A Spatial Diversity Scheme

For Fixed Point Indoor Wireless Communication

A Thesis Submitted to the

College of Graduate Studies and Research

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the Department of Electrical Engineering

University of Saskatchewan

Saskatoon, Saskatchewan

by

Neil Gerein

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Degree of Master of Science from the University of Saskatchewan, the author agrees that the libraries of this University may make it freely available for inspection. The author further agrees that permission for copying of this thesis in any manner, in whole or in part for scholarly purposes may be granted by the professor who supervised this thesis work or, in his absence, by the Head of the Department or the Dean of the College of Graduate Studies and Research at the University of Saskatchewan. Any copying, publication, or use of this thesis, or parts thereof, for financial gain without the author's written permission is strictly prohibited. Proper recognition shall be given to the author and the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

Requests for permission to copy or to make any other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical Engineering,

57 Campus Drive,

University of Saskatchewan,

Saskatoon, Saskatchewan,

Canada S7N 5A9

ABSTRACT

The ease with which indoor wireless systems can be installed has become their main selling feature. A desirable application for wireless systems is the transmission of compressed digital music in an indoor shopping mall environment. The indoor environment, with its many walls and highly reflective surfaces, has a high level of multipath. High levels of slowly changing multipath can cause deep fades, and therefore reduce the reliability of the system.

The proper use of multiple receiving elements is one way to mitigate the deep fades caused by multipath. The main objective of this thesis is to study a simple and cost effective approach to combining the signals from several receiving elements. A novel diversity combining approach using 2 receiving elements is presented. The novel diversity combining approach consists of periodically changing the phase of one of the two received signals.

A set of simulations was developed to study the effectiveness of the novel diversity combining method in mitigating deep multipath fades. The relative performances of two different implementations of the diversity combining were compared to a baseline test case that did not include diversity combining. In both of the simulated implementations, the diversity combining approach proved to be an effective means of mitigating the multipath fading phenomenon.

A proof-of-concept, bench-top hardware prototype was also developed. The transmitter and receiver were implemented in Field Programmable Gate Arrays (FPGAs). The laboratory testing of the hardware successfully illustrated the feasibility of the proof-of-concept system.

ACKNOWLEDGEMENTS

I extend my gratitude to Dr. J. Eric Salt for his guidance throughout my studies. I also wish to thank the Natural Sciences and Engineering Research Council (NSERC) and TRLabs for the financial backing and resources to complete this research. Thank you also to my colleague Jason Dielschneider for the many productive technical conversations.

Thank you to my wife, Janna, for putting up with the long nights of me working on the computer.

Special thanks go to my parents, Marcy and the late Walter Gerein, for providing years of encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A/D	Analog to Digital
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPF	Band Pass Filter
BPSK	Binary Phase Shift Keying
BW	Bandwidth
D/A	Digital to Analog
DPSK	Differential Phase Shift Keying
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
HB	Half Band
Hz	Hertz
IF	Intermediate Frequency
IIR	Infinite Impulse Response
ISI	Intersymbol Interference
ISM	Industrial Science and Medical
kb/s	Kilobits per Second
LAN	Local Area Network
LO	Local Oscillator

LPF	Low Pass Filter
Mchip/s	Megachips per Second
MHz	Megahertz
MPEG	Moving Picture Experts Group
NSERC	Natural Sciences and Engineering Research Council
NZ	Nyquist Zone
PN	Pseudorandom Noise
PSD	Power Spectral Density
RF	Radio Frequency
SAW	Surface Acoustical Wave
SRRC	Square Root Raised Cosine
VCO	Voltage Controlled Oscillator

1 Introduction

1.1 Motivation for Research

There is a demand for fixed-point to fixed-point indoor wireless communications. An indoor wireless system offers a number of advantages over its wired counterpart, including allowing a communications link to be set up quickly without the difficulty and expense of installing data transmission lines. Indoor wireless systems are showing up in an increasing number of applications, including voice and video communications, local area networks (LANs) for computers, and wireless security systems.

The main difficulty with the indoor wireless channel is multipath fading. Signals are reflected off of walls, furniture, and people. Instead of obtaining the original transmitted signal, the receiver detects a summation of the transmitted signal and all reflected signals. It is possible that the signals add destructively causing a signal fade. The problem is compounded further if the channel is slowly changing, i.e. people and objects are moving through the environment. The changing nature of the channel may even be such that the direct line of sight signal is completely obstructed.

For this thesis the intended application is distribution of compressed digital music in a shopping mall. The targeted bit rate is 192 kilobits per second (kb/s), which allows for high quality MPEG encoded music. The Industrial Science and Medical (ISM) frequency band from 902-928 MHz is the intended transmission band. To further conform to Industry Canada rules, direct sequence spread spectrum (DSSS) technology will be employed, the spreading gain will be at least 11, and the transmitted power will be kept under 1 watt. Following the Industry Canada rules for use of the ISM eliminates

the need for the user to obtain a transmitting license to operate the device. This thesis will focus on the special case of having just one transmitter and receiver.

1.2 Overview of Multipath Fading

The multipath phenomenon occurs when radio waves reach the receiving antenna by more than one path. Each path has a unique attenuation factor and time delay [1]. There may also be some phase shift due to reflections.

The effect of multipath can be illustrated with a simple example using a sinusoidal carrier with no data modulation. Consider the static case with a direct path and a slightly attenuated reflected path. Figure 1.1 shows the case where the carrier of the reflected signal has a phase shift of 0 degrees with respect to the carrier of the direct signal. The two signals add constructively to form the composite signal at the receiving antenna. This case is not a concern for the receiver.

Figure 1.2 shows the case where the carrier of the reflected sinusoid is now 180 degrees out of phase with respect to the carrier of the direct path sinusoid. The two signals now add destructively to form the weak composite signal. The amplitude of the weak signal may be low enough to preclude reception by the receiver. If the receiver was tracking previously, the receiver may now lose lock. This case is undesirable and mitigation is warranted.

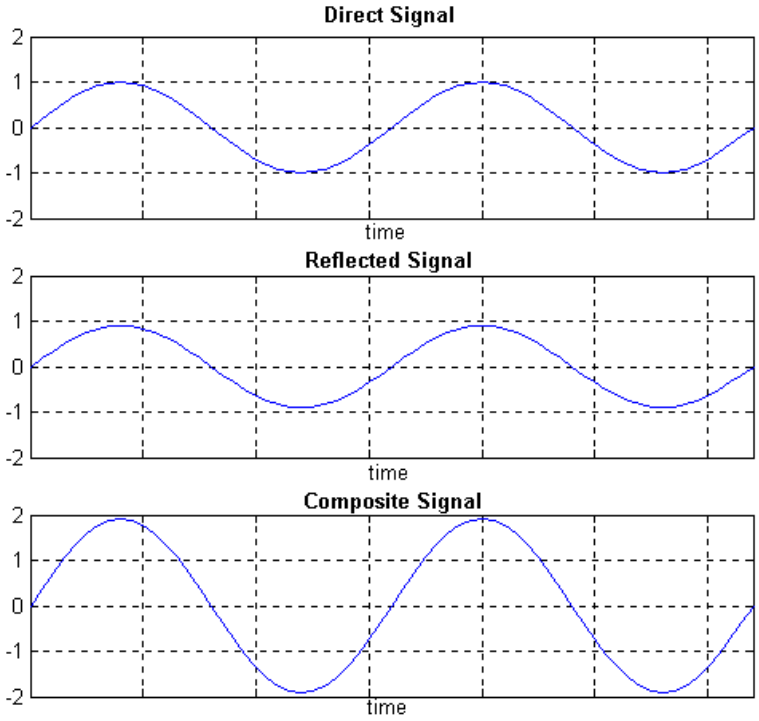

Figure 1.1 Constructive Multipath Interference

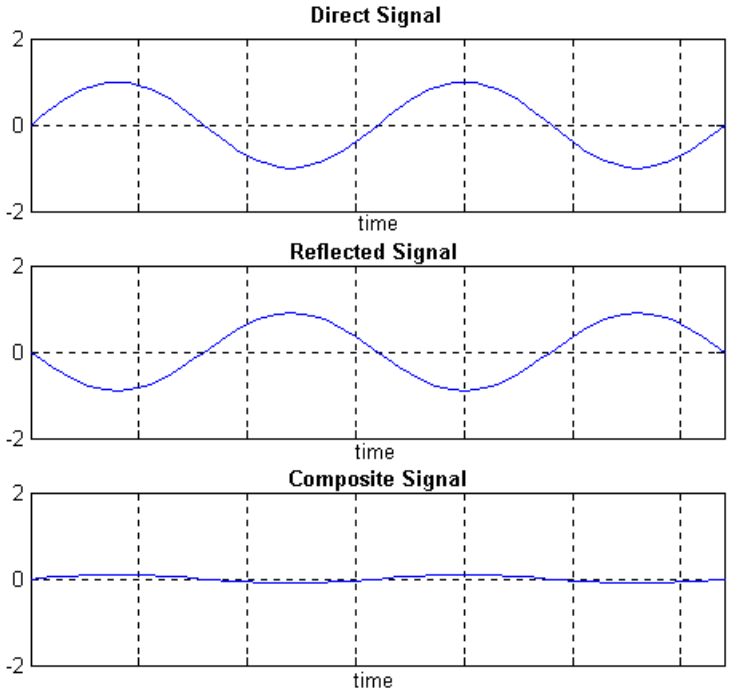

Figure 1.2 Destructive Multipath Interference

In the dynamic case where the communications channel is changing, the path lengths (and hence the time delays) of the reflected signals change. Phasor diagram representation may be used to show how the signals interact to form the composite signal [1][2].

As people and objects move through the shopping mall, the signal path lengths change over time since the scattering bodies are at different locations. The phasors at the top of Figure 1.3, and the corresponding signal envelope in the lower portion of the figure, show how the received signal may exhibit periods of constructive addition as well as periods of almost complete cancellation. This phenomenon is referred to as multipath fading. Multipath fading is a concern for almost all communication systems. The deep fade may cause data loss, and depending on the receiver, loss of carrier and timing synchronization. It is highly desirable to minimize the duration of the fade and the amount of amplitude attenuation (the depth of fade). Spatial diversity is a method of multipath mitigation that uses multiple receiving elements to mitigate multipath fading.

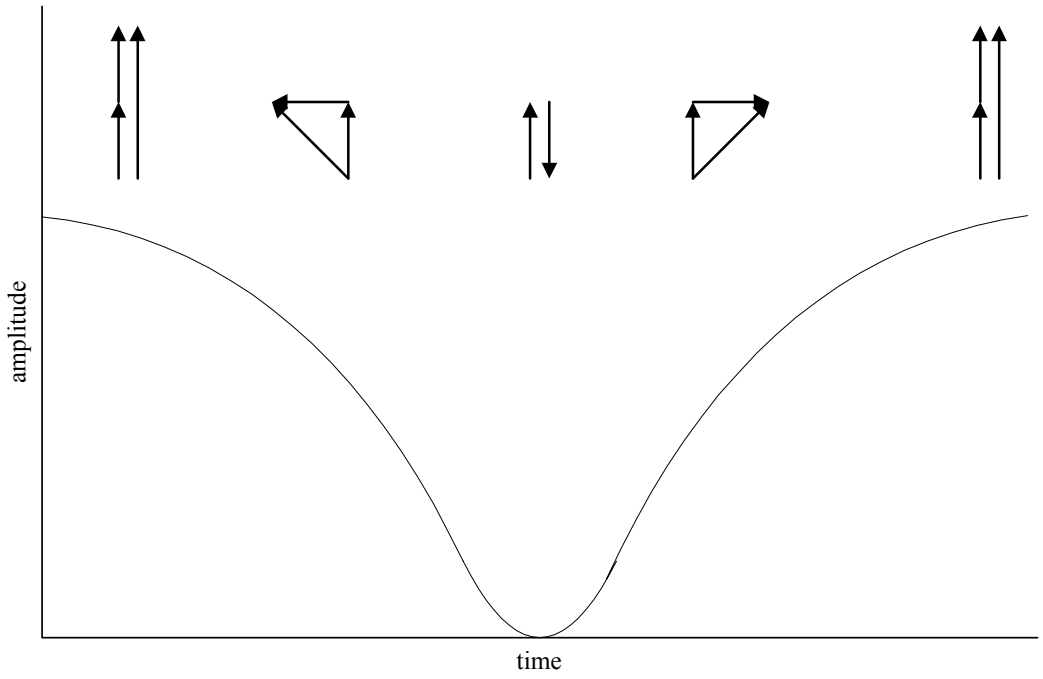

Figure 1.3 Signal Envelope Fading Due to Multipath Interference

1.3 Research Objectives

The objective of this thesis is to simulate and implement a novel spatial diversity scheme for indoor wireless communications. The intended application is low cost distribution of compressed digital music. Therefore, the diversity scheme should be easy and economical to implement, and offer effective mitigation against the indoor multipath channel.

The diversity system will be simulated in software. The Simulink software package from The Mathworks will be used for the simulations. Simulink provides a powerful tool for the simulation of communication systems in the presence of multipath signals. The purpose of the simulations will be to study the relative effectiveness of the diversity scheme in mitigating multipath fades. Two implementations of the diversity scheme

will be compared to a non-diversity receiver. The figure of merit for the simulation will be the bit error rate (BER).

A simple hardware breadboard proof-of-concept version of the spatial diversity scheme will then be implemented in an Altera Field Programmable Gate Array (FPGA) using the Verilog hardware description language. FPGA's allow for rapid prototyping prior to the expense of a full Application Specific Integrated Circuit (ASIC) development. The purpose of the hardware implementation will be to demonstrate that the diversity scheme can indeed be implemented in a practical manner. A simple version of timing recovery will be implemented in the hardware. The laboratory testing of the hardware system will be tested using two signal paths to show proof of concept.

1.4 Thesis Organization

This thesis is organized into eight chapters. Chapter 1 is an introduction to the thesis and describes the motivation for the research.

Chapter 2 discusses necessary background information. Topics in this chapter include sampling theory, digital mixers, spread spectrum techniques and pseudo-noise (PN) sequences, as well as the basic design parameters of the system to be implemented.

Chapter 3 provides an overview of spatial diversity techniques. It describes the existing techniques of equal gain combining, switch combining, and maximal ratio combining. It then outlines the proposed diversity combining technique.

Chapter 4 examines in detail the implementation of the proposed diversity combining technique. The system block diagram is studied in detail and the FPGA implementation

is described. The approximate BER of the receiver without diversity combining is derived.

Chapter 5 provides the details of the implementation of the Simulink simulation.

Chapter 6 covers the testing performed using the Simulink simulations. Using the simulation, the effect of diversity switching on the received signal is examined. Eye diagrams are used to show the effectiveness of the diversity combining technique against deep multipath fades. The relative BER performance of the system is also presented.

Chapter 7 is an overview of the end-to-end testing of the FPGA implementation of the system. The results of the hardware testing are presented.

Chapter 8 will conclude the thesis with a summary of the work undertaken and a proposal for future work.

2 Background Theory

2.1 Phase Shift Keying Modulation

A modulation scheme is needed to transmit information over a communication channel. Among the various modulation methods are; amplitude modulation (data encoded by changing the amplitude of the signal), frequency modulation (data encoded by changing the frequency of the signal), and phase modulation (data encoded by changing the phase of the signal). The modulation method chosen for this thesis is binary phase shift keying (BPSK).

As the word binary implies, BPSK can be used to encode one of two values. BPSK encodes data onto a carrier by changing the phase of the signal by 180 degrees to indicate either a data value of one or a data value of zero. This is illustrated in Figure 2.1 where the data values 1 and 0 were modulated with a 4 Hz sine wave. The first 0.5 seconds of the figure represents the data value 1. At 0.5 seconds the data value changes from 1 to 0 and thus the sine wave changes phase by 180 degrees.

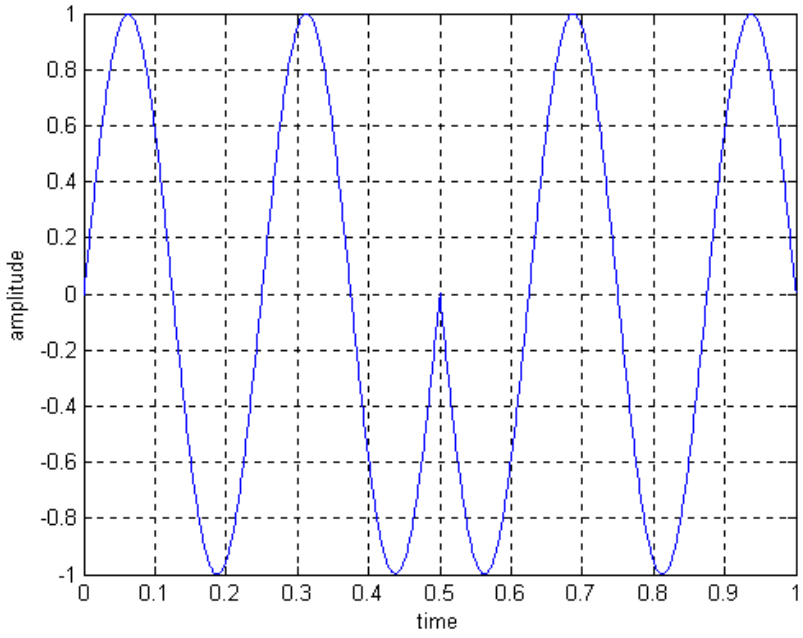

Figure 2.1 Time Domain BPSK Signal

The BPSK signal can be represented mathematically as

$$s(t) = d(t)\cos(2\pi f_c t) \tag{2.1}$$

where *d(t)* is the message signal, and f_c is the carrier frequency (Hz).

The power spectral density (PSD) of the BPSK signal is given as [10]:

$$PSD(f) = \left[\frac{\sin(\pi T_b(f - f_c))}{\pi T_b(f - f_c)}\right]^2 \tag{2.2}$$

where T_b is the bit duration, which is related to the data rate R by $R=1/T_b$. Equation (2.2) gives a null-to-null bandwidth of $2.00R$. The power spectrum for a 181.8 kb/s data rate signal is shown in Figure 2.2. Figure 2.2 was generated using the Simulink simulator described in Chapter 5.

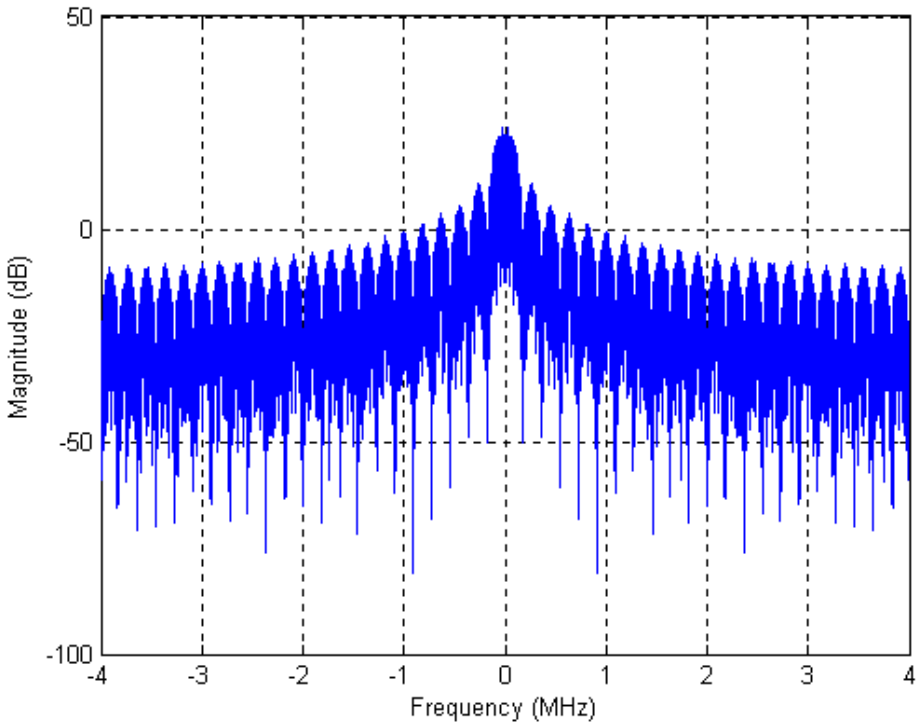

Figure 2.2 BPSK Power Spectrum (random data)

2.2 Sampling Theory

Nyquist's sampling theorem states that to adequately sample with an analog to digital converter (A/D) the sampling rate must be at least twice the bandwidth of the signal. This criteria is represented by:

$$F_s \geq 2BW \tag{2.3}$$

where F_s is the sampling frequency, and BW is the bandwidth of the signal being sampled.

The spectrum of the sampled signal need not be located at baseband, and it may in fact be located at some higher intermediate frequency (IF). This is commonly known as IF sampling and is illustrated in Figure 2.3, where the desired signal is centred about 70

MHz. In Figure 2.3 a number of frequency regions called Nyquist zones (NZ) are defined, each of them occupying a bandwidth of $F_s/2$, where F_s is the sampling frequency of 8 MHz. The Nyquist zones begin with Nyquist zone 1, and increase by single integer numbers. In Figure 2.3 the bandwidth of the signal is 2 MHz.

A second sampling criterion may be defined as:

$$F_s = \frac{4 f_c}{2NZ - 1} \tag{2.4}$$

where f_c is the centre frequency of the desired spectrum.

Assuming that the analog bandwidth of the A/D is greater than the highest frequency in the desired spectrum, and that both equations (2.3) and (2.4) are satisfied, the desired spectrum centred at 70 MHz (Nyquist zone 18) in Figure 2.3 will be converted to Nyquist zone 1 and a centre frequency of 2 MHz.

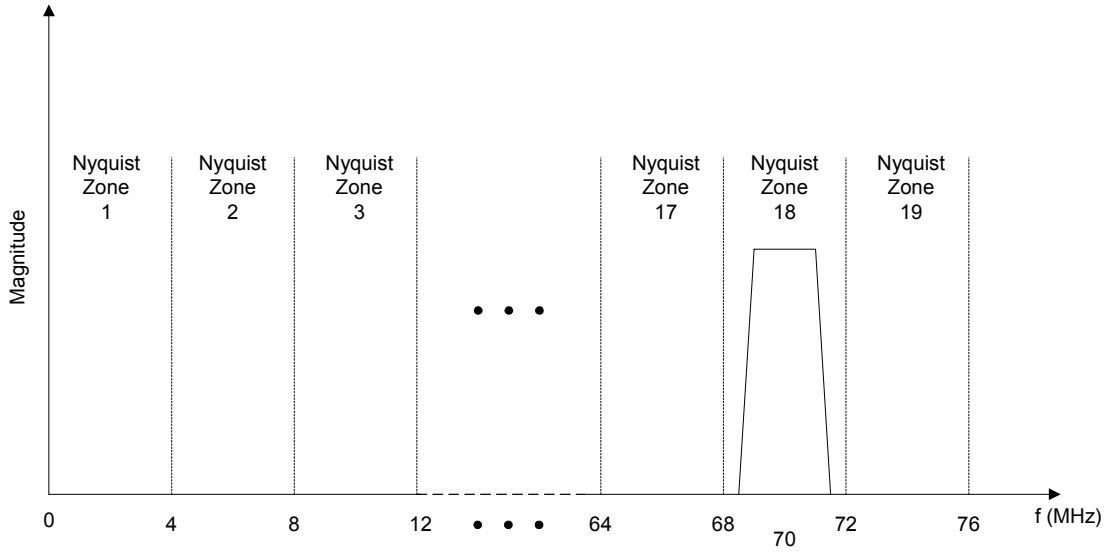

Figure 2.3 Frequency Spectrum Showing IF Sampling

2.3 Digital Mixer

Digital mixers are used in this thesis for frequency translation within the simulation and the FPGAs. Analog mixers are susceptible to local oscillator (LO) feed-through and intermodulation products. Therefore, digital mixers are the preferred solution whenever possible. An ideal digital mixer multiplies the input signal with a perfect sinusoid. Since multiplication is a very resource intensive operation in digital logic, a simple implementation is desirable.

Consider the discrete time sinusoid:

$$\sqrt{2}\cos\left(\frac{2\pi n}{4}+\frac{\pi}{4}\right). \tag{2.5}$$

With an 8 MHz sample rate, the sinusoid in (2.5) may be used to translate the spectrum in Nyquist zone 1 in Figure 2.3 from 2 MHz down to baseband. Table 2.1 shows that the sinusoid in (2.5) takes on the values of +1 and –1, indicating multiplication by the sinusoid can be implemented by a simple negation in the FPGA.

Table 2.1 Evaluation of Digital Mixer Sinusoid

n	**value of** $\sqrt{2}\cos\left(\frac{2\pi n}{4}+\frac{\pi}{4}\right)$
0	1
1	-1
2	-1
3	1

2.4 Spectrum Spreading and De-spreading

The use of direct sequence spread spectrum (DSSS) in communications systems is well documented in the open literature. Use of the ISM band at 900 MHz requires that the

transmitted signal use a spreading sequence with a length of at least 10. The use of DSSS in the ISM band allows for a lower peak power transmitted power within the band. DSSS also allows multiple users to co-exist in the same band.

DSSS is implemented by multiplying each bit of the input data sequence with a pseudorandom noise code. The individual values of the pseudorandom noise code are referred to as chips, rather than bits, to indicate that they carry no information by themselves.

The spreading sequence chosen for this thesis was a length 11 Barker code. The Barker code satisfies the Industry Canada rules for the length of the spreading sequence. The Barker code is [1 1 1 –1 –1 –1 1 –1 –1 1 –1] (using the mapping (+1:0) = (+1:-1) and has the autocorrelation function shown in Figure 2.4.

Consider the following example to illustrate the effect of the spreading operation in the frequency domain. A 181.8 kb/s BPSK (shown in Figure 2.2) is spread using a 2 megachip per second (Mchip/s) Barker spreading code. The resulting time domain signal is still a BPSK signal, but now the changes in phase happen at the 2 Mchip/s rate. The spread BPSK power spectrum, as described previously by equation (2.2) and shown in Figure 2.2, is illustrated in Figure 2.5. Notice that due to the length 11 Barker spreading code, the spectrum is now 11 times wider than the original BPSK spectrum shown in Figure 2.2. Besides the major null at +/- 2 MHz, minor nulls are noticeable in Figure 2.5 at +/- 1 MHz and +/- 3 MHz. These minor nulls are due to the magnitude of the line spectra of the power spectrum of the Barker sequence shown in Figure 2.6.

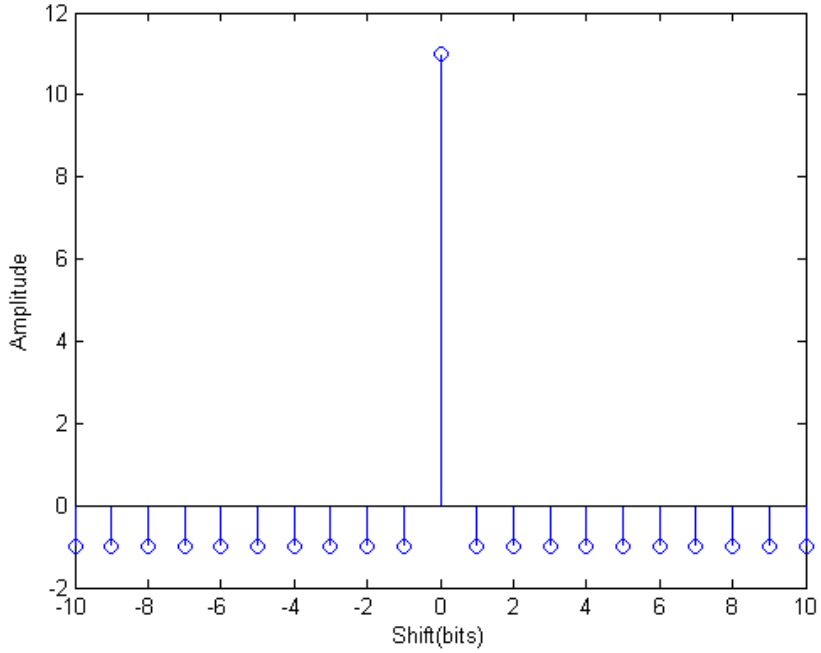

Figure 2.4 Autocorrelation of Barker Sequence

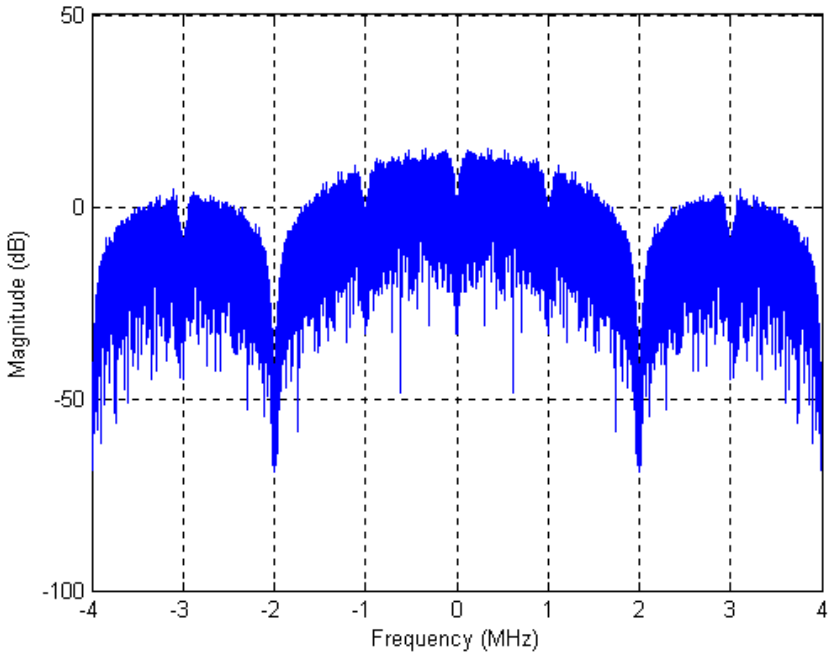

Figure 2.5 Power Spectrum of Random Data Spread With Barker Spreading Code

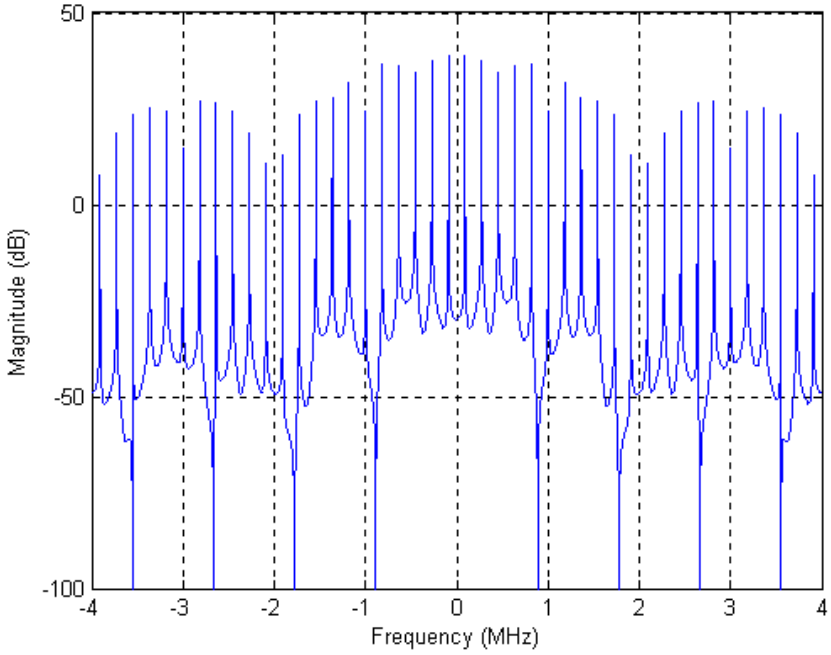

Figure 2.6 Power Spectrum of Barker Code (no random data)

2.5 Differential Encoding and Detection

Differential detection can be called a partially coherent technique, in that the phase reference for the present signaling interval is provided by a delayed version of the signal that occurred during the previous signaling interval [10]. The main reason for using differential detection is that the overall design is simplified.

