power spectrum plot of signals *cosout1* and *sinout1* which are outputs signals from Splitter Stage 1.

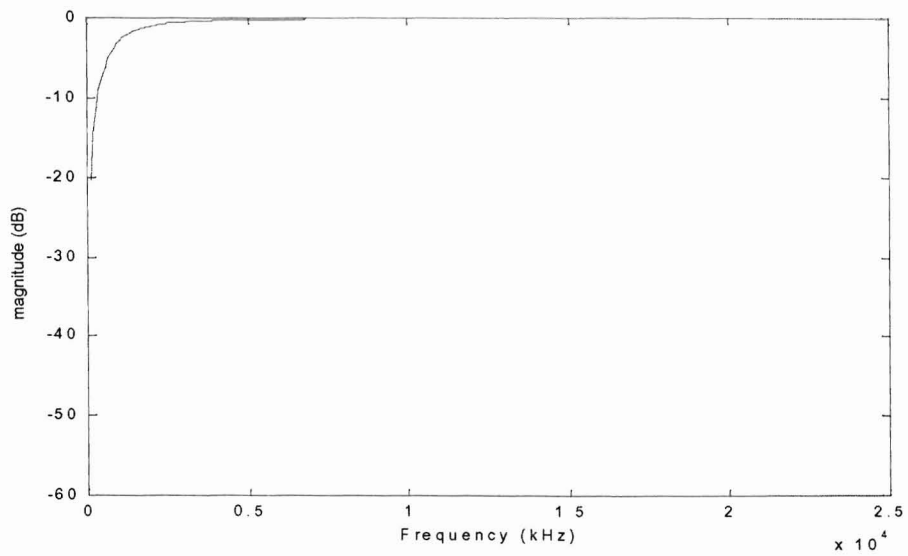


**Figure 6-4.** Normalized Power spectrum plot of *cosout1* and *sinout1*.

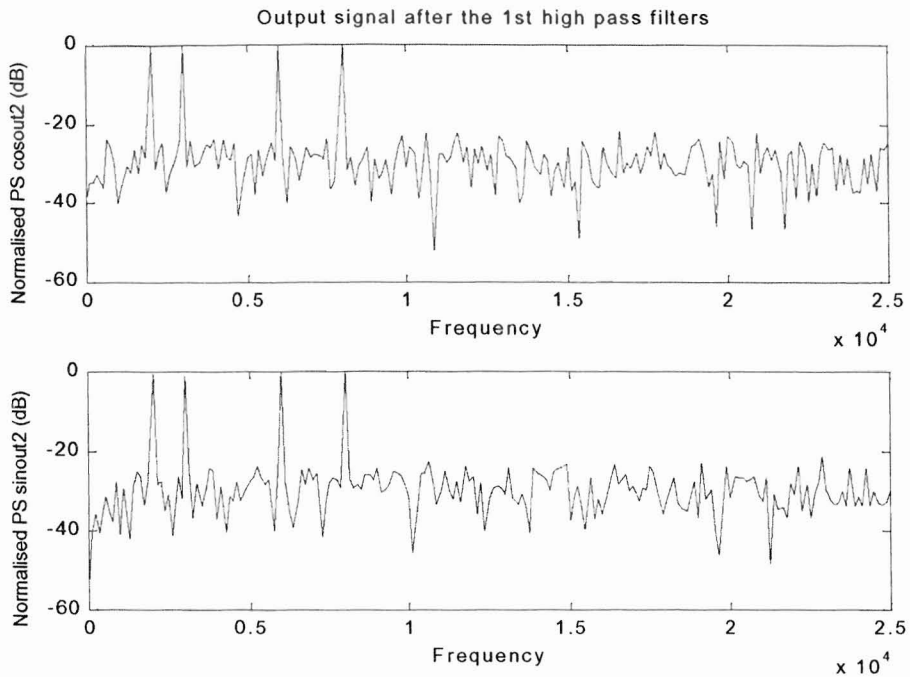
At this stage, the signals *cosout1* and *sinout1* are then passed to Filter Stage 1, which is the 1st order IIR high-pass filter discussed in Chapter 5. The high-pass filter was designed to have a 3dB cutoff frequency of 500Hz which provides a narrow stopband region. Figure 6-5 shows the block diagram of the high-pass filter system when input signals *cosout1* and *sinout1* are passed through it. This operation produces output signals *cosout2* and *sinout2* respectively. Figure 6-6 shows the magnitude vs frequency plot of the 1st order IIR high-pass filter. Figure 6-7 shows the normalized power spectrum plot of signals *cosout2* and *sinout2* coming out of Filter Stage 1.



**Figure 6-5.** Block diagram of Filter Stage 1.

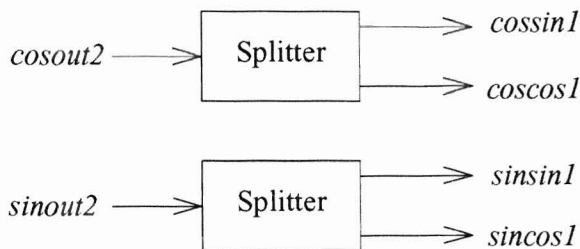


**Figure 6-6.** Magnitude response of IIR high-pass filter with 3dB cutoff frequency of 500Hz.

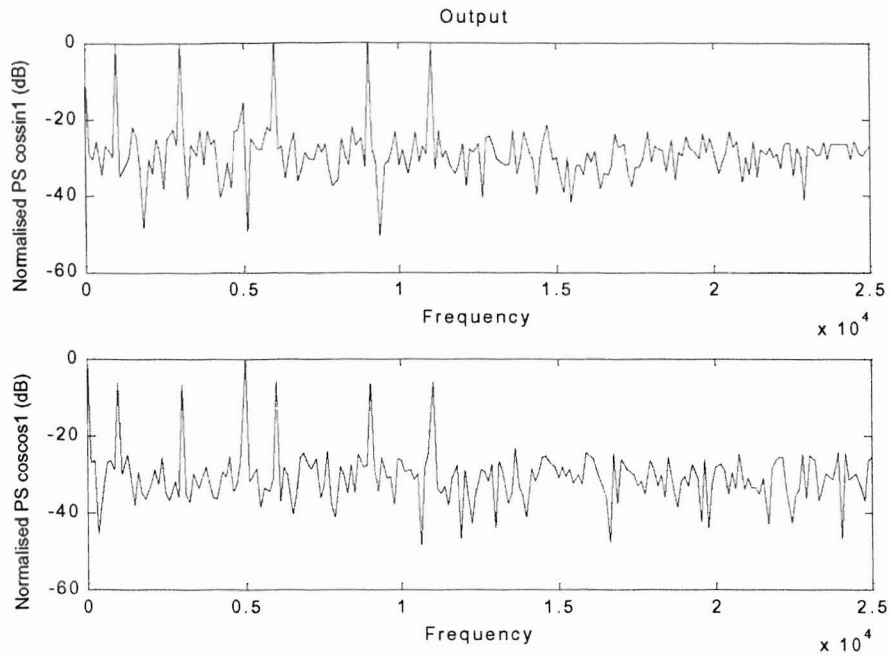


**Figure 6-7.** Normalized power spectrum plot of signals *cosout2* and *sinout2*.

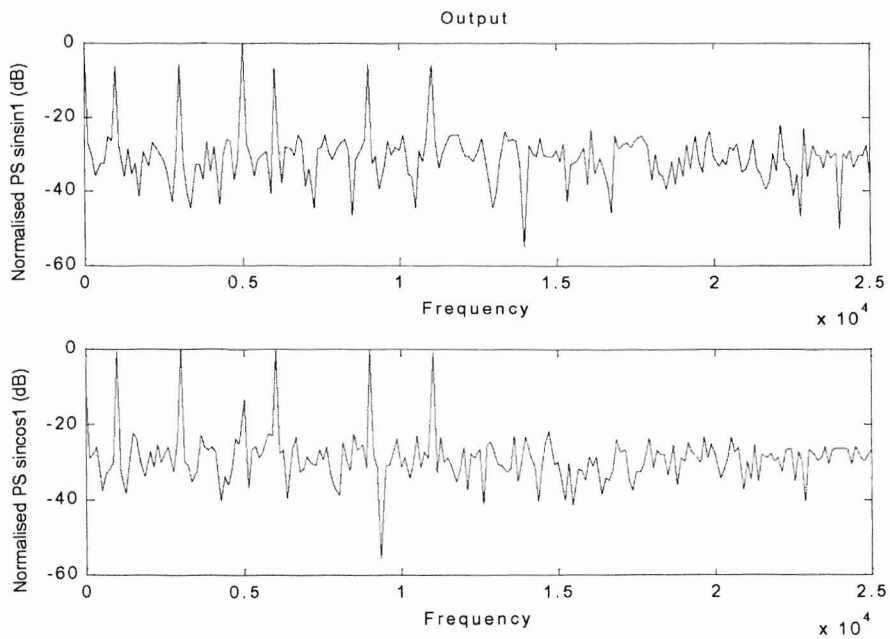
After Filter Stage 1, the outputs *cosout2* and *sinout2* are passed through Splitter Stage 2. Figure 6-8 shows the block diagram of the resulting outputs obtained from passing input signals *cosout2* and *sinout2* into Splitter Stage 2. Figures 6-9 and 6-10 shows the normalized power spectrum plot of the outputs of Splitter Stage 2.



**Figure 6-8.** Block diagram of Splitter Stage 2.



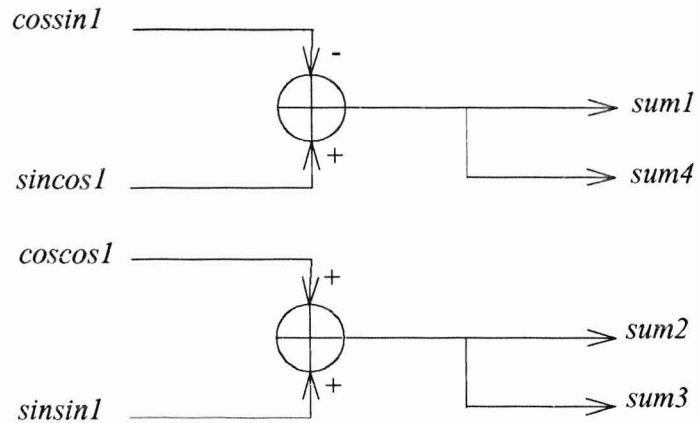
**Figure 6-9.** Normalized power spectrum plot of signals cossin1 and coscos1.



**Figure 6-10.** Normalized power spectrum plot of signals sinsin1 and sincos1.

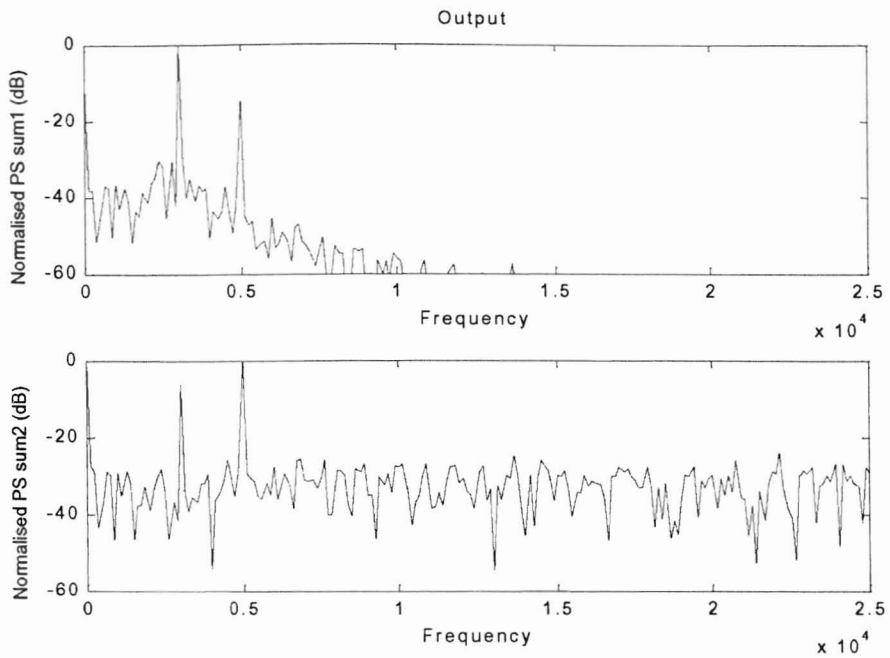


The output signals from *Splitter Stage 2* are then passed to the *Intermediate Stage*. Figure 6-11 shows the block diagram of the resulting outputs obtained when passing the output signals from *Splitter Stage 2* through the *Intermediate Stage*. Figure 6-12 shows the normalized power spectrum plot of the output signals emerging out of the *Intermediate Stage*.

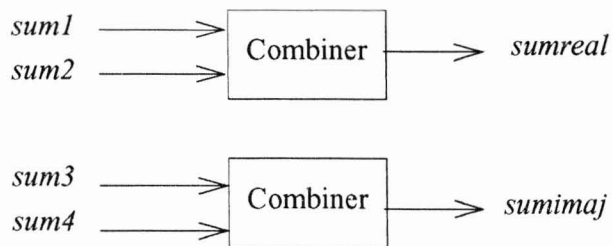


**Figure 6-11.** Block diagram of Intermediate Stage.

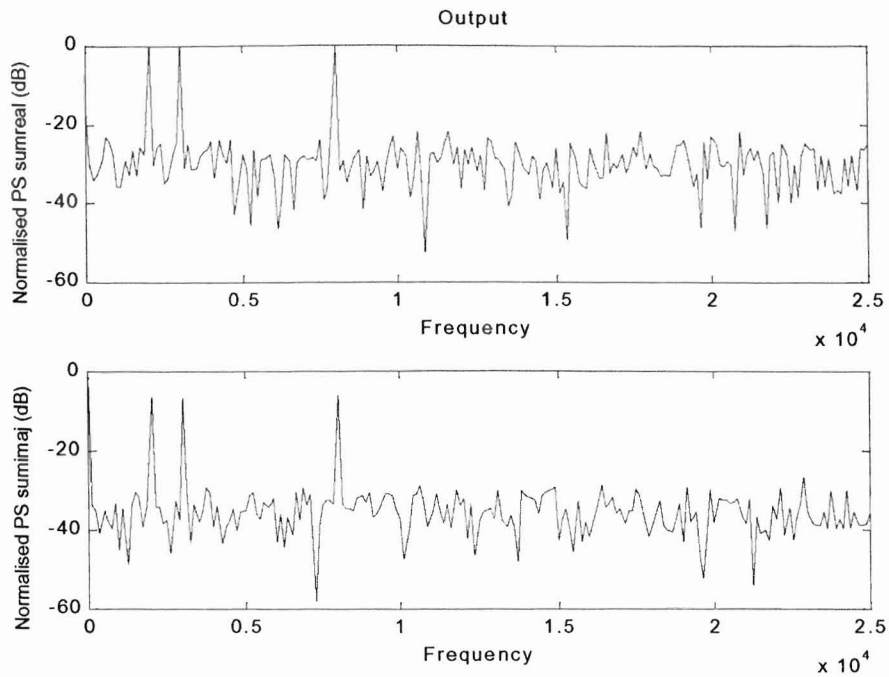
The output signals from the *Intermediate Stage* is then passed to *Combiner Stage 1*. Figure 6-13 shows the block diagram of the resulting outputs obtained when the output signals from the *Intermediate Stage* is passed through *Combiner Stage 1*. Figure 6-14 shows the normalized power spectrum plots of the output signals from *Combiner Stage 1*.