A differential encoder is used in the transmitter, and is shown in Figure 2.7. On the receiving side, the current bit needs to be compared with the previous bit (a delayed bit) to determine the proper output state. A differential decoder is illustrated in Figure 2.8. This has the advantage that the delayed bit is used as a phase reference for the system, and traditional carrier recovery is not needed. Differential detection works as long as

there is no appreciable phase difference from bit to bit. This is generally the case in the slowly fading indoor environment.

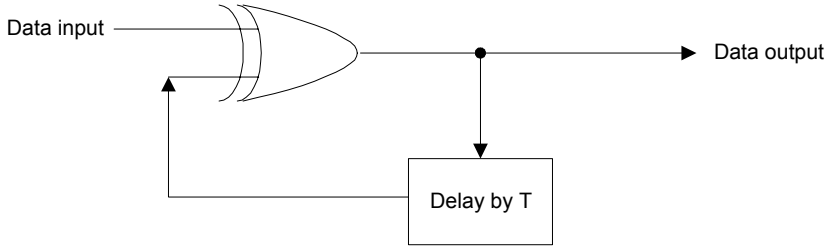

Figure 2.7 Differential Encoder

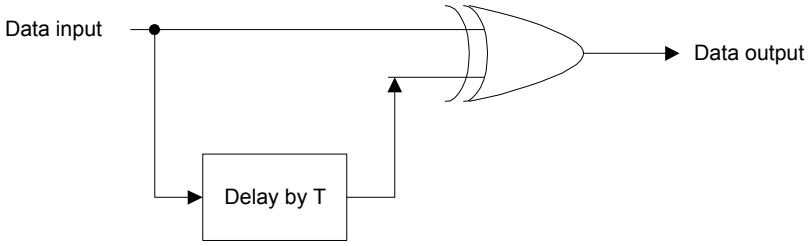

Figure 2.8 Differential Decoder

In this thesis, another benefit of the differential detection scheme is that the previous bit contains the reference PN sequence. This has the advantage that a separate Barker code generator and associated spread spectrum chip acquisition mechanism is not needed in the receiver. An example of this is shown in Figure 2.9. The top plot in the figure shows the received signal as a series of Barker sequences. The middle plot is the received Barker sequence delayed by one bit period (or in other words, delayed by the length of one Barker sequence). The bottom plot, which is the product of the top plot and the middle plot, is the recovered and despread input sequence. For illustrative purposes, the plots in Figure 2.9 are shown with baseband data.

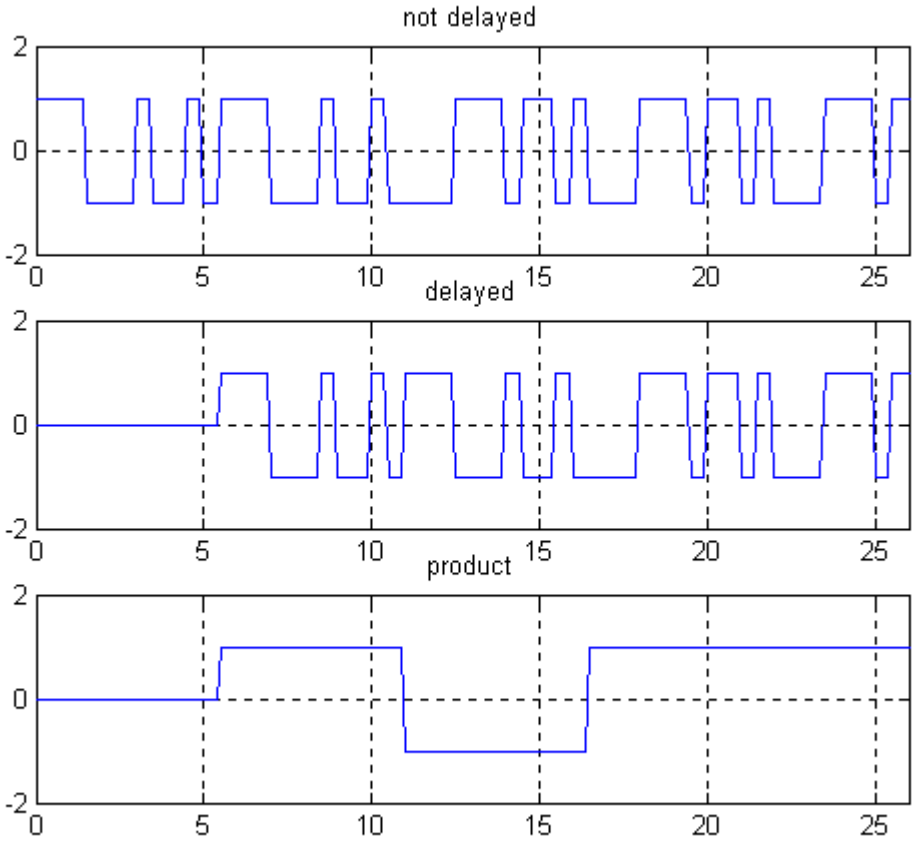

Figure 2.9 Differential Detection of Barker Sequence

2.6 System Design

This subsection describes the selection of the bit rates and sample clocks, selection of the IF, and the frequency plan of the transmitter and receiver.

The first step in the overall system design is the selection of the bit rate. For the intended application of the distribution of high quality digital music, a bit rate of 192 kb/s is desired. After the bit rate is selected, the chipping rate can be calculated using the spreading factor of 11 from the Barker sequence. A sampling frequency to handle the selected chipping rate is selected, and the final IF centre frequency is chosen.

The overall system design was actually done in somewhat of a reverse order. The IF centre frequency, f_c, was chosen to be 70 MHz because of the availability of surface acoustic wave (SAW) filters available at this frequency. Using a value of 70 MHz for f_c in equation (2.4) gives an appropriate choice of sampling frequency, F_s, of 8 MHz. To ease computation burden, the Simulink simulation assumed the data was in Nyquist zone 1, with an IF centre frequency of 2 MHz.

For ease of digital implementation, and to allow for the use of digital mixers, the maximum chipping rate of the signal used is set to be $F_s/4$. This gives a chipping rate of 2 MHz.

Since a length 11 Barker sequence is used to spread the data bit, the bit rate is:

$$R_{bit} = (2 \cdot 10^6 \text{ chips/s}) / (11 \text{ chips/bit}) = 181.8 \cdot 10^3 \text{ bits/s}. \qquad (2.6)$$

The final bit rate of 181.8 kb/s is less than the desired bit rate of 192 kb/s, however, it is considered sufficient for the proof-of-concept system.

Since this thesis is intended to be a bench-top proof-of-concept system, the up-conversion for the FPGA implementation is to the 70 MHz IF only. Figure 2.10 shows the frequency plan of the FPGA based transmitter and receiver. The 2 Mchip/s baseband data is translated to a centre frequency 2 MHz by using the digital mixer with an LO frequency of 2 MHz. The digital mixer creates an image frequency at 6 MHz. The 6

MHz image frequency is removed by the low pass filter (LPF) after the digital to analog (D/A) converter. The remaining 2 MHz spectrum is up converted to the 70 MHz IF using an analog mixer and a local oscillator (LO) of 68 MHz. The analog mixer causes an image frequency at 66 MHz that is removed using the SAW band pass filter (BPF). After the SAW filter, the desired spectrum at 70 MHz remains. After the diversity combining, another SAW filter is used in the receiver section to bandlimit the noise. Using IF sampling as described in Section 2.1, the spectrum of the signal after the A/D is centred at 2 MHz. Digital mixing translates the spectrum to baseband and creates an image frequency at 4 MHz. The image at 4 MHz is removed using a half band (HB) filter, leaving the desired signal spectrum centred at baseband.

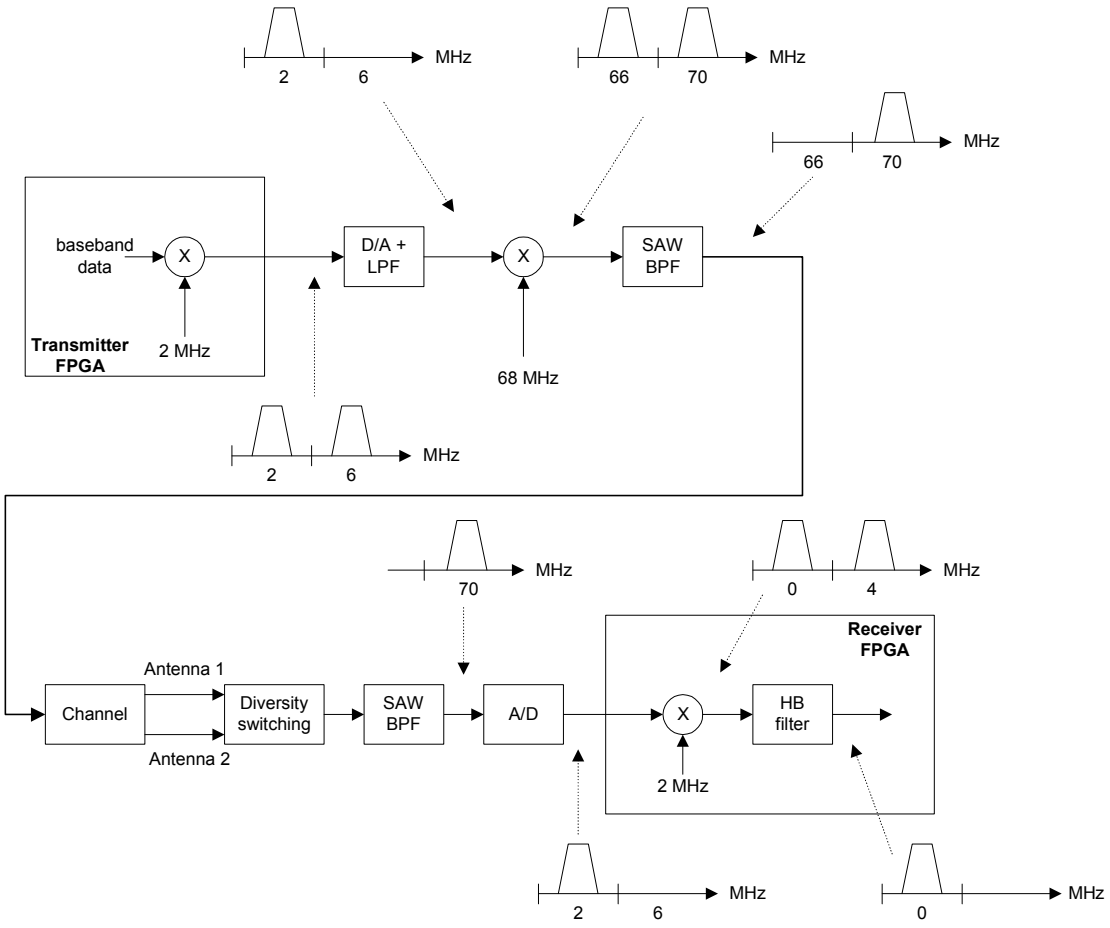

Figure 2.10 Frequency Plan of FPGA Implementation

3 Spatial Diversity Overview

Due to a large number of scatterers and reflectors, the indoor wireless channel is particularly susceptible to multipath fades. Diversity techniques are used to mitigate the effects of the multipath phenomenon. The first part of this chapter will describe the fading characteristics of the indoor channel. The concept of diversity reception will be introduced and a number of spatial diversity combining techniques will then be discussed. The chapter concludes with the introduction of the proposed radio frequency (RF) combining technique.

3.1 The Indoor Channel

For a communication system to exhibit acceptable performance, a certain minimum signal level is needed at the receiver. When the signal level is less than this minimum threshold, the system is in a fade and the BER performance of the system is close to ½. To make the system more reliable, the length of time that these fades occur must be decreased.

Due to multipath reflections, the channel impulse response looks like a series of impulses with different amplitudes. As shown by the vertical arrows in Figure 3.1, a typical indoor delay profile has early reflections arriving with almost identical power [3]. Since these reflections are relatively large the resulting fades are quite deep and almost complete destructive cancellation can occur at the receiver. Later reflections tend to have lower relative power and the expected power per unit time decays as shown by the solid line in Figure 3.1.

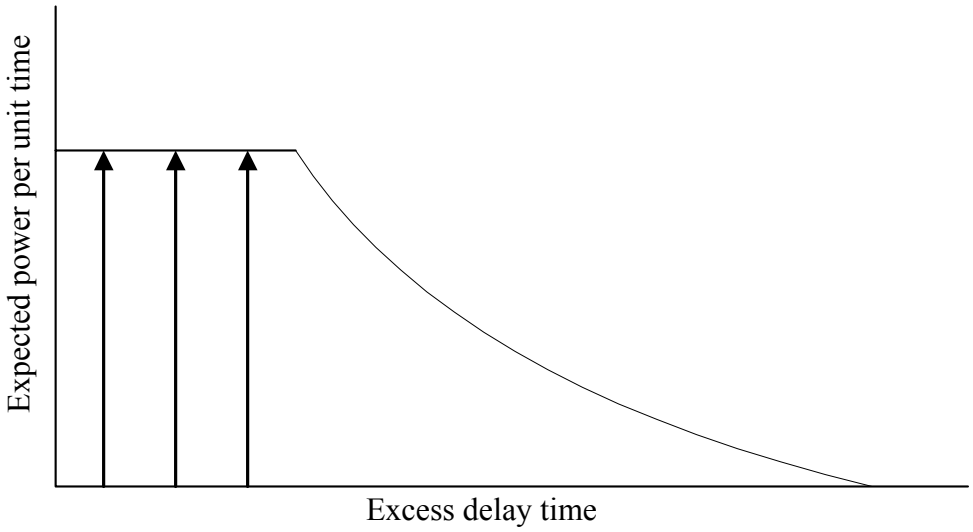

Figure 3.1 Typical Indoor Delay Profile

When the transmitter and receiver are in a fixed position, the multipath environment may be relatively static and due largely to walls and furniture. The fading may also be slowly changing due to the movement of people. Due to this static and/or slow movement, when a fade occurs it may last for several thousand bits. This makes error correction coding impractical for indoor applications, and other methods to reduce the fades must be explored.

3.2 Diversity Techniques

It is desirable to minimize the amount of time the received signal is in a fade. If the receiver is supplied with L independent copies of the transmitted signal, then the probability that all L paths fade is p^L, where p is the probability that any one signal is in a fade. Diversity is the method by which these L independent versions are supplied.

Frequency diversity transmits the message on multiple carrier frequencies spaced sufficiently far apart so as to provide independent fading versions of the channel. Time

diversity transmits the message in different time slots, providing signal repetition [4]. Both of these methods are wasteful in that they use up excessive channel bandwidth.

Another method to mitigate the effects of multipath is spatial diversity. With spatial diversity, multiple receiving antennas are spaced at least half a wavelength of the carrier frequency apart to ensure that the signals reaching them are statistically independent [5]. The advantage of spatial diversity over frequency and time diversity is that the message carrying signal does not have to be rebroadcast. Spatial diversity techniques can use post-detection combining (expensive since more than one receiver is required), or the more economical method of pre-detection combining. The next few subsections describe various pre-detection combining techniques where signals from the multiple antennas are combined to form a composite signal.

3.2.1 Passive Combining

In its most simple form, diversity combining consists of placing an additional antenna one half wavelength of the carrier frequency, or more, apart from the first antenna, and summing the signals together as shown in Figure 3.2. This implementation increases the amount of RF signal that is received, and the probability of both antennas experiencing a fade simultaneously is reduced. The problem with this combining method is that the signal at both antennas may be strong, but out of phase with each other. In this case, destructive interference still occurs.

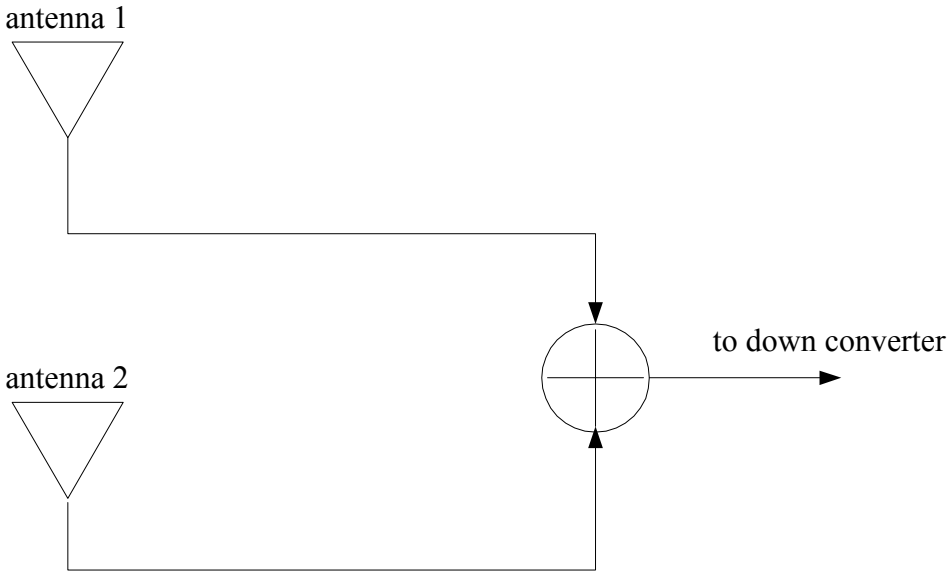

Figure 3.2 Passive Combining

3.2.2 Selection Combining

Another popular diversity combining technique, known as selection combining, is shown in Figure 3.3. In this method, the receiver monitors the level of the incoming signal using switch logic. When the signal level drops below a predefined threshold a switch changes the path to the other antenna. This method performs better than passive combining in that the two signal paths cannot add destructively. The problem with selection combining is that the switching does not take place until a fade has already occurred. There is also the possibility that the signal at the other antenna will be at an even lower signal level.

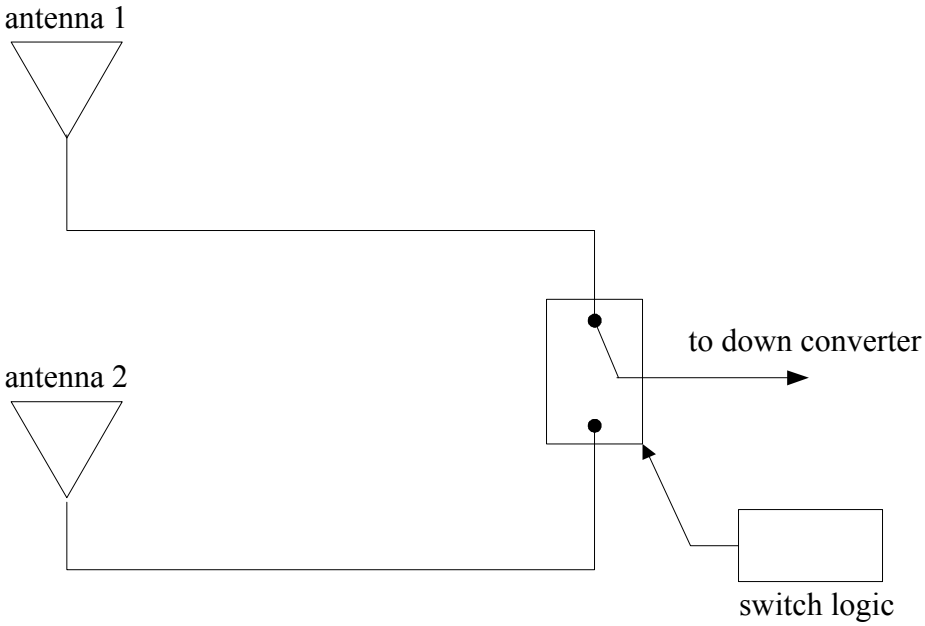

Figure 3.3 Selection Combining

3.2.3 Equal Gain Combining

Another commonly used combining technique is that of equal gain combining, shown in Figure 3.4. In this method each of the L branch signals have a corresponding receiver. The receivers in Figure 3.4 introduce the phase shifts ϕ_1 and ϕ_2 so that the signals are co-phased and add constructively. The designation of equal gain comes from the fact that amplitudes of all branches are equally weighted.

Equal gain combining performs better than selection combining because all available branches are used. Unlike selection combining, it has the undesirable feature that the overall receiver complexity is dependent on the number of resolvable paths L [6].

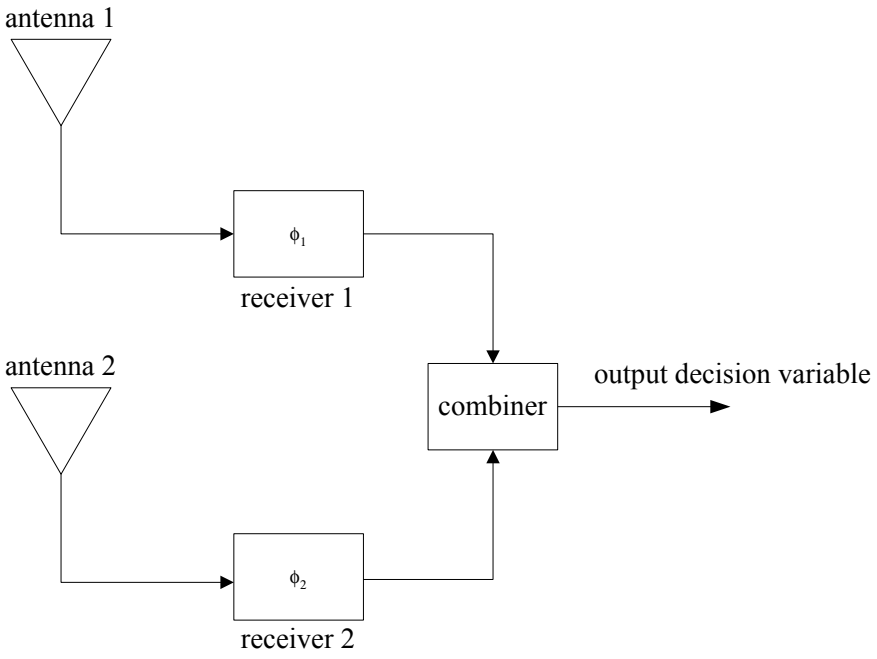

Figure 3.4 Equal Gain Combining

3.2.4 Maximal Ratio Combining

Shown in Figure 3.5, maximal ratio combining is similar to equal gain combining in that the signals are co-phased in the branch receivers by applying phase shifts of ϕ_1 and ϕ_2. Maximal ratio combining differs from equal gain combining in that each branch is weighted by a factor, indicated by α_1 and α_2 in Figure 3.5. The effect of these scaling factors is that a strong signal carries a larger weight than a weak signal [4].

The difficulty in implementing maximal ratio combining is that it is assumed that the weighting factors and phase shifts are known exactly. This is a non-trivial task to implement in a digital receiver, considering that the weighting factors and phase shifts change in real-time. Receiver complexity is once again dependent on the number of resolvable paths.

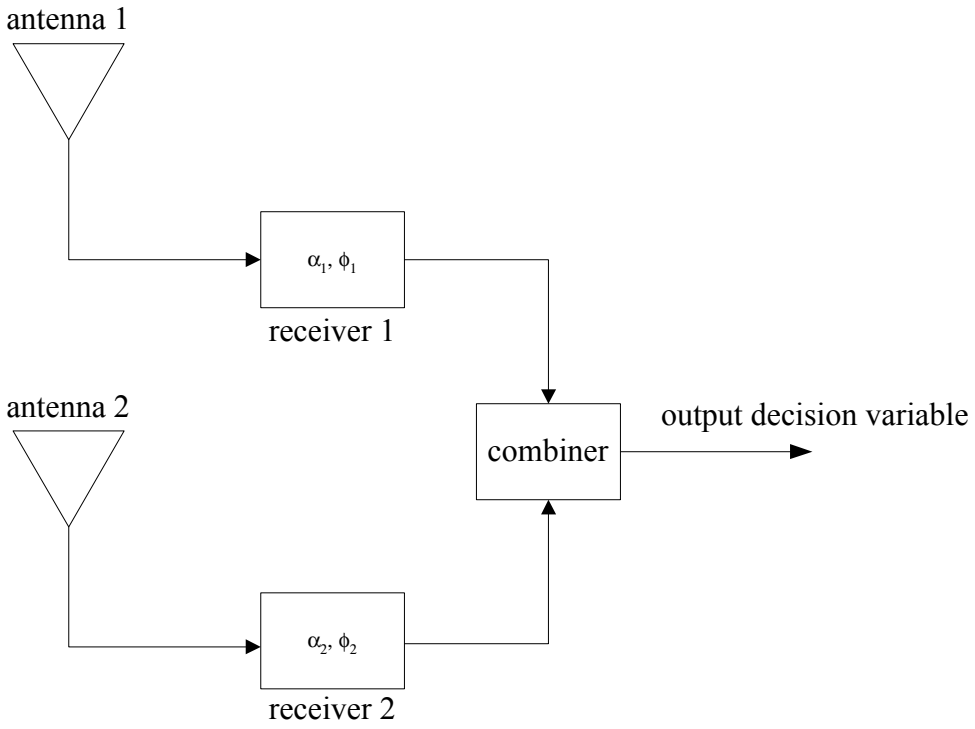

Figure 3.5 Maximal Ratio Combining

3.3 Proposed RF Combining

The new spatial diversity combining technique [7] studied in this thesis combines the signals at RF, and will be referred to as RF diversity combining. The block diagram for RF diversity combining is shown in Figure 3.6. Both branch signals are used with the branch from antenna 2 taking one of two paths; one direct, and the other through a 180 degree phase shift. A switch operating at the bit clock rate ensures that each path is active for one half of a bit period. This is the baseline architecture studied in this thesis. Instead of switching between the paths using a hard 180 degree phase shift, a 'softer' phase shift may be implemented by mixing the signal with a sinusoid. This is the second implementation of the diversity combining that will be studied in this thesis.

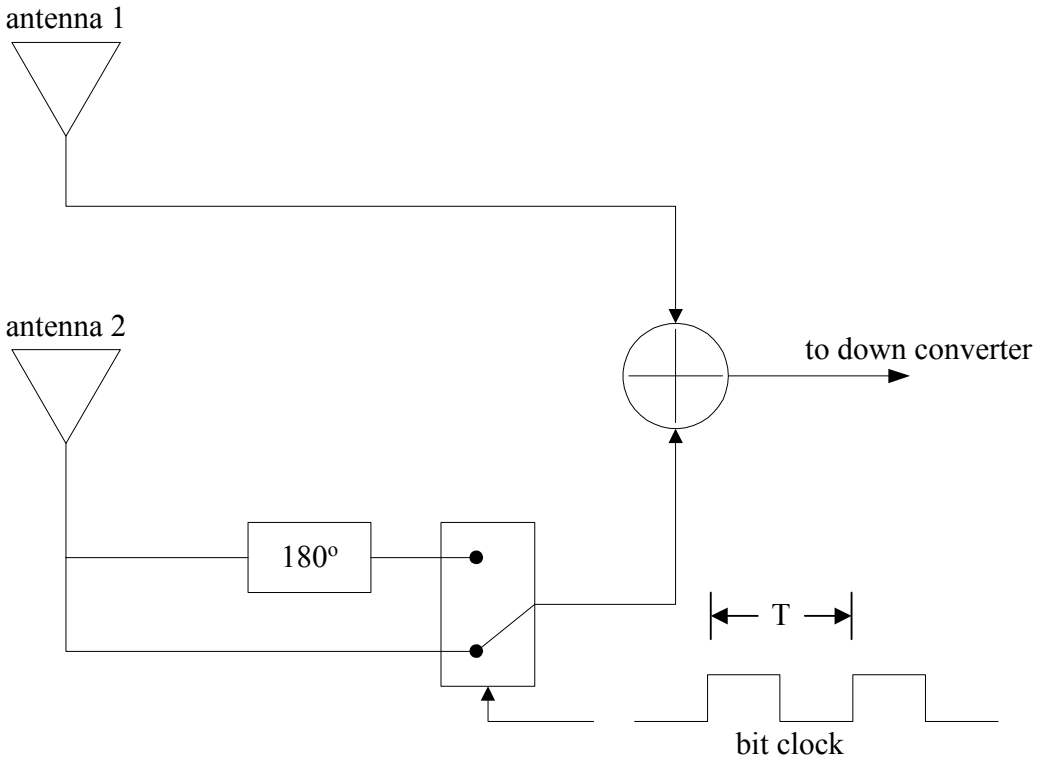

Figure 3.6 RF Diversity Combining

The general theory of operation of the switch is illustrated in Figure 3.7. The top graph shows the signal from antenna 1 entering the summer. The middle graph shows the signal from antenna 2 entering the summer after the 180 degree phase switching operation. The bottom graph shows the output of the summation block. Even though there is a fade for the first half of the bit period, there is enough signal strength for the second half of the bit period to detect the signal.