**Figure 6-12.** Normalized power spectrum plot of signals sum1(=sum4) and sum2 (=sum3).

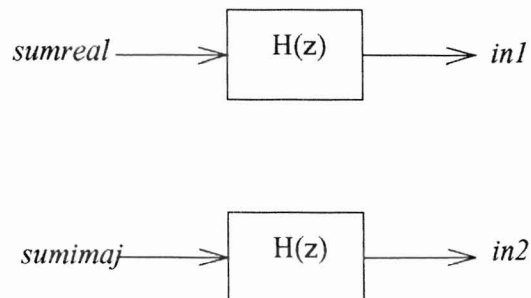


**Figure 6-13.** Block diagram of Combiner stage 1.

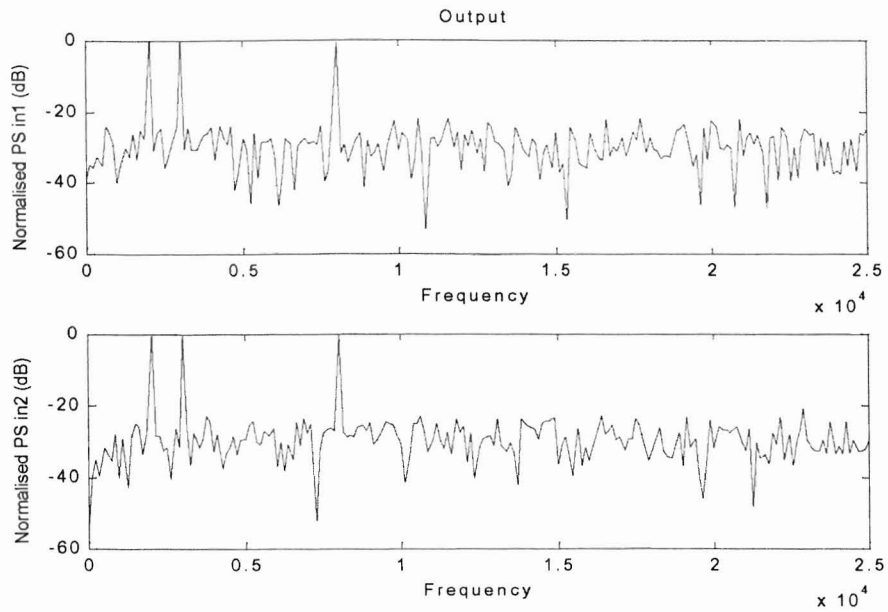


**Figure 6-14.** Normalized Power spectrum plot of signals *sumreal* and *sumimaj*.

The signals *sumreal* and *sumimaj* are then passed through *Filter Stage 2*. Figure 6-15 shows the block diagram of *Filter Stage 2*. Figure 6-16 shows the resulting normalized power spectrum plots of output signals *in1* and *in2* coming from *Filter Stage 2*.

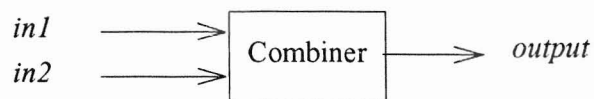


**Figure 6-15.** Block diagram of *Filter Stage 2*.

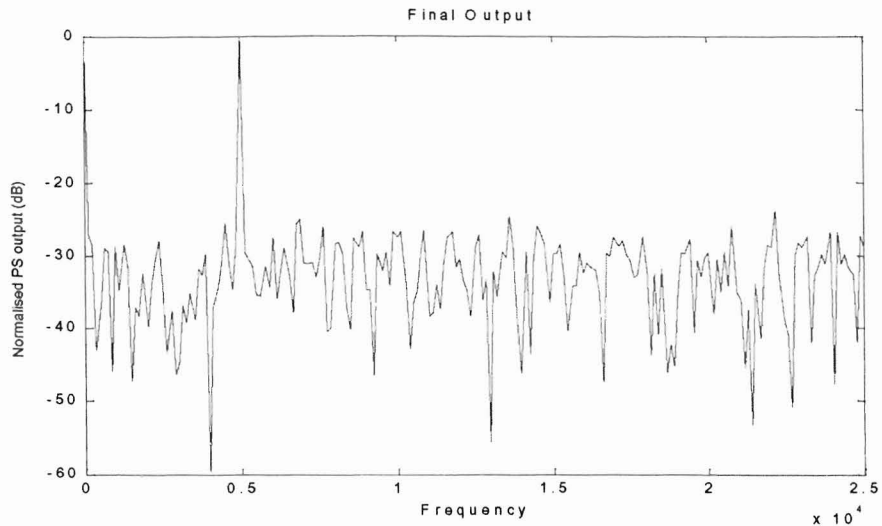


**Figure 6-16.** Normalized power spectrum plot of signals *in1* and *in2*.

Finally, the signals *in1* and *in2* are passed to *Combiner Stage 2*. Figure 6-17 shows the block diagram of *Combiner Stage 2* and Figure 6-18 shows the resulting normalized power spectrum plot of the *output* signal.



**Figure 6-17.** Block diagram of *Combiner Stage 2*.

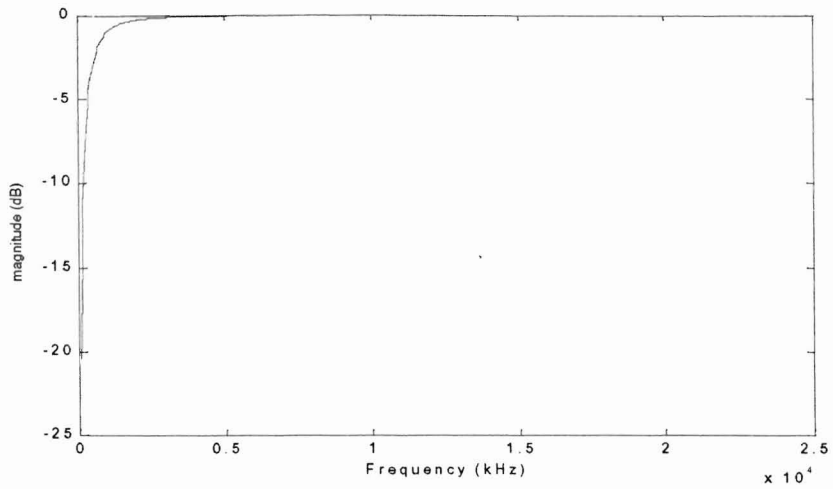


**Figure 6-18.** Normalized power spectrum plot of the output signal.

As observed in Figure 6-18, the simulated NBI signal of 3kHz appears to have been removed from the system, leaving behind only the desired signal at 5kHz, as expected.

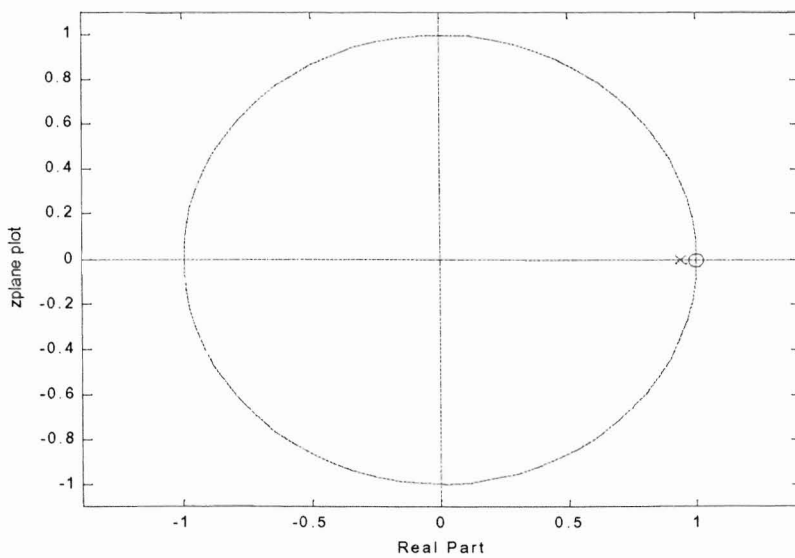
## 6.1 Filter translation

The following section details the MatLab simulation of the virtual translation of the fixed-coefficient IIR high-pass filter in frequency. Figure 6-19 shows the original frequency response of the high-pass filter with a 3dB cutoff frequency of 500Hz prior to translation in the frequency domain.



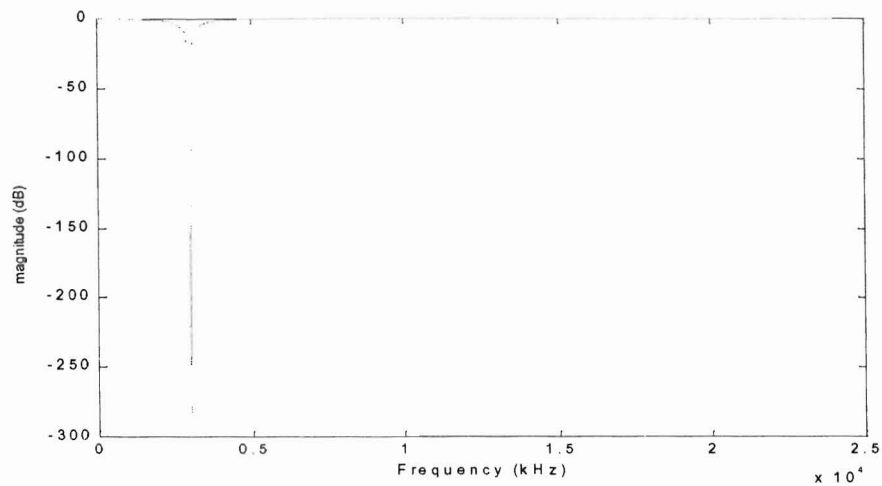
**Figure 6-19.** Frequency response of 1st order high-pass filter with 3dB cutoff at 500Hz.

Figure 6-20 shows the pole-zero plot of the 1st order high-pass filter prior to translation in the frequency domain.

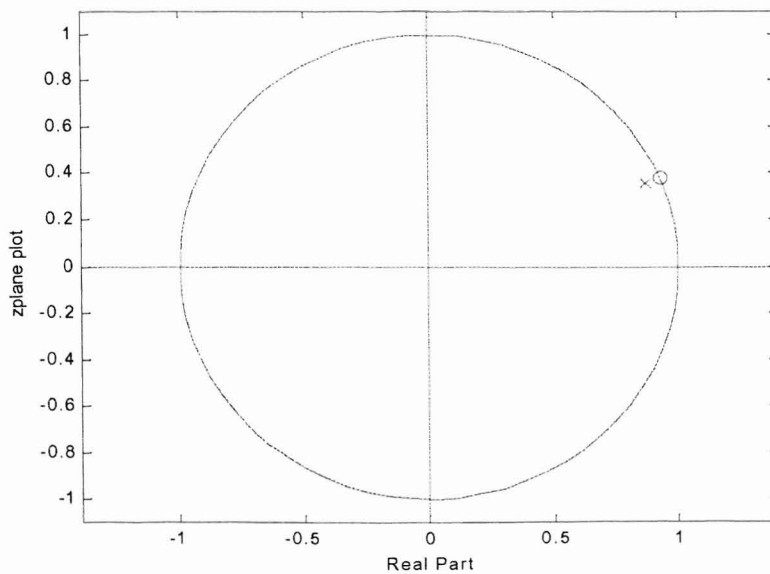


**Figure 6-20.** Pole-zero plot of original IIR high-pass filter prior to translation.

Figure 6-21 shows the filter 'up-translated' in the frequency domain by the heterodyning frequency of 3kHz. Observable is the fact that the high-pass filter turns into a notch filter centered at 3kHz with twice the bandwidth of the stopband region as compared to the high-pass filter.



**Figure 6-21.** Filter up-translated by 3kHz.



**Figure 6-22.** Pole-zero plot of high-pass filter up-translated in frequency.

Figure 6-23 shows the high-pass filter 'down-translated' in frequency by the heterodyning frequency of 3kHz and its relative pole-zero plot is shown on Figure 6-24.

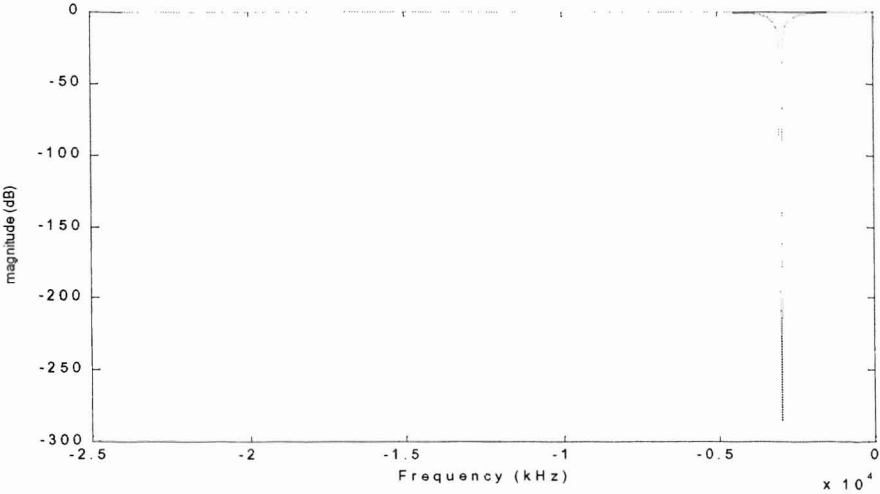


Figure 6-23. Frequency response of high-pass filter downtranslated by 3kHz.

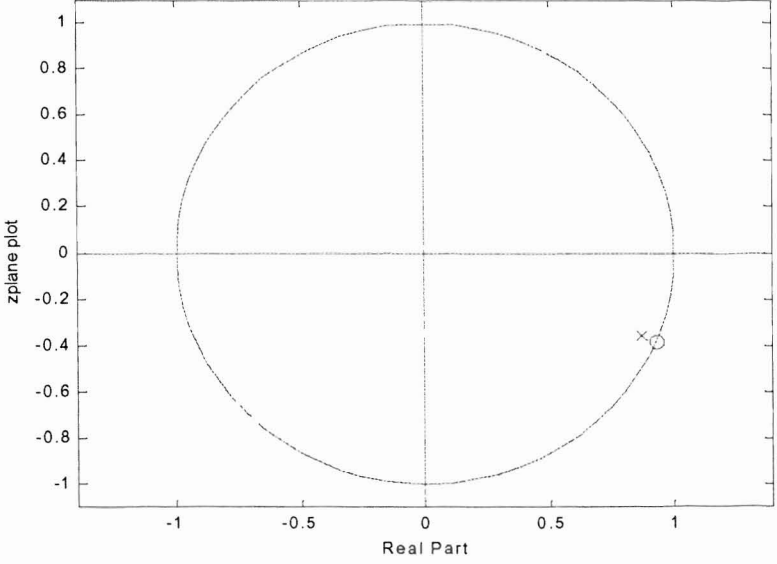
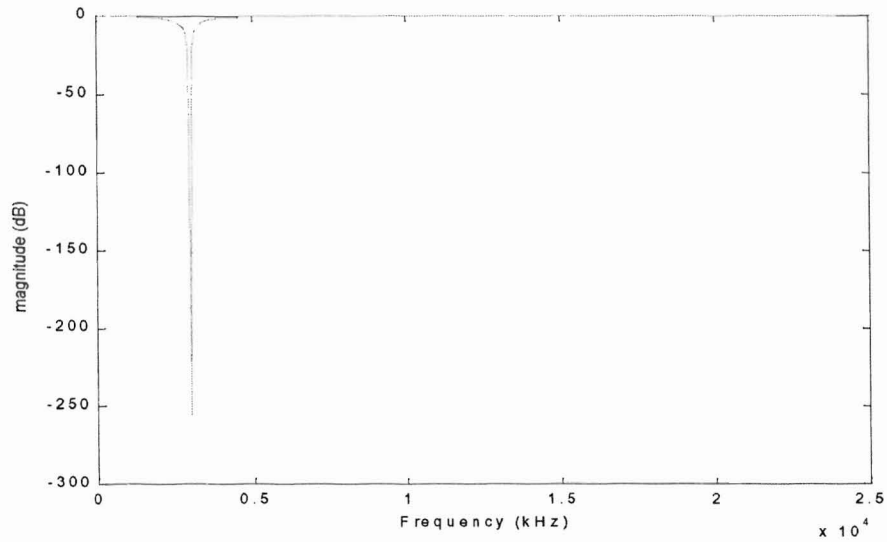


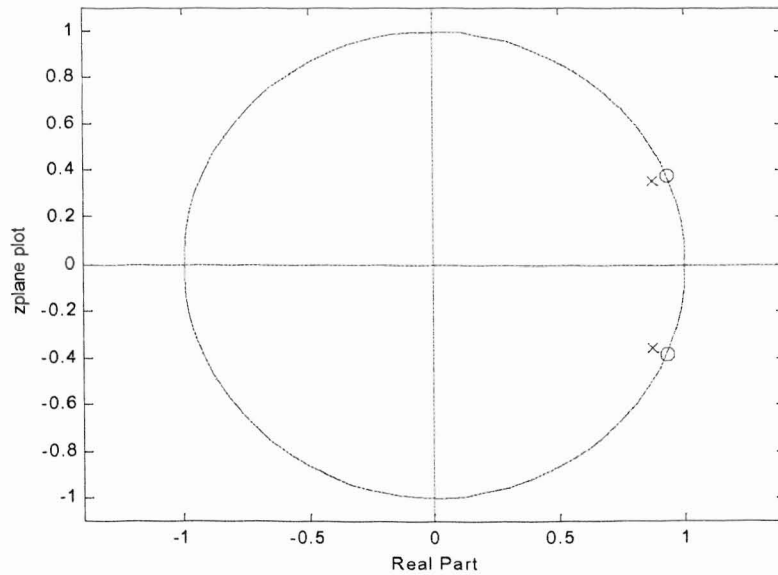
Figure 6-24. Pole-zero plot of filter down-translated by 3kHz.



Finally, the final transfer function of the entire system essentially creates a notch filter centered about the heterodyning frequency, which is 3kHz. Figure 6-25 shows the frequency response plot of the filter derived from the final transfer function in equation 5.10 of Chapter 5.



**Figure 6-25.** Final position of translated filter.



**Figure 6-26.** Final pole and zero positions of translated filter.

The plots shown in Figures 6-25 and 6-26 proves that the FTHN filter system appears to have been translated in frequency in order to eliminate the interference frequency at 3kHz. It is essential to understand that the filter system did not and cannot translate in frequency. In actuality, the undesired signal was translated into the stopband region of the filter, thereby removing it from the transmitted signal. Any and every image created through the heterodyning process have also been eliminated.

# 7 Hardware Choice

The platform for the hardware implementation of the FTHNF structure discussed in this thesis was chosen to be Field Programmable Gate Arrays (FPGA's). The implementation of the design using FPGA's was chosen since it allows for rapid prototyping and with its relatively quick turn-around time, it proves to be an appropriate solution for the initial scratch-work before the design is to be fabricated. Advancements in programmable logic devices families have enabled FPGA's to be used as a viable option for implementation of DSP algorithms and filter structures in the market today. This is primarily due to the fact that the device can be attainable off-the-shelf and its functionality can be changed on-the-fly. For the purpose of this thesis, we will be discussing the structure of the Virtex 2.5V XCV800 Programmable Gate Arrays obtained from Xilinx which is used to test the design presented.

## 7.1 Field Programmable Gate Array (FPGA)

An FPGA is a prefabricated programmable logic device which allows logic implementation by electrically programming the interconnects and personalizing the basic cells within the confines of the user's laboratory. It provides, for many applications, an adequate number of transistors in a single chip package for the functional blocks, switches for the routing network, and the memory capacity to control both. There are 4 main categories of FPGAs in the commercial market today: symmetrical array, row-based, hierarchical Programmable

Logic Device (PLD) and sea-of-gates. The difference between these structures is in the chip-level architecture, granularity of the function unit, method of programming and the interconnection structure [9]. Currently there are four technologies in use which are: static RAM cells, anti-fuse, EPROM transistors, and EEPROM transistors. Depending upon the application, one FPGA technology may have features desirable for that application [8]. The device can implement a given circuit by electrically programming its interconnects and by configuring the functionality of its basic cell structure [5]. The ease of reprogrammability coupled with its hardware emulation status makes the FPGA an ideal choice for any hardware implementation. In the next section, a brief overview of the architectural components of the chosen FPGA is introduced.

## **7.2 Xilinx XCV800 FPGA structure**

A Virtex 2.5V FPGA is basically made-up of 3 components:

1. Configurable Logic Blocks (CLB) which enables the construction of the purposed logic functions.
2. Input/Output Blocks which contribute as the interface between the FPGA and the outside world.
3. Block SelectRAM which complement the LUTs in the CLB.

These components are connected to each other via the programmable interconnects contained within the FPGA package [4].

## **7.3 Configurable Logic Blocks (CLBs)**

The basic underlying structure of an FPGA is in its CLB's. Each CLB is made up of 4 Logic Cells (LC), each of which includes a 4-input function generator/Look-up table (LUT), carry logic and a storage element. As a note, the Virtex architecture is actually divided into smaller parts known as 'Slices'. Each CLB contains 2 Slices. Figure 7-1 shows the Virtex CLB [4].

### **7.3.1 Function Generator/Look-Up table (LUT)**

The function generators available in the Virtex series FPGA are implemented as 4-input LUTs. Each LUTs therefore serves the purpose of storing the truth table for the desired implementation of up to 4 variables or when combined, can implement the truth table of up to 6 variables. These LUTs can also be configured to serve as synchronous 16x1-bit Random Access Memory (RAM) or when combined, can also implement a 32x1-bit RAM (or a 16x2-bit RAM) [4].

### **7.3.2 The Arithmetic Logic**

Each CLB contains dedicated carry logic to provide fast arithmetic carry capabilities needed when implementing high-speed arithmetic functions. The Virtex CLB supports 2 separate carry chains, one per Slice. The dedicated carry path can also be used to cascade function generators for implementing wide logic functions [4].

### 7.3.3 Storage Elements

Each Slice in the Virtex CLB contains 2 storage elements which can be configured as D-type flip-flops or as level-sensitive latches. The inputs to the D flip-flops can be driven either by the function generators within the slice or directly from slice inputs, bypassing the function generators [4].

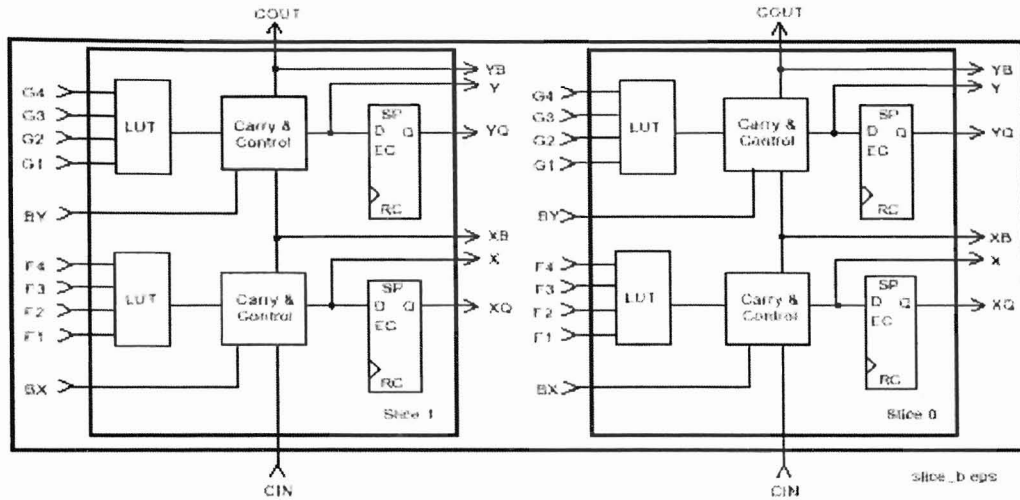


Figure 7-1. 2 Slice Virtex CLB

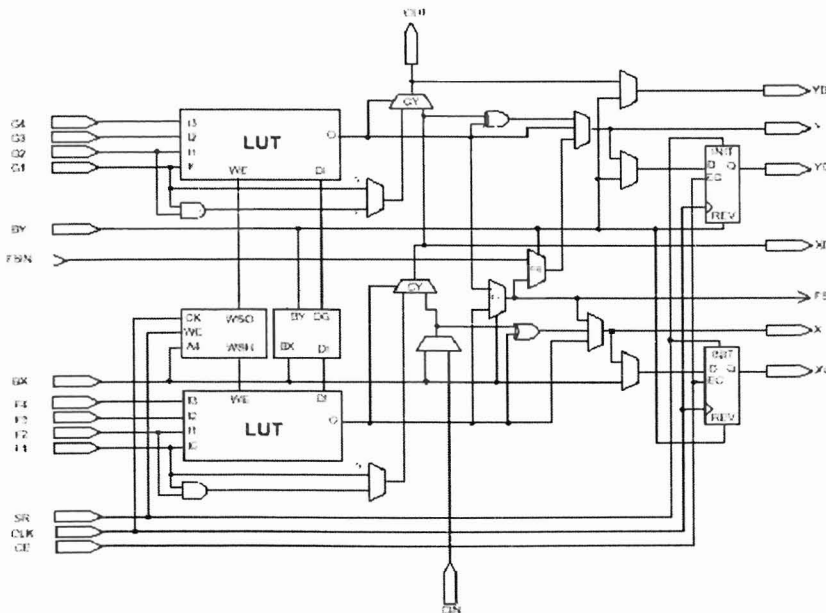


Figure 7-2. Detailed view of the Virtex Slice.

## 7.4 Input/Output Blocks

The Input/Output Blocks (IOBs) in the Virtex Series FPGA provide an interface between the external package pins and the internal logic of the FPGA. Each of the IOB controls one package pin and can be configured for Input, Output or Bi-directional signals. Figure 7-3 shows the simplified block diagram of the IOB [4].

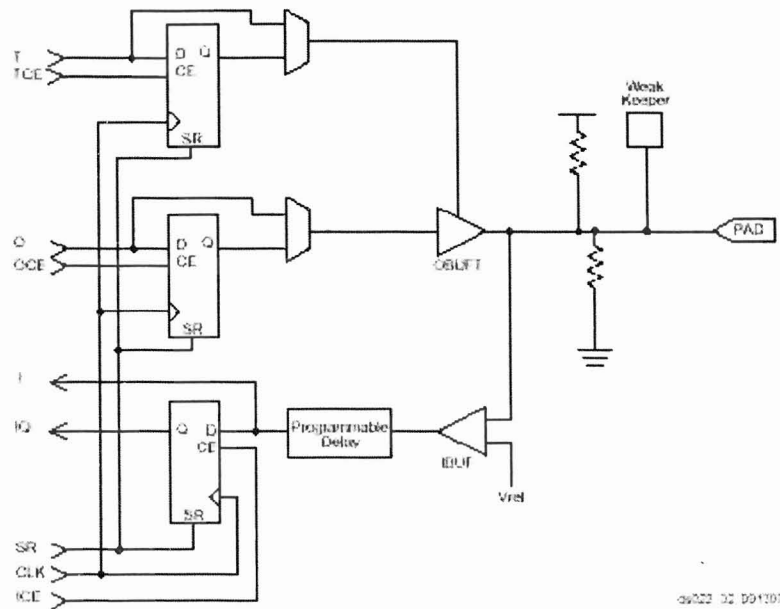


Figure 7-3. Virtex Input/Output (I/O) Block

## 7.5 Block SelectRAM

Virtex FPGAs incorporate several large Block SelectRAM memories. These Block SelectRAMs complement the distributed LUT SelectRAMs that provide shallow RAM structures implemented in CLBs. Each Block SelectRAM cell is a fully synchronous dual-ported 4096-bit RAM with independent control signals for each port [4].

## 7.6 CLB Configuration

To configure a FPGA for a given application, the design needs to be translated into configuration data. The configuration data is obtained by converting the netlist produced during the design stage into a bitstream file using a recognized development software for the given device. The CLBs are then configured by loading the bitstream file into the internal configuration memory. The method of configuring the FPGA determines the type of bitstream file. FPGAs can be configured by a PROM. The serial PROM is the most common. The FPGA can either actively read its configuration data out of external serial or byte-parallel PROM (master mode), or the configuration data can be written into the FPGA (slave and peripheral mode) [8]. Since the device Virtex XCV800 device is based on the SRAM technology, the configuration bits are stored in the RAM. The outputs of the RAMs controls the switches which implements the connectivity and functionality of the design in hardware.



# 8 Hardware Implementation

We will begin the discussion of the hardware implementation of the FTHN filter system with the 8-bit sine and cosine generator followed by the 8-bit 2's complement registered-multiplier. These 2 components are the foundation of the entire heterodyne filter structure presented in this thesis.

The structures discussed in this Chapter have been implemented using two separate reconfigurable logic tools. A third-party synthesis tool from Synplicity called *Synplify Pro* was used for the front-end portion of the design phase and compared to the integrated tool provided by Xilinx, called *Foundation Series*. For the back-end portion of the design phase, the integrated development tool from Xilinx was used to map, place and route and configure the design onto the FPGA chip. Along the course of this chapter, a comparison of the 2 reconfigurable logic tools is provided to demonstrate the hardware reduction capable when using Synplify Pro for the front-end phase as compared to the Xilinx integrated tool. This is due to the fact that the synthesis tool performance of *Synplify Pro* greatly outweighs the performance of the integrated development tool from Xilinx [12].

## 8.1 Sine/Cosine Generator

The Sine/Cosine generator functions to produce both  $\sin(\theta)$  and  $\cos(\theta)$  simultaneously for multiplication with the incoming signal. Theta ( $\theta$ ) for the sine and cosine values were pre-calculated using the following expression [11]:

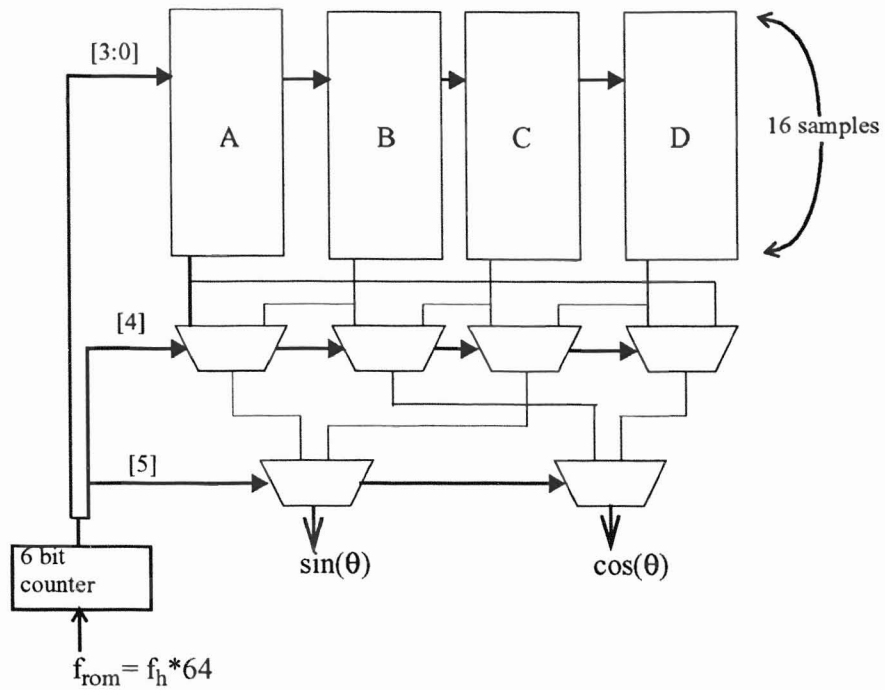
$$\theta = \frac{2\pi \cdot n}{2^m} \quad (8.1)$$

$n$  = # of samples per period

input width ( $m$ ) =  $\log_2(n)$

and stored in a Look-up Table (LUT). Theta ( $\theta$ ) in equation 8.1 represents all values in radians for  $n$  number of samples needed to represent a sine or cosine wave of one period. These values are converted into  $m$ -bit, twos complement numbers and stored in the LUTs.

For the purposes of this thesis, we've opted to have 64 samples to represent an entire sine/cosine wave. Since the sine wave values trails the cosine wave values by  $90^\circ$  (a quarter wave), we can produce both the sine and cosine values simultaneously from these 64 samples simply with the use of multiplexers. This is because these 64 samples are actually divided into 4 look-up tables containing 16 ( $2^4$ ) samples, equivalent to a quarter-wave, of the pre-calculated values of  $\sin(\theta)$  and  $\cos(\theta)$  within the sine/cosine generator structure. A simple 6-bit (4-bits for addressing and 2-bits for multiplexer control) counter can be used to address the 64 sample values and at the same time provide the control logic to produce the sample values out of the separate sine and cosine branches. Figure 8-1 shows the block diagram of the sine/cosine generator structure.



**Figure 8-1.** Sine/Cosine generator

The frequency of the sine and cosine signal produced by the sine/cosine generator is determined by its input frequency and the total number of samples per period,  $n$ , stored within the structure. The following equation relates the heterodyning frequency,  $f_h$ , to the input frequency of the sine/cosine generator,  $f_{rom}$ :

$$f_{rom} = f_h \cdot n \tag{8.2}$$

where  $n=64$  in this case

Figure 8-2 shows the synthesized circuit for the sine/cosine generator using *Synplify Pro*.