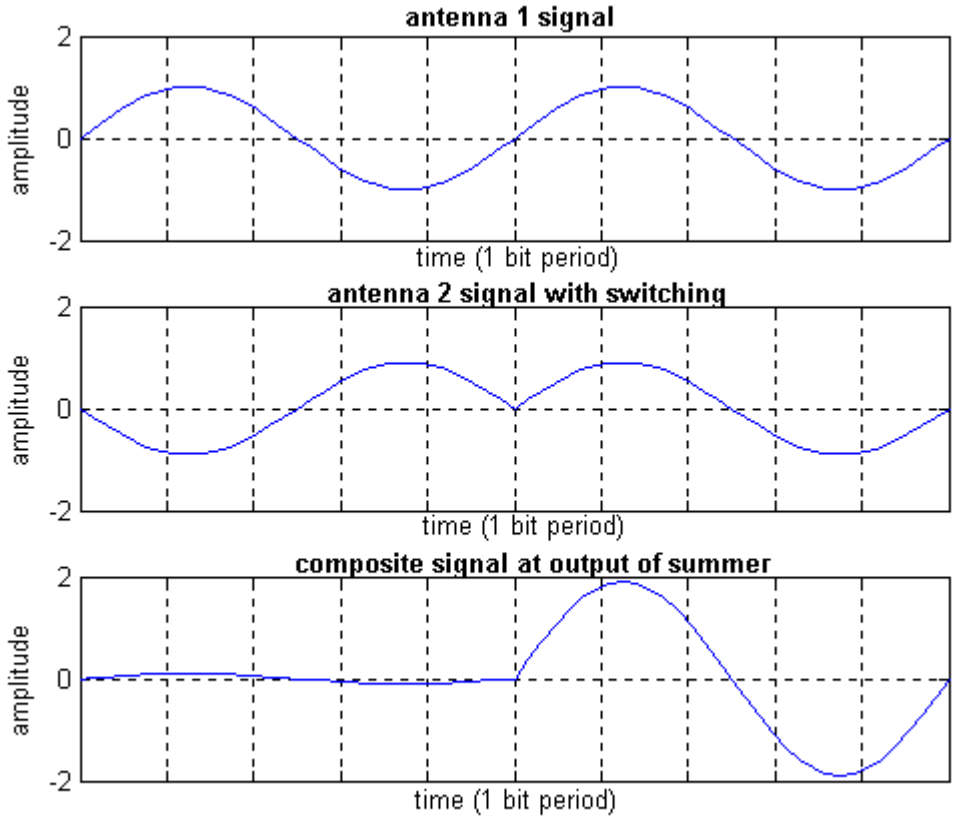

Figure 3.7 Signals for RF Diversity Combining

With RF diversity combining, weighting factors and phase shifts do not have to be calculated, resulting in less digital signal processor (DSP) hardware. In addition, the signal quality estimation used in other methods is not necessary. The switching at higher frequencies results in only one down converter and analog to digital converter (A/D) being needed, reducing the overall cost of the system. RF combining can also be extended to more than two antennas.

In the fixed-point indoor wireless application, the carrier phase does not change appreciably over two consecutive bit periods. This, combined with differential detection, makes traditional carrier recovery unnecessary.

The remainder of this section will examine the special case of two receiving antennas. The general form for the received signal in a communications system is [7]:

$$r_i(t) = A_i \cos(\omega_0 t + \phi_i) \tag{3.1}$$

where:

A_i is the amplitude of the carrier received at receiving element *i*

ω_0 is the angular frequency of the carrier

ϕ_i is the phase in radians of the carrier received at receiving element *i.*

When *N* spatial diversity receiving elements are used, the total summed composite signal may be described as:

$$s(t) = \sum_{i=1}^{N} A_i \cos(\omega_0 t + \phi_i)\,. \tag{3.2}$$

The energy of the combined bit over the bit period *T* is then given by:

$$E_b = \int_0^T s^2(t)dt\,. \tag{3.3}$$

Substituting equation (3.2) into (3.3) gives the following double sum:

$$E_b = \int_0^T \sum_{j=1}^{N}\sum_{i=1}^{N} A_i A_j \cos(\omega_0 t + \phi_i)\cos(\omega_0 t + \phi_j)dt\,. \tag{3.4}$$

Using the trigonometric identity cos*(a)*cos*(b)*=1/2[cos*(a+b)* + cos*(a-b)*], equation (3.4) can be rewritten as:

$$E_b = \int_0^T \sum_{j=1}^{N}\sum_{i=1}^{N} \frac{A_i A_j}{2}\left[\cos(2\omega_0 t + \phi_i + \phi_j) + \cos(\phi_i - \phi_j)\right]dt\,. \tag{3.5}$$

For communication systems, the carrier frequency ω_0 is much greater than the bit frequency ($1/T$). This will cause the integration over a bit period of the term containing

$\cos(2\omega_0 t + \phi_i + \phi_j)$ in equation (3.5) above to go to zero. Also recognizing that when *i=j*, $\cos(\phi_i - \phi_j)$ = cos(0) = 1, and equation (3.5) can then be rearranged and simplified to read:

$$E_b = \sum_{i=1}^{N} \int_0^T \frac{A_i^2 dt}{2} + \sum_{j=1}^{N} \sum_{\substack{i=1 \\ i \neq j}}^{N} \int_0^T \frac{A_i A_j}{2} \cos(\phi_i - \phi_j) dt \,. \tag{3.6}$$

Referring to (3.6), the summand for the single sum is independent of carrier phase, and increases with increasing *N*. Even though each amplitude A_i will change in time, the probability that all values *A* will be small at the same time goes down as *N* increases. Further examination shows that the summand for the double sum is dependent on the difference of the carrier phases ϕ_i and ϕ_j. Cases exist where the double sum contains enough negative terms to cause the entire equation representing E_b to go to a value close to, or equal to, zero. This would cause signal fading to occur. Constraining the summand for the double sum in (3.6), such that the total of the double sum approaches zero, is one method to mitigate the signal fading.

The phase of the signal received at each antenna can be perturbed, or changed, with a phase shifter circuit. This can be done quite easily as will be explained. Let the phase shifter for element *i* change the phase by $\Psi_i(t)$. The received signal defined in (3.1) is then given as:

$$r_i(t) = A_i \cos(\omega_0 t + \phi_i + \Psi_i(t)) \,. \tag{3.7}$$

With this new phase adjustment term, the equation for the received bit energy becomes:

$$E_b = \sum_{i=1}^{N} \int_0^T \frac{A_i^2 dt}{2} + \sum_{j=1}^{N} \sum_{\substack{i=1 \\ i \neq j}}^{N} \int_0^T \frac{A_i A_j}{2} \cos(\phi_i - \phi_j + \Psi_i(t) - \Psi_j(t)) dt \,. \tag{3.8}$$

The double sum in equation (3.8) is now a function of the newly introduced phase adjustments. Careful selection of phase adjustment functions can force the double sum term to zero for all combinations of carrier phase *i* and *j*.

The patented invention in [7] describes the properties for a group of phase adjustment functions that will force the double sum in equation (3.8) to zero. For the purpose of illustration, consider a system with only two antennas (see Appendix B for the general case). For one antenna, the phase adjustment function is a constant equal to zero. For the second antenna, the phase perturbation is periodic at the bit rate, with value 0 radians for the first half of the bit period and value π radians for the second half of the bit period. The differences in the phase perturbations are

$$\Psi_i(t) - \Psi_j(t) = 0 - 0 \text{ for the first 50\% of the bit period} \tag{3.9}$$

and $$\Psi_i(t) - \Psi_j(t) = 0 - \pi \text{ for the second 50\% of the bit period.} \tag{3.10}$$

Following the above rules and designating $\Psi_i(t) = \Psi_1$, E_φ is used to represent the double sum term of equation (3.8) and can be written as:

$$E_\varphi = \sum_{j=1}^{2} \sum_{\substack{i=1 \\ i \neq j}}^{2} \int_0^T \frac{A_i A_j}{2} \cos(\phi_i - \phi_j + \Psi_i(t) - \Psi_j(t)) dt \,. \tag{3.11}$$

Expanding equation (3.11) gives:

$$E_\varphi = \int_0^{T/2} \frac{A_2 A_1}{2} \cos(\phi_2 - \phi_1) dt + \int_0^{T/2} \frac{A_1 A_2}{2} \cos(\phi_1 - \phi_2) dt$$

$$+ \int_{T/2}^{T} \frac{A_2 A_1}{2} \cos(\phi_2 - \phi_1 + \pi) dt + \int_{T/2}^{T} \frac{A_1 A_2}{2} \cos(\phi_1 - \phi_2 + \pi) dt \,. \tag{3.12}$$

Using the trigonometric identity:

$$\cos(a+\pi) = \cos(a-\pi) = -\cos(a)\,, \tag{3.13}$$

equation (3.12) is found to be equal to zero. Referring back to equation (3.8), this is the desired result.

As mentioned previously, this thesis will study two types of phase perturbation; perturbation by a square wave signal (180 degree hard phase shift as described in equation (3.12)), and perturbation by a sinusoid.

4 RF Combining Implementation

Chapter 4 begins by describing a top-level block diagram of the system as it is implemented in the FPGA based transmitter and receiver. The implementation of the necessary digital filters is also described. Many of the concepts presented in this chapter are also applicable to the Simulink simulation described in Chapter 5.

4.1 Transmitter Block Diagram

The hardware implementation of the transmitter used in this thesis is shown in Figure 4.1. A single 8 MHz sample clock is provided to both the FPGA and the D/A converter. A binary data input line is supplied to the FPGA. A clocking network within the FPGA provides both the chipping clock for the data spreader (2 MHz clock), and the 181.1 kb/s clock for sampling the data input (181.8 kb/s). The input data is provided to a differential encoder and data spreading block where it is spread by the 11 chip Barker sequence described in Section 2.4.

The output of the differential encoder and data spreader is then used as the input to the square root raised cosine (SRRC) filter. The SRRC filter is used to band limit the signal and to mitigate intersymbol interference (ISI). It is described in more detail in Section 4.5.

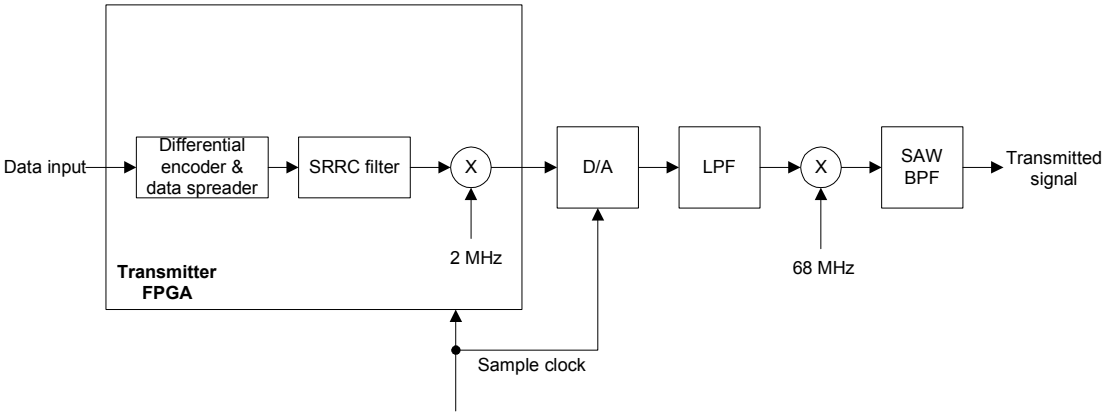

Figure 4.1 Block Diagram of Transmitter

The output of the SRRC is up-converted to a centre frequency of 2 MHz using the digital mixer described in section 2.3. The modulated and translated data spectrum is then provided to the 10-bit Analog Devices AD9731 D/A converter. A low pass filter (LPF) is used as a reconstruction filter, and to remove the image spectrum centred at 6 MHz that was shown in Figure 2.10.

To mix the spectrum centred at 2 MHz to 70 MHz, an analog mixer is driven by a 68 MHz local oscillator (LO). A SAW band-pass filter (BPF) is used to eliminate the image frequency at 66 MHz created by the mixer. The output of the SAW filter is then passed directly to the receiver block.

4.2 Receiver Block Diagram

The hardware implementation of the receiver used in this thesis is shown in Figure 4.2. The signal from the transmitter block is split into two paths, one experiences a delay and the other does not. The direct and delayed signals model the output from two receiving antennas. The analog delay that is used to model the second antenna signal is

implemented using sections of RF coaxial cable. The analog delay is measured using a network analyzer. A switch operating at twice the bit rate controls the path between the analog delay and the summer. For the first half of the bit period, the output from the analog delay block is input directly into the summer. For the second half of the bit period, the output from the analog delay block is shifted in phase by 180 degrees. The switch and phase shift by 180 degrees is implemented using a discrete off-the-shelf BPSK modulator. A SAW BPF is used to filter the signal to a reasonable bandwidth. An amplifier is used to increase the signal power to recover from the insertion loss caused by the SAW filter. The amplifier also sets the signal to an appropriate level for use by the A/D converter.

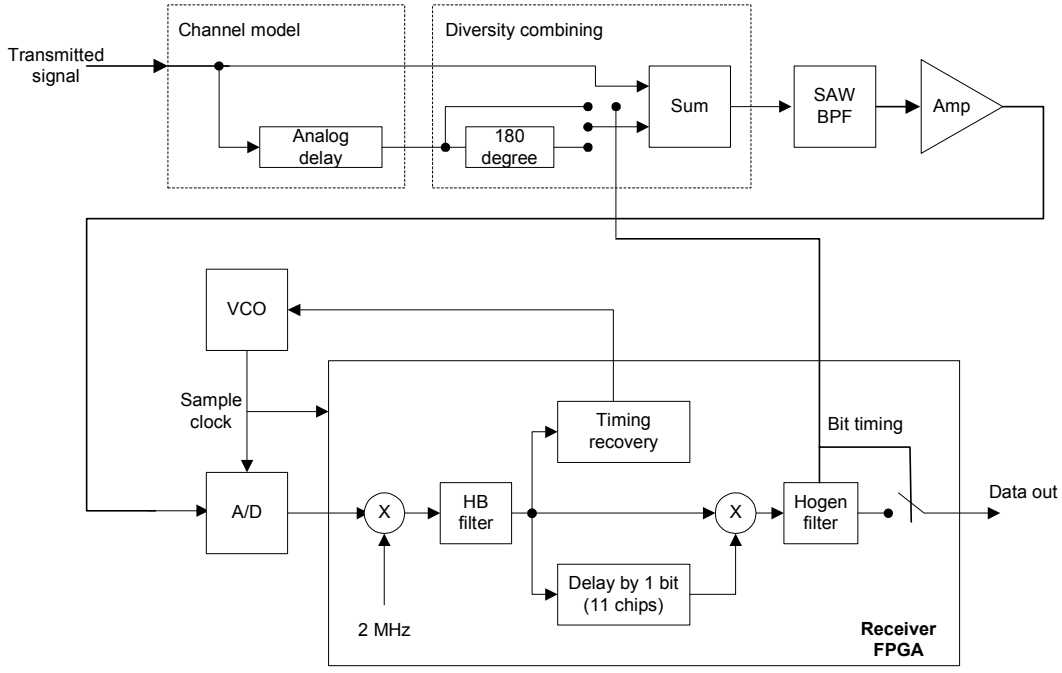

Figure 4.2 Block Diagram of Receiver

The 10-bit, Analog Devices model AD9200, A/D converter is clocked at 8 MHz. The sample clock for the A/D is controlled with a voltage-controlled oscillator (VCO).

The VCO setup in Figure 4.2 is simplified for the sake of clarity. In actuality, the output from a 20 MHz M-TRON model MV16V2CD VCO is divided by two to and output as a 10 MHz signal. The 10 MHz signal is used as the reference input for an HP 33120 arbitrary waveform generator. The HP 33120 supplies the 8 MHz square wave sample clock to the A/D converter and the FPGA.

The 10-bit digital input signal to the FPGA is centred at 2 MHz as shown in Figure 2.10. A digital mixer within the FPGA is used to down-convert the spectrum centred at 2 MHz to baseband. The digital mixer creates an image frequency centred at 4 MHz. The image frequency is removed using the half-band (HB) filter described in section 4.4.

After the HB filter, differential detection is used to recover the originally spread data sequence. A Hogenauer filter is used as an economical implementation of a sum and clear module. The Hogenauer filter is sampled to produce the output decision value. The Hogenauer filter is described in more detail in section 4.3.

The timing recovery module determines the optimum sampling point. The timing recovery module sends a control signal to the VCO and controls the overall timing of the receiver. The timing recovery module is described in section 4.6

4.3 Hogenauer Filter

An integrate-and-dump module is commonly used in spread spectrum receivers. The integration is synchronized to a bit time boundary and the output is sampled just prior to dumping. An overall positive value indicates a logic 1, and an overall negative value indicates a logic 0. The difficulty with an integrate-and-dump module is that the receiver must know the precise time to perform the dump. A Hogenauer filter is used in

this thesis as an implementation of an integrate-and-dump module. The advantage of the Hogenauer filter is that no "dumping" is necessary.

An implementation of the digital equation described by equation (4.1) is shown in Figure 4.3. A five input summation block is used for the implementation.

$$y(n) = x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4)\,. \tag{4.1}$$

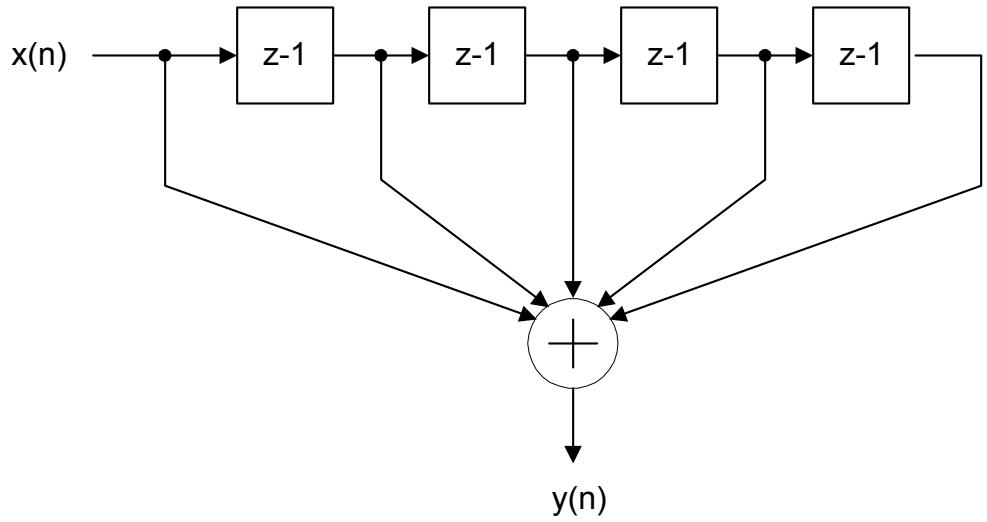

Figure 4.3 Implementation of Digital Equation (4.1)

Since the gain of each of the filter coefficients is 1, equation (4.1) can be written in the equivalent form:

$$y(n) = u(n) - u(n-5) \tag{4.2}$$

where *u(n)* is the unit step function. Equation (4.2) is illustrated graphically in Figure 4.4.

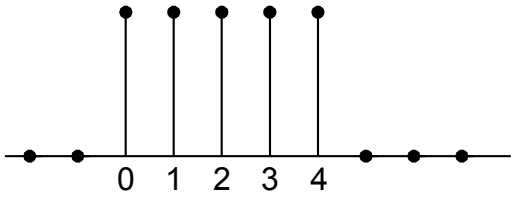

Figure 4.4 Graphical Representation of Unit Step Response

The filter in Figure 4.3 may be implemented more economically using equation (4.2). This economical implementation is shown in Figure 4.5 [8]. It uses two 2-input summers, which are more economical to implement in digital logic than the five input summer shown previously.

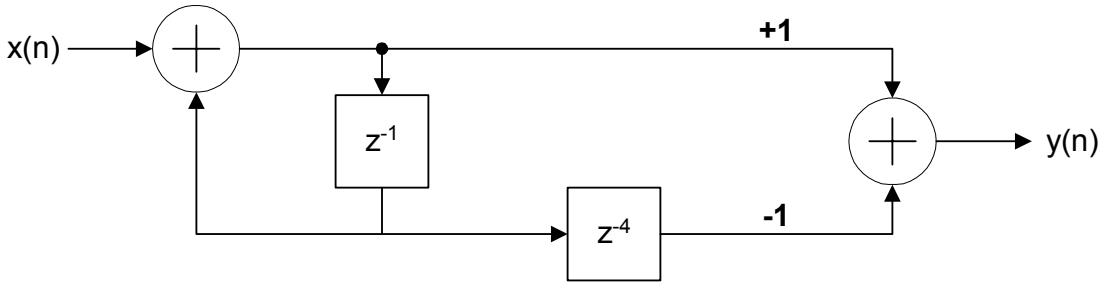

Figure 4.5 Generic Hogenauer Filter

For the receiver design in this thesis, the Hogenauer filter is used to integrate over the length of the Barker spreading code (11 chips). With 8 MHz samples, each chip is sampled 4 times and the Hogenauer filter can be described by the equation:

$$y(n) = u(n) - u(n-44). \tag{4.3}$$

This Hogenauer filter can be implemented using the specific filter design shown in Figure 4.6.

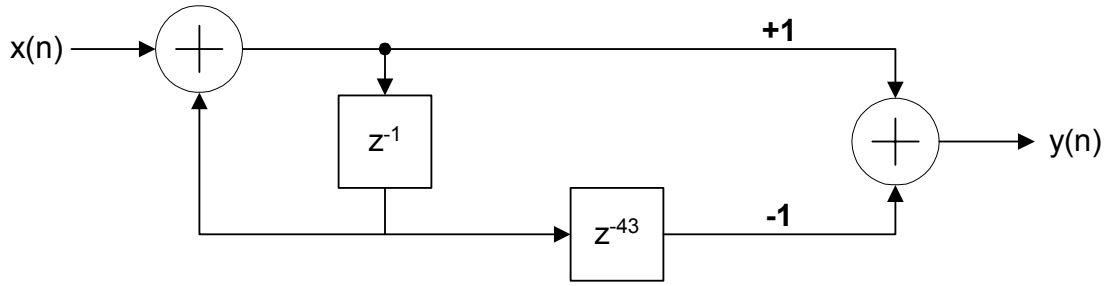

Figure 4.6 Hogenauer Filter Implementation

The Hogenauer filter must be sampled at the appropriate time to produce the decision variable. The design of the decision variable can best be explained through the use of an example. Given a repeating input data sequence of '+1 -1 +1 –1 –1' into the transmitter of Figure 4.1, and assuming for the moment no diversity combining, the output of the Hogenauer filter in the receiver is shown as the solid line in Figure 4.7. The Hogenauer filter is sampled at the peaks to determine the value of the decision variable. A positive value of the decision variable indicates a '+1' (data value 1) was sent, a negative value of the decision variable indicates a '-1' (data value 0) was sent.

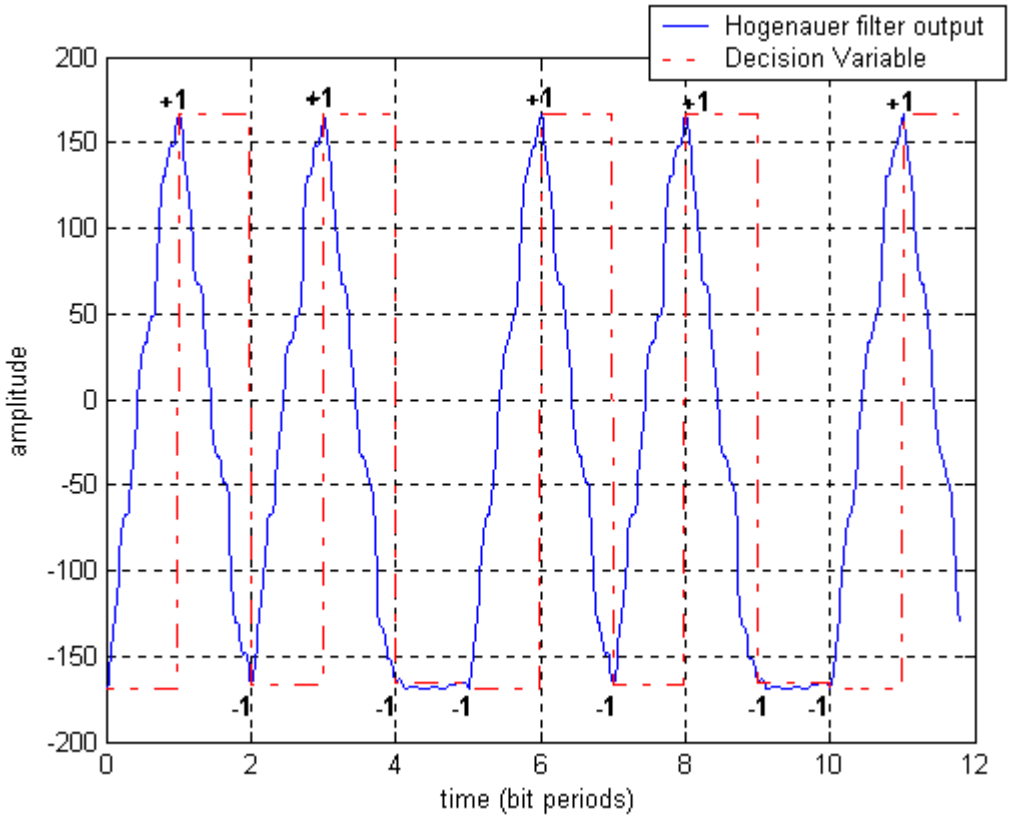

Figure 4.7 Hogenauer Filter Output with Repeating '+1 –1 +1 –1 –1' Data Input

4.4 Half-band Filter

A filter is needed to eliminate the image frequency that appears after the final digital down conversion in the FPGA receiver shown in Figure 4.2. Lagrange half-band filters were chosen for this application since they are very efficient to implement compared to standard finite impulse response (FIR) filters. Other properties of the Lagrange half band filters include a cut-off frequency of $\pi/2$, and equal pass and stop band ripples. Lagrange half-band filters have an impulse response given by:[9]

$$h_i(2n-1) = \frac{(-1)^{n+i-1}\prod_{k=1}^{2i}(i-k+1/2)}{(i-n)!(i-1+n)!(2n-1)} \tag{4.4}$$

with the value of $h(0) = 0.5$.

Through a trade-off of coefficient length versus filter performance, the value i was chosen to be 3. The number of coefficients is then defined as $N = 4i - 1 = 11$. Solving equation (4.4) for the 11 different coefficient values, with the middle coefficient being 0.5, gives the impulse response shown in Figure 4.8. The linear and logarithmic magnitude responses of the filter are shown in Figure 4.9 and Figure 4.10 respectively. The half-band filter was implemented with the filter structure shown in Figure 4.11.