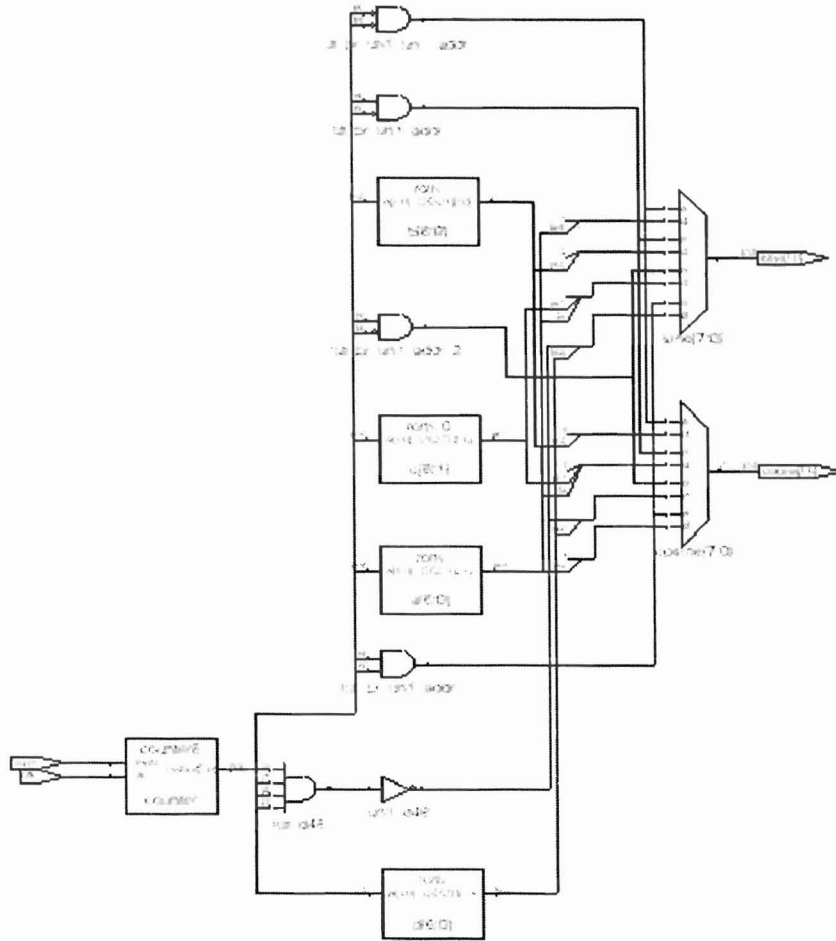
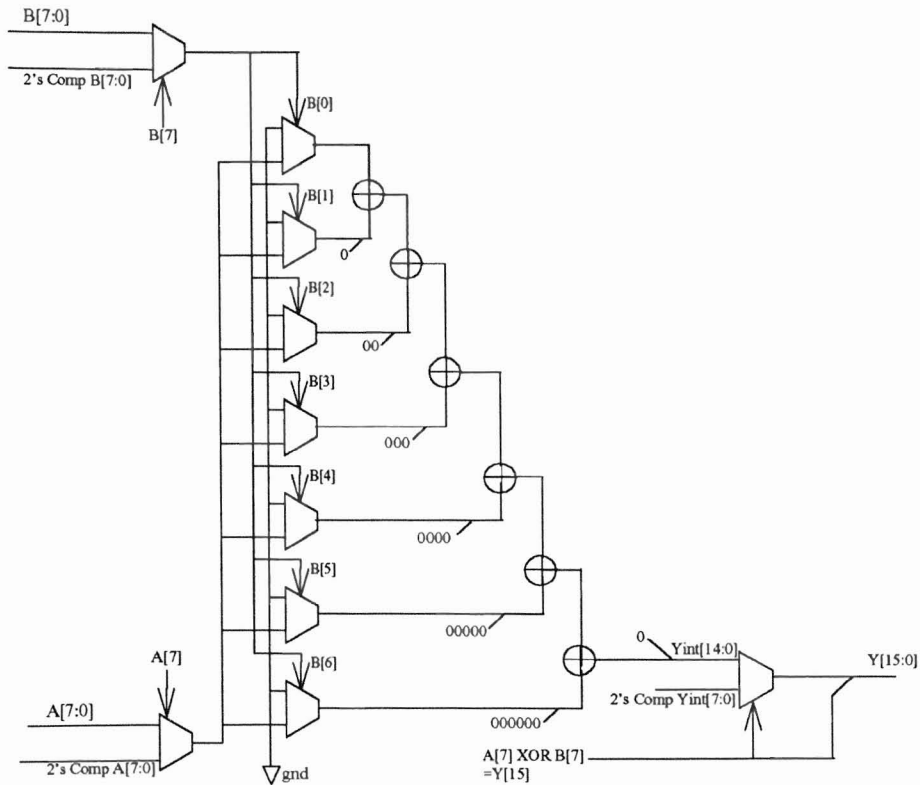


Figure 8-2. Synthesized sine/cosine generator circuit using Synplify Pro.

## 8.2 8-bit 2's Complement Multiplier

The next significant part of the splitter and combiner structures is the multiplier circuit. The multiplier in the FTHN filter system functions to multiply an incoming signal with the output from the sine/cosine generator at the heterodyning frequency,  $f_h$ . This operation immitates the analog mixer circuit commonly found in many analog communication struc-

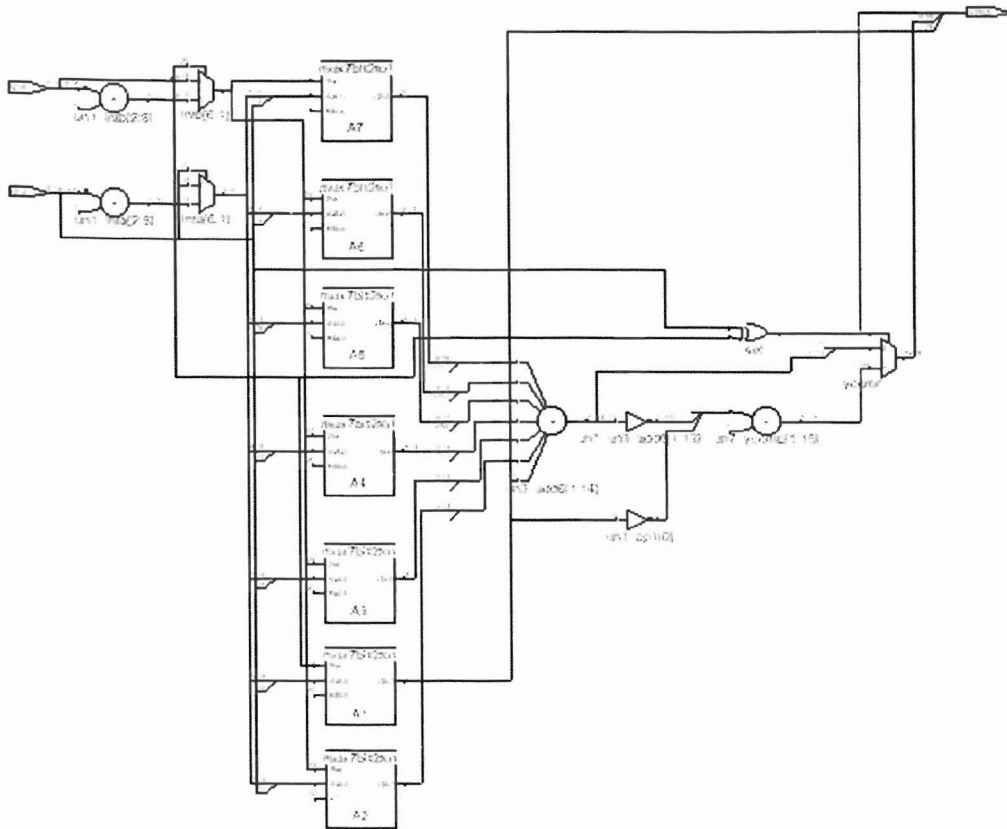
tures. Since the output of the sine/cosine generator and the incoming digital signal are in 2's complement format, we'll have to design a 2's complement multiplier to perform the multiplication process. To do so, we can easily convert an 8-bit sign-magnitude multiplier into an 8-bit 2's complement structure to serve as required, at the cost of some hardware increase which is trivial. Figure 8-3 shows the block diagram for the proposed 8-bit, 2's complement multiplier.



**Figure 8-3.** 8-bit 2's complement multiplier block diagram.

8-bit registers were then placed at inputs and output of the 8-bit 2's complement multiplier in order to ensure synchronization between the values produced by the sine/cosine generator and the incoming signal. The resulting structure was implemented using *Synplify Pro*

for the front-end phase of the design. Figure 8-4 shows the the 8-bit, 2's complement multiplier created using *Synplify Pro*.

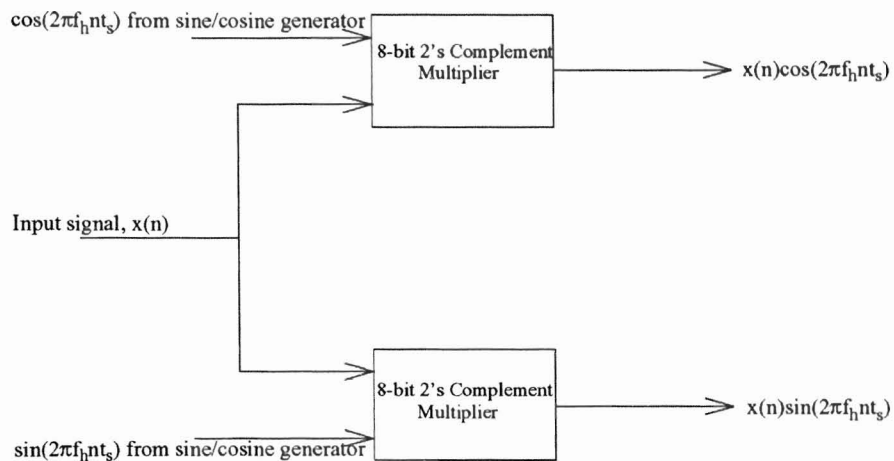


**Figure 8-4.** 8-bit, 2's complement multiplier implementation using Synplify Pro.

### 8.3 Splitter

The Splitter circuit serves to split the incoming signal into 2 branches which is the in-phase branch and the quadrature branch. This structure functions to multiply the output signal from the sine/cosine generator which is at the heterodyning frequency,  $f_h$ , with any incoming signal to produce the sum and difference frequencies between the two signals. A previous implementation places the sine/cosine generator within the Splitter circuit which would

imply that a sine/cosine generator is incurred for each Splitter or Combiner in the circuit [2]. Since the heterodyning frequency produces both a sine and a cosine for any desired frequency, considerable hardware can be saved by making the sine/cosine generator explicit to the Splitter and Combiner circuits. This way, only one sine/cosine generator would be needed for the entire FTHN filter system. Figure 8-5 shows the block diagram of the splitter circuit.



**Figure 8-5.** Block diagram of the Splitter circuit.

Figure 8-6 shows the Splitter circuit implemented using Synplify Pro. Input ports *insin* and *incos* shown in Figure 8-6 represents the connection to the *sine* and *cosine* output ports of the sine/cosine generator respectively. As mentioned before, there is no need to duplicate the sine/cosine generator for each Splitter or Combiner component in the FTHN filter system resulting in considerable savings in terms of hardware.

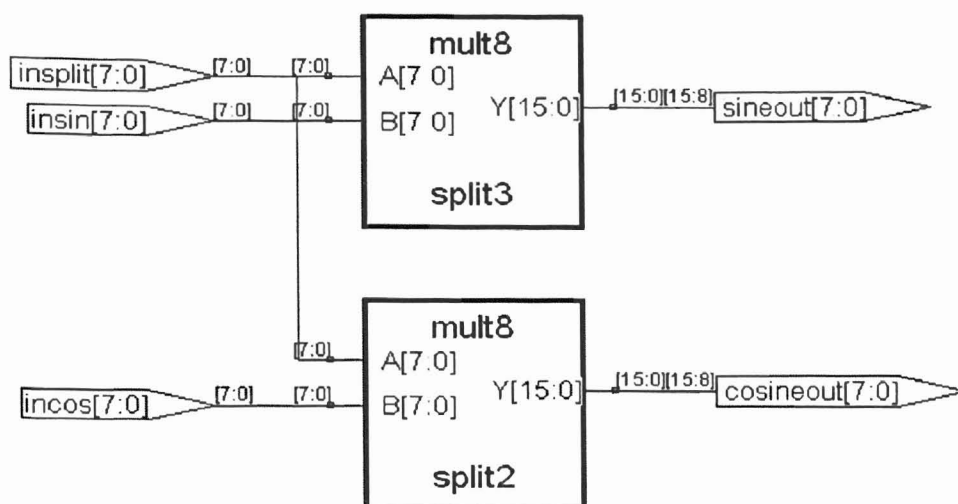


Figure 8-6. Synthesized Splitter circuit using Synplify Pro.

## 8.4 Fixed-coefficient Prototype Filter

As mentioned before, the first-order high-pass filter can be described using the following transfer function:

$$H(z) = \frac{k(z^{-1} - 1)}{z^{-1} - r}, \text{ where } k = 0.9683 \text{ and } r = 0.9366 \quad (8.3)$$

It is obvious from Equation 8.3 that the coefficients  $k$  and  $r$  will need to be implemented using fixed multipliers. Assuming an 8-bit number implementation, the closest binary representation of the two coefficients are as follows:

$$k = 0.1111100 = 0.96875$$

$$r = 0.1111000 = 0.93750$$



To implement these coefficients directly, the  $k$  coefficient would require 4 adders, while the  $r$  coefficient would require 3 adders. However, a more hardware-efficient implementation of these multipliers can be obtained using the Canonic Sign Digit (CSD) representation of the coefficients coupled with the Dempster-McLeod (DM) method for constant integer multiplication using a minimum adders. The CSD representation of  $k$  and  $r$  are:

$$k = 1.0000(-1)00 = 1 - (2^{-5}) = 0.96875$$

$$r = 1.000(-1)000 = 1 - (2^{-4}) = 0.93750$$

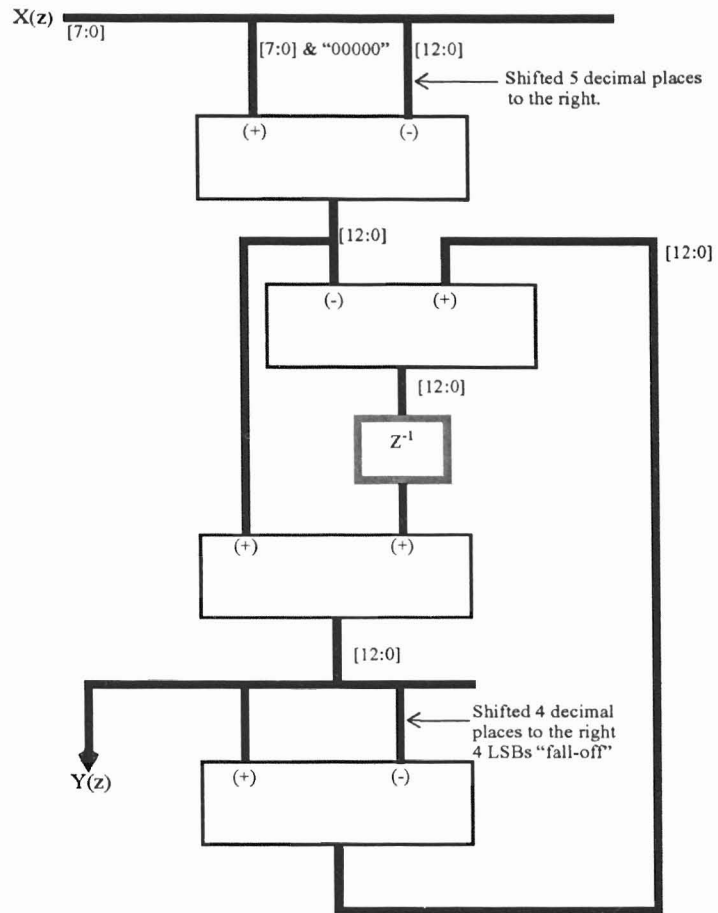
To further explain the implementation of these coefficients as fixed multipliers, let's assume that our 8-bit input is given as  $X$ . Therefore, the multiplication of our input  $X$  with either  $k$  or  $r$  can be described as:

$$kX = (1 - 2^{-5})X = X - (2^{-5})X$$

$$rX = (1 - 2^{-4})X = X - (2^{-4})X$$

where  $(2^{-5})X$  is simply the binary input shifted 5 decimal places to the right and  $(2^{-4})X$  is the binary input shifted 4 decimal places to the right.

Figure 8-7 shows the block diagram of the 1st-order high-pass filter implementation. Each adder/subtractor in Figure 8-7 is made 13 bits wide in order to accommodate the shifted bits in the structure. The final output,  $Y$ , is truncated to 8-bits.



**Figure 8-7.** Block diagram of 1st-order high-pass filter

Figure 8-8 shows the 1st-order high-pass filter implementation using *Synplify Pro*.

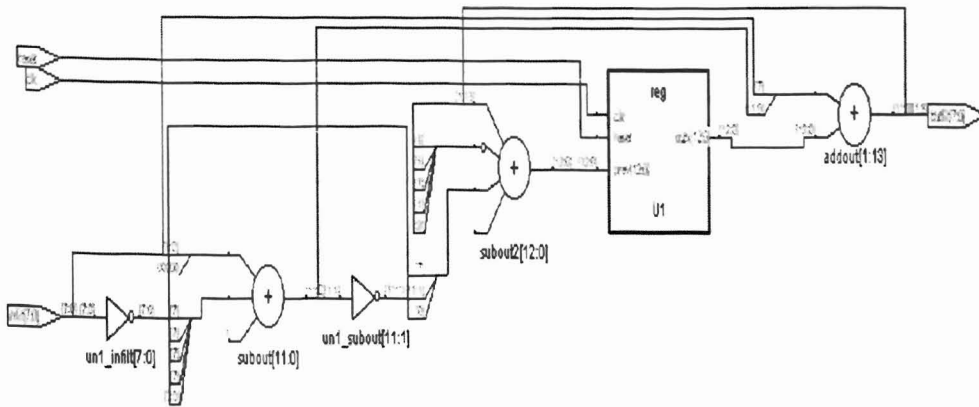


Figure 8-8. Synthesized 1st order high-pass filter using Synplify Pro.