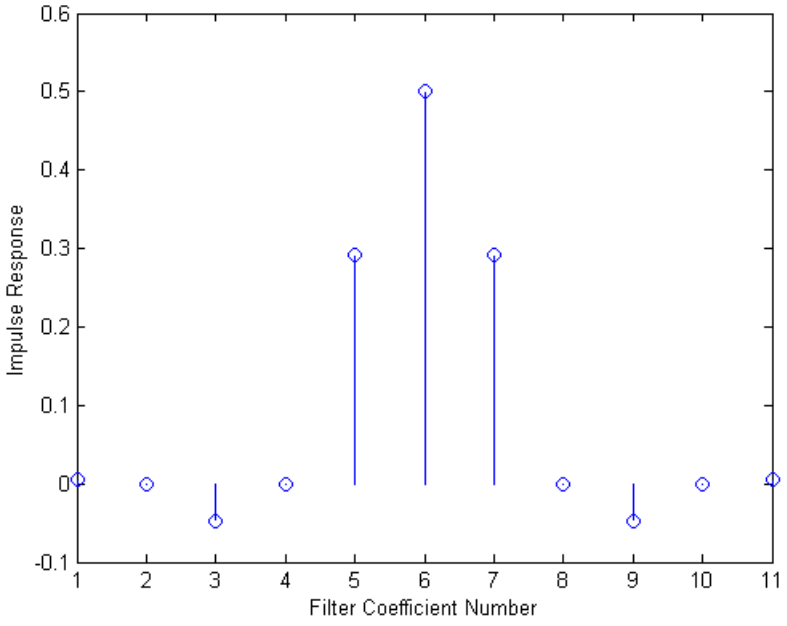

Figure 4.8 Lagrange Half-band Filter Impulse Response

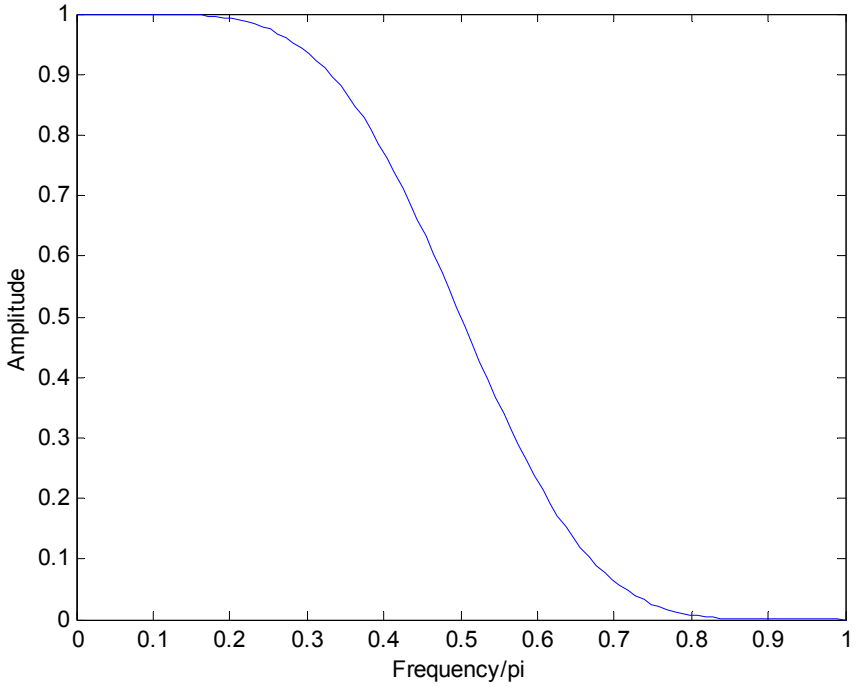

Figure 4.9 Lagrange Half-band Filter Magnitude Response (linear)

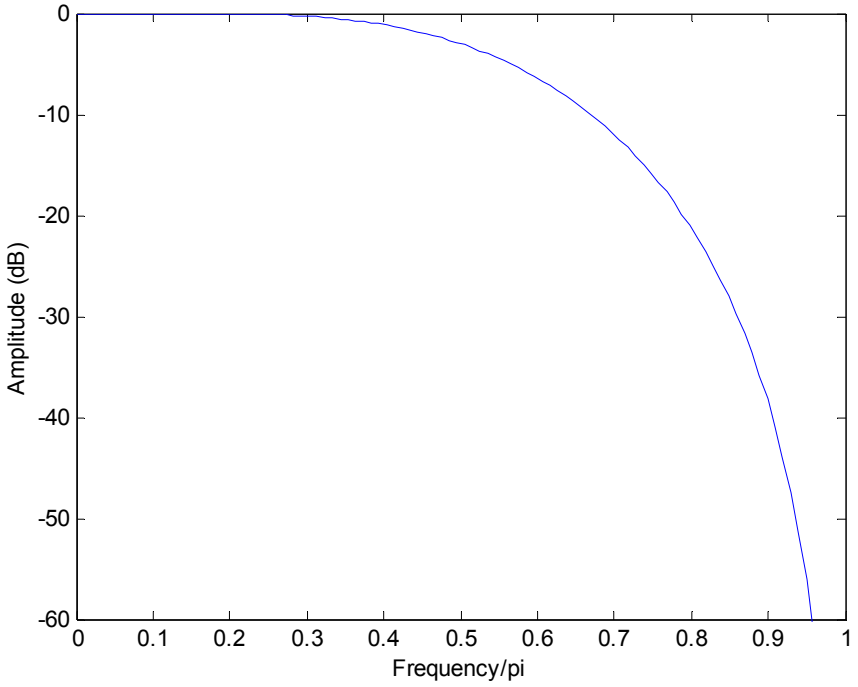

Figure 4.10 Lagrange Half-band Filter Magnitude Response (logarithmic)

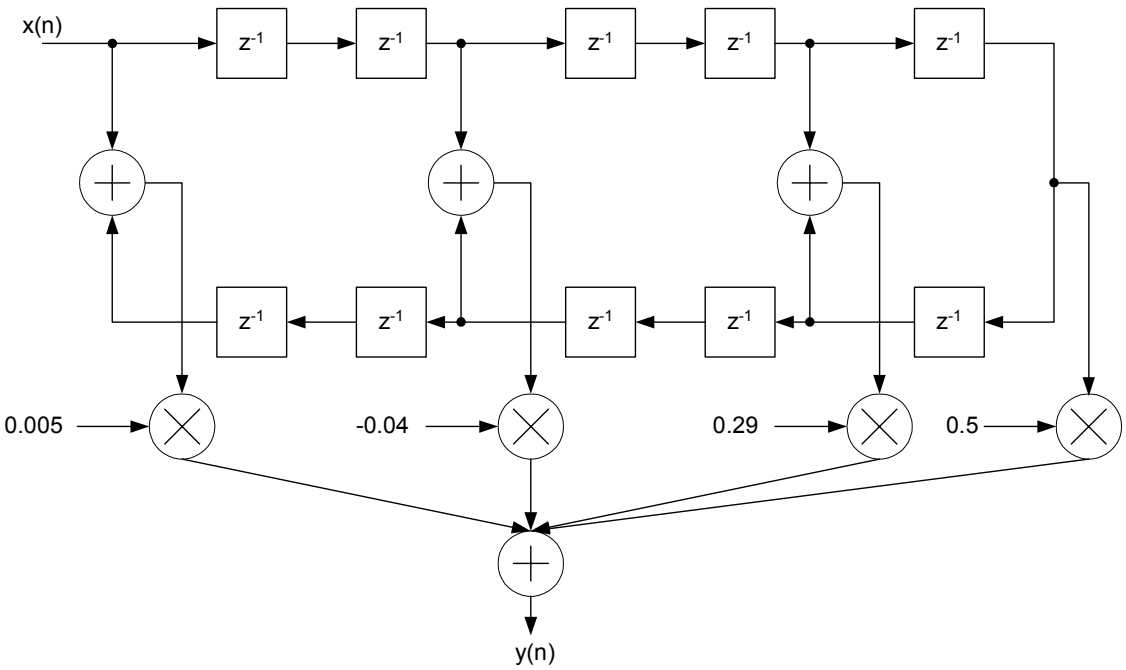

Figure 4.11 Half-band Filter Implementation

4.5 Square Root Raised Cosine Filter

The modulation method used in this thesis is the differential form of BPSK. The rectangular pulses used for BPSK lead to a relatively large transmitted bandwidth. If these rectangular pulses are filtered improperly as they pass through the communication system, they will spread in the time domain and smear into adjacent time slots and cause intersymbol interference (ISI). Pulse shaping filters can be used to limit the bandwidth of the signal in the frequency domain and to reduce ISI. The raised cosine filter is one such filter. The raised cosine filter can be described as [10]:

$$H(f)=\begin{cases} 1, & |f|<(1-r)f_0 \\ \frac{1}{2}\left\{1+\cos\left(\frac{\pi(|f|-(1-r)f_0)}{2rf_0}\right)\right\}, & (1-r)f_0<|f|<(1+r)f_0 \\ 0, & |f|>(1+r)f_0 \end{cases} \tag{4.5}$$

where f_0 is the 6-dB bandwidth of the filter and r is the rolloff factor.

The frequency responses for three different raised cosine filters with f_0=1/4 cycles/sample and rolloff factors 1, 0, and 0.35 are shown in Figure 4.12.

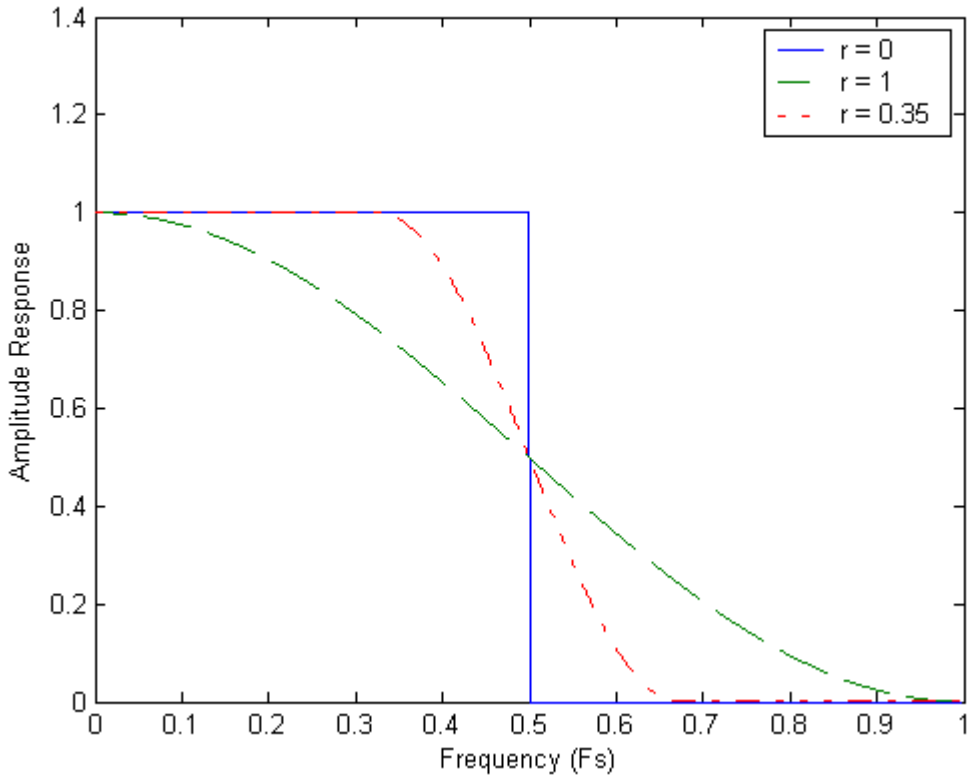

Figure 4.12 Raised Cosine Filter Response

The corresponding time domain impulse response of a raised cosine filter is:

$$h(t) = 2f_0\left(\frac{\sin(2\pi f_0 t)}{2\pi f_0 t}\right)\left[\frac{\cos(2\pi r f_0 t)}{1-(4rf_0 t)^2}\right]. \quad (4.6)$$

In practical systems, the raised cosine filter is split between the transmitter and the receiver, with each implementing a square root raised cosine (SRRC) filter. The cascaded response of the two filters is the raised cosine filter response. For this thesis, the rolloff factor was selected to be 0.35. This value was chosen because it provides a reasonable tradeoff between the implementation cost and the amount of bandlimiting that the filter provides. Figure 4.13 shows the time domain response, with the input impulses being the 11-chip Barker sequence (upsampled by 4 and shown as 'x' in the figure) and the rolloff factor being set to 0.35. The output of the transmit SRRC filter is shown as the triangle data points. This sequence is then used as the input to a receiver SRRC filter to produce the solid line in Figure 4.13.

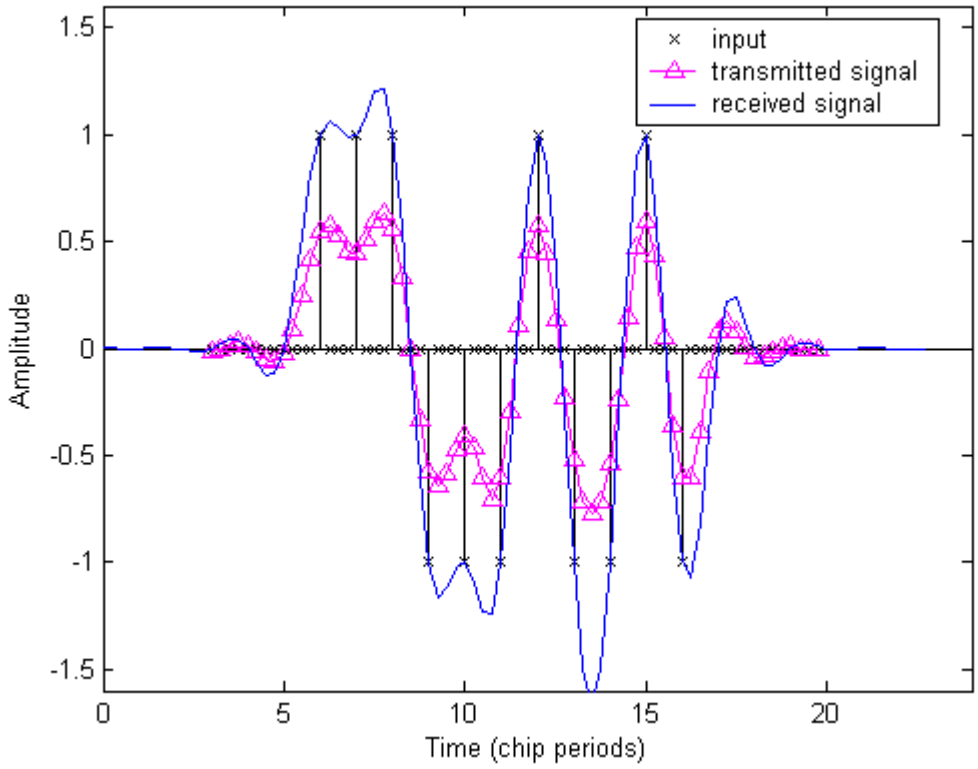

Figure 4.13 SRRC Time Domain Response

For the specific design implemented in this thesis, recall the 4 MHz wide null-to-null bandwidth of the BPSK signal that was a result of spreading the input data sequence with the Barker sequence (see Figure 2.5). The use of a SRRC filter reduces the occupied spectrum of the signal to 2 MHz. The power spectrum of the Barker-spread random data after passing the SRRC filter is shown in Figure 4.14.

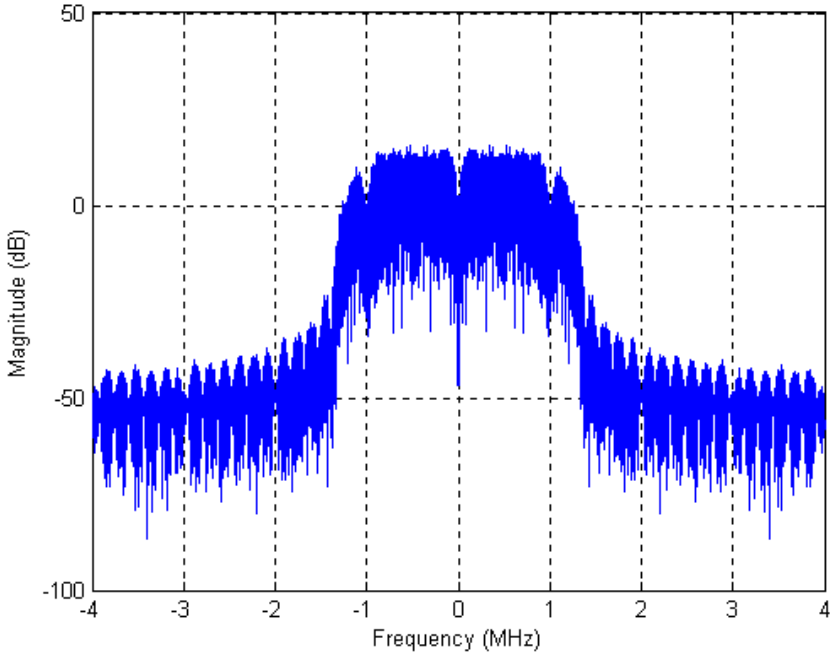

Figure 4.14 Power Spectrum of SRRC Filtered Spread Signal

The SRRC filters were designed using MATLAB. Ideal SRRC filter impulse responses are of infinite length and must therefore be truncated when implemented in a real system. A reasonable number of input chips are needed to create the finite length impulse response. A value of 6 chips was chosen. Prior to the input to the filter, the signal is upsampled from the 2 MHz chipping rate to the 8 MHz sample rate by inserting three zeros after every chip. The impulse response of a SRRC filter has an odd number of coefficients. To facilitate implementation in hardware, an even number of coefficients is used. This is accomplished by designing the filter for a sample rate of double the desired sample rate, and then down sampling the impulse response by a factor of 2 [9]. The MATLAB filter design is given in the listing below:

```
% SRRC filter design
rolloff = 0.35; % rolloff factor
len = 6;        % length of filter in input (chips)
rate = 4;       % desired upsampling rate (2 MHz * 4 to get 8 MHz)
% design the filter at twice the rate, length will be odd
h_long = rcosfir(rolloff, [-len/2 len/2], 2*rate, 1, 'sqrt');
% take every other coefficient so that length will be even
h = h_long(2:2:length(h_long));
```

The impulse response of the SRRC filter is shown in Figure 4.15.

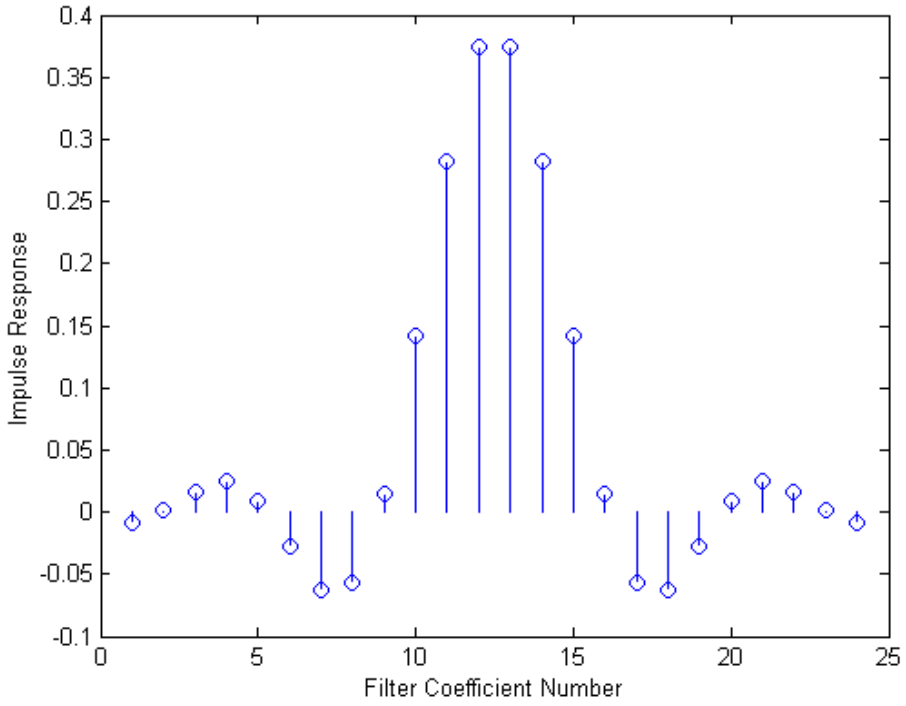

Figure 4.15 Square Root Raised Cosine Filter Impulse Response

For implementation in the FPGA, the coefficient values are scaled to represent integer arithmetic. The first 12 scaled coefficients are given in Table 4.1. The remaining 12 values are symmetric.

Table 4.1 SRRC Scaled Filter Coefficients

Coefficient number	Value
1	-8
2	1
3	16
4	24
5	9
6	-27
7	-60
8	-54
9	14
10	136
11	271
12	359

In the transmitter FPGA, the SRRC output is implemented as follows. The transmitter keeps track of the current chip and the previous 5 chips (total of 6 chips). With 6 chips, and 4 samples per chip, this accounts for the 24 samples in the impulse response shown in Figure 4.15. An internal counter keeps track of which of the 4 samples per chip the transmitter is processing. If the input chip is +1, the output is equal to the impulse response value. If the input chip is –1, the output is the negated value of the impulse response. The total output of the filter is then the combined contribution from the current chip and the previous 5 chips, depending on the current sample number within the chip. Table 4.2 shows an example of the output resulting from an input sequence of [+1 –1 +1 +1 –1 –1].

Table 4.2 Example SRRC Output

count	**+1**	**-1**	**+1**	**+1**	**-1**	**-1**	**total output**
1	-8	-9	14	359	54	-24	**386**
2	1	27	136	271	60	-16	**479**
3	16	60	271	136	27	-1	**509**
4	14	54	359	14	-9	8	**440**

4.6 Timing Recovery

The sample clock timing recovery used in this thesis is an early minus late scheme. With this method, three different sample points are taken of an arbitrary received chip as shown in Figure 4.16. Two of the points are called Early and Late. If the difference between Early and Late is zero then the third sampling point (labeled Prompt) will be at the optimum point for sampling the decision value.

If the Early sample is greater than the Late sample (as shown in Figure 4.17) then the sample clock speed must be momentarily decreased to return the system to the state shown in Figure 4.16. If the Early sample is less than the Late sample (as shown in Figure 4.18) then the sample clock speed must be momentarily increased to return the system to the state shown in Figure 4.16.

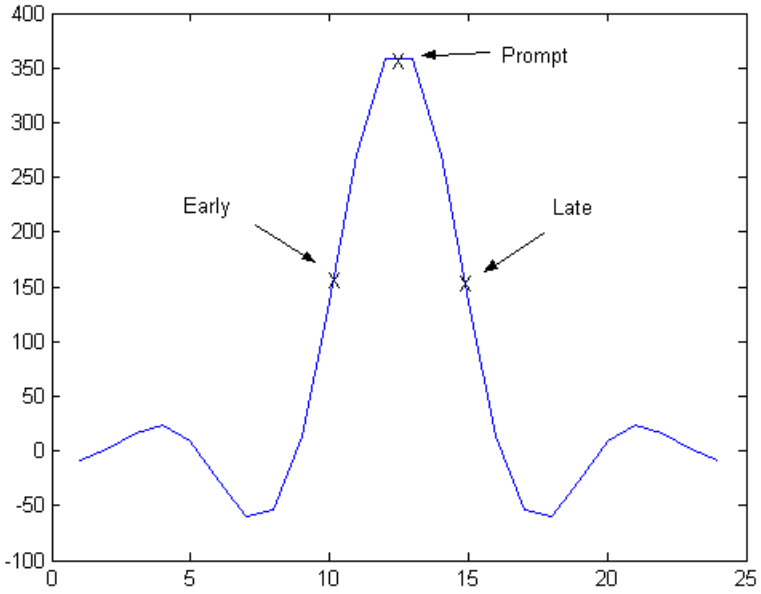

Figure 4.16 Early-Late Sampling of a Received Signal (timing recovered)

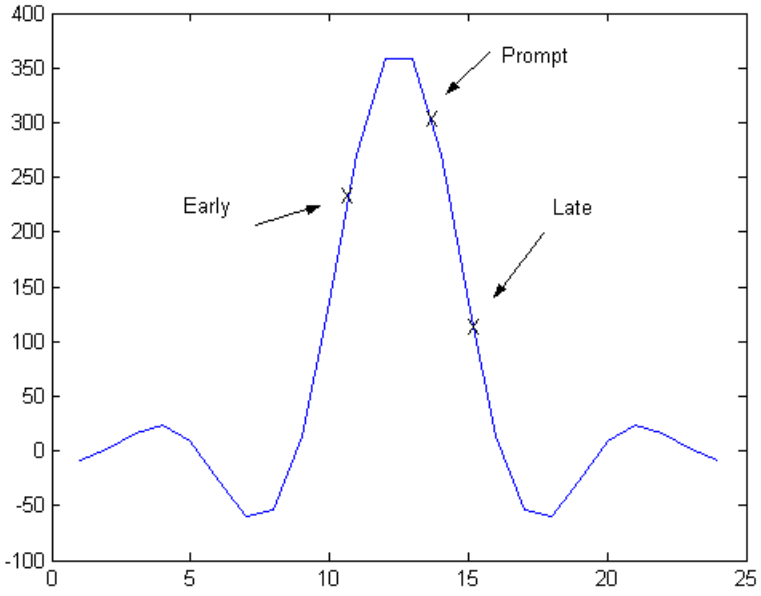

Figure 4.17 Early is Greater Than Late (slow down)

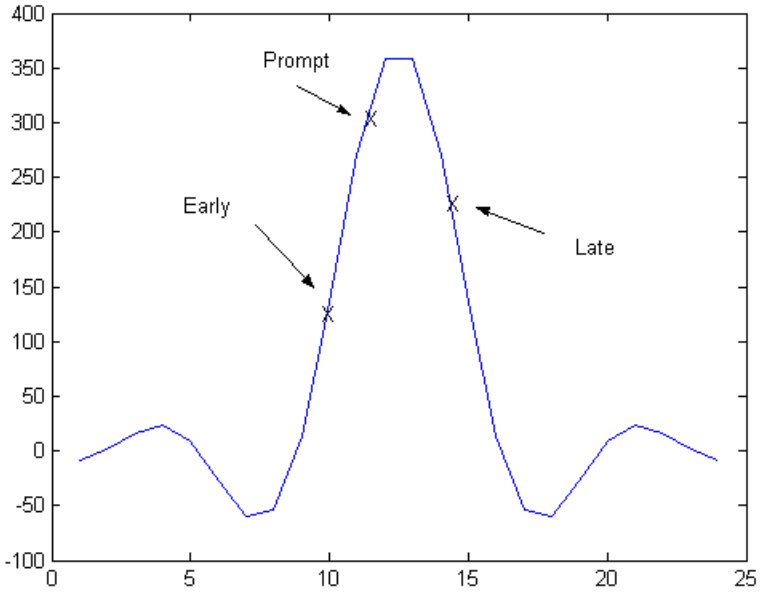

Figure 4.18 Early is Less Than Late (speed up)

The transmitted data is random and does not consist of a single pulse. The received data will sometimes consist of a set of consecutive '1' values, and sometime it will consist of a set of consecutive '-1' values. The system therefore needs to use some form of long-term averaging of Early and Late samples for the control signal. The method of taking the long-term average is not appropriate for burst mode transmission, but is appropriate for continuous mode transmission as in the transmission of digital music. Taking the variance of the Early minus Late signal is one possible method of generating the control signal. For this thesis, the absolute value of the Early minus Late signal was used to as the control signal. An absolute value measurement is easier to implement in hardware than a variance measurement. It has been shown in [11] that the absolute value method provides essentially the same results as the variance method.

The sample timing clock control block diagram for the receiver is shown in Figure 4.19. The digital processing creates the Early, Prompt, and Late samples, calculates the absolute value of Early minus Late, determines the long-term average value, and implements the control system to force the Early minus Late value to zero. The output of the control filter (a digital value) is input to a D/A converter to supply a voltage to a voltage controlled oscillator (VCO). The VCO supplies the sample clock to the A/D and the FPGA and closes the feedback system.

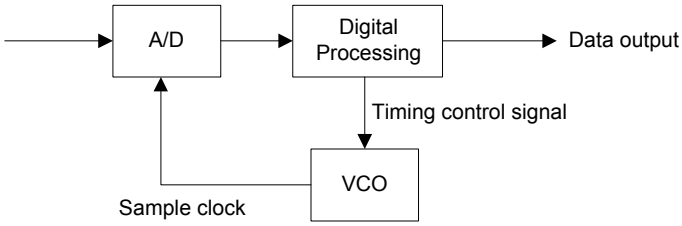

Figure 4.19 Sample Clock Control

A digital filter was used to determine the average of the absolute values of the Early minus Late signal over a very large number of samples. The digital filter used is given as

$$y(n) = x(n) + a \cdot y(n-1), \tag{4.7}$$

where n is the sample index, and a is the gain factor applied to the previous samples. The value of a was chosen so that the old results will eventually contribute zero to the current output. The a value was chosen through experimental trials and was set to a value of 63/64.

For the FPGA implementation, the sample timing recovery was used to adjust the sample clock within the duration of a chip. The other timing signal that must be

recovered is the time at which the output of the Hogenauer filter must be sampled to produce the decision variable to decide if a data value of '1' or a '0' was transmitted. For testing purposes, the relative start time of the data bit was sent across a hardware wire from the transmitter to the receiver. The sampling of the Hogenauer filter was done at 22 samples of the 8 MHz clock later. This crude method was suitable for the initial testing of the proof-of-concept prototype. The next development step after the proof-of-concept system would be to enhance the receiver to sample the Hogenauer filter output, without knowledge of the data bit timing being provided directly from the transmitter.

4.7 Bit-Error Rate Performance

In this section the expected BER performance of the DSSS DPSK receiver without diversity combining is derived. This will provide a metric against which the simulation will be tested to verify its operation. For general ease the derivation will be presented using a discrete analysis.[1]

A general block diagram of the DSSS DPSK receiver without diversity combing is shown in Figure 4.20. The general received signal in Figure 4.20 may be described as:

$$r(n) = s(n) + y(n) \tag{4.8}$$

where *s(n)* is the transmitted signal, *y(n)* is the noise, and *n* is the chip index. One full data bit would therefore consist of *N* individual chips, where *N* represents the spreading gain (and equivalently the number of chips delayed in the delay and multiply block).

[1] This discrete analysis assumes that the sample rate is fixed.

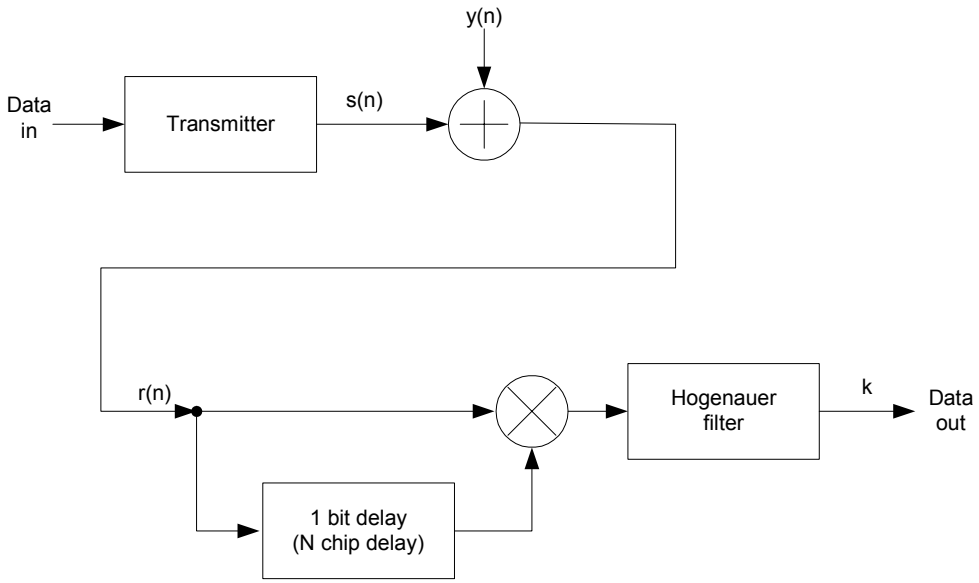

Figure 4.20 Block Diagram for BER Derivation

The output of the Hogenauer filter (shown as *k* in Figure 4.20) is:

$$k(\zeta) = \sum_{n=1}^{N} \big(s(n) + y(n,\zeta)\big)\big(s(n-N) + y(n-N,\zeta)\big), \tag{4.9}$$

where the symbol ζ is used to indicated that the noise is a sample function of a stochastic process. Expanding out (4.9) gives:

$$k(\zeta) = \sum_{n=1}^{N} s(n-N)y(n,\zeta) + \sum_{n=1}^{N} s(n)y(n-N,\zeta) + \sum_{n=1}^{N} y(n,\zeta)y(n-N,\zeta) + \sum_{n=1}^{N} s(n)s(n-N)$$

$$= k_1(\zeta) + k_2(\zeta) + k_3(\zeta) + k_4 . \tag{4.10}$$

To determine the BER of the modulation scheme it is necessary to find the variance of (4.10). Note that the term k_4 in (4.10) is not a sample function of the stochastic process. The variance of each term on the right-hand side of (4.10) will be found individually. The variance of $k_1(\zeta) = \sigma_{11}^2$ and can be calculated as follows:

$$\sigma_{11}^2 = E\left[\sum_{m=1}^{N}\sum_{n=1}^{N} s(n-N)y(n)s(m-N)y(m)\right]$$

$$= \sum_{m=1}^{N}\sum_{n=1}^{N} s(n-N)s(m-N)E\left[y(n)y(m)\right]$$

$$= \sum_{m=1}^{N}\sum_{n=1}^{N} s(n-N)s(m-N)R_{yy}(n-m)$$

$$= \sum_{n=1}^{N} s(n-N)\sum_{m=1}^{N} s(m-N)R_{yy}(n-m) \qquad (4.11)$$

and let:

$$x(m) = s(m-N) \quad \text{for } m=1,\ldots N$$

and $$x(m) = 0 \quad \text{otherwise.} \qquad (4.12)$$

To continue with the calculation it is helpful to make use of the Fourier transform properties that multiplication in the time domain is equivalent to convolution in the Fourier domain, and convolution in the time domain is equivalent to multiplication in the Fourier domain. Let:

$$X(\omega) = \Im(x(m))$$

and $$S_{yy}(\omega) = \Im(R_{yy}(n-m)) \qquad (4.13)$$

where $\Im$ is the Fourier transform, and the Fourier transform of the autocorrelation function is known as the power spectrum $S_{yy}(\omega)$. Substituting (4.12) into (4.11) and concentrating on the last two terms gives:

$$\sum_{m=1}^{N} x(m)R_{yy}(n-m)\,. \qquad (4.14)$$

Realizing that the autocorrelation function R_{yy} is essentially zero for values outside the range of *m=1* to *m=N*, the limits of the summation in (4.14) can be extended from $-\infty$ to $+\infty$. Introducing the complex exponential and evaluating at ω=0 gives:

$$= \sum_{m=-\infty}^{\infty} x(m) R_{yy}(n-m) e^{-j\omega} \Big|_{\omega=0}$$

$$= \Im(x(m) R_{yy}(n-m)) \Big|_{\omega=0}$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) S_{yy}(\omega) e^{j\omega n} d\omega \qquad (4.15)$$

where the convolution property of the Fourier transform has been used.

The variance of $k_1(\zeta)$ can then be written as:

$$\sigma_{11}^2 = \sum_{n=1}^{N} s(n-N) \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) S_{yy}(\omega) e^{j\omega n} d\omega$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) S_{yy}(\omega) \sum_{n=1}^{N} x(n) e^{j\omega n} d\omega$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) X(-\omega) S_{yy}(\omega) d\omega$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega) S_{yy}(\omega) d\omega. \qquad (4.16)$$

The variance σ_{11}^2 given by (4.16) can now be evaluated in the Fourier domain using the value of $S_{yy}(\omega)$ (equal to the double sided noise power density $N_0/2$ in watts/Hz) and the value of $S_{xx}(\omega)$ (equal to the energy per bit E_b). The calculation for σ_{22}^2 is carried out in the same manner as the calculation for σ_{11}^2. The result in both cases is the same:

$$\sigma_{11}^2 = \sigma_{22}^2 = \frac{N_0 E_b}{2}. \tag{4.17}$$

The calculation for σ_{33}^2 is done in a similar manner. Starting with:

$$\sigma_{33}^2 = \sum_{n=1}^{N}\sum_{m=1}^{N} E\left[y(n)y(n-N)y(m)y(m-N)\right]. \tag{4.18}$$

The expected value in (4.18) is a set of jointly Gaussian, zero-mean random variables and may therefore be evaluated using Isserlis's formula. Isserlis's formula of a fourth order joint moment for a set of Gaussian zero-mean random variables $A_1(\zeta)$ to $A_4(\zeta)$ is:

$$\begin{aligned} E\left[A_1(\zeta)A_2(\zeta)A_3(\zeta)A_4(\zeta)\right] &= E\left[A_1(\zeta)A_2(\zeta)\right]E\left[A_3(\zeta)A_4(\zeta)\right] \\ &\quad + E\left[A_1(\zeta)A_3(\zeta)\right]E\left[A_2(\zeta)A_4(\zeta)\right] \\ &\quad + E\left[A_2(\zeta)A_3(\zeta)\right]E\left[A_1(\zeta)A_4(\zeta)\right]. \end{aligned} \tag{4.19}$$

Evaluating (4.18) using the Isserlis's formula shown in (4.19) gives:

$$\begin{aligned} \sigma_{33}^2 &= \sum_{n=1}^{N}\sum_{m=1}^{N}\left[R_{yy}(N)R_{yy}(N) + R_{yy}(n-m)R_{yy}(n-m) + R_{yy}(n-m-N)R_{yy}(n-m+N)\right] \\ &\cong \sum_{n=1}^{N}\sum_{m=1}^{N} R_{yy}(n-m)R_{yy}(n-m) \end{aligned} \tag{4.20}$$

where it is assumed the first and last terms from Isserlis's formula are approximately equal to zero due to the autocorrelation of the noise approaching zero at N chips away from the origin.

It is possible to reduce the double sum in (4.20) to a single sum. Consider a two-dimensional space as shown in Figure 4.21 to represent the double sum for an illustrative example case of $N = 4$. Letting $\tau = m - n$, when $\tau = 0$ the calculation of (4.20) along the diagonal in Figure 4.21 is:

$$N \cdot R_{yy}^2(0). \tag{4.21}$$

Similarly, for the other values of τ, the evaluation of (4.20) becomes:

$$\sigma_{33}^2 \cong \sum_{\tau=-N}^{N}(N-|\tau|)R_{yy}^2(\tau)$$

$$\cong N\sum_{\tau=-N}^{N}R_{yy}^2(\tau) \tag{4.22}$$

where the absolute value sign is removed since it is assumed that $R_{yy}^2(\tau)$ goes to zero quickly.

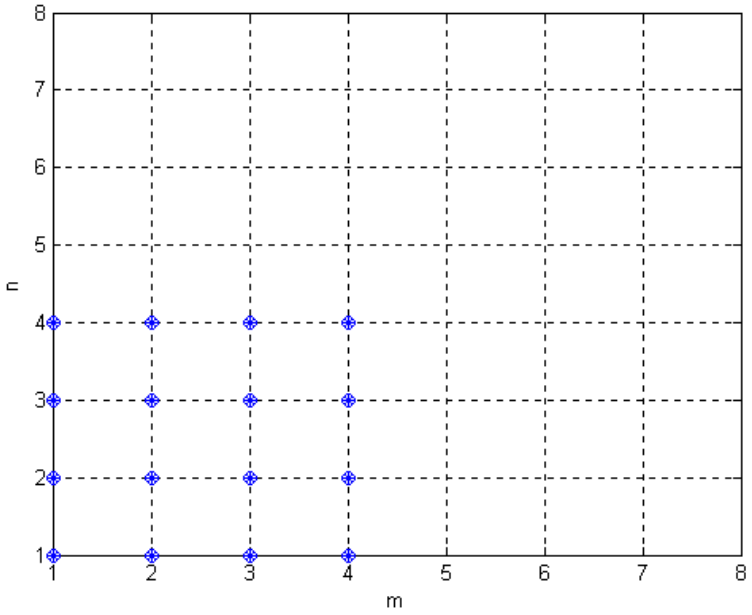

Figure 4.21 Example Double Sum Sample Space for N = 4

The variance in (4.22) can be evaluated using the Fourier transform if the complex exponential is introduced and evaluated at $\omega = 0$:

$$\sigma_{33}^2 \cong N\sum_{m-n=-N}^{N}R_{yy}^2(m-n)e^{-j\omega n}\Big|_{\omega=0}. \tag{4.23}$$

Recalling that multiplication in the time domain is equivalent to convolution in the Fourier domain, (4.23) can be written as the circular convolution:

$$\sigma_{33}^2 \cong N\frac{1}{2\pi}\int_{-\pi}^{\pi} S_{yy}(\varphi)S_{yy}(\omega-\varphi)d\varphi\,. \tag{4.24}$$

Evaluating (4.24) at $\omega = 0$, and recognizing that the power spectrum is an even spectrum, (4.24) becomes

$$\sigma_{33}^2 \cong N\frac{1}{2\pi}\int_{-\pi}^{\pi} S_{yy}^2(\omega)d\omega \cong N(\frac{N_0}{2}\frac{N_0}{2})\,. \tag{4.25}$$

The probability of error is given by:

$$P(error) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{0} e^{-\frac{(x-E_b)^2}{2\sigma^2}}dx\,. \tag{4.26}$$

Using the change in variable $z = \frac{x-E_b}{\sigma}$ we have:

$$P(error) = \frac{1}{\sqrt{2\pi}}\int_{\frac{E_b}{\sigma}}^{\infty} e^{-\frac{z^2}{2}}dz$$

$$= Q\left(\frac{E_b}{\sigma}\right). \tag{4.27}$$

Substituting in the total variance of (4.10) and realizing that the energy per bit E_b=$E_{chip}N$ (energy per chip times number of chips per bit) gives:

$$Q\left(\frac{E_b}{\sigma}\right) = Q\left(\frac{E_{chip}N}{\sqrt{N_0E_{chip}N + N\frac{N_0}{2}\frac{N_0}{2}}}\right)$$

$$= Q\left(\sqrt{\frac{E_{chip}N \cdot E_{chip}N}{N_0 E_{chip}N\left(1+\frac{N_0}{4E_{chip}}\right)}} \right)$$

$$= Q\left(\frac{E_b / N_0}{1+\frac{N}{4E_b / N_0}} \right). \qquad (4.28)$$

Note that the bandwidth of the received signal may be included in (4.28) by realizing that the spreading gain N is equal to $T_{bit}B$, where T_{bit} is the bit period in seconds, and B is the bandwidth in Hz.

Figure 4.22 is the plot of the BER given by (4.28) with a spreading gain of 11. For comparison, the BER curves for an optimum DPSK receiver (without DSSS) and BPSK are also given [10]. At higher values of E_b/N_o, differential detection of DSSS with a spreading gain of 11 performs about 3 dB worse than optimum DPSK.

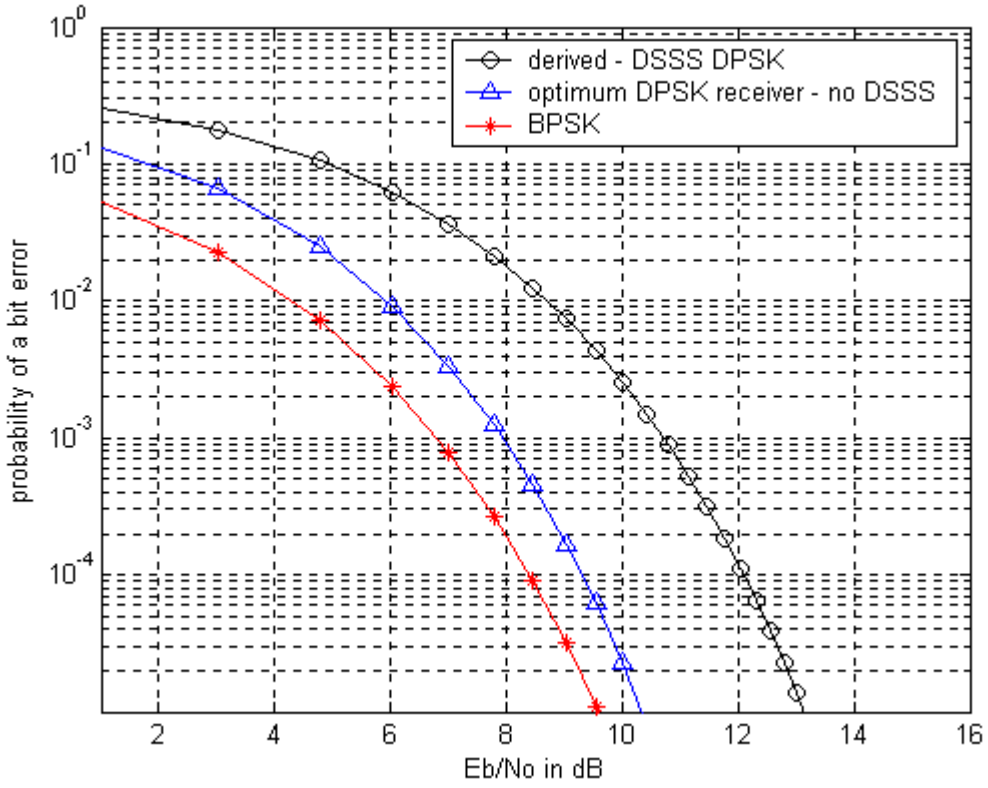

Figure 4.22 Probability of a Bit Error for Spread Spectrum DPSK

5 Simulink Implementation

The Simulink software package from The Mathworks provides a platform for simulation of discrete time communication systems. An initial simulation was developed to determine the bit error rate performance of the receiver in the presence of additive white Gaussian noise (AWGN), in the absence of fading, and without diversity combining. After this initial simulation was confirmed to be working, additional simulations were created for the RF diversity combining configurations, including square wave shifting, and sinusoidal shifting.

Simulink is capable of running simulations in continuous time mode, discrete time mode, and a mixed continuous/discrete mode. Ideally the simulations would handle the effects of timing recovery with a sample clock controlled by a VCO. Unfortunately, the discrete portions of the Simulink simulation for the receiver must run at a constant sample rate. The Simulink simulations are therefore focused on the effects of the diversity combining and ideal timing recovery is assumed. The timing recovery described in Section 4.6 is included in the FPGA implementation.

5.1 Transmitter and Channel Overview

The Simulink block diagram for the transmitter is shown in Figure 5.1. The "Uniform random data" block supplies random data of amplitude '1' or '0' with a uniform distribution. This data is called "Transmitted_Bits" and is also used by the BER calculation to compare the transmitted bits to the received bits.

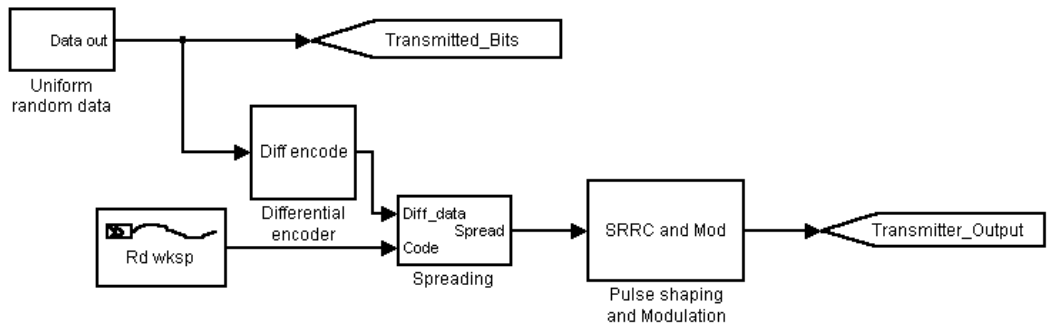

Figure 5.1 Simulink Transmitter Block Diagram

The uniform random data is input to the differential encoder block that is shown in Figure 5.2. The differential encoding is the exclusive OR of the input data and the previous output of the exclusive OR delayed by one bit period.

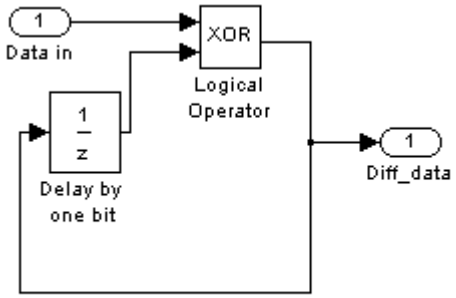

Figure 5.2 Differential Encoder Block

The differentially encoded data is spread with the Barker spreading sequence in the "Speading" block shown in Figure 5.3. The Barker sequence is read from the MATLAB workspace with the "Rd wksp" block.

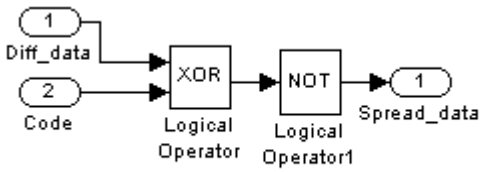

Figure 5.3 Spreading Block

The "Pulse Shaping and Modulation" block is shown in Figure 5.4. The '1' and '0' data values from the spreading block are mapped to '1' and '-1' values. These values are then put through the SRRC filter described in section 4.5. The SRRC filter block also upsamples the input by a factor of 4. A gain block is used to set the signal power to 1 watt to ease computation within the simulation of the E_b/N_0. The shaped pulses out of the SRRC filter are then mixed with a 2 MHz sine wave to produce an IF signal centred at 2 MHz for output from the block.

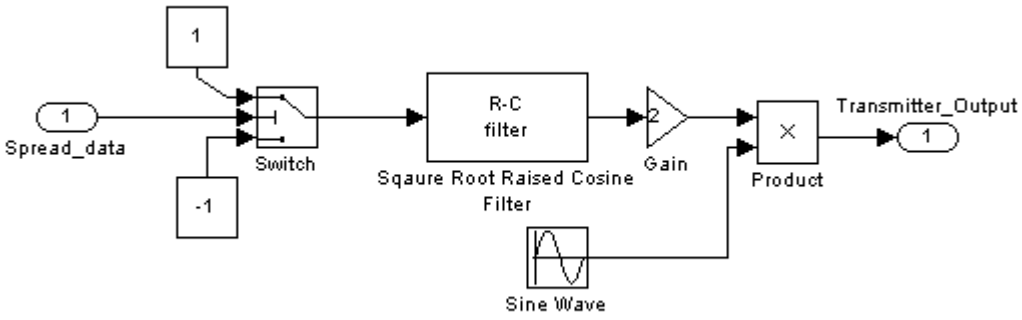

Figure 5.4 Pulse Shaping and Modulation Block

The signal output from Figure 5.1 labeled "Transmitter_Output" is a noise-less signal. To simulate the effects of the multipath communication channel, the signal "Transmitter_Output" is input to the channel block shown in Figure 5.5. For the

simple one path case with no diversity switching, the details of the channel block are shown in Figure 5.6 (for the single path case the "Antenna_2" output is not used). The AWGN block runs in the signal to noise ratio (SNR) mode. One of the block parameters in SNR mode is the input signal power, which in this case is set to 1 watt by using the gain block after the SRRC filter shown in Figure 5.4.

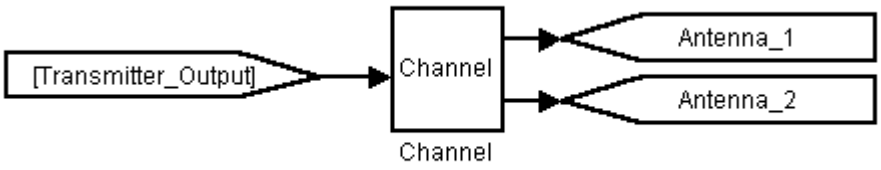

Figure 5.5 Channel Block

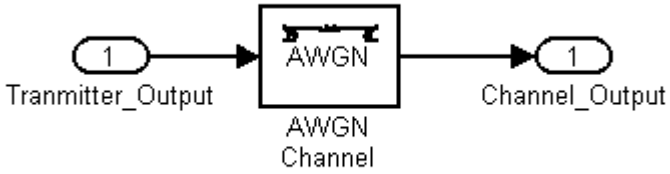

Figure 5.6 Channel Block For Simple One Path Case

For the multipath case with diversity combining, the channel block diagram is as shown in Figure 5.7. For the diversity combining case a line of sight path is used (using the same AWGN channel block shown previously in Figure 5.6) and a secondary fading path (lower part of Figure 5.7). For the secondary path, AWGN is first added to the passband signal. Simulink includes a 'Multipath Rayleigh Fading Channel' block that processes complex-valued baseband signals. Within this fading block, the amplitude

values are drawn from a Rayleigh distribution. The varying phase values for the 'Multipath Rayleigh Fading Channel' block are drawn from the Doppler spectrum from the Jakes channel model [12]. To allow the 'Multipath Rayleigh Fading Channel' block to be used with the real-valued passband signal from the AWGN block, a real/imaginary-to-complex block is used and the imaginary input is set to a value of zero. Similarly, the complex-valued output from the 'Multipath Rayleigh Fading Channel' block is converted to real and imaginary parts before being passed out of the block, with the imaginary part not used (terminated in the block to prevent Simulink warning messages). The outputs from the channel block shown in Figure 5.7 are the two antenna signals that are used as inputs for the receiver block described in the next section.

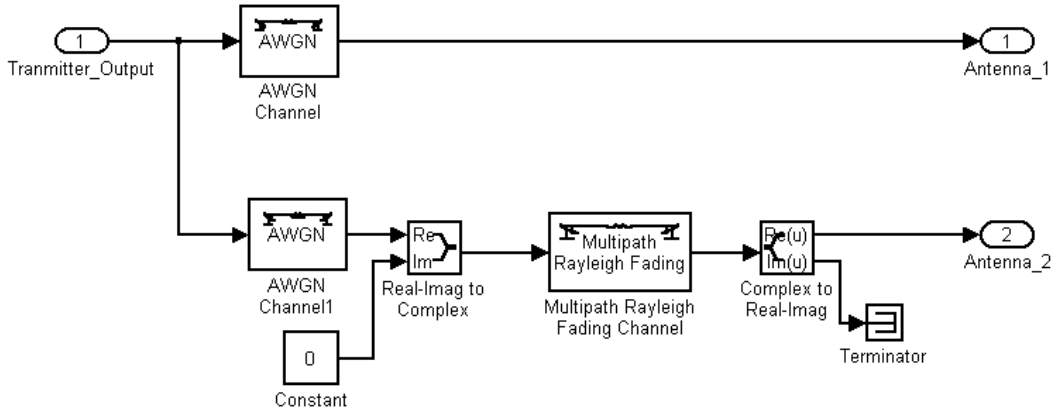

Figure 5.7 Channel Block For Two-Paths

5.2 Receiver Overview

The Simulink block diagram for the receiver is shown in Figure 5.8. The two inputs to this block are the two separate antenna signals that are output from the block shown in Figure 5.5.

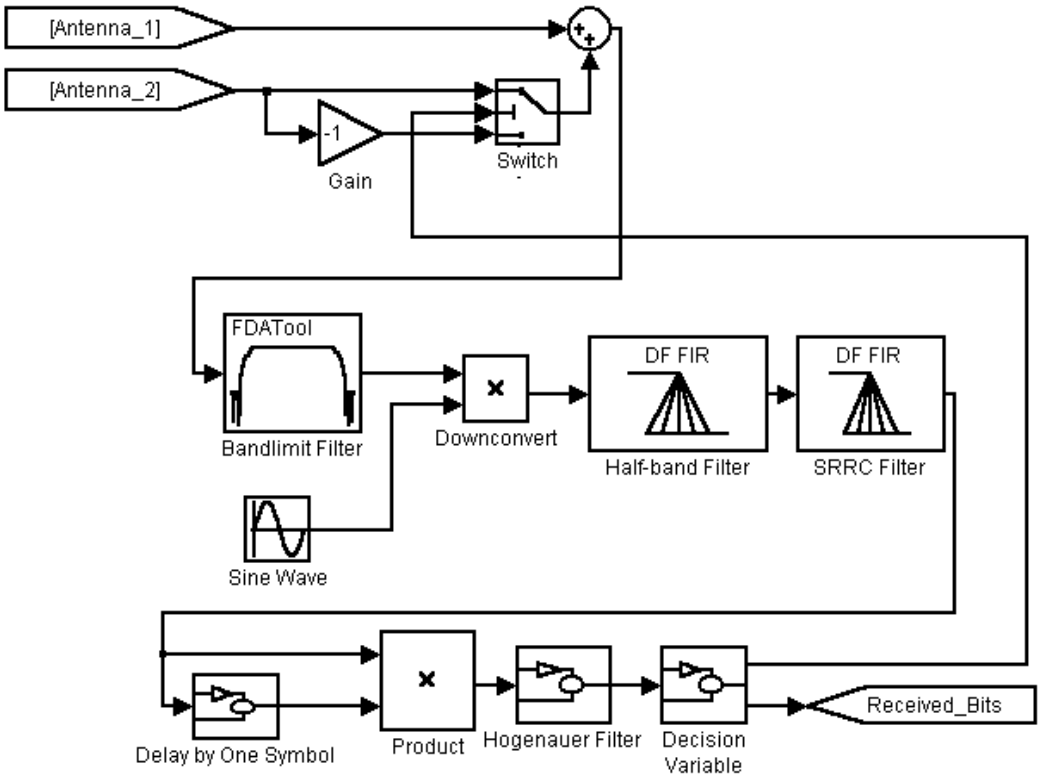

Figure 5.8 Simulink Receiver Block Diagram (square wave diversity switching)

The actual RF diversity switching is implemented in two different ways. Diversity switching with a 180 degree phase shift is shown in Figure 5.8. This is equivalent to mixing the signal with a square wave. The 'Switch' element is controlled with a 50/50 duty cycle signal, with the rising edge of the signal coinciding with the start of a bit transition. A time domain snapshot of the inputs to the summation block, after the

diversity switching, is given in Figure 5.9 (with the signal sampled at 8 MHz). For this figure, the power of the switching path was set to be almost equal to the power of the non-switching path.

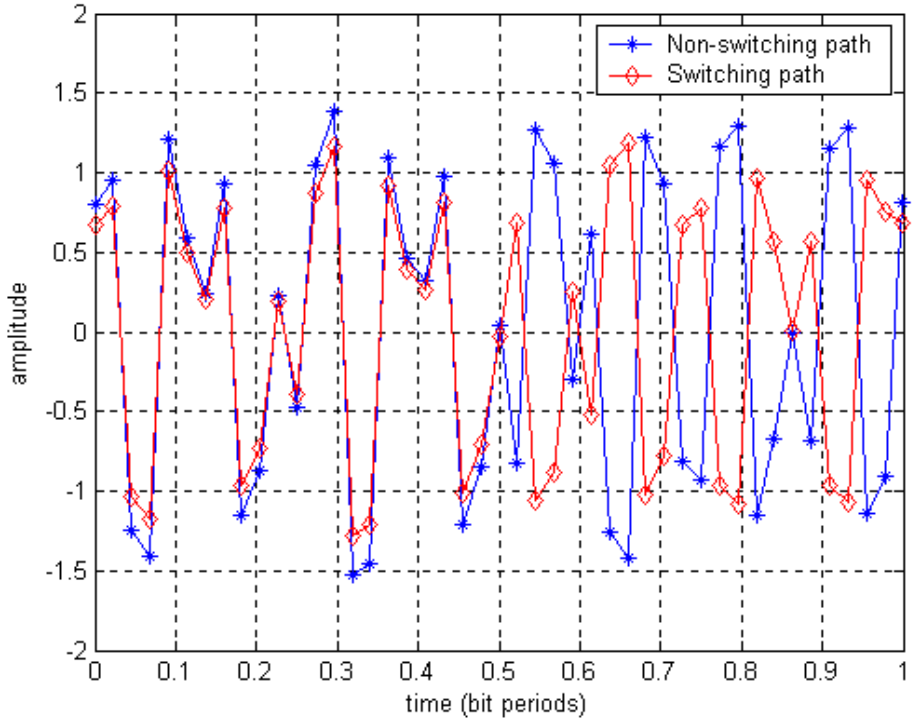

Figure 5.9 Diversity Switching (square wave switching)

Figure 5.10 shows the diversity switching implementation with the switching operation implemented by mixing the secondary fading path with a sine wave. In the simulation, the phase of the sine wave controlling the mixing operation of the Antenna_2 signal is aligned to the bit clock. A time domain snapshot of the inputs to the summation block after the sine wave diversity switching is given in Figure 5.11. Figure 5.11 was produced using the same input signal as was used for Figure 5.9.

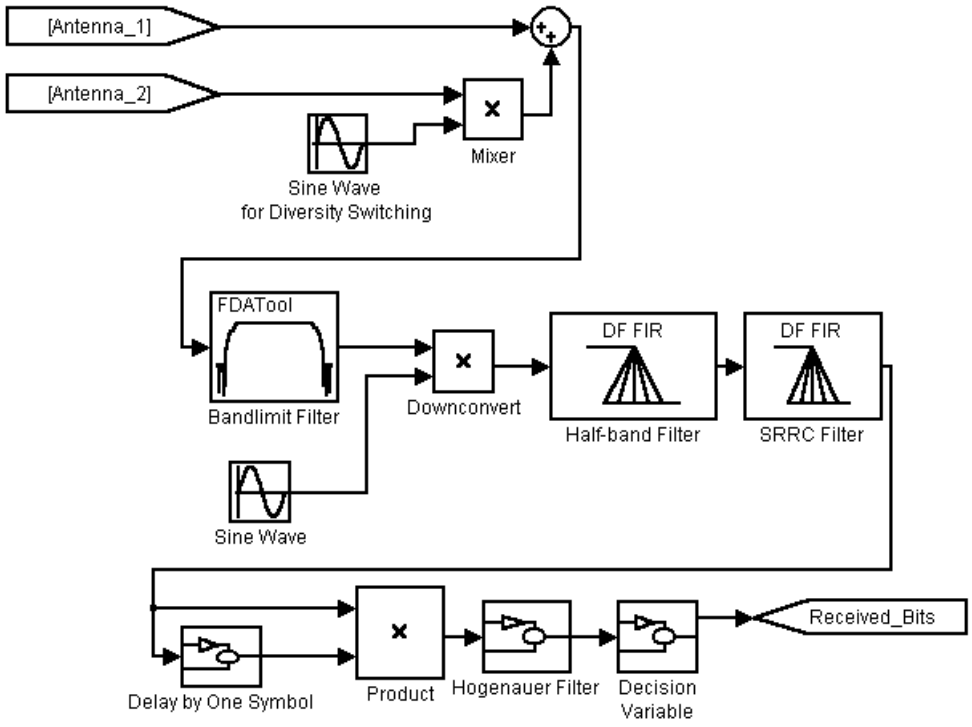

Figure 5.10 Simulink Receiver Block Diagram (sine wave diversity switching)

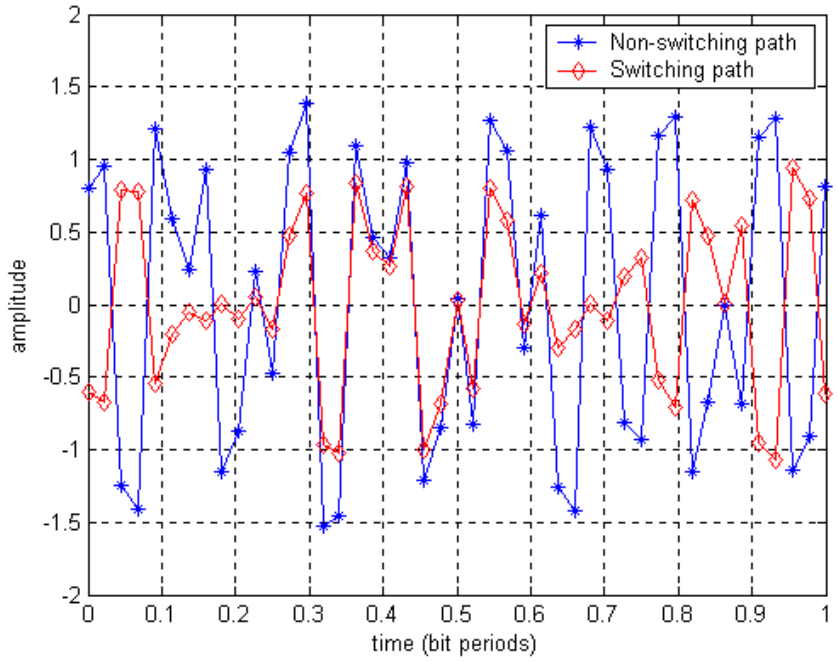

Figure 5.11 Diversity Switching (sine wave switching)

After the summing block, the next block in the receiver is the intermediate frequency (IF) filter (labeled 'Bandlimit Filter' in Figure 5.8). This is implemented as an order 46 (11.5 chips long) finite impulse response (FIR) bandpass filter, with a centre frequency of 2 MHz and a passband of width 2 MHz. The magnitude response of the filter is shown in Figure 5.12. The filter group delay is 23 samples. The group delay is an important parameter in Simulink simulations because to get meaningful BER results the transmitted bits and received bits must be lined up in time. One way to line up the transmitted and received bits is to know the combined delay through the simulation. This simulation also uses ideal timing recovery, and therefore the total delay through the simulation is needed to set the rising edge of the signal that samples the Hogenauer filter to produce the decision variable.