## 8.5 Combiner

The Combiner structure multiplies incoming signals with the outputs of the sine/cosine generator at the heterodyning frequency,  $f_h$ , and subsequently sums the results created from the multiplication process. Figure 8-9 shows the block diagram of the Combiner structure.

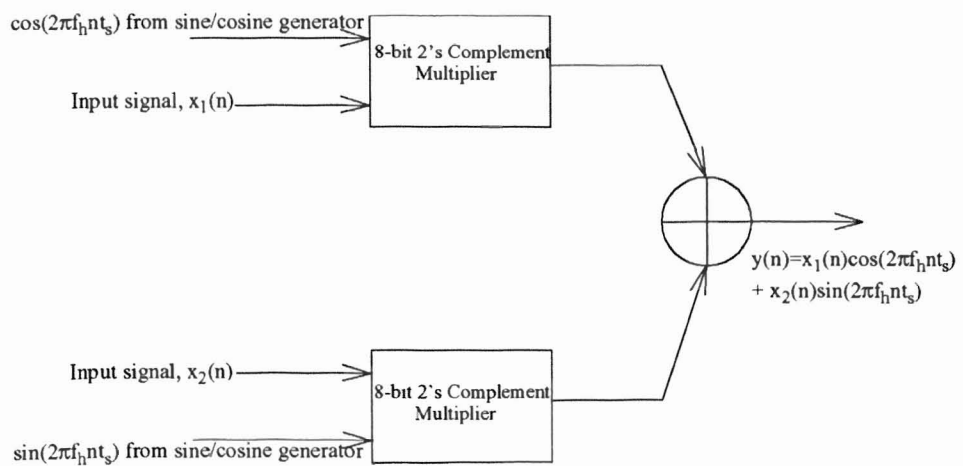


Figure 8-9. Block diagram of Combiner.

Figure 8-10 shows the Combiner structure implemented using Synplify Pro.

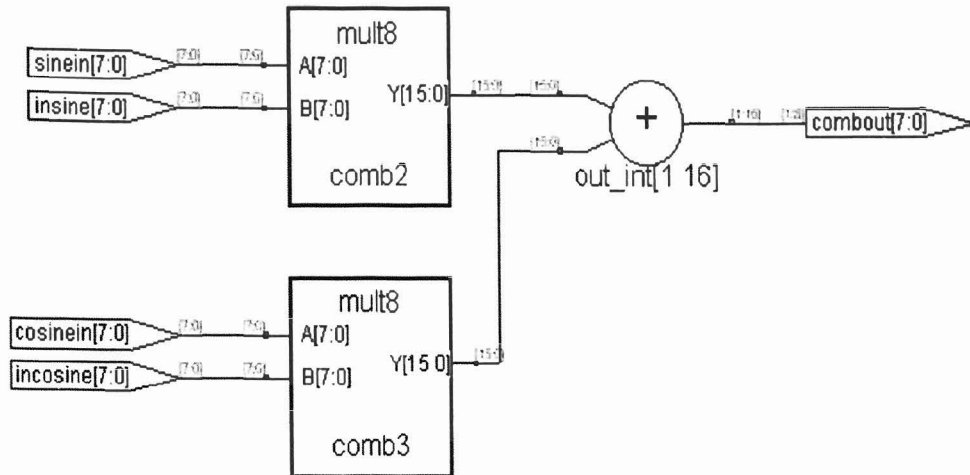


Figure 8-10. Synthesized structure of Combiner using Synplify Pro.

## 8.6 Final Structure

The final structure of the Fully Tunable Heterodyne Notch Filter is shown in Figure 8-11.

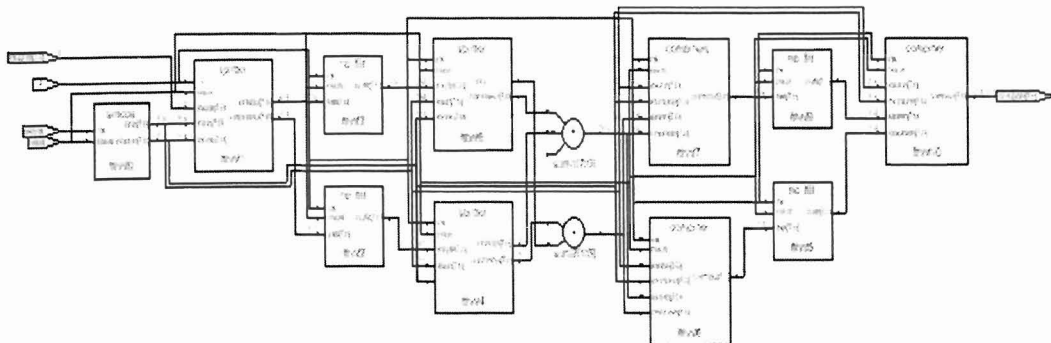


Figure 8-11. Final synthesized structure of FTHN filter using Synplify Pro.

## 8.7 Resource Usage

The following tables lists the resource usage for each of the implementations mentioned in this Chapter. A comparison between *Xilinx* and *Synplify Pro* is shown to further stress the efficiency of using Synplify Pro's synthesis tool compared to the integrated development tool offered by Xilinx. Table 8-1 compares the slice count for the sine/cosine generator and the multiplier.

**Table 8-1.** Resource usage comparison for sine/cosine generator and multiplier.

Component	# of Slices using Xilinx's Foundation Series	# of Slices using Synplicity's Synplify Pro
Sine/Cosine generator	36	31
8-bit 2's complement Multiplier	54	53
(Total # of Slices)	90	84

**Table 8-2.** Resource usage comparison for FTHN filter components.

Component	# of Slices using Xilinx's Foundation Series	# of Slices using Synplicity's Synplify Pro
Splitter	103	106
High-Pass Filter	22	34
Combiner	115	108
(Total # of Slices)	240	238

**Table 8-3.** Resource usage comparison for entire structure.

Component	# of Slices using Xilinx's Foundation Series	# of Slices using Synplicity's Synplify Pro
Fully Tunable Heterodyne Notch Filter (Total # of Slices)	1073	817

# 9 Final Results

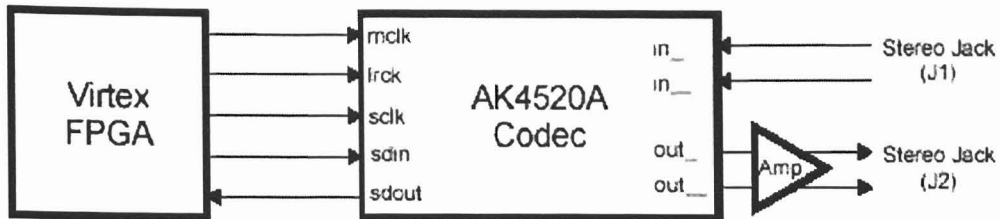
## 9.1 Experimental Results

The Fully Tunable Heterodyne Notch (FTHN) filter structure was implemented using the Xilinx Virtex 800 family FPGA chip. This section provides the setup and the analyses of the final results obtained as compared to the simulation results shown earlier in Chapter 6, using MatLab.

### 9.1.1 Experimental Setup

The FTHN filter structure was downloaded and configured through a PC parallel port onto a Xilinx XSV-800 V1.1 board. This board contains the Virtex 800 FPGA chip embedded in a framework for processing video and audio signals [13]. The board also has the capability of processing stereo audio signals with up to 20 bits of resolution and a bandwidth of 20 KHz using its AK4520A stereo codec, which is external to the FPGA. This option allows us to input an analog signal from a function generator which will then be converted into a bit stream by the codec for use by the FPGA [14]. This conversion process is controlled via the interface shown in Figure 9-1. The interface aids in transforming the incoming analog signal into a serial bit stream for use by a particular design contained within the FPGA chip. The serial bit streams produced when an analog signal is digitized will be syn-

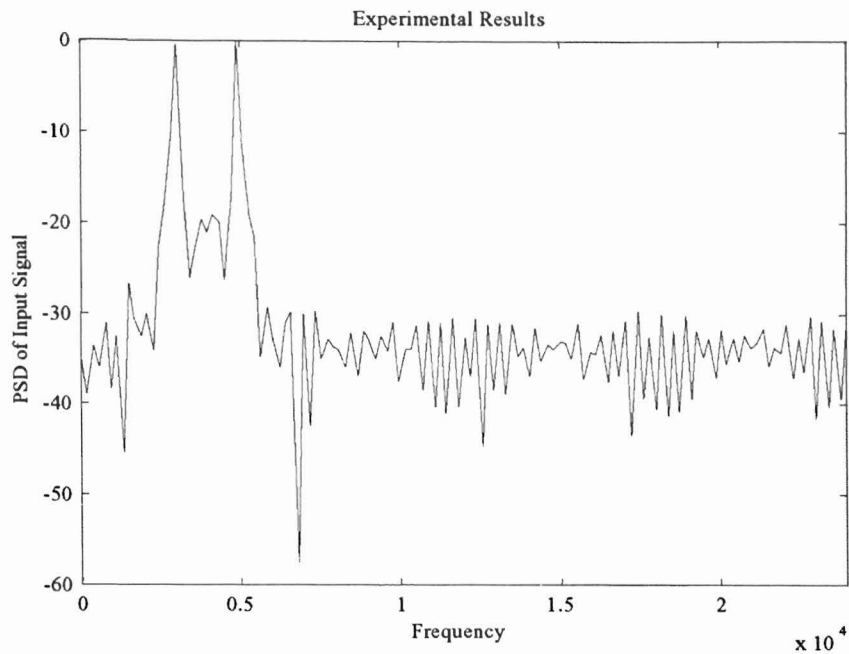
chronized with a clock from the FPGA that enters the codec on the SCLK signal shown in Figure 9-1 [14].



**Figure 9-1.** Interface between codec and FPGA on the XSV-800 V1.1 board.

However, in order for the interface between the FPGA and the codec to work accordingly, a handshaking module will need to be instantiated within the high-level vhdl code for the design. This handshaking module creates the necessary control signals shown in Figure 9-1 above, in order for the codec to perform the proper Analog-to-Digital (A/D) or Digital-to-Analog (D/A) conversions.

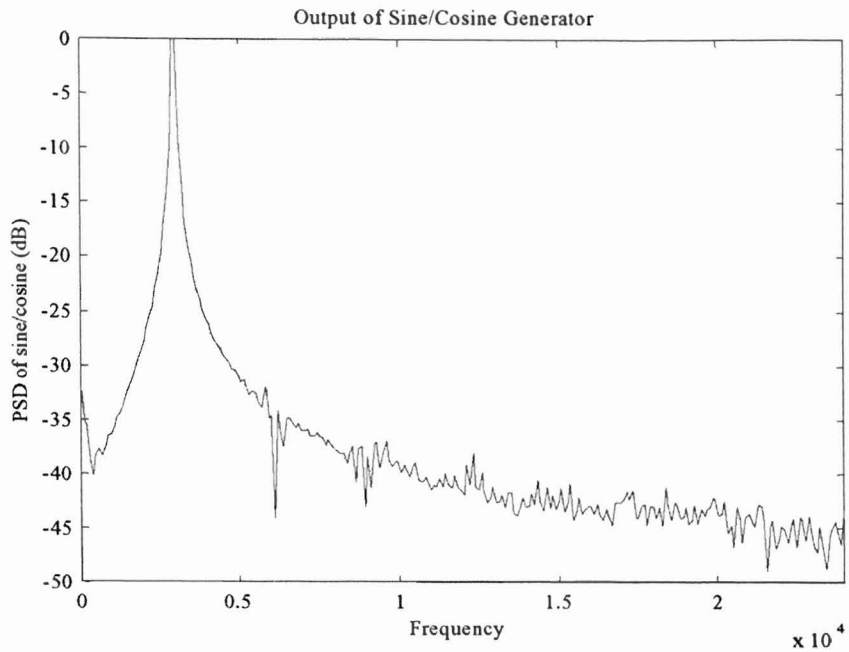
The output of the design was recorded using the HP *LogicWave* Logic Analyzer. In order to observe and compare the power spectrum plots obtained from the experiments with those obtained using MatLab, these 8-bit output values were first converted to its equivalent fractional numbers before performing a 512-point Fast Fourier Transform (FFT) on them. The post-FFT values were then normalized on the magnitude dB scale and plotted against frequency to obtain the Power Spectrum vs. Frequency plots. Figure 9-2 shows the power spectrum plot of the input signal.



**Figure 9-2.** 512 point, normalized Power spectrum plot of the input signal.

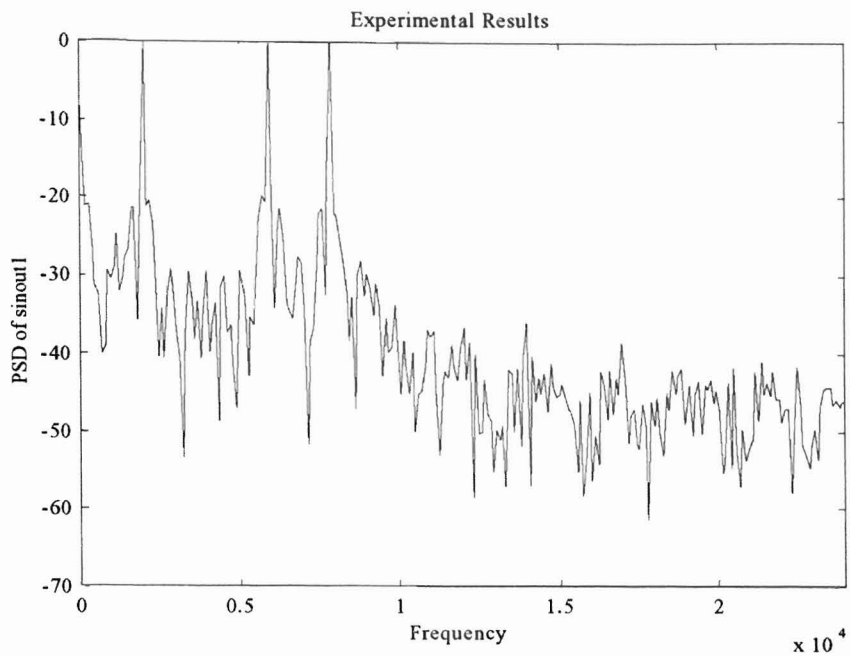
As shown in Figure 9-2, the input signal contains frequency components at 3kHz and 5kHz, represented by the two signal tones. The true average noise power for the signal is clearly around -35dB which is sufficient for an analog signal that has been digitized for use. This means that the Signal-to-Noise (SNR) ratio of the digitized input signal is clearly at an acceptable level.





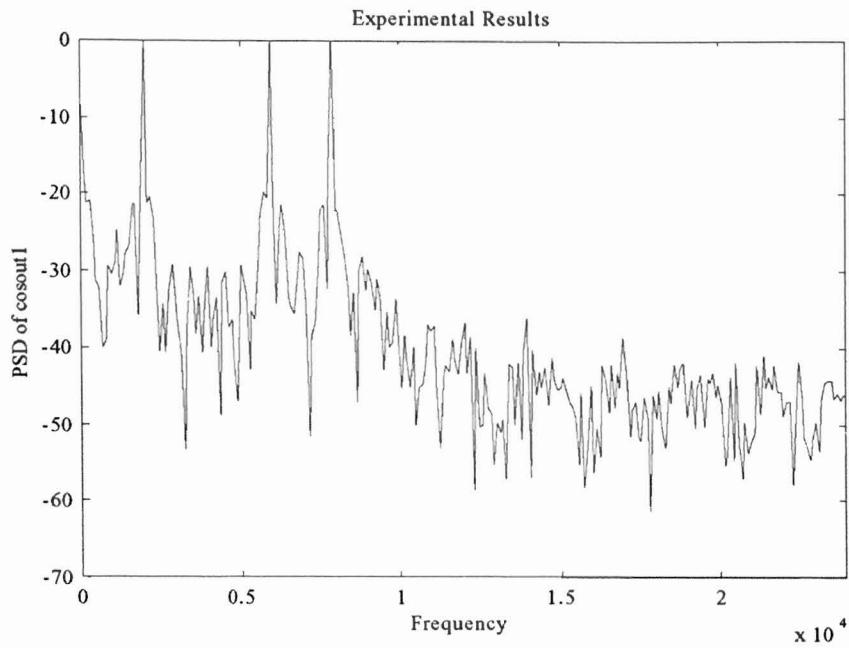
**Figure 9-3.** 512 point, normalized Power spectrum plot of the heterodyning signal.

For the purposes of this thesis, the tunable heterodyning frequency,  $f_h$ , was set at 3kHz. Figure 9-3 shows the 512-point power spectrum plot of the sine/cosine signal tone at the heterodyning frequency. We can see from the diagram that the true average noise power of the signal is well below -45dB. This proves that the sine/cosine generator implementation was highly successful in producing the required signal tone at the heterodyning frequency.



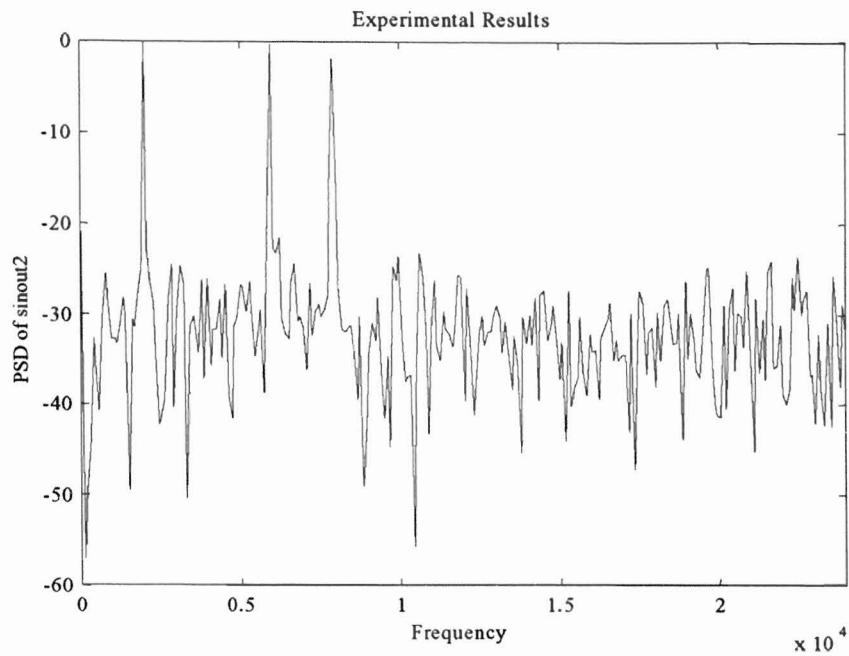
**Figure 9-4.** 512 point, normalized Power spectrum plot of signal *sinout1*.

Figure 9-4 shows the 512-point power spectrum plot of signal *sinout1*. Signal *sinout1* is essentially the spectrum resulting from the product of the input signal and the sine signal at the heterodyning frequency (3kHz). The resulting frequency components are at 2kHz, 6kHz and 8kHz as expected. The true average noise power is about -50dB.



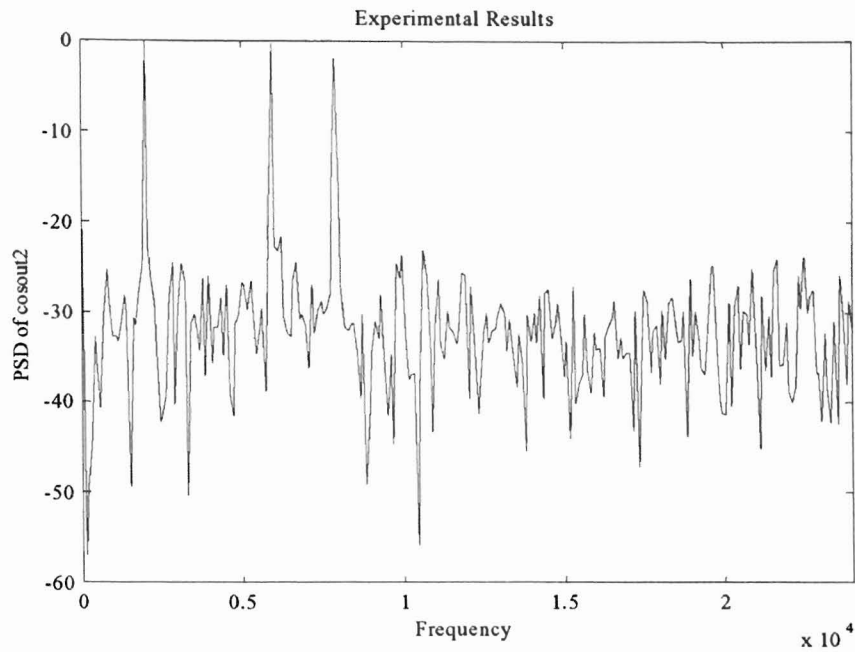
**Figure 9-5.** 512 point, normalized Power spectrum plot of signal *cosout1*.

Figure 9-5 shows the 512-point power spectrum plot of signal *cosout1*. Signal *cosout1* is essentially the spectrum resulting from the product of the input signal and the cosine signal at the heterodyning frequency of 3kHz. The resulting frequency components are also at 2kHz, 6kHz and 8kHz as expected. The true average noise power is about -50dB.



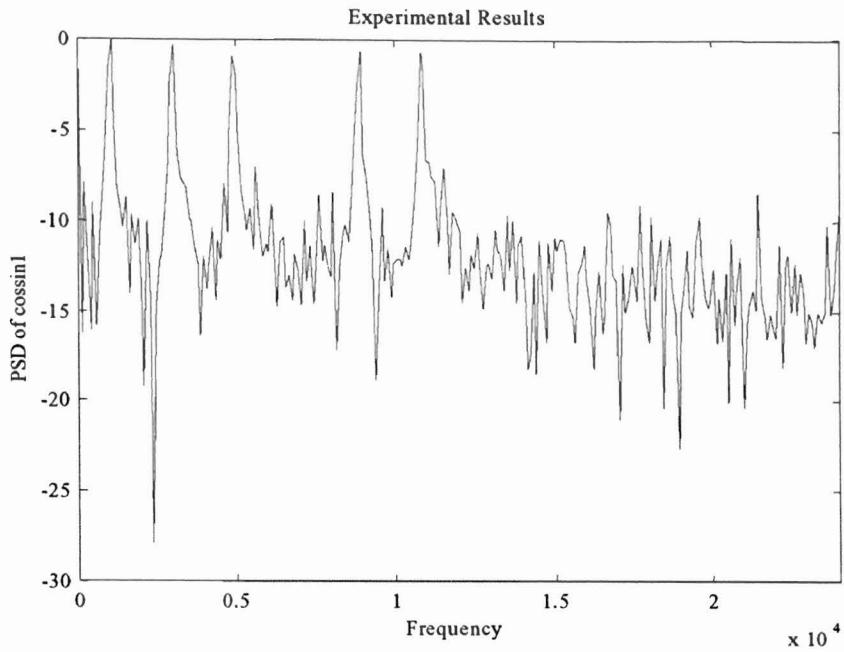
**Figure 9-6.** 512 point, normalized Power spectrum plot of signal *sinout2*.

Figure 9-6 shows the 512-point power spectrum plot resulting from passing signal *sinout1* through the prototype high-pass filter. The frequency components at 2kHz, 6kHz and 8kHz remain due to the characteristics of the high-pass filter. Here, the true average noise power is about -35dB. The increase in the noise power level can be attributed to ?



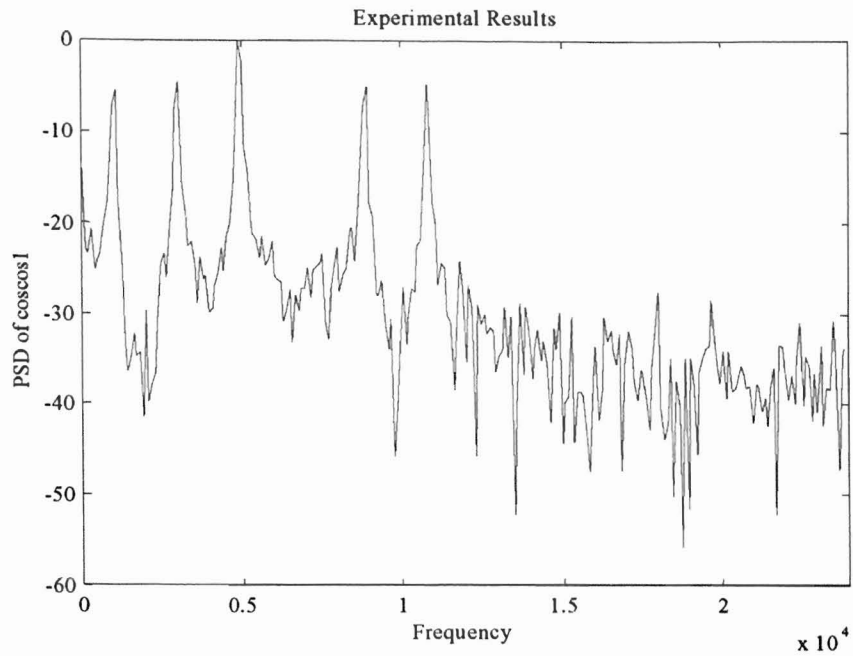
**Figure 9-7.** 512 point, normalized Power spectrum plot of signal *cosout2*.

Figure 9-7 shows the 512-point power spectrum plot resulting from passing signal *cosout1* through the prototype high-pass filter. The frequency components at 2kHz, 6kHz and 8kHz remain due to the characteristics of the high-pass filter. The true average noise power is about -35dB.



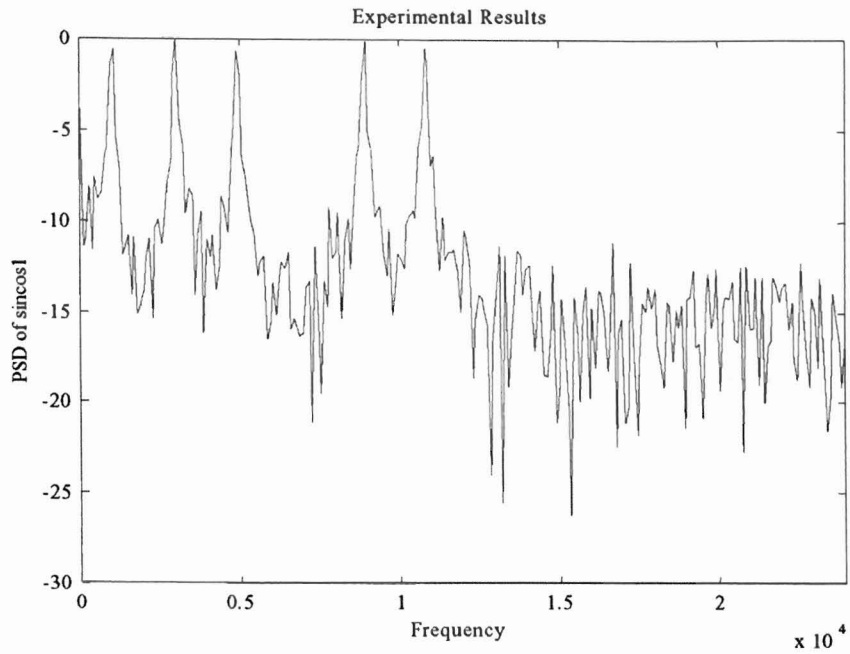
**Figure 9-8.** 512 point, normalized Power spectrum plot of signal *cossin1*.

Figure 9-8 shows the 512-point power spectrum plot of signal *cosout2* after being heterodyned with the sine signal at the heterodyne frequency. Initially, signal *cosout2* contained frequency components at 2kHz, 6kHz and 8kHz. After the heterodyne operation, the resulting signal *cossin1*, shown above, contains frequency components at 1kHz, 3kHz, 5kHz, 9kHz and 11kHz respectively.



**Figure 9-9.** 512 point, normaized Power spectrum plot of signal *coscos1*.

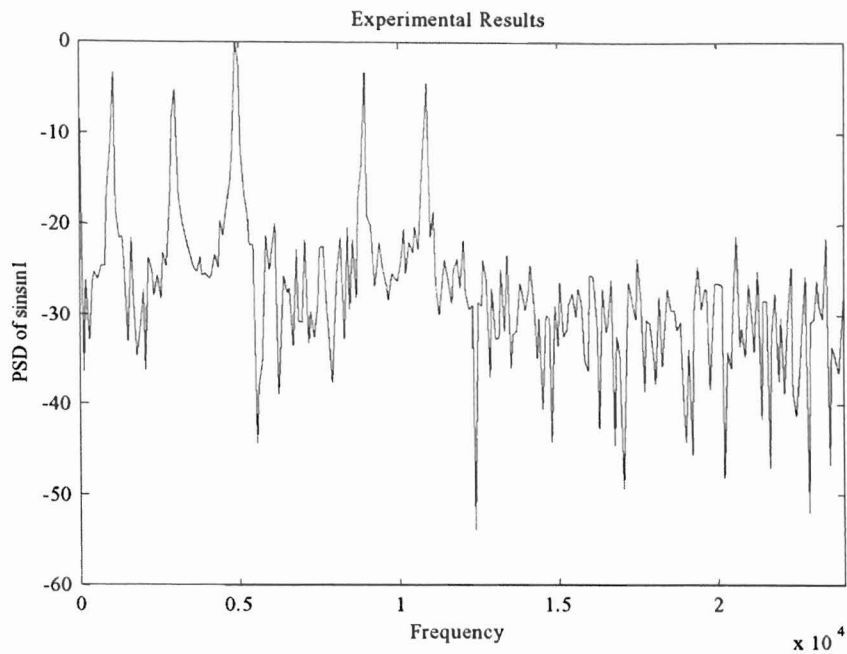
Figure 9-9 shows the 512-point power spectrum plot of signal *cosout2* after being heterodyned with the cosine signal at the heterodyne frequency. Initially, signal *cosout2* contained frequency components at 2kHz, 6kHz and 8kHz. After the heterodyne operation, the resulting signal *coscos1* contains frequency components at 1kHz, 3kHz, 5kHz, 9kHz and 11kHz respectively.



**Figure 9-10.** 512 point, normalized Power spectrum plot of signal *sincos1*.

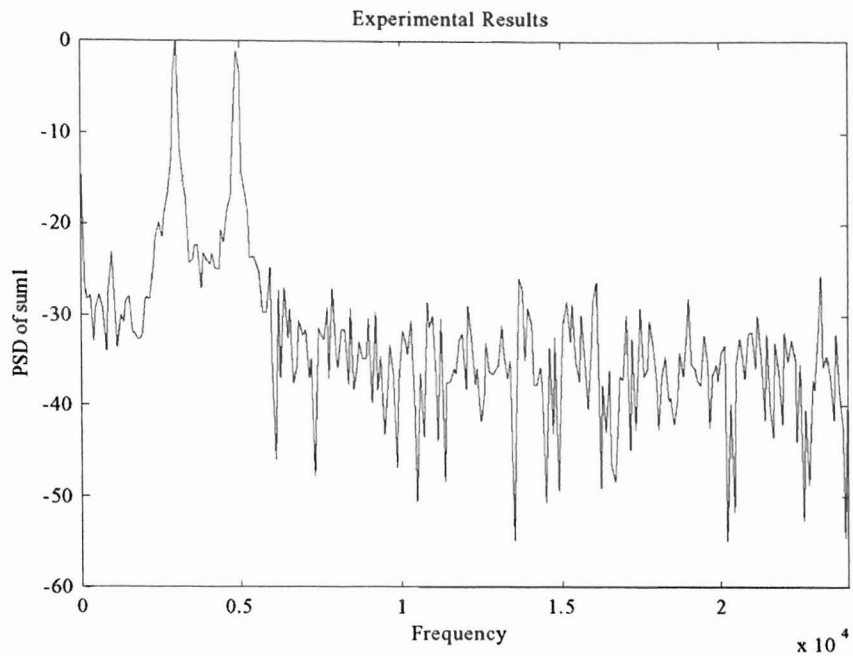
Figure 9-10 shows the 512-point power spectrum plot of signal *sinout2* after being heterodyned with the cosine signal at the heterodyne frequency. Initially, signal *sinout2* contained frequency components at 2kHz, 6kHz and 8kHz. After the heterodyne operation, the resulting signal *sincos1*, shown above, contains frequency components at 1kHz, 3kHz, 5kHz, 9kHz and 11kHz respectively.