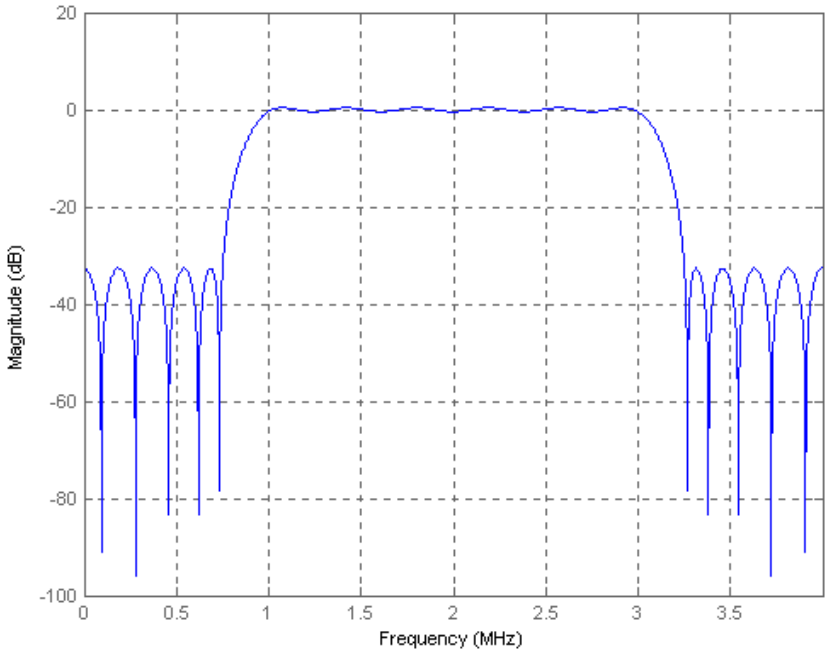

Figure 5.12 Simulink IF Filter Magnitude Response (dB)

The centre frequency of the signal after the IF filter is 2 MHz. The signal is mixed with a local sine wave using the 'Downconvert' block to mix the signal to baseband. A direct form FIR filter is used to implement the half-band filter that was described in Section 4.4. The half-band filter is used to remove the upper sideband image created by the mixing operation. The half-band filter is immediately followed by the SRRC receive filter, as described in Section 4.5.

After the SRRC, the signal is delayed and multiplied to perform the differential detection. The delay is by one bit (44 samples at 8 MHz).

The Hogenauer filter implementation is the same as was shown in Figure 4.6.

The implementation of the decision variable is shown in Figure 5.13. A sample and hold circuit is used to sample the output of the Hogenauer filter. The desired sample point is when the Hogenauer filter is at its peak value in the absence of noise. This simulation assumes ideal timing recovery. A pulse generator supplies the rising edge trigger signal. The generated pulse has a 50/50 duty cycle and a period of one bit period. To ensure that the sample and hold occurs at the optimum sampling time, the phase delay of the pulse generator is set to be equal to the combined group delay of all the filters in the system. This pulse generator circuit is labeled as 'Diversity_Switch_Control' in Figure 5.13. This signal is output from the decision variable block, and it is used to control the diversity combining switch shown in Figure 5.8. After the Hogenauer filter output is sampled by the sample and hold block, a switch is used to map positive values to data value '1' and negative values to data value '0'. The sequence of '1' and '0' data values is output from the block as the signal 'Received_Bits'.

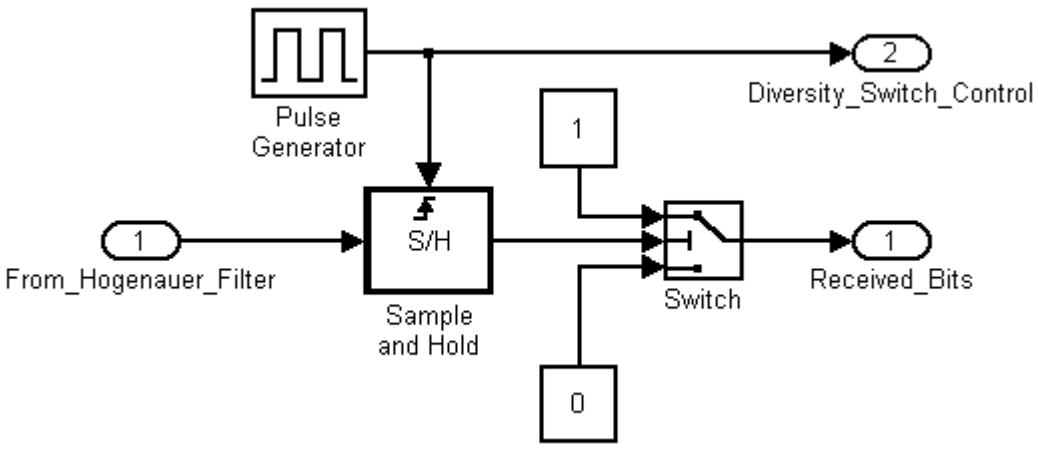

Figure 5.13 Decision Variable Block Diagram

5.3 Bit Error Rate Determination

The main goal of the simulation is to determine the BER performance of the system. To calculate the BER, each received bit is compared to the corresponding transmitted bit to determine if an error has occurred. The total number of errors per total number of transmitted bits is the BER. The Simulink block that performs this calculation is shown in Figure 5.14. The transmitted bits are denoted by the variable name 'Transmitted_Bits', and are taken from the transmitter input as was shown in Figure 5.1. The input data is at a rate of 181.8 kb/s, but because of the way that Simulink inherits sample rates, the line from which 'Transmitted_Bits' is taken off of has a sample rate of 8 MHz. Therefore, the sample rate of 'Transmitted_Bits' is reduced by a factor of 44 to return the signal to the bit rate of 181.8 kb/s. The 'Received_Bits' variable is from the receiver, and it is the output of the decision variable as shown in Figure 5.8. In Simulink, this signal is also at a sample rate of 8 MHz and must be downsampled by a

factor of 44 to be able to compare it to the input data. The error rate calculation is then performed using a native Simulink block, and the BER result is output to the MATLAB workspace for analysis.

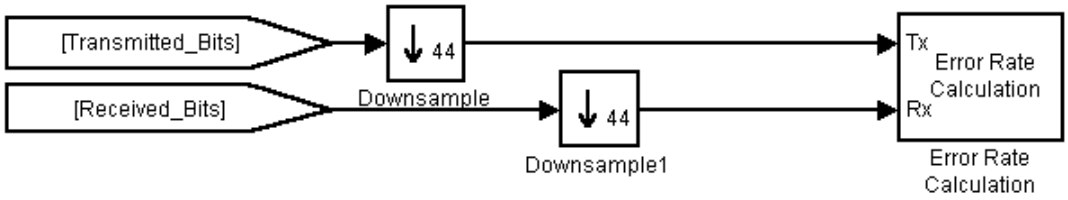

Figure 5.14 Simulink BER Calculation Block

6 Simulation Testing

This chapter uses the simulation to examine the relative performance of the RF diversity combining method during periods of deep multipath fades. The chapter begins with a discussion of the change in bandwidth of the signal due to the RF diversity combining. Eye diagrams are then presented to show how the RF diversity combining technique can "re-open" the eye during a deep fade. The simulation tests for the BER testing are described and the results are presented.

6.1 Effects of Diversity Switching on Signal Bandwidth

The phase perturbations caused by the diversity combining cause the bandwidth of the signal to spread out in a manner similar to a mixing operation. To see this effect, test signals were taken off the simulation at various points in the system to look at the power spectrum of the signal. The effects of AWGN were minimized by setting the E_b/N_0 to a very high value.

6.1.1 Square Wave Diversity Switching

To begin, recall the shape of the power spectrum of the input data sequence given in Figure 2.2. The shape of the power spectrum of the signal after SRRC filtering in the transmitter, but prior to mixing to the simulation IF of 2 MHz is shown in Figure 6.1. The same signal after mixing to the 2 MHz IF is shown in Figure 6.2.

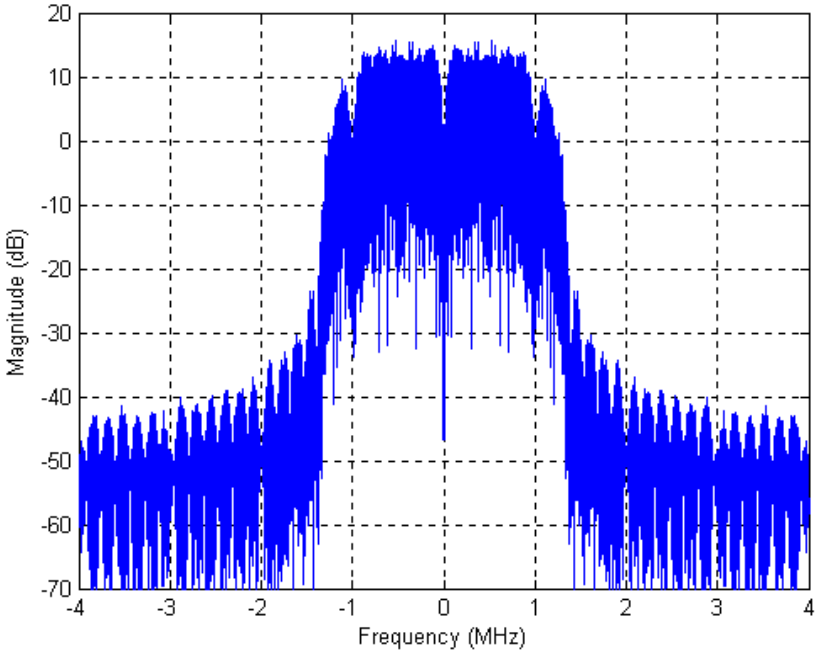

Figure 6.1 Power Spectrum of SRRC Filtered Signal

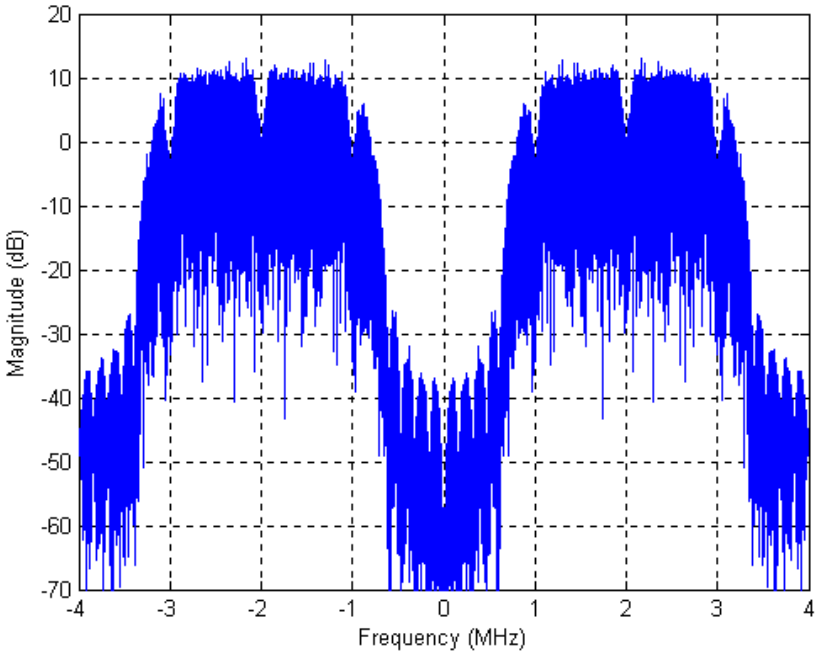

Figure 6.2 Power Spectrum of SRRC Filtered Signal After Mixing

Assume for the moment that square wave RF diversity switching is being used. With no multipath fading, the power spectrum of the signal after the diversity switch, but before combining with the non-switched arm, is shown in Figure 6.3. The spectrum of the signal now spills over into adjacent bands. The net effect is that power will be lost when the signal is passed through the receiver bandpass filter.

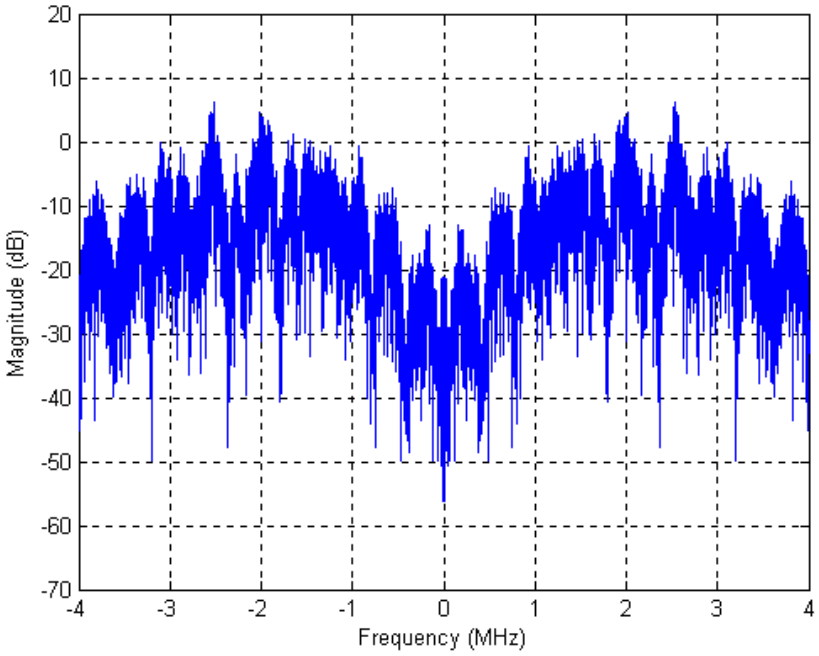

Figure 6.3 Power Spectrum (no fading) After Switch and Before Combining

The reason for the harsh looking spectrum of Figure 6.3 can be better understood by looking at the spectrum of the 181.8 kHz square wave switching signal shown in Figure 6.4. The peaks in Figure 6.4 correspond to the humps in the power spectrum of Figure 6.3. The switching operation has effectively spread the signal to higher and lower frequencies.

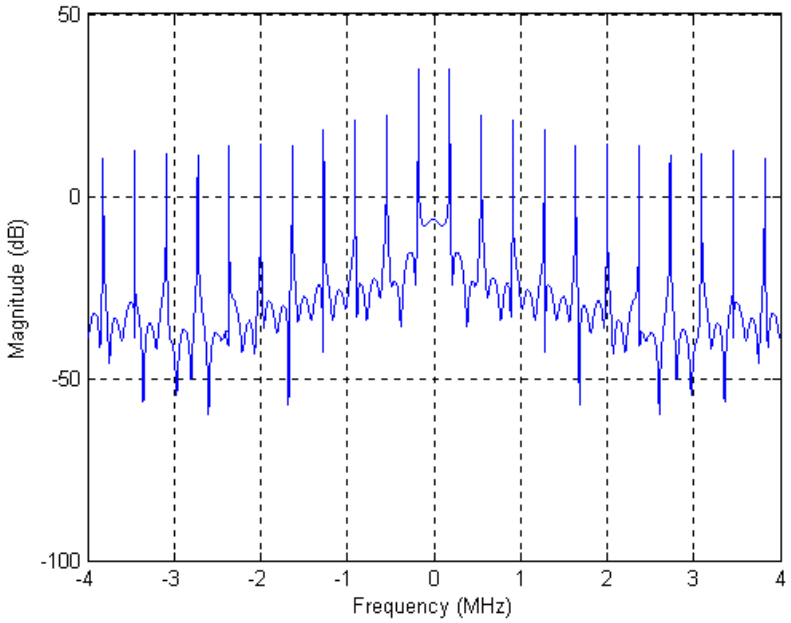

Figure 6.4 Power Spectrum of Switching Operation (constant input)

The switched diversity arm is combined with the non-switched arm to get the combined signal power spectrum shown in Figure 6.5. The resulting signal is filtered by the receiver IF filter. The power that was spread out of the bands centred at +/- 2 MHz in Figure 6.3 is now lost due to the IF filtering (see Figure 6.6).

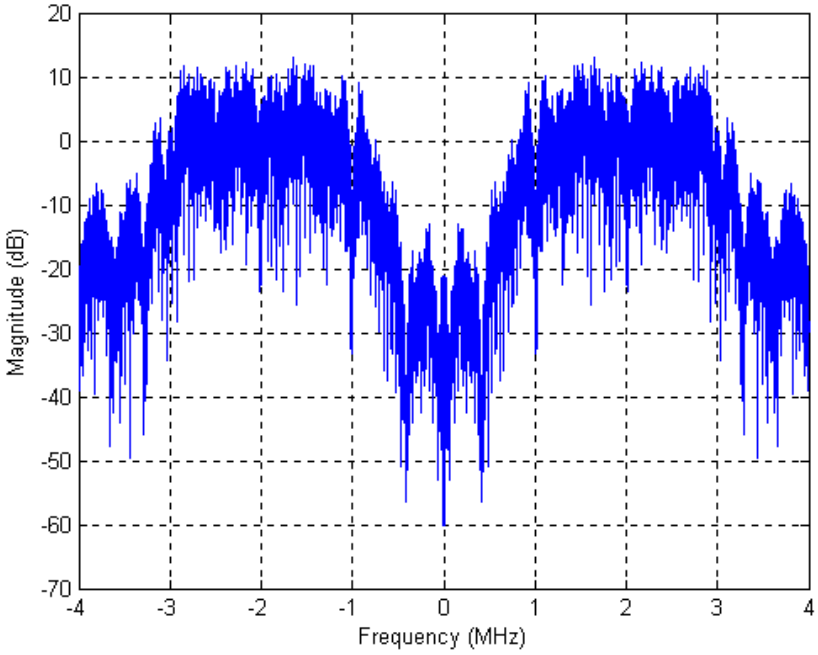

Figure 6.5 Power Spectrum of Signal After Diversity Combining

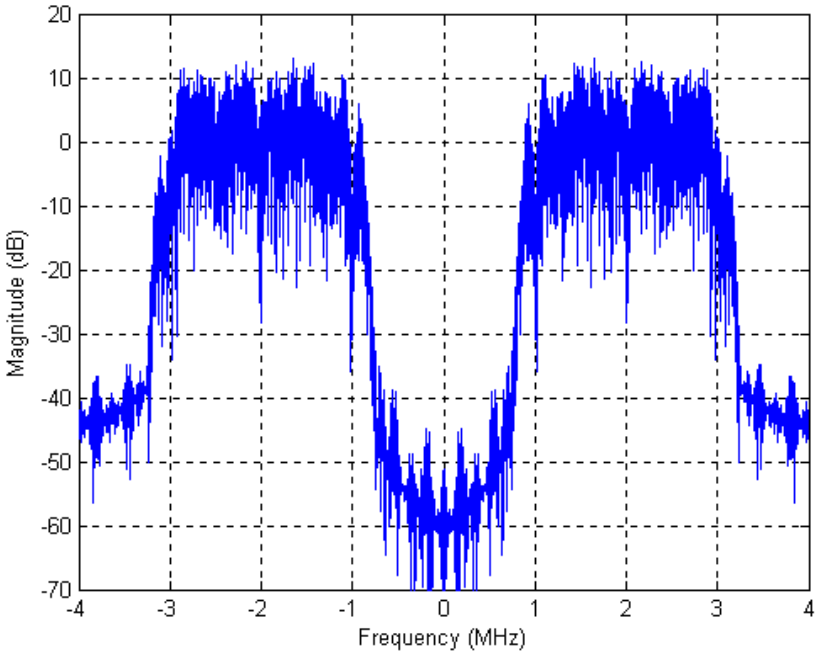

Figure 6.6 Power Spectrum of Switching Diversity Combining (after IF filter)

6.1.2 Sinusoidal Diversity Switching

The bandwidth effect of using a sinusoid to impart the phase perturbations is now studied. First consider the power spectrum of the sinusoidal phase perturbation signal by itself (Figure 6.7). Due to the finite windowing effects of the Fast Fourier Transform used to plot Figure 6.7, the lines in the power spectrum do not appear as ideal spectral lines. When compared to the square wave power spectrum shown in Figure 6.4, the sinusoidal case has only a single spectral line for each of the positive and negative frequencies. This results in relatively less spreading of the signal within the diversity switching arm (Figure 6.8). The end result is that less energy is lost after IF filtering.

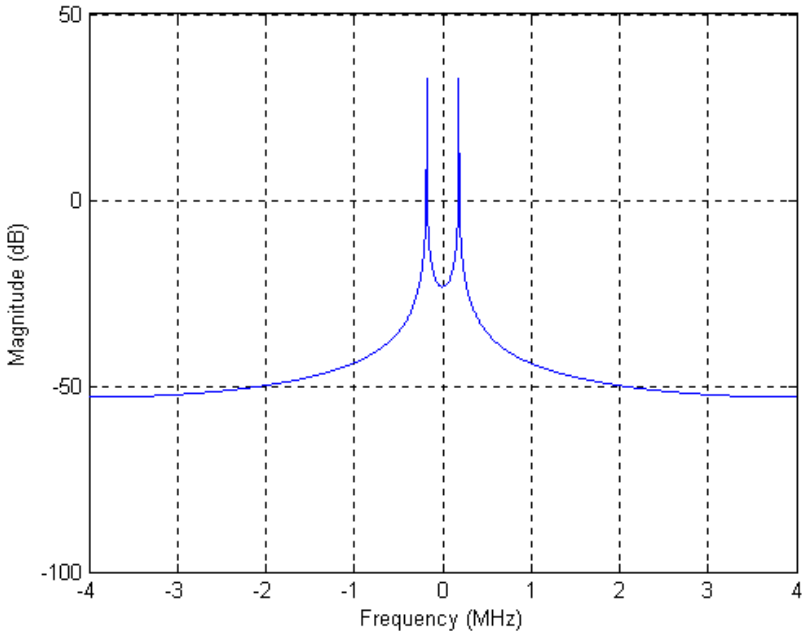

Figure 6.7 Power Spectrum of Sinusoidal Switching Operation (constant input)

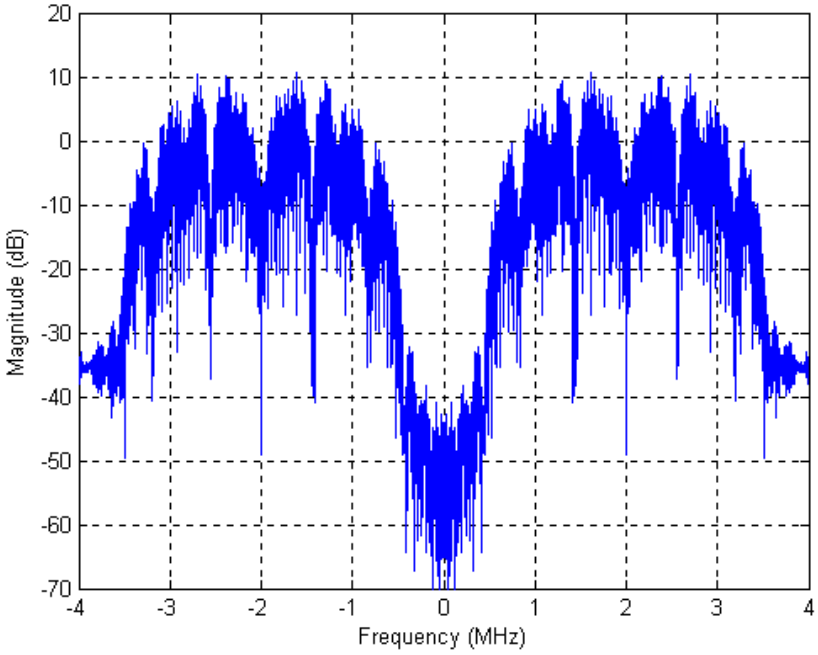

Figure 6.8 Power Spectrum After Sinusoidal Switching and Before Combining

6.2 Eye Diagrams

Eye diagrams provide a useful tool for determining the effects of a communication channel on the received signal. An eye diagram is created by displaying multiple sweeps of a received signal on a scope, with each sweep being triggered by a clock signal. The Simulink eye diagram tool was used to look at the eye diagram at the output of the Hogenauer filter, prior to the signal being sampled by the decision variable. Figure 6.9 shows the eye diagram for a baseline case of no signal fading and no diversity combining.

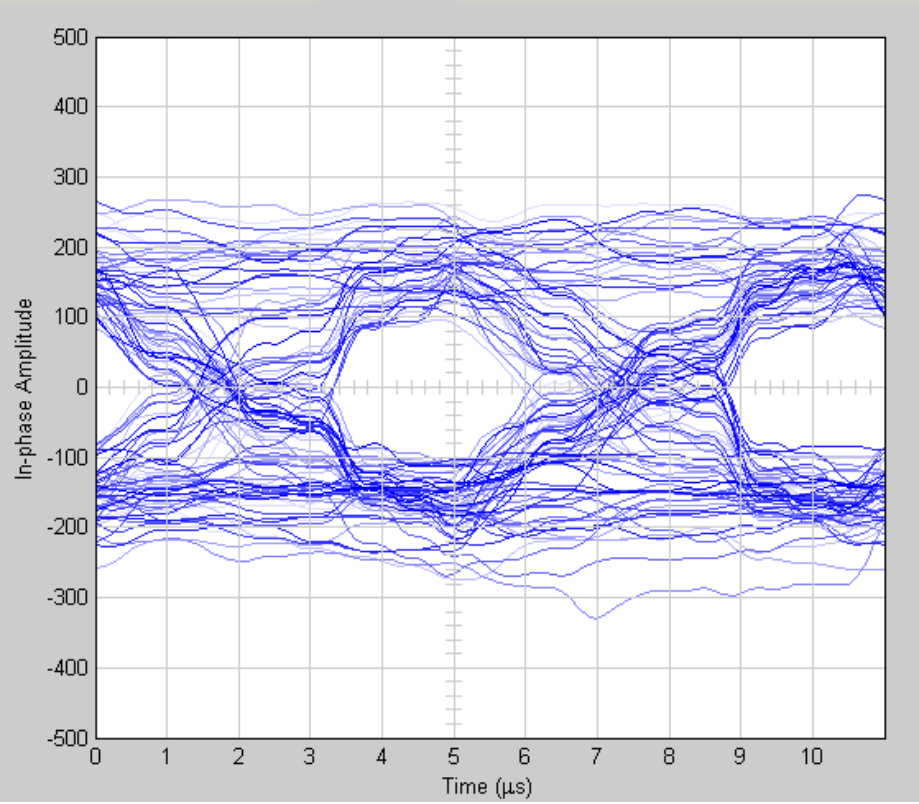

Figure 6.9 Eye Diagram of Output of Hogenauer Filter (no fading or switching)

To evaluate the effect that RF diversity combining has on the eye diagram, the simulation was set to generate an E_b/N_0 of 17 dB, with fading, but with switching on the second path turned off. A deep fade happens at 0.008 seconds into the simulation. The deep fade causes the eye diagram to close, as shown in Figure 6.10. The same simulation was then run with RF diversity combining enabled. The eye diagram for the diversity combining case with square wave switching, using the same random data 'seed' value, taken at the same point in time as the eye diagram in Figure 6.10, is shown in Figure 6.11. Notice how the 'eye' is opened due to the use of RF diversity combining, allowing a relatively decent sampling of the decision variable.