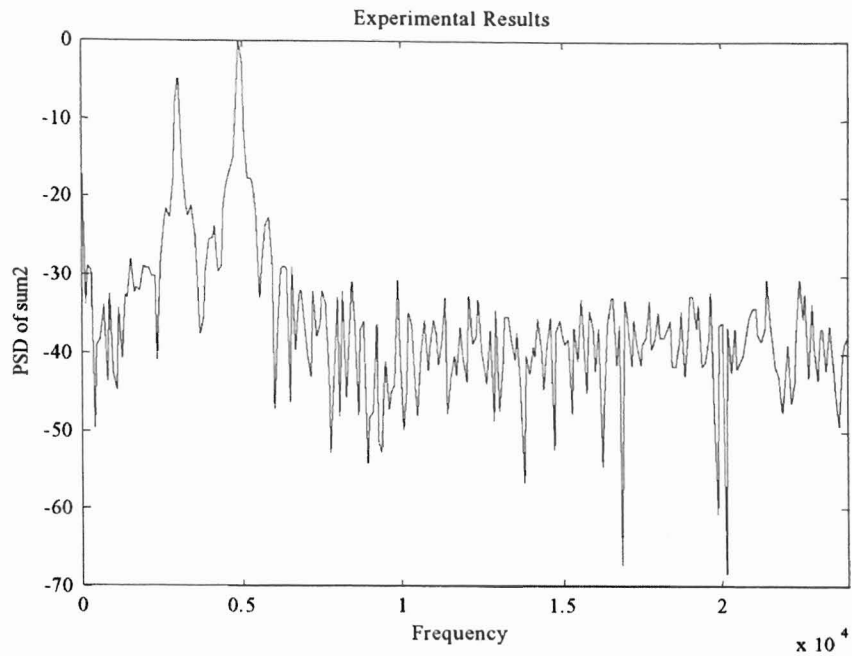
**Figure 9-11.** 512 point, normalized Power spectrum plot of signal *sinsin1*.

Figure 9-11 shows the 512-point power spectrum plot of signal *sinout2* after being heterodyned with the cosine signal at the heterodyne frequency. Initially, signal *sinout2* contained frequency components at 2kHz, 6kHz and 8kHz. After the heterodyne operation, the resulting signal *sinsin1*, shown above, contains frequency components at 1kHz, 3kHz, 5kHz, 9kHz and 11kHz respectively.



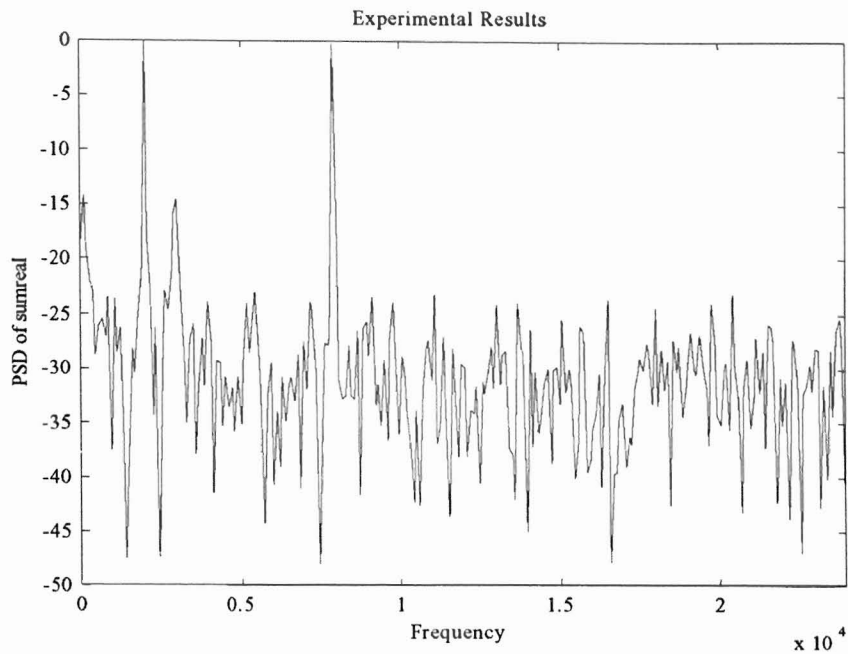
**Figure 9-12.** 512 point, normalized Power spectrum plot of signal *sum1*.

Figure 9-12 above shows the 512-point normalized power spectrum plot of signal *sum1*. Signal *sum1* was obtained from the subtraction of signal *cosin1* from signal *sincos1*. This subtraction process yielded the two original input frequency components of 3kHz and 5kHz respectively. The true average noise power level here is about -35dB.



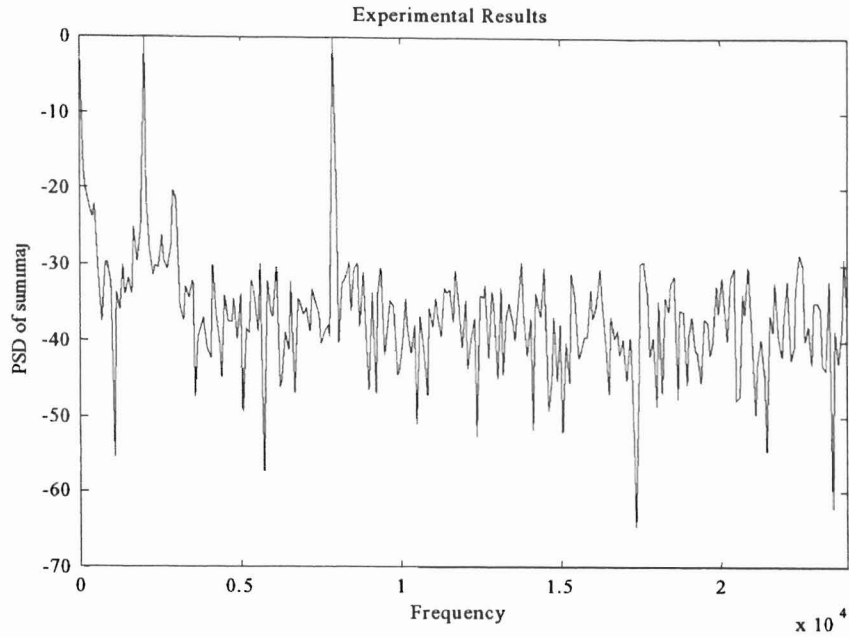
**Figure 9-13.** 512 point, normalized Power spectrum plot of signal *sum2*.

Figure 9-13 above shows the 512-point normalized power spectrum plot of signal *sum2*. Signal *sum2* was obtained from the addition of signal *coscos1* and signal *sinsin1*. This addition process yields the two original input frequency components of 3kHz and 5kHz respectively. The true average noise power level here is about -45dB.



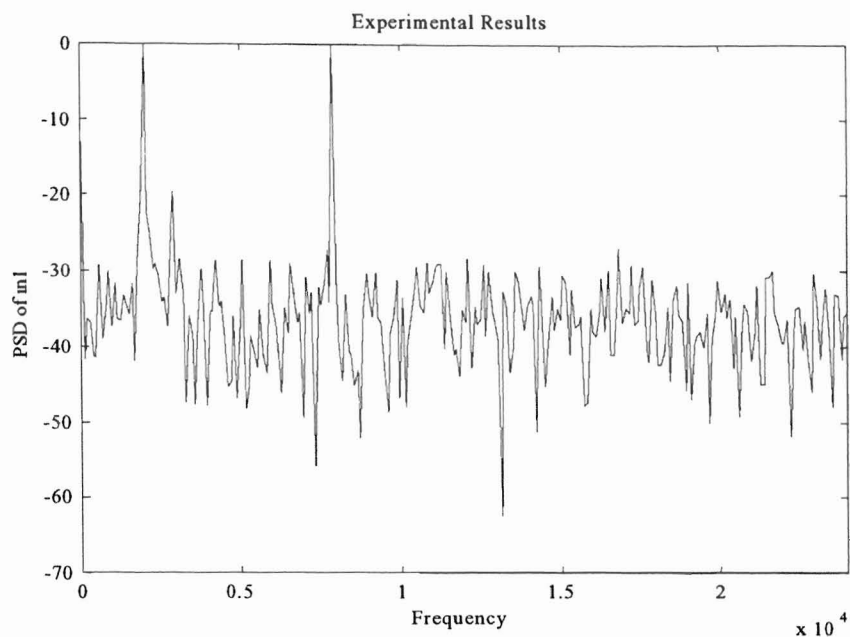
**Figure 9-14.** 512 point, normalized Power spectrum plot of signal *sumreal*.

Figure 9-14 above shows the resulting power spectrum plot of signal *sum1* and *sum2* after being passed through a combiner stage. Essentially, signal *sum1* was heterodyned with a sine at the heterodyning frequency and signal *sum2* was heterodyned with a cosine signal at the heterodyning frequency. The result of these two heterodyning process is then summed to produce signal *sumreal* which is shown above.



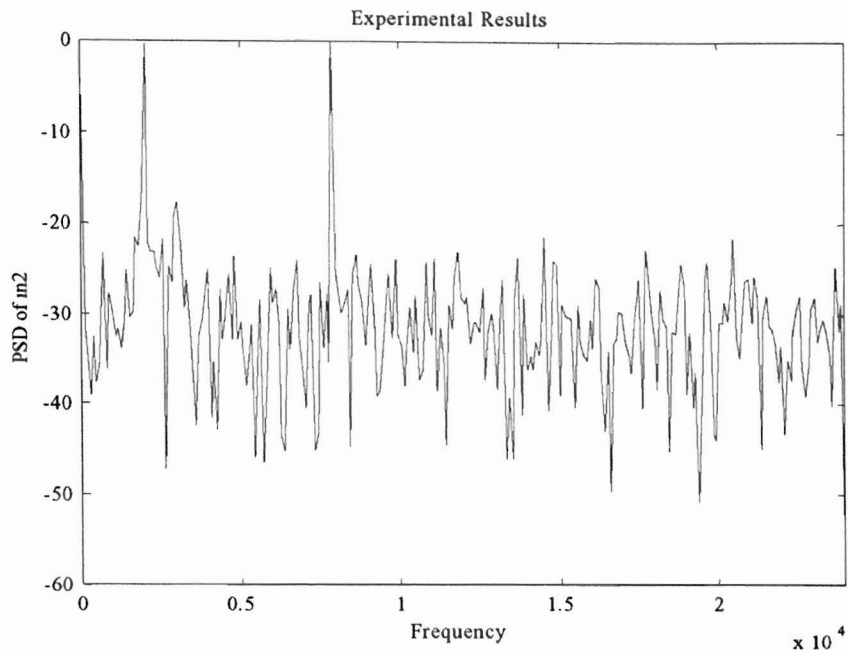
**Figure 9-15.** 512 point, normalized Power spectrum plot of signal *sumimaj*.

Figure 9-14 above shows the resulting power spectrum plot of signal *sum3* and *sum4* after being passed through a combiner stage. Essentially, signal *sum3* was heterodyned with a cosine at the heterodyning frequency and signal *sum4* was heterodyned with a sine at the heterodyning frequency. The result of these two heterodyning process is then subtracted to produce signal *sumimaj* which is shown above.



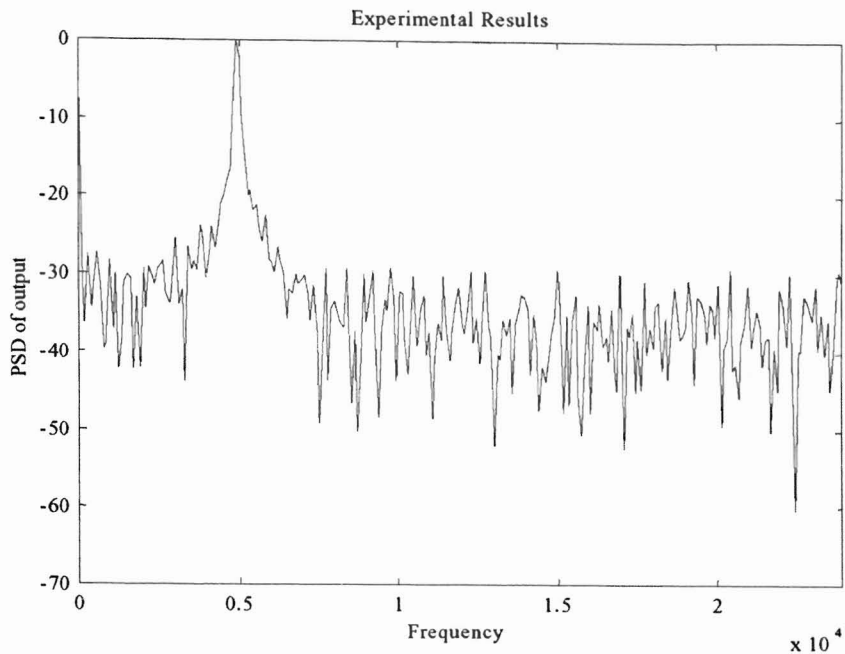
**Figure 9-16.** 512 point, normalized Power spectrum plot of signal *in1*.

Shown in Figure 9-16 above is the power spectrum plot of signal *in1*. Signal *in1* was obtained from passing signal *sumreal* through the second prototype high-pass filter stage.



**Figure 9-17.** 512 point, normalized Power spectrum plot of signal *in2*.

Shown in Figure 9-17 above is the power spectrum plot of signal *in2*. Signal *in2* was obtained from passing signal *sumimaj* through the second prototype high-pass filter stage.



**Figure 9-18.** 512 point, normalized Power spectrum plot of signal output.

Figure 9-18 shows the final power spectrum plot of the output signal obtained after passing two input signals having frequency components 3kHz and 5kHz through the Fully Tunable Heterodyne Notch Filter system. Clearly observable is the fact that the designated interference frequency component of 3kHz has been completely removed from the spectrum. Also noticeable from the PSD plot of Figure 9-18 is that the true average noise power level at the resulting output of the system is about -40dB. This proves that a digital version of a Fully Tunable Heterodyne Notch Filter structure can be successfully and efficiently implemented using FPGAs.



# 10 Future Work

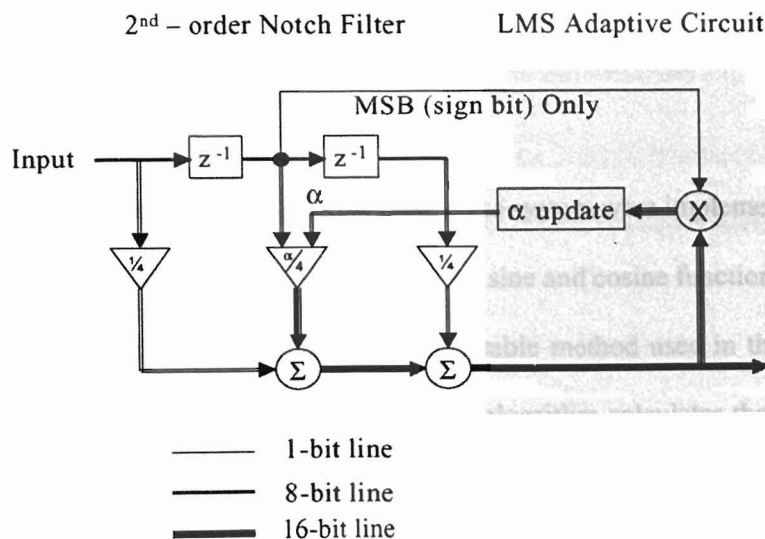
The Fully Tunable Heterodyne Notch (FTHN) filter system discussed in this thesis has been proven to be effective in the complete removal of narrowband interference signals stemming from a single source. Results for the 8-bit implementation discussed in this thesis prove to be sufficient to provide the necessary elimination of the narrowband interference signal in the channel.

## 10.1 Adaptive Heterodyne Notch Filter

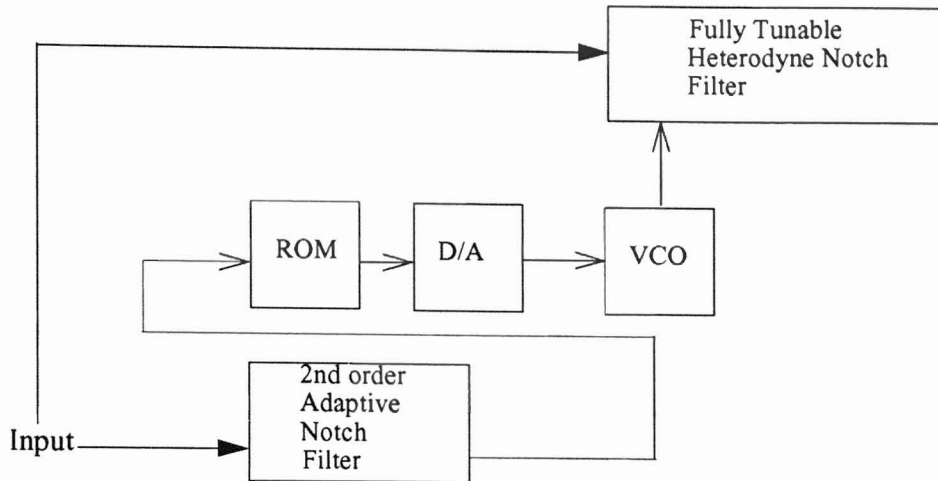
An adaptive version of the filter system discussed in this thesis can be implemented to provide automatic tracking and removal of any narrowband interference signals in the channel. The Least Mean Squared (LMS) algorithm is a fine candidate to be employed in implementation of a completely adaptive version of the proposed filter system. Since the implementation requires the removal of narrowband interference signals from broadband signals, the algorithm can be used to calculate the error between the two signals and hence, the gradient which will then be used to update the tuning parameter ( $\omega_0$ ) of the tunable filter. In short, the LMS algorithm can be used to drive the heterodyne signal frequency needed to translate the interference signal to the stopband of the notch filter, but the filter itself maintains its position in frequency.