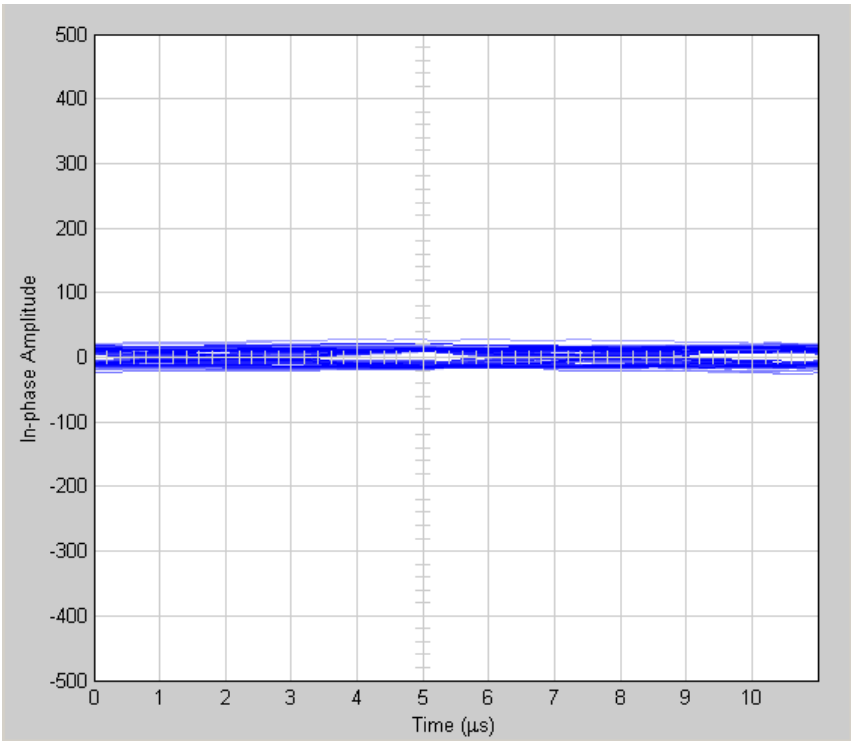

Figure 6.10 Eye Diagram with Fading (no switching)

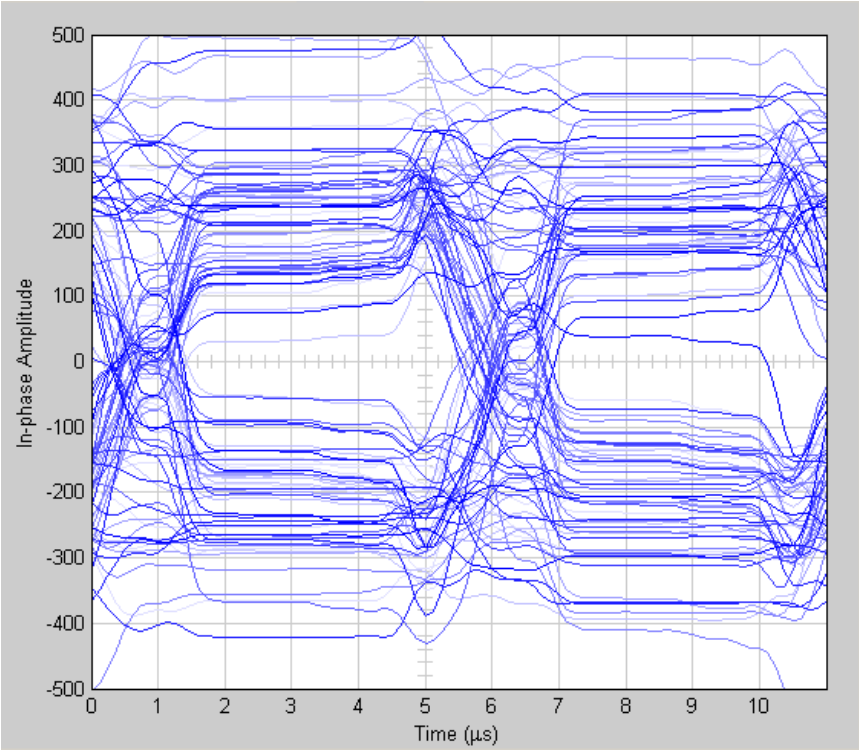

Figure 6.11 Eye Diagram with Fading (square wave switching)

The simulation was run again, this time with sinusoidal switching. The resulting eye diagram is shown in Figure 6.12. Comparing Figure 6.12 and Figure 6.11, one notices that the sinusoidal switching generally results in a more open 'eye', and therefore better performance would be expected from the sinusoidal switching.

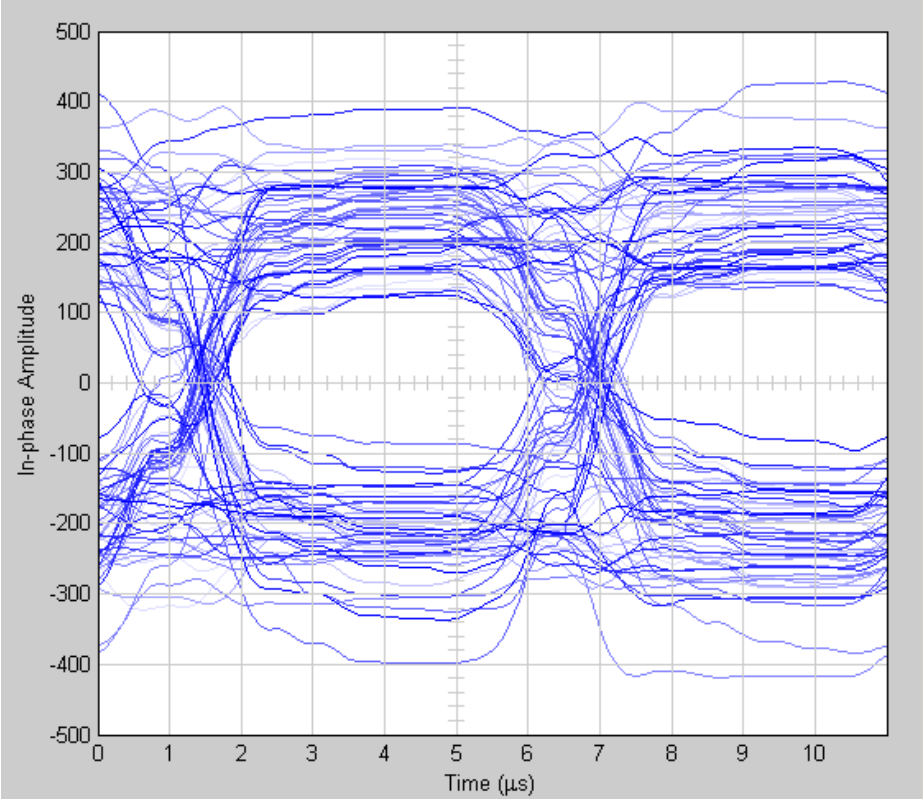

Figure 6.12 Eye Diagram with Fading (sinusoidal switching)

6.3 Simulink BER Testing

The simulation was used to perform BER testing. To verify that the simulation was working, a baseline simulation was run with no fading and with no RF diversity combining. The goal was to confirm the analysis of Section 4.7 and the resulting BER given in (4.28). The Simulink simulation was run 18 times to simulate E_b/N_o values from 1 dB to 18 dB in 1 dB increments. The BER measurements were saved for each

simulation run. Figure 6.13 compares the theoretical value derived in (4.28) to the simulated values. As can be seen in the figure, the results are in agreement with theory to within 2 to 3 dB.

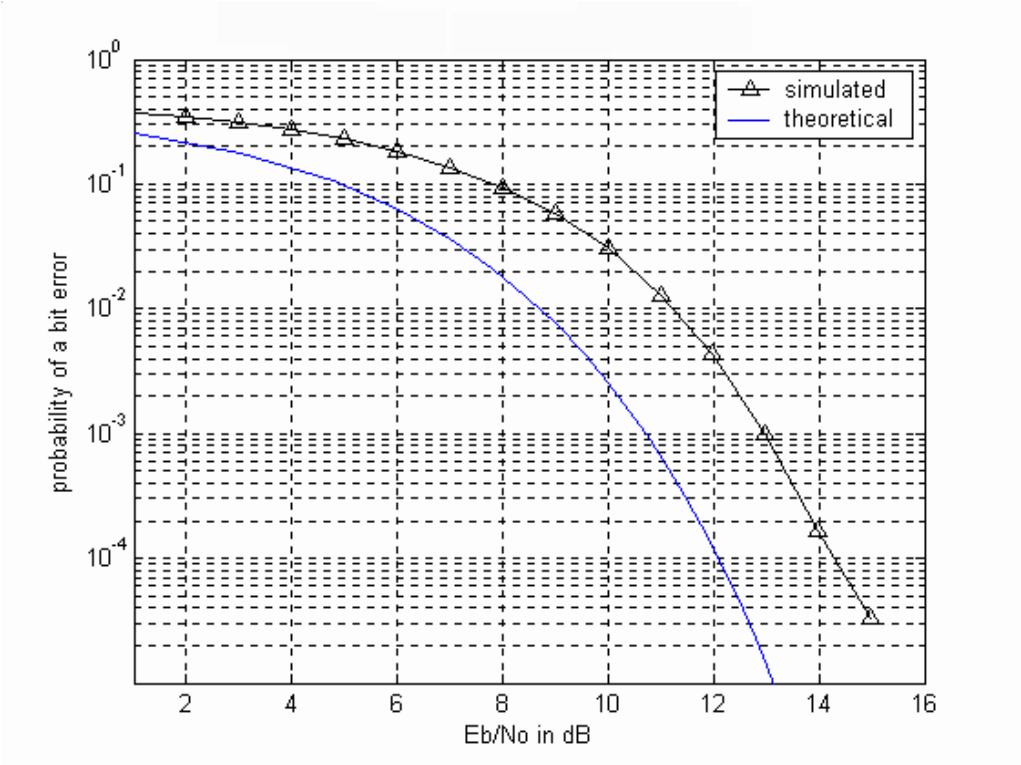

Figure 6.13 Theoretical vs. Simulated Probability of Error (no fading or diversity combining)

To evaluate the effectiveness of diversity combining against multipath fades, the simulation was run with fading and no diversity combining. The simulation was then run with diversity combining with square wave switching, and again with sinusoidal switching. The results of the BER testing are shown in Figure 6.14. For reference, the non-fading simulation case from Figure 6.13 is repeated in Figure 6.14 (triangle data points). Without diversity combining, the receiver experiences a high number of bit errors, even for large values of E_b/N_0 (square data points). With square wave switching

diversity combining, the deep fades are mitigated and the overall BER is much improved (* data points). The BER is slightly improved by approximately 0.5 dB for the case of sinusoidal switching diversity combining (circle data points). This improvement in performance is likely due to the sinusoidal switching causing less power loss through the IF filter.

Based on the eye diagrams shown in Figure 6.11 and Figure 6.12, it was expected that the relative performance improvement of the sinusoidal switching over square wave switching would be greater than 0.5 dB. One possible explanation why the performance improvement was not greater is that the square wave switching may have performed better than expected, due to the fact that the sampling of the Hogenauer filter to produce the decision variable was 'forced' to be the correct time within the simulation.

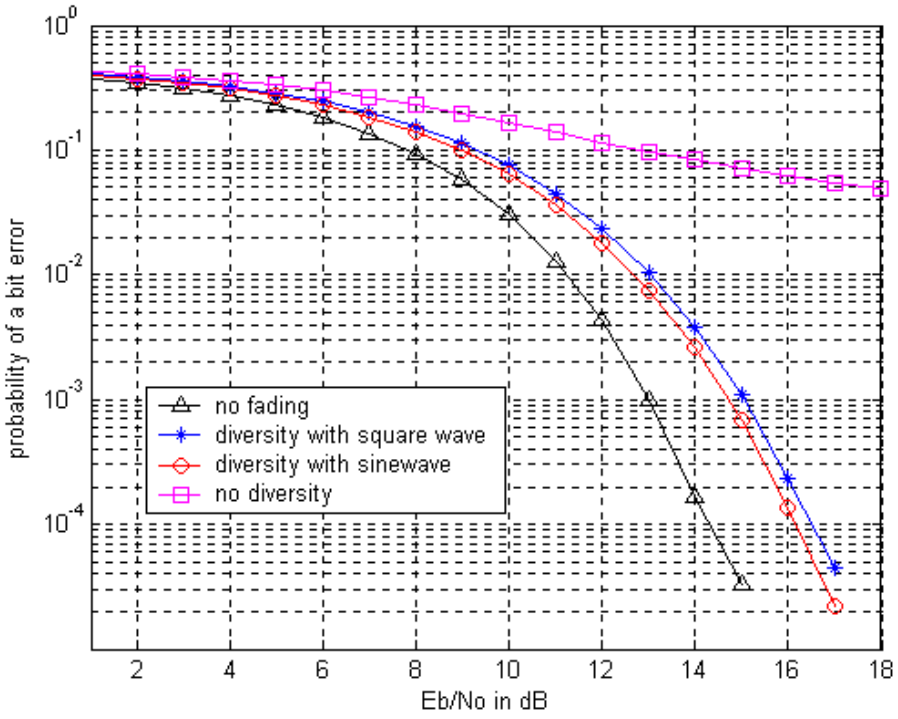

Figure 6.14 Simulated BER Test Results Using Diversity Combining in a Fading Channel

After all the simulation and hardware tests (described in the next chapter) were completed, the implementation was revisited. A final simulation was run, this time without the final down conversion to baseband. Essentially, the 'Sine wave', 'Downconvert' and 'Half-band Filter' blocks were removed from the receiver shown in Figure 5.8. The simulation was performed with no fading and no diversity combining. This new simulation is labelled as 'simulated – passband' in Figure 6.15 to indicate that the differential detection is done at the IF of 2 MHz. For reference, Figure 6.15 also contains the data that was plotted previously in Figure 6.13. The 'simulated-passband' data agrees with the theoretical performance to within 1 dB. Differential detection at the IF of 2 MHz should therefore form the basis of the design for the next version of the system.

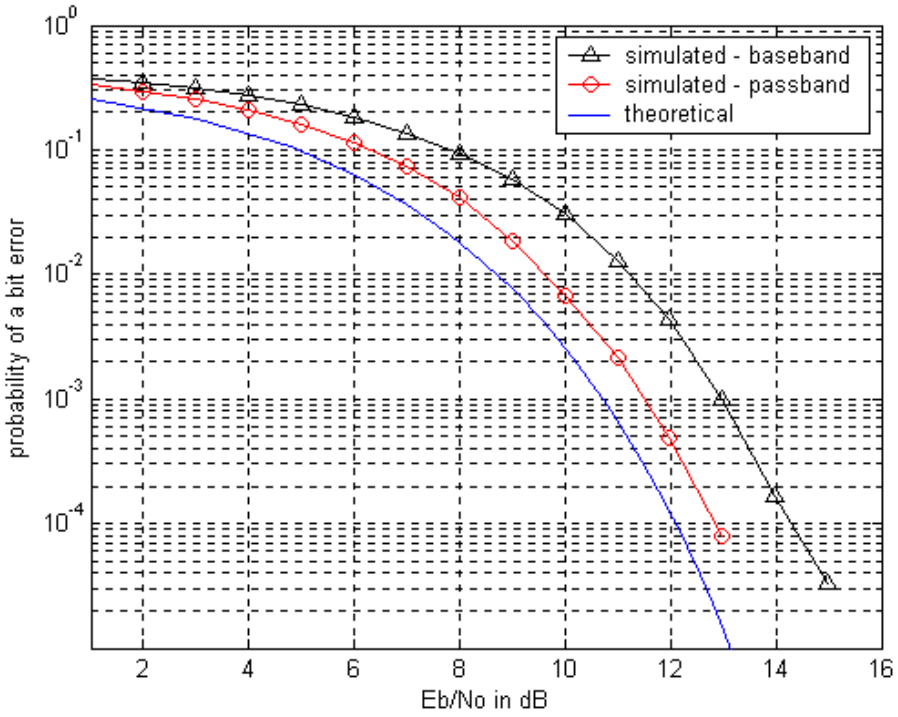

Figure 6.15 Theoretical vs. Simulated Baseband and Passband Probability of Error (no fading or diversity combining)

7 FPGA Testing

This section describes the end-to-end system test of the hardware implementation of the spatial diversity system. As mentioned previously, the up conversion was to 70 MHz. To simplify the use of the test equipment, sample clock and data rates were set to multiples of 1024. The clock rate was set to a nominal value of 8.192 MHz, the chip rate was set to 2.048 MHz, and the bit rate was set to 186.181 kHz.

For the end-to-end system testing a FireBERd bit error rate analyzer was used to supply the data to the transmitter. The FireBERd was also used to compare the transmitted data to the received data at the output of the receiver to determine the BER. Since an actual indoor communications channel was not used, noise needed to be added to the test setup. An HP3764A Digital Transmission Analyzer was used to supply the noise. The HP3764A outputs a PN sequence at 170 mega-bits per second. The output of the HP3764A appears as noise at 70 MHz. A block diagram of the test setup is shown in Figure 7.1.

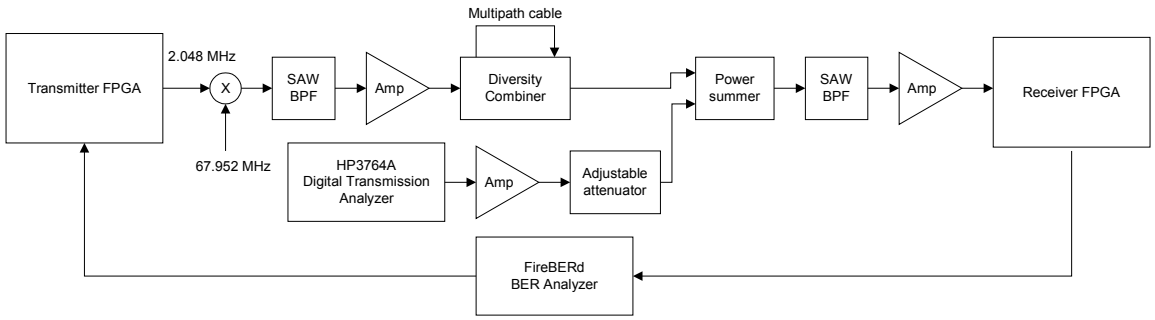

Figure 7.1 System Test Setup

Power measurements were taken prior to the beginning of the tests. An HP436A Power Meter was used to determine the power at the input to the A/D at the receiver. The HP3764A was set to deliver a constant power, and the adjustable attenuator was then used to adjust the noise power.

To determine the E_b/N_0, the energy per bit (E_b) was first determined. The signal power from the FireBERd was measured using the power meter. The bit rate was also used in determining the E_b. For example, with the signal level from the FireBERd measured to be –10 dBm, the value of E_b was:

$$E_b = -10dBm - 10\log(186181) = -10dBm - 52.7dB = -62.7dBm\,. \qquad (6.1)$$

The noise power was read using the power meter. The noise density was calculated, taking into consideration that the 3 dB bandwidth of the SAW filters was 3 MHz. For example, if the noise power measurement from the power meter was –15 dBm, then the noise power density was:

$$N_0 = -15dBm - 10\log(3MHz) = -15dBm - 64.8dB/Hz = -79.8dBm/Hz\,. \qquad (6.2)$$

Three different cables were used to model three different multipath delays. Each cable had a different electrical delay, and therefore caused a different amount of multipath when inserted into the test circuit of Figure 7.1. For convenience these cables were labelled Cable A, Cable B, and Cable C. Testing with a fourth cable, Cable D, was attempted. Partway into the testing Cable D was found to be defective and the results from those tests are omitted.

The test setup allowed the diversity combining circuit to be in one of three states:

State 1. Switching turned on, with diversity path switching between 0 and π radians, with a 50% duty cycle.

State 2. Diversity path set to 0 radians phase, with a 100% duty cycle.

State 3. Diversity path set to π radians phase, with a 100% duty cycle.

Table 7.1 shows how the diversity combining circuit was effective in mitigating deep fades. For Cable A and Cable B, deep fades were experienced when the diversity combining path had a delay of π radians. For Cable C, a slight fade was experienced when the diversity combining path had a delay of π radians. In all cases, the diversity switching mitigated the fade. Note that for all cases there was a "better" signal strength from either state 2 or state 3. Without complex signal processing to determine the actual phase delay combination, the receiver does not "know" the optimum phase delay for combining. The RF diversity switching acts as an effective and simple means to mitigate fading.

Table 7.1 Measured Signal Strength for Various Diversity Combining States

	State 1 (switching)	**State 2 (0 radians)**	**State 3 (π radians)**
Cable A	-16.2 dBm	-12.8 dBm	-28.5 dBm
Cable B	-16.1 dBm	-13.0 dBm	-23.8 dBm
Cable C	-15.8 dBm	-14.2 dBm	-17.6 dBm

During another set of tests, the BER was measured for various values of E_b/N_0. The results are shown in Figure 7.2, Figure 7.3, and Figure 7.4 for Cables A, B, and C respectively. The theoretical value of BER plotted in the figures was calculated using equation (4.28) from Section 4.7. In each of the three figures, the theoretical and actual values agree to within 2 to 3 dB. Considering that the typical path loss in the indoor

environment would be on the order of several tens of dBs, a 2 to 3 dB loss is deemed acceptable for illustrating the proof-of-concept.

The differential detection in the FPGA receiver was done at baseband, and not the IF centred at 2 MHz. Referring to the simulation results shown in Figure 6.15, a performance improvement of 1 to 2 dB would be expected by implementing the differential detection at the IF centre frequency of 2 MHz. Other differences between the simulated and FPGA implementation can be attributed to differences in the multipath fading modeling and implementation losses.

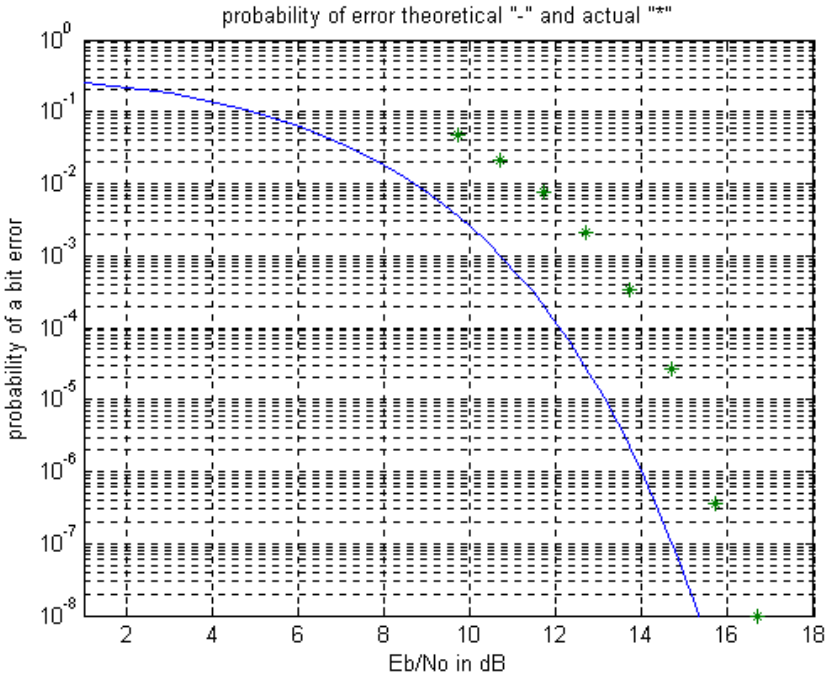

Figure 7.2 Probability of a Bit Error for Cable A

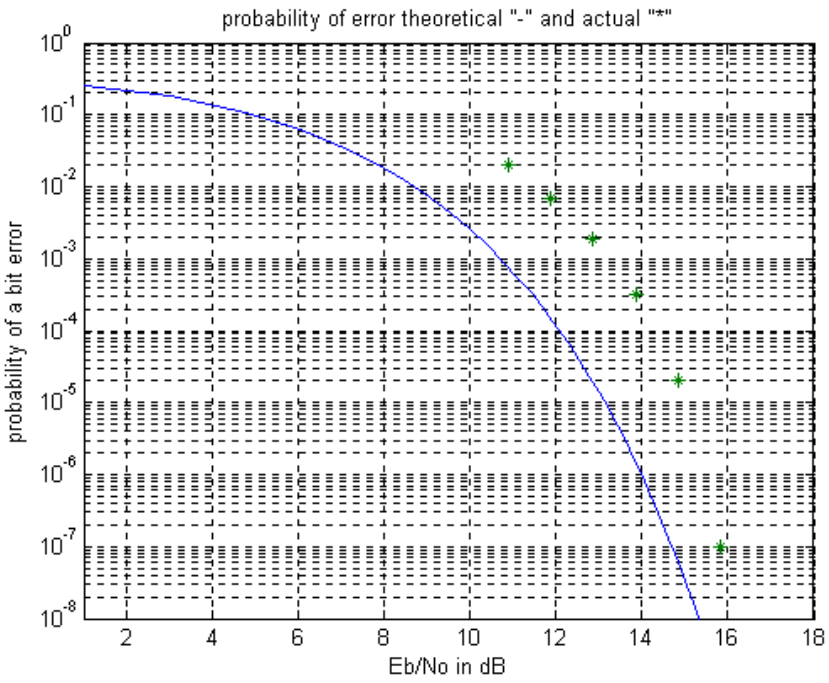

Figure 7.3 Probability of a Bit Error for Cable B

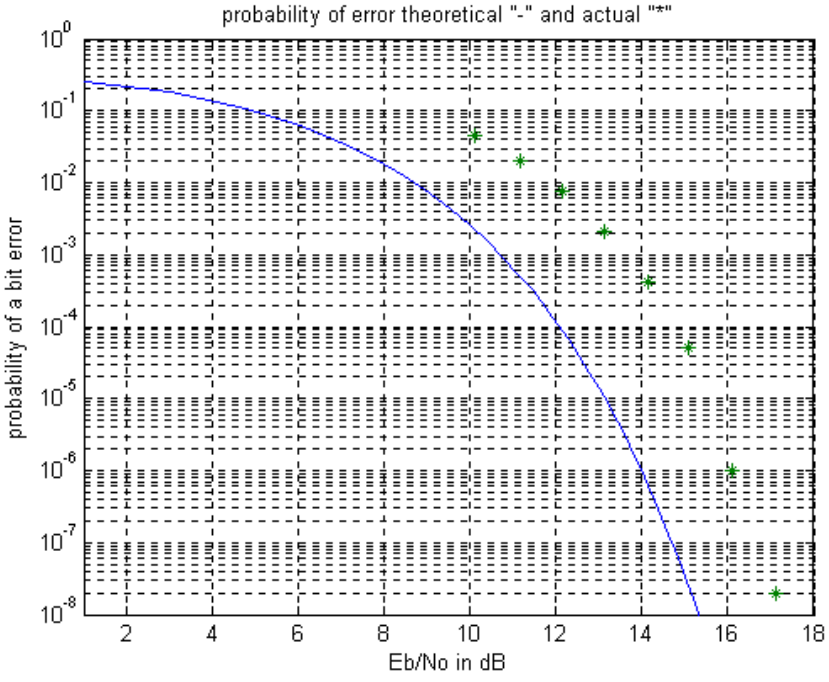

Figure 7.4 Probability of a Bit Error for Cable C

8 Summary and Conclusions

8.1 Summary and Conclusions

This thesis examined the performance of a novel spatial diversity scheme suitable for indoor wireless communications. The relative system performance of the RF diversity combining scheme was evaluated using a Simulink simulation. A simple hardware breadboard proof of concept prototype was also developed.

The intended application was the distribution of compressed digital music in a shopping mall environment. To conform to Industry Canada rules for the ISM band, a DSSS spreading factor of 11 was used. To make the system economical, a simple DPSK transmitter and receiver was used. No description of the BER performance of a DSSS DPSK receiver could be found in the open literature. The theoretical BER was therefore derived. Simulink simulations showed that the performance of the baseline receiver, without diversity combining, agreed to within 2 to 3 dB for differential detection at baseband, and to within 1 dB for differential detection at the IF of 2 MHz.

Simulink simulations were conducted to determine the relative performance of two implementations of the RF diversity combining; diversity switching using a square wave switching signals, and diversity combining using a sinusoidal switching signal. The BER performance for both cases showed that the novel diversity method was very effective in mitigating deep multipath fades. The sinusoidal switching implementation performed better than the square wave implementation by approximately 0.5 dB. The performance difference is attributed to the spectral spreading effect of the diversity switch and the subsequent IF filtering.

The square wave switching version of the RF diversity combining system was implemented as a proof-of-concept breadboard prototype. The phase shift by 180 degrees was implemented using a simple off-the-shelf BPSK modulator. The BPSK modulator was driven by a 50% duty cycle square wave, with a clocking rate equal to the bit rate. The system was implemented in Altera FPGAs, using the Verilog hardware description language. The Flex10k Altera FPGAs used for the transmitter and receiver each had an equivalent gate count of 100,000 gates. The transmitter used only 4% of the device resources, or 4000 equivalent gates. The receiver used 33% of the device resources, or 33,000 equivalent gates. These low gate counts, coupled with the low cost BPSK modulator used for the diversity switching, illustrated that the system could be implemented in an economical manner. End to end system testing showed that the hardware implementation of the spatial diversity scheme was effective against deep fades.

8.2 Future Work

This thesis was an initial proof of concept system. The opportunity exists to improve upon the system. Some of the possible areas for future work are:

- Expand the simulation testing to include the study of other types of modulation (i.e. only switch the signal by 90 degrees instead of 180 degrees).
- Expand the spatial diversity to use more than 2 receiving antennas.
- The breadboard prototype was a 'wire-wrapped' system. The version of the system should use printed circuit boards.

- Implement the system hardware using differential detection at the IF frequency of 2 MHz.
- The approximate equivalent gate counts of the transmitter and receiver have now been established. The next version of the design should be moved to smaller and cheaper next generation FPGA.
- Perform further hardware debugging to fully implement timing recovery of the Hogenauer filter output.
- Extend the system to transmission and testing at the final RF frequency band of 900 MHz. This will allow for qualitative testing in a representative indoor environment.

REFERENCES

[1] S. Haykin, *Communications Systems*, New York : John Wiley and Sons Inc., 3rd Edition, 1994, pp. 737-747.

[2] J.D. Parsons, *The Mobile Radio Propagation Channel*', New York : John Wiley and Sons Inc., 1992.

[3] J.P.M.G. Linnartz, *Wireless Communication, The Interactive Multi-Media CD-ROM*', Red Bank: Baltzer Science Publishers, 1999.

[4] J.G. Proakis, *Digital Communications*, New York: McGraw-Hill Inc., 3rd Edition, 1995, pp.\ 758—833.

[5] J.H. Winters, J. Salz, R.D. Gitlin, "The Impact of Antenna Diversity on the Capacity of Wireless Communication Systems", *IEEE Transactions on Communications*, Vol. 42, No. 2, April 1994, pp.1740-1751.

[6] T. Eng, N. Kong, "Comparison of Diversity Combining Techniques for Rayleigh-Fading Channels", *IEEE Transactions on Communications*, Vol. 44, No. 9, September 1996, pp.1117-1129.

[7] J.E. Salt, "Receiver Combiner For Spatial Diversity Digital Communications", United States Patent #6,389,085, May 14, 2002.

[8] E.B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE Transactions on Acoustics, Speech, and Signal Processing,* April 1991.

[9] N.J. Fliege, *Multirate Digital Signal Processing,* New York: John Wiley & Sons, Inc., 1994.

[10] L.W. Couch II, *Digital and Analog Communication Systems,* New York: Macmillan Publishing Company, 4th Edition, 1993.