The conversion of a tunable filter to an adaptive one can be done by slaving the tunable filter to a simple second-order adaptive notch filter [17][18][19]. Figure 10-1 shows the

basic block diagram of the second-order notch filter used [20]. The incoming signal containing a mixture of the wanted and unwanted signal would be fed into the inputs of the tunable filter and the adaptive notch filter. The purpose of this second-order notch filter is to produce the required parameter ( $\alpha$ ) which moves the notch to the required location of the interfering frequency and at the same time is subsequently mapped to the heterodyning frequency,  $f_h$  needed to tune the tunable filter. Figure 10-2 shows the complete Adaptive Heterodyne Notch (AHN) filter system [17]. Thus, it is evident that we need an adaptive filter to transform our tunable filter into an adaptive one. Perhaps the possibility transforming a tunable filter into an adaptive one without the use of another adaptive filter could be the topic for another thesis.



**Figure 10-1.** Basic block diagram of second-order adaptive notch filter



**Figure 10-2.** Complete Adaptive Heterodyne Notch Filter.

## 10.2 CORDIC algorithm

The much needed sine and cosine functions used in the system were implemented via look-up tables. However, a new method of calculating the sine and cosine functions for a particular frequency can be used to replace the look-up table method used in this thesis. The COordinate Rotation DIgital Computer (CORDIC) algorithm calculates the angle of sine and cosine using phasors to yield  $\sin(\theta)$  and  $\cos(\theta)$  [10][16]. This implementation requires the use of barrel-shifters to calculate the angles required to produce a sine or cosine at the required heterodyning frequency,  $f_h$ . For an FPGA-based implementation however, this implementation may prove to consume considerable hardware but may provide a more efficient implementation for VLSI-based architectures.

## 10.3 Multiple-source Interferences

The filter system discussed in this thesis involves the attenuation of interferences stemming from a single source. It is highly likely that several interference sources could corrupt the transmission channel at any given point in time and will need to be attenuated before the desired signal is to be extracted.

One course of implementation would be to try and cascade the system discussed in this thesis in order to attenuate each interference up to a certain number. However, simple cascading of the structure discussed in this thesis will not work. A suitable method of attenuating more than one NBI signal is by using adaptive filtering techniques attributable to Kwan and Martin [22]. This method can be further refined to reduce hardware complexity by modifying the gradient-search algorithm employed in the structure[23].

With the capability to remove from 3 upto 10 interference sources from a channel, the hardware implementation of this modified Kwan-Martin structure is a suitable thesis subject.

## Bibliography

- [1] Nelson, Karl E. *Development and Analysis of an Adaptive Heterodyne Filter*, PhD Thesis, University of California Davis, California, 2001.
- [2] Azam, Asad. *Hardware Implementation of Tunable Heterodyne Band-Pass Filters*, MS Thesis, Oklahoma State University, Stillwater, 2001.
- [3] Mundra, Pritpal S. et al, *Radio Frequency Interference - An Aspect For Designing A Mobile Radio Communication System*, IEEE transactions, 1992.
- [4] Xilinx. *The Xilinx Data Book 2000*. Xilinx, Inc., San Jose, CA. 2000.
- [5] Pak K. Chan & Mourad, Samiha. *Digital Design using Field Programmable Gate Arrays*, Prentice Hall, NJ 1994, pp 1-13.
- [6] Ziemer, R.E & Tranter, W.H. *Principles of Modern Communications - Systems, Modulation, and Noise*, Third Edition, Houghton Mifflin Company, 1990, pg. 569.
- [7] Jachimczyk, Withold. *Spread Spectrum*, <http://www.ece.wpi.edu/courses/ee535/hwk11cd95/witek/witek.html>
- [8] Virtual Computer Corporation, <http://www.vcc.com/fpga.html>
- [9] Oldfield, John V. and Dorf, Richard C., *Field Programmable Gate Arrays: Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems*, 1994.
- [10] Andraka, Ray, *A Survey of CORDIC algorithms for FPGA based computers*, Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Feb. 22-24, 1998, Monterey, CA. pp191-200
- [11] Xilinx Product Specification, *Sine/Cosine Look Up Table V1.0.2*, LogiCore, San Jose, CA, 95124, October, 1999.

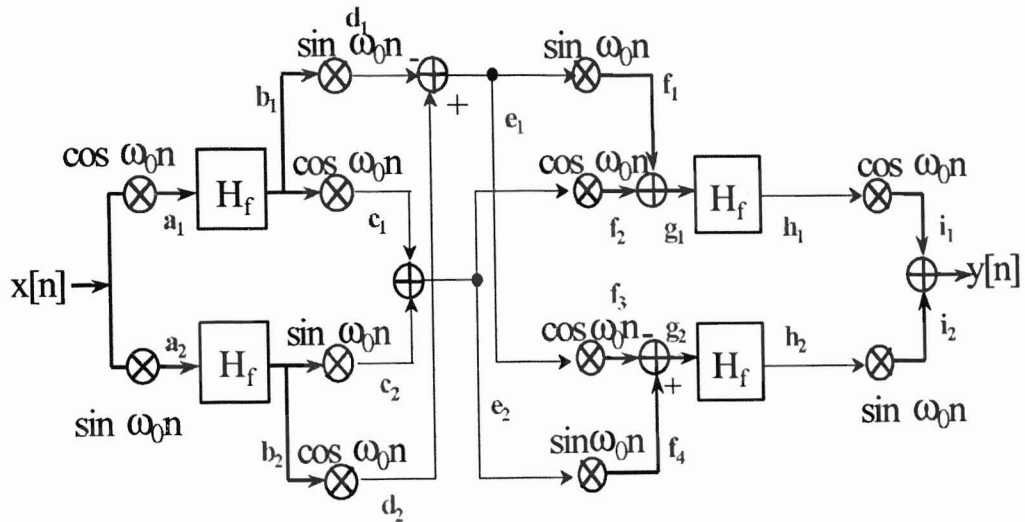
- [12] Matveev, M., *Evaluation of the RTL Synthesis tool for PLD/FPGA Design*, [http://www-collider.physics.ucla.edu/cms/trigger/talks/0108Rice / Matveev\\_SYN2001.pdf](http://www-collider.physics.ucla.edu/cms/trigger/talks/0108Rice/Matveev_SYN2001.pdf), RICE University, August 10, 2001.
- [13] Xess Corporation, *XSV-800 Virtex Prototyping Board with 2.5V, 800,000-gate FPGA*, [http://www.xess.com/prod014\\_4.php3](http://www.xess.com/prod014_4.php3)
- [14] Xess Corporation, *XSV-800 Board V1.1 Manual*, [http://www.xess.com/manuals/xsv-manual-v1\\_1.pdf](http://www.xess.com/manuals/xsv-manual-v1_1.pdf)
- [15] Poor, Vincent H and Rusch, Leslie A., *Narrowband Interference Suppression in Spread Spectrum CDMA.*, IEEE Personal Communications Magazine, Third Quarter 1994., pp.14-27
- [16] Yu Hen Hu, *CORDIC-based VLSI architecture for Digital Signal Processing*, IEEE Signal Processing Magazine, July 1992.
- [17] Soderstrand, et. al, *FPGA Implementation of Adaptive Heterodyne Filters*, Proceedings 34<sup>th</sup> IEEE Asilomar Conference on Signals Systems and Computers, Pacific Grove, California, Volume 1, 2000, pp.375 -378
- [18] K.E. Nelson, P.V.N. dao and M.A.Soderstrand, *A Modified Fixed-Point Computational Gradient Descent Gray-Markel Notch Filter Method for Sinusoidal Detection and Attenuation*, IEEE International Symposium on Circuits and Systems, Hong Kong, China, June 1997.
- [19] K.E. Nelson and M.A. Soderstrand, *Adaptive Filtering Using Heterodyne Frequency Translation*, Proceedings 40th IEEE International Midwest Symposium on Circuits and Systems, Sacramento, CA, August 1997.
- [20] M.A. Soderstrand, *Adaptive Filters*, ECEN5050.377 Lecture Notes, Department of Electrical and Computer Engineering, Oklahoma State University, OK.
- [21] Ziemer, Roger E. and Peterson, Roger L. *Introduction to Digital Communication*, 2nd Edition, Prentice Hall, pp.562-573.
- [22] Kwan, T and Martin, K, *Adaptive detection and enhancement of multiple sinusoids using a cascade IIR filter*, IEEE Transactions. Circuits and Syst., pp. 937-945, July 1989.

- [23] Soderstrand, et. al, *Suppression of multiple narrow-band interference using real-time adaptive notch filters*, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, Volume: 44, Issue: 3 , March 1997, pp.217 -225.

## Appendix A

### Fully Tunable Heterodyne Notch Filter Derivation

Part of the contents of this appendix was obtained from Karl Nelson's Thesis [1] and completed here.



$$a_1(z) = \frac{1}{2} \left[ X \left( z e^{j\omega_0} \right) + X \left( z e^{-j\omega_0} \right) \right]$$

$$a_2(z) = \frac{1}{2j} \left[ X \left( z e^{j\omega_0} \right) - X \left( z e^{-j\omega_0} \right) \right]$$

$$b_1(z) = \frac{1}{2} H(z) \left[ X \left( z e^{j\omega_0} \right) + X \left( z e^{-j\omega_0} \right) \right]$$

$$b_2(z) = \frac{1}{2j} H(z) \left[ X \left( z e^{j\omega_0} \right) - X \left( z e^{-j\omega_0} \right) \right]$$



$$c_1(z) = \frac{1}{4}H\left(ze^{j\omega_0}\right)\left[X\left(ze^{2j\omega_0}\right) + X(z)\right] + \frac{1}{4}H\left(ze^{-j\omega_0}\right)\left[X(z) + X\left(ze^{-2j\omega_0}\right)\right]$$

$$c_2(z) = -\frac{1}{4}H\left(ze^{j\omega_0}\right)\left[X\left(ze^{2j\omega_0}\right) - X(z)\right] + \frac{1}{4}H\left(ze^{-j\omega_0}\right)\left[X(z) - X\left(ze^{-2j\omega_0}\right)\right]$$

$$e_2(z) = c_1(z) + c_2(z)$$

$$\begin{aligned} e_2(z) &= \frac{1}{4}H\left(ze^{j\omega_0}\right)X\left(ze^{2j\omega_0}\right) + \frac{1}{4}H\left(ze^{j\omega_0}\right)X(z) + \frac{1}{4}H\left(ze^{-j\omega_0}\right)X(z) + \\ &\frac{1}{4}H\left(ze^{-j\omega_0}\right)X\left(ze^{-2j\omega_0}\right) - \frac{1}{4}H\left(ze^{j\omega_0}\right)X\left(ze^{2j\omega_0}\right) + \frac{1}{4}H\left(ze^{j\omega_0}\right)X(z) + \\ &\frac{1}{4}H\left(ze^{-j\omega_0}\right)X(z) - \frac{1}{4}H\left(ze^{-j\omega_0}\right)X\left(ze^{-2j\omega_0}\right) \end{aligned}$$

$$e_2(z) = \frac{1}{2}H\left(ze^{j\omega_0}\right)X(z) + \frac{1}{2}H\left(ze^{-j\omega_0}\right)X(z)$$

$$d_1(z) = \frac{1}{4j}H\left(ze^{j\omega_0}\right)\left[X\left(ze^{2j\omega_0}\right) + X(z)\right] - \frac{1}{4j}H\left(ze^{-j\omega_0}\right)\left[X(z) + X\left(ze^{-2j\omega_0}\right)\right]$$

$$d_2(z) = \frac{1}{4j}H\left(ze^{j\omega_0}\right)\left[X\left(ze^{2j\omega_0}\right) - X(z)\right] + \frac{1}{4j}H\left(ze^{-j\omega_0}\right)\left[X(z) - X\left(ze^{-2j\omega_0}\right)\right]$$

$$e_1(z) = d_2(z) - d_1(z)$$

$$e_1(z) = -\frac{1}{2j}H\left(ze^{j\omega_0}\right)X(z) + \frac{1}{2j}H\left(ze^{-j\omega_0}\right)X(z)$$

$$\begin{aligned} f_1(z) &= \frac{1}{4}H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) - \frac{1}{4}H(z)X\left(ze^{j\omega_0}\right) - \frac{1}{4}H(z)X\left(ze^{-j\omega_0}\right) + \\ &\frac{1}{4}H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right) \end{aligned}$$

$$\begin{aligned} f_2(z) &= \frac{1}{4}H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) + \frac{1}{4}H(z)X\left(ze^{j\omega_0}\right) + \frac{1}{4}H(z)X\left(ze^{-j\omega_0}\right) + \\ &\frac{1}{4}H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right) \end{aligned}$$

$$f_3(z) = -\frac{1}{4j}H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) + \frac{1}{4j}H(z)X\left(ze^{j\omega_0}\right) - \frac{1}{4j}H(z)X\left(ze^{-j\omega_0}\right) + \frac{1}{4j}H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right)$$

$$f_4(z) = \frac{1}{4j}H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) + \frac{1}{4j}H(z)X\left(ze^{j\omega_0}\right) - \frac{1}{4j}H(z)X\left(ze^{-j\omega_0}\right) - \frac{1}{4j}H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right)$$

$$g_1(z) = f_1(z) + f_2(z)$$

$$g_1(z) = \frac{1}{2}H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) + \frac{1}{2}H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right)$$

$$g_2(z) = \frac{1}{2j}H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) - \frac{1}{2j}H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right)$$

$$h_1(z) = \frac{1}{2}H(z)\left[H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) + H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right)\right]$$

$$h_2(z) = \frac{1}{2j}H(z)\left[H\left(ze^{2j\omega_0}\right)X\left(ze^{j\omega_0}\right) - H\left(ze^{-2j\omega_0}\right)X\left(ze^{-j\omega_0}\right)\right]$$

$$i_1(z) = \frac{1}{4}H\left(ze^{j\omega_0}\right)\left[H\left(ze^{3j\omega_0}\right)X\left(ze^{2j\omega_0}\right) + H\left(ze^{-j\omega_0}\right)X(z)\right] + \frac{1}{4}H\left(ze^{-j\omega_0}\right)\left[H\left(ze^{j\omega_0}\right)X(z) + H\left(ze^{-3j\omega_0}\right)X\left(ze^{-2j\omega_0}\right)\right]$$

$$i_2(z) = -\frac{1}{4}H\left(ze^{j\omega_0}\right)\left[H\left(ze^{3j\omega_0}\right)X\left(ze^{2j\omega_0}\right) - H\left(ze^{-j\omega_0}\right)X(z)\right] + \frac{1}{4}H\left(ze^{-j\omega_0}\right)\left[H\left(ze^{j\omega_0}\right)X(z) + H\left(ze^{-3j\omega_0}\right)X\left(ze^{-2j\omega_0}\right)\right]$$

$$Y(z) = i_1(z) + i_2(z)$$

$$Y(z) = H\left(ze^{j\omega_0}\right)H\left(ze^{-j\omega_0}\right)X(z)$$

VITA

Dhinesh Sasidaran

2

Candidate for the Degree of  
Master of Science

Thesis: HARDWARE IMPLEMENTATION OF A FULLY TUNABLE HETERODYNE NOTCH FILTER IN FPGA

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Sandakan, Malaysia on January 27<sup>th</sup>, 1975, the son of Stanley Sasidaran and Kanniammal.

Education: Graduated from High school in Malaysia; received an associate degree from an affiliated school in Subang Jaya, Malaysia, as part of Oklahoma State University twinning program; received a Bachelor's of Science degree in Electrical Engineering with Computer option from Oklahoma State University, Stillwater, Oklahoma in May 1999. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in May 2002.

Experience: During the Master degree worked as a Research and Teaching Assistant with the Department of Electrical Engineering at Oklahoma State University, 1999 to present.