[11] Jason Dielschneider, "16 QAM Radio Link on the 5.7 GHz ISM Band", *M.Sc. Thesis*, Department of Electrical Engineering, University of Saskatchewan, September 2001.

[12] W. C. Jakes, *Microwave Mobile Communications*, New York: IEEE Press, 1974

A. APPENDIX – VERILOG FILES

A.1. Transmitter File

```
/*****************************************************************
* File:  mod.v
* Author: Neil D. Gerein
* Description:  Modulator for the RF diversity combining modem.
*                     Keep track of the last 6 symbols, 4 samples per
*                     symbol.  The pulse shaping filter is a square
*                     root raised cosine filter with an even number
*                     of samples.  The rolloff factor is 0.35.
*
*
* Created:      March 15/1999
* Revision: August 12/1999 -- Data can now be from external input or
*                                 a known data sequence.
*****************************************************************/

`define zero          2'b00
`define one           2'b01
`define two                 2'b10
`define three         2'b11

  module
mod(clk,data_input,data_sel,da_clk,clk_by_4,symbol_clock,data_bit,filt_
out,tristateA,tristateB);
  // data_input is connected to the BER detector
      input clk,data_input;
  // data_sel selected internal or external data input
      input data_sel;
  // inputs to be tristated for the second DB25 connector
      input [11:0] tristateA,tristateB;
      output clk_by_4,symbol_clock,data_bit,da_clk;
      output [9:0] filt_out;
      reg clk_by_4,bit_delay,bit_in,data_bit;
      reg [1:0] state;              // where we are in the symbol
  // keep track of the last 6 symbols
      reg [5:0] symbols,temp_symbols;
      reg [9:0] filt_out,temp_out,temp_mod;
      reg [9:0] temp5,temp4,temp3,temp2,temp1,temp0;
      reg [10:0] temp01,temp23,temp45;    // used to prevent overflow


      assign da_clk = clk;

      // keep track of where we are in the symbol
      upsamp4 StateCount (clk,clk_by_4,state);

      // spread the data
      // bit_in is the output of the data_spread module that is used as
input to the filter
      data_spread                                                DataSpread
(clk,clk_by_4,data_sel,data_input,bit_in,symbol_clock);
```

```
// shift in the newest input
always @ (posedge clk)
  begin
      if (state==`three)
   // calculate this on state three so its ready for state zero
        begin
            symbols     <= symbols >> 1;
            symbols[5]  <= bit_in;
            data_bit    <= bit_in;  // for testing
        end
  end

// calculate the filter output
always @ (negedge clk)
  begin
      if (symbols[5]==0)
        begin
            case (state)
            `zero:      temp5 <=  10'b00_0000_1000;   // 8
            `one:       temp5 <=  10'b11_1111_1111;   // -1
            `two:       temp5 <=  10'b11_1111_0000;   // -16
            `three:     temp5 <=  10'b11_1110_1000;   // -24
            default:    temp5 <= 10'b00_0000_0000;    // 0
            endcase
        end
      else
        begin
            case (state)
            `zero:      temp5 <=  10'b11_1111_1000;   // -8
            `one:       temp5 <=  10'b00_0000_0001;   // 1
            `two:       temp5 <=  10'b00_0001_0000;   // 16
            `three:     temp5 <=  10'b00_0001_1000;   // 24
            default:    temp5 <= 10'b00_0000_0000;    // 0
            endcase
        end

      if (symbols[4]==0)
        begin
            case (state)
            `zero:      temp4 <=  10'b11_1111_0111;   // -9
            `one:       temp4 <=  10'b00_0001_1011;   // 27
            `two:       temp4 <=  10'b00_0011_1100;   // 60
            `three:     temp4 <=  10'b00_0011_0110;   // 54
            default:    temp4 <= 10'b00_0000_0000;    // 0
            endcase
        end
      else
        begin
            case (state)
            `zero:      temp4 <=  10'b00_0000_1001;   // 9
            `one:       temp4 <=  10'b11_1110_0101;   // -27
            `two:       temp4 <=  10'b11_1100_0100;   // -60
            `three:     temp4 <=  10'b11_1100_1010;   // -54
            default:    temp4 <= 10'b00_0000_0000;    // 0
```

```
      endcase
  end

if (symbols[3]==0)
  begin
      case (state)
      `zero:      temp3 <=  10'b11_1111_0010;   // -14
      `one:       temp3 <=  10'b11_0111_1000;   // -136
      `two:       temp3 <=  10'b10_1111_0001;   // -271
      `three:     temp3 <=  10'b10_1001_1001;   // -359
      default:    temp3 <= 10'b00_0000_0000;    // 0
      endcase
  end
else
  begin
      case (state)
      `zero:      temp3 <=  10'b00_0000_1110;   // 14
      `one:       temp3 <=  10'b00_1000_1000;   // 136
      `two:       temp3 <=  10'b01_0000_1111;   // 271
      `three:     temp3 <=  10'b01_0110_0111;   // 359
      default:    temp3 <= 10'b00_0000_0000;    // 0
      endcase
  end

if (symbols[2]==0)
  begin
      case (state)
      `zero:      temp2 <=  10'b10_1001_1001;   // -359
      `one:       temp2 <=  10'b10_1111_0001;   // -271
      `two:       temp2 <=  10'b11_0111_1000;   // -136
      `three:     temp2 <=  10'b11_1111_0010;   // -14
      default:    temp2 <= 10'b00_0000_0000;    // 0
      endcase
  end
else
  begin
      case (state)
      `zero:      temp2 <=  10'b01_0110_0111;   // 359
      `one:       temp2 <=  10'b01_0000_1111;   // 271
      `two:       temp2 <=  10'b00_1000_1000;   // 136
      `three:     temp2 <=  10'b00_0000_1110;   // 14
      default:    temp2 <= 10'b00_0000_0000;    // 0
      endcase
  end


if (symbols[1]==0)
  begin
      case (state)
      `zero:      temp1 <=  10'b00_0011_0110;   // 54
      `one:       temp1 <=  10'b00_0011_1100;   // 60
      `two:       temp1 <=  10'b00_0001_1011;   // 27
      `three:     temp1 <=  10'b11_1111_0111;   // -9
      default:    temp1 <= 10'b00_0000_0000;    // 0
      endcase
  end
else
```

```
            begin
                case (state)
                `zero:      temp1 <=  10'b11_1100_1010;   // -54
                `one:       temp1 <=  10'b11_1100_0100;   // -60
                `two:       temp1 <=  10'b11_1110_0101;   // -27
                `three:     temp1 <=  10'b00_0000_1001;   // 9
                default:    temp1 <= 10'b00_0000_0000;    // 0
                endcase
            end

          if (symbols[0]==0)
            begin
                case (state)
                `zero:      temp0 <=  10'b11_1110_1000;   // -24
                `one:       temp0 <=  10'b11_1111_0000;   // -16
                `two:       temp0 <=  10'b11_1111_1111;   // -1
                `three:     temp0 <=  10'b00_0000_1000;   // 8
                default:    temp0 <= 10'b00_0000_0000;    // 0
                endcase
            end
          else
            begin
                case (state)
                `zero:      temp0 <=  10'b00_0001_1000;   // 24
                `one:       temp0 <=  10'b00_0001_0000;   // 16
                `two:       temp0 <=  10'b00_0000_0001;   // 1
                `three:     temp0 <=  10'b11_1111_1000;   // -8
                default:    temp0 <= 10'b00_0000_0000;    // 0
                endcase
            end

          // calculate the signed output
          temp01 = {temp0[9],temp0} + {temp1[9],temp1};
          temp23 = {temp2[9],temp2} + {temp3[9],temp3};
          temp45 = {temp4[9],temp4} + {temp5[9],temp5};
          temp_mod = temp01 + temp23 + temp45;


      // modulate by cos(2*pi*n/4+pi/4) to shift the spectrum from
baseband to 2 MHz
          if ((state==`zero)||(state==`three))
                temp_out = temp_mod;
          else
                temp_out = ~temp_mod + 1;


          // convert to unsigned value to send to the D/A
          filt_out[9] <= ~temp_out[9];
          filt_out[8:0] <= temp_out[8:0];


      end

  endmodule
```

A.2. Internal Sample Clock File

```
// Puts out the bit_clk on every fourth input clock

module upsamp4(clk, bit_clk, state);

          input clk;
          output bit_clk;
          output [1:0] state;

          reg [1:0] state, next_state;
          reg   bit_clk;

          `define zero            2'b00
          `define one             2'b01
          `define two             2'b10
          `define three           2'b11


          // find the next state
          always @ (posedge clk)
            begin
                case(state)
                      `zero:      state<=`one;
                      `one:       state<=`two;
                      `two:       state<=`three;
                      `three:     state<=`zero;
                      default:    state<=`three;
                endcase
            end

          // process the output logic
          always @ (state)
            begin
                case(state)
                      `zero:      bit_clk <= 1;
                      `one:       bit_clk <= 1;
                      `two:       bit_clk <= 0;
                      `three:     bit_clk <= 0;
                      default:    bit_clk <= 0;
                endcase
            end

          endmodule
```

A.3. Data Spreading File

```
/*****************************************************************
* File:  data_spread.v
* Author: Neil D. Gerein
* Description:  If data_sel is high then the input data_in is the
*         data sequence to be spread, else use a known data
*         sequence generated internally.
*
*
* Created:      August 12/1999
* Revision:
*****************************************************************/

  module
data_spread(clk,clk_by_4,data_sel,data_in,data_out,symbol_clock);
      input clk,clk_by_4,data_sel,data_in;
      output data_out,symbol_clock;
      reg data_out;
      reg [7:0] sequence;     // the known data sequence to be spread
      reg [10:0] barker; // length 11 Barker seq used for spreading

      reg temp,last_temp,temp_encode,symbol_clock;
      // need flag since reset_sample_count is high for 4 clk periods
      reg reset_sample_count,keep_counting_flag;
      reg [3:0] count;
      reg [5:0] sample_count; // needed to count up to 44

      always @ (posedge clk)
        begin
            if ((reset_sample_count == 1) && (keep_counting_flag == 0))
              begin
                  // will rollover to zero at end of always block
                  sample_count = 6'b111111;
                  symbol_clock = 1;
                  keep_counting_flag = 1;
              end
            else if (sample_count == 21)
              begin
                  symbol_clock = 0;
                  keep_counting_flag = 0;
              end
            sample_count = sample_count + 1;
        end


      always @ (posedge clk_by_4)
        begin
            // set the initial values
            if (sequence==0)
              begin
                  sequence = 8'b1101_0110;
                  // init to d'10 so the system can start up quickly
                  count = 4'b1010;
              end
            else
```

```
            begin
                if (count==4'b1011)
                  begin
                      if (data_sel == 0)
                      // use internal known data sequence
                        begin
                            temp = sequence[7];
                            sequence = sequence << 1;
                            sequence[0] = temp;
                        end
                      else
                        begin
                            // use external input data
                            temp = data_in;
                        end
                      // differentially encode and spread
                      temp_encode = temp ^ last_temp;
                      if (temp_encode==0)
                        // zero so output the inverted Barker
                      sequence
                            barker = 11'b000_1110_1101;
                      else
                      // one so output the Barker sequence
                            barker = 11'b111_0001_0010;
                      last_temp = temp_encode;
                      count = 0;
                      reset_sample_count = 1;
                  end
                else
                      reset_sample_count = 0;
              end
            data_out = barker[10];
            barker = barker << 1;
            count = count + 1;
      end


endmodule
```

A.4. Receiver File

```
/*****************************************************************
* File:  demod.v
* Author: Neil D. Gerein
* Description: Demodulator for the RF diversity combining modem.
*
* * Created:    March 15/1999
* Revision: July 16/1999 -- Added symbol timing code.
*   July 19/1999 -- Redefined in/outs.
*   August 17/1999 -- Added half-band filter and timing recovery.
*   August 24/1999 -- Added code to reset timing.
*****************************************************************/

  module
demod(clk,reset_timing,in_bits,out_bit,symbol_clock,symbol_clock_A,timi
ng_control,timing_control_clk);
      input clk,reset_timing;
      input [9:0] in_bits;
      output [9:0] timing_control;
      output out_bit,timing_control_clk;
      // two pins for testing purposes
      output symbol_clock,symbol_clock_A;
      wire ad_clk;
      reg out_bit,symbol_clock;
      reg zero_flag,locked;
      reg symbol_time_reset;
      reg [1:0] state;  // keeps track of where we are in the symbol
      reg [5:0] count,symbol_time_count;
      reg [9:0] in_bits_temp;
      wire [9:0] in_bits_twos,filter_out;
      wire [19:0] dm_out;
      wire [19:0] hogen_out;
      wire hogen_MSB;
      // hogen_sign keeps track of whether the hogenauer filter output
      // was positive or negative for the last sample
      reg last_hogen_sign;
      assign ad_clk = clk;
      assign hogen_MSB = hogen_out[19];
      assign symbol_clock_A = symbol_clock;

  /*---------------------------------------------------------------*/

      // change the input from unsigned to signed

      assign in_bits_twos[9] = ~in_bits[9];
      assign in_bits_twos[8:0] = in_bits[8:0];

      // multiply input by cos(2*pi*n/4+pi/4)
      // to shift the spectrum from 2 MHz to baseband

      always @ (posedge clk)
        begin
            state = state + 1;
            if ((state==2'b00)||(state==2'b11))
```

```
                in_bits_temp = in_bits_twos;
          else
                in_bits_temp = ~in_bits_twos + 1;
      end
    // filter off the upper sideband using the half-band filter

    halfband HB1 (clk,in_bits_temp,filter_out);

/*-----------------------------------------------------------------*/

    // do the timing recovery
    // timing_control word is in unsigned format
    // coming out of the timing module

    timing Time (clk,filter_out,timing_control,timing_control_clk);

/*-----------------------------------------------------------------*/
    // feed a/d input into delay_mult block
    // dm_out is already in twos complement

    delay_mult DMult1 (clk,count,filter_out,dm_out);

    hogenauer H1 (clk,dm_out,hogen_out);

/*-----------------------------------------------------------------*/
    // do some simple symbol timing recovery
    // Keep track of the 44 samples in the symbol
    // using the following resetable counter.
       // Once the symbol boundaries are found then run the system
       // blindly until user resets.

    always @ (posedge clk)
      begin
          if ((symbol_time_reset == 1) && (locked == 0))
                symbol_time_count <= 0;
          else if (symbol_time_count == 43)
            begin
                symbol_time_count <= 0;
                if (reset_timing == 0)
                      // now locked so don't check for zero crossings
                      locked <= 1;
                else
                      // user wants to reset timing
                      locked <= 0;
            end
          else
                symbol_time_count <= symbol_time_count + 1;
      end

    // reset the counter when the hogenauer filter crosses zero

    always @ (negedge clk)
      begin
          if (hogen_out[19] != last_hogen_sign)  // zero crossing
                symbol_time_reset <= 1;
          else
                symbol_time_reset <= 0;
```

```
            last_hogen_sign <= hogen_out[19];
        end

    // If the zero crossings are known then sample the hogenauer
    // filter in the 22 samples to ensure that the output is
    // taken near a max or min.
    always @ (posedge clk)
            if (symbol_time_count == 21)
              begin
                  out_bit = hogen_out[19];
              end

/*-----------------------------------------------------------------*/

    // output the symbol clock

    always @ (posedge clk)
      begin
            if (symbol_time_count == 0)
                  symbol_clock = 1;
            else if (symbol_time_count == 22)
                  symbol_clock = 0;
            else
                  symbol_clock = symbol_clock;
      end


endmodule
```

A.5. Half-band Filter File

```
/*****************************************************************
* File:  halfband.v
* Author: Neil D. Gerein
* Description: A Lagrange half-band filter to eliminate
* the image at pi.
* The coefficients are h=[6 0 -5 0 300 512 300 0 -5 0 6].
*
*
* Created:      August 17/1999
* Revision:
*****************************************************************/

module halfband(clk,x0,y_out);
    input clk;
    input [9:0] x0;
    output [9:0] y_out;
    reg [9:0] x1,x2,x3,x4,x5,x6,x7,x8,x9,x10;
    reg [9:0] y_out;
    reg [10:0] sum1,sum2,sum3;
    reg [14:0] sum4;
    reg [19:0] sum5,sum6;
    reg [13:0] x_h0,x_h0_temp,x_h2,x_h2_temp,x_h4_temp1;
    reg [17:0] x_h4_temp2;
    reg [19:0] x_h4;
    reg [18:0] x_h5;


    always @ (posedge clk)
      begin
          // calculate partial sums
          sum1 = {x0[9],x0} + {x10[9],x10};
          sum2 = {x2[9],x2} + {x8[9],x8};
          sum3 = {x4[9],x4} + {x6[9],x6};
          // multiply by coefficients
          // using canonical signed digit with minimum number of adds
          x_h0        = (sum1<<2) + (sum1<<1);      // 6 = 2^2 + 2^1
          x_h2_temp   =(sum2<<2) + sum2;            // 5 = (2^2 + 1)
          x_h2        = ~x_h2_temp + 1;             // negate for -5
          x_h4_temp1  = (sum3<<2) + sum3;           // 5 = (2^2 + 1)
          if (x_h4_temp1[13] == 1)
          // 15 = 2^4 - 1
                x_h4_temp2 = (x_h4_temp1<<4) - {4'b1111,x_h4_temp1};
                else
          // 15 = 2^4 - 1
                x_h4_temp2 = (x_h4_temp1<<4) - {4'b0000,x_h4_temp1};
                x_h4        = (x_h4_temp2<<2);         //300 =
2^2*5*15
          x_h5        = (x5<<9);
          // create the final sums
          sum4 = {x_h0[13],x_h0} + {x_h2[13],x_h2};
          sum5 = x_h4 + {x_h5[18],x_h5};
          if (sum4[14] == 1)
                sum6 = {5'b11111,sum4} + sum5;
```

```
            else
                  sum6 = {5'b00000,sum4} + sum5;
            y_out = sum6[19:10];
            // advance everything through the buffer
            x10 = x9;
            x9 = x8;
            x8 = x7;
            x7 = x6;
            x6 = x5;
            x5 = x4;
            x4 = x3;
            x3 = x2;
            x2 = x1;
            x1 = x0;
      end

endmodule
```

A.6. Hogenauer Filter File

```
// Hogenauer filter
//  y(n) = x(n) + x(n-1) + x(n-2) + x(n-3) ... + x(n-43)
//   which for a Hogenauer filter is equivalent to
//  y(n) = u(n) - u(n-44)


module hogenauer(clk,x_n,y_n);
    input clk;
    input [19:0] x_n;
    output [19:0] y_n;
    reg [19:0] y_n;
    reg [19:0] x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_10,
                   x_11,x_12,x_13,x_14,x_15,x_16,x_17,x_18,x_19,x_20,
                   x_21,x_22,x_23,x_24,x_25,x_26,x_27,x_28,x_29,x_30,
                   x_31,x_32,x_33,x_34,x_35,x_36,x_37,x_38,x_39,x_40,
                   x_41,x_42,x_43,x_44;
    always @ (posedge clk)
      begin
          // use upper 10 bits of x_n and do sign extention
          if (x_n[19]==0)
                x_1 <= {10'b00_0000_0000,x_n[19:10]} + x_1;
          else
                x_1 <= {10'b11_1111_1111,x_n[19:10]} + x_1;
          if (x_n[19]==0)
                y_n <= {10'b00_0000_0000,x_n[19:10]} + x_1 - x_44;
          else
                y_n <= {10'b11_1111_1111,x_n[19:10]} + x_1 - x_44;

          x_2 <= x_1;
          x_3 <= x_2;
          x_4 <= x_3;
          x_5 <= x_4;
          x_6 <= x_5;
          x_7 <= x_6;
          x_8 <= x_7;
          x_9 <= x_8;
          x_10 <= x_9;
          x_11 <= x_10;
          x_12 <= x_11;
          x_13 <= x_12;
          x_14 <= x_13;
          x_15 <= x_14;
          x_16 <= x_15;
          x_17 <= x_16;
          x_18 <= x_17;
          x_19 <= x_18;
          x_20 <= x_19;
          x_21 <= x_20;
          x_22 <= x_21;
          x_23 <= x_22;
          x_24 <= x_23;
          x_25 <= x_24;
          x_26 <= x_25;
```

```
            x_27 <= x_26;
            x_28 <= x_27;
            x_29 <= x_28;
            x_30 <= x_29;
            x_31 <= x_30;
            x_32 <= x_31;
            x_33 <= x_32;
            x_34 <= x_33;
            x_35 <= x_34;
            x_36 <= x_35;
            x_37 <= x_36;
            x_38 <= x_37;
            x_39 <= x_38;
            x_40 <= x_39;
            x_41 <= x_40;
            x_42 <= x_41;
            x_43 <= x_42;
            x_44 <= x_43;
        end

endmodule
```

A.7. Timing Recovery Module File

```
/*****************************************************************
* File:  timing.v
* Author: Jason Dielschneider
* Description:  Timing recovery module.  Takes the absolute value of
*                     the input before calculating early-late.
*
*
*
* Created:
* Revision: August 17/1999 -- Modified for BPSK modem -- Neil Gerein
*****************************************************************/

module timing(clk,timing_in,control_out,control_out_clk);
    parameter Gain_Filter_Width = 19;
    parameter bits_gain_factor = 1;
    parameter a_value = 63;
    parameter a_bits_chopped = 7;

    input clk;
    input [10:1] timing_in;
    output [10:1] control_out;
    output control_out_clk;
    reg control_out_clk;
    reg [2:1] count;
    reg [10:1] early_in,late_in;
    reg [10:1] difference;
    reg [Gain_Filter_Width:1] diff_out;
    reg [10:1] abs_out,control_out;
    reg [10:1] accum;

    // create a variable count to determine the position of early and
late samples

    always @ (negedge clk)
          count = count + 1;

    // take absolute value of the input

    always
      begin
          if (timing_in[10] == 1)
            begin
                abs_out[10] = 1'b0;
                abs_out[9:1] = ~timing_in[9:1];
            end
          else
                abs_out = timing_in;
      end


    /* sample the input into variable for early, sample and late
inputs
                count=0           early sample
                count=1           sample point sample
```

```
                count=2           late sample
                count=3           process the data        */

      always @ (posedge clk)
        begin
            if (count == 2'd0)                  // sample early input
                  early_in = abs_out;
            else if (count == 2'd1)
                  control_out_clk = 1;
            else if (count == 2'd2) // sample late input
                  late_in = abs_out;
            else if (count == 2'd3)
              begin
                  difference[10:1] = early_in[10:1] - late_in[10:1];
                  control_out_clk = 0;
              end
        end

      // make the call to the filter module

      digital_filter DF1
(clk,count,diff_out[Gain_Filter_Width:1],difference[10:1]);
            defparam    DF1.Filter_Width = Gain_Filter_Width;
            defparam    DF1.bits_chopped = a_bits_chopped;
            defparam    DF1.a = a_value;

      // output the control signal is D/A unsigned format

      always @ (negedge control_out_clk)
        begin
            control_out[10] = !diff_out[Gain_Filter_Width];
            control_out[9:1] = diff_out[Gain_Filter_Width-1-
              bits_gain_factor:Gain_Filter_Width-bits_gain_factor-9];
        end


  endmodule
```

A.8. Low Pass Digital File

```
/*****************************************************************
* File:  digital_filter.v
* Author: Jason Dielschneider
* Description:  Extremely sharp lowpass digital filter used for
*                     timing recovery   .
*
*
* Created:
* Revision: August 17/1999 -- Modified for BPSK modem -- Neil Gerein
*****************************************************************/

module digital_filter(clk,count,filter_out,filter_in);
    parameter Filter_Width = 20;
    parameter a = 2047;
    parameter bits_chopped = 12;

    input clk;
    input [2:1] count;
    input [10:1] filter_in;
    output [Filter_Width:1] filter_out;
    reg [Filter_Width:1]
filter_out,diff_temp,filter_out_big,prev_diff_temp;
    reg [Filter_Width+11:1] a_scaled,b_scaled;

    // the input to this filter changes on count = 3
    // input is 2's complement arithmetic
    // filter output occurs on count = 3

    always @ (posedge clk)
      begin
          if (count == 2'b01)
            begin
                diff_temp <= {{(Filter_Width-
10){filter_in[10]}},filter_in[10:1]} +
                  a_scaled[Filter_Width+bits_chopped-1:bits_chopped];
            end
          if (count == 2'b00)
            begin
                prev_diff_temp[Filter_Width:1] <=
diff_temp[Filter_Width:1];    // implement the delay block
            end
          if (count == 2'b11)
            begin
                filter_out_big[Filter_Width:1] <=
diff_temp[Filter_Width:1] - b_scaled[Filter_Width+11:12];
            end

      end


    // multiply the late output by 'a' for the pole

    mult M1 (prev_diff_temp,a,a_scaled);
          defparam M1.Num_of_Bits_A = Filter_Width;
```

```
        defparam M1.Num_of_Bits_B = bits_chopped;

    // multiply by 0.8 for the zero of the filter
    // 1638 = (0.8) * 2048

    mult M2 (pref_diff_temp,12'd1638,b_scaled[Filter_Width+11:1]);
        defparam M2.Num_of_Bits_A = Filter_Width;
        defparam M2.Num_of_Bits_B = 12;

    // assign filter_out_big to filter_out

    always @ (negedge clk)
        filter_out = filter_out_big;


endmodule
```

B. APPENDIX – GENERAL CASE OF RF DIVERSITY COMBINING

Section 3.3 described the special case of the RF diversity combining with two receiving antennas. This appendix describes the general case of the RF diversity combining.

Recall from section 3.3 the equation for the received symbol energy:

$$E_b = \sum_{i=1}^{N} \int_0^T \frac{A_i^2 dt}{2} + \sum_{j=1}^{N} \sum_{\substack{i=1 \\ i \neq j}}^{N} \int_0^T \frac{A_i A_j}{2} \cos(\phi_i - \phi_j + \Psi_i(t) - \Psi_j(t)) dt . \quad \text{(B.1)}$$

The patented invention in [7] describes the properties for a group of phase adjustment functions that will force the double sum in equation (B.1) to zero. Let f_0 be a function that is a value of 1 for 100% of the time. The phase perturbations may be defined by the following properties.

1. The phase perturbation of received signal i controlled by the function f_0 may be expressed as a constant:

$$\Psi_i(t) = \Psi_i \text{ where } \Psi_i \text{ could be equal to 0.} \quad \text{(B.2)}$$

2. The phase perturbation of a received signal i, controlled by some orthogonal function other than f_0, may be expressed as:

$$\Psi_i(t) = 0 \text{ for 50\% of the time}$$

$$\Psi_i(t) = \Psi_i \text{ for 50\% of the time.} \quad \text{(B.3)}$$

3. The phase perturbation phase differences between a received signal i having phase perturbations controlled by f_0, and any other received signal j that has phase perturbations controlled by some orthogonal function, may be expressed as:

$$\Psi_i(t) - \Psi_j(t) = \Psi_i \text{ for 50\% of a symbol period}$$

$$\Psi_i(t) - \Psi_j(t) = \Psi_i - \Psi_j \text{ for 50\% of a symbol period.} \quad \text{(B.4)}$$

4. The phase perturbation difference between any two received signals *i* and *j*, neither of which is controlled by f_0, may be expressed as:

$$\Psi_i(t) - \Psi_j(t) = 0 \text{ for 25\% of a symbol period}$$

$$\Psi_i(t) - \Psi_j(t) = \Psi_i \text{ for 25\% of a symbol period}$$

$$\Psi_i(t) - \Psi_j(t) = -\Psi_j \text{ for 25\% of a symbol period}$$

$$\Psi_i(t) - \Psi_j(t) = \Psi_i - \Psi_j \text{ for 25\% of a symbol period.} \quad \text{(B.5)}$$

Following the above four rules and designating $\Psi_i(t) = \Psi_1$, E_φ is used to represent the double sum term of (B.1) and can be written as:

$$E_\varphi = 2\sum_{j=2}^{N} \int_0^{T/2} \frac{A_1 A_j}{2} \left[\cos(\phi_1 - \phi_j + \Psi_1) + \cos(\phi_1 - \phi_j + \Psi_1 - \Psi_j)\right] dt +$$

$$\sum_{j=2}^{N} \sum_{\substack{i=2 \\ i \neq j}}^{N} \frac{A_i A_j}{2} \int_0^{T/2} \left[\cos(\phi_i - \phi_j) + \cos(\phi_i - \phi_j + \Psi_i) + \cos(\phi_i - \phi_j - \Psi_j) + \cos(\phi_i - \phi_j + \Psi_i - \Psi_j)\right] dt \quad \text{(B.6)}$$

Equation (B.6) can be forced to zero if Ψ_i is π for *i*=2,3,…N by using the trigonometric identity:

$$\cos(a + \pi) = \cos(a - \pi) = -\cos(a). \quad \text{(B.7)}$$

Substituting π for Ψ_i in (B.6) gives:

$$E_\varphi = 2\sum_{j=2}^{N}\int_0^{T/2}\frac{A_1A_j}{2}\left[\cos(\phi_1-\phi_j+\Psi_1)+\cos(\phi_1-\phi_j+\Psi_1-\pi)\right]dt+$$

$$\sum_{j=2}^{N}\sum_{\substack{i=2\\ i\neq j}}^{N}\frac{A_iA_j}{2}\int_0^{T/2}\left[\cos(\phi_i-\phi_j)+\cos(\phi_i-\phi_j+\pi)+\cos(\phi_i-\phi_j-\pi)+\cos(\phi_i-\phi_j+\pi-\pi)\right]dt\,.$$

(B.8)

Substituting the trigonometric identity given in (B.7) into (B.8) gives the following equation:

$$E_\varphi = 2\sum_{j=2}^{N}\int_0^{T/2}\frac{A_1A_j}{2}\left[\cos(\phi_1-\phi_j+\Psi_1)-\cos(\phi_1-\phi_j+\Psi_1)\right]dt+$$

$$\sum_{j=2}^{N}\sum_{\substack{i=2\\ i\neq j}}^{N}\frac{A_iA_j}{2}\int_0^{T/2}\left[\cos(\phi_i-\phi_j)-\cos(\phi_i-\phi_j)-\cos(\phi_i-\phi_j)+\cos(\phi_i-\phi_j)\right]dt$$

(B.9)

Inspection of (B.9) shows that the E_φ goes to zero for all values of Ψ_i. Referring back to (B.1) this is the desired result.