Carrier Frequency Offset Architecture for an OFDM System

By: Shauna Rauhala, B.Eng.

A thesis submitted to the Faculty of Graduate Studies
and Research in partial fulfillment of the
requirements for the degree of
Master of Applied Science (M.A.Sc.)

Department of Electronics
Faculty of Engineering
Carleton University
Ottawa, Canada
September, 2002

© Copyright
Shauna Rauhala, 2002

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-79714-7

Canada

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis

Carrier Frequency Offset Architecture for an OFDM System

Submitted by Shauna Rauhala, B.Eng.
in partial fulfillment of the requirements for the degree of
Master of Applied Science

Dr. J.S. Wight, Thesis Supervisor

Chair, Department of Electronics

Abstract

The ability to track carrier frequency offsets in IEEE 802.11a architectures is imperative for the accurate decoding of high data-rate information. Three methods to detect and correct a carrier frequency offset in an IEEE 802.11a architecture are proposed and compared. Each tracking frequency offset (TFO) design exploits the use of the pilot symbols found in an OFDM (Orthogonal Frequency Division Multiplexed) data symbol. The systems are simulated with multi-path and noise conditions that could be expected in an indoor wireless local area network. Performance is based on a comparison of Bit Error Rate (BER) curves, tracking ability and Packet Error Rate curves. Simulation results show that the best TFO architecture is capable of accurately detecting and correcting typical frequency offsets occurring in the data portion of a packet.

Acknowledgements

There are many thanks that must be extended to many people who have aided in the completion of this degree. First of all, I would like to thank IceFyre Semiconductor, without whose support this thesis would not have been possible. Within IceFyre, I would like to extend a special thanks to the Systems department and to Jung Yee for this opportunity. As well, a thank you to Phil Guillemette, Hossain Pezeshki and Michael Moher who took the time to do their best to answer my questions, no matter how unrelated to their work at that moment. A special thanks to Neil Birkett for his encouragement and inspiration. I would also like to thank Zahir Jaffer for adding some humor to my life when the simulations became difficult due to reasons beyond my control. Which leads me to a big thank you to Claude Evraire for his IT support at least once a week.

Outside of IceFyre there are many people in my life who gave me the strength to pull through this. I would like to thank my parents, my husband, my best friend and sister Laura, and my best friend Indira for their encouragement and motivation throughout my University career. Words can not express my gratitude and love for them. I would also like to thank Jorge Aguirre - just for being himself through the years we've been in school together - Jorge, your outlook on life is refreshing; laughter is always good for the soul, and an excellent stress reliever. We have definitely shared lots of laughter and friendship, hopefully with more to come. Last, but certainly not least, I would like to thank Jim Wight for enticing me to do a Masters in the first place and for providing the extra boost of confidence and reassurance just when I think everything is about to fall apart. When I started this degree I wasn't sure I was ready for it, but now I thank you, Jim, for believing in me.

And, of course, a thank you to NSERC and to Carleton University for their assistance with this degree.

Table of Contents

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	vii
List of Tables	ix
List of Acronyms	x
1 Introduction.....	11
1.1 Motivation.....	11
1.2 Objective.....	12
2 Background	14
2.1 Channel	14
2.2 OFDM.....	16
2.2.1 General Description	16
2.2.2 OFDM Architecture.....	18
2.3 Specifications for 802.11a	21
2.3.1 Packet Formation	21
2.3.2 Parameters.....	24
2.3.3 Transmitter Specifications	25
2.3.4 Channel Model.....	26
2.4 Effects of Timing and Frequency Offsets.....	28
2.4.1 Frequency Offsets and Carrier Phase Offsets	28
2.4.2 Sampling Offsets.....	30
2.5 Tracking the Frequency Offsets.....	32
3 System Description.....	34
3.1 Top Level.....	34
3.2 Channel	36
3.3 Stage I.....	37
3.4 Stage II.....	38
3.5 Tracking.....	42
3.5.1 Method #1	43
3.5.2 Method #2.....	46
3.5.3 Method #3	50
3.5.4 Complexity.....	52
3.6 Overall System.....	52
4 Results and Testing.....	55
4.1 Simulation Setup.....	55
4.1.1 Platforms.....	55
4.1.2 Testbench Generation	57
4.2 Testing.....	60
4.2.1 Description of General Tests	60
4.2.2 Specific Tests	61

4.2.3	Performance Measurements.....	62
4.3	Results.....	66
4.3.1	Expected BER Curves.....	66
4.3.2	Simulation time.....	68
4.3.3	54Mbps results.....	68
4.3.4	A Comparison of Different Data Rates.....	76
4.3.5	Limitations	87
4.3.6	Scatterplot Results	88
4.3.7	ACQFO Results	100
4.3.8	Packet Error Rate Results	102
5	Conclusion	106
5.1	Discussion of Results.....	106
5.2	Future Work	108
6	Appendix A.....	109
7	Appendix B.....	111
8	References.....	116

List of Figures

Figure 2.1: OFDM Signal in Time Domain.....	17
Figure 2.2: OFDM signal definition – Frequency Domain [5].....	17
Figure 2.3: General OFDM transmit/receive chain	19
Figure 2.4: 802.11a Packet structure.....	22
Figure 2.5: IFFT inputs and outputs. ‘Subc’ means subcarrier.....	23
Figure 2.6: Channel Impulse Response for 100ns delay spread [1]	26
Figure 2.7: Channel Model for IEEE 802.11a system [1]	27
Figure 2.8: (a) Frequency offset (f_o) is constant for all frequencies (f_n) within one OFDM symbol (b) Frequency offset ($w_o t$) causes rotation of constellation from one OFDM symbol to the next.....	29
Figure 2.9: Sampling Offset causes linear phase variation over frequency within one OFDM symbol	30
Figure 3.1: Top level block diagram.....	34
Figure 3.2: Acquisition and Tracking Blocks	35
Figure 3.3: Channel model.....	36
Figure 3.4: Packet Detect Architecture.....	38
Figure 3.5: Preamble Processing.....	41
Figure 3.6: Correction of frequency offset	42
Figure 3.7: Tracking Method #1	44
Figure 3.8: Tracking Method #2	47
Figure 3.9: Tracking Method #3	49
Figure 3.10: Method #1 with acquisition.....	53
Figure 3.11: Method #2 with acquisition.....	53
Figure 3.12: Method #3 with acquisition.....	54
Figure 4.1: Inputs and Outputs of Main Code	58
Figure 4.2: Top level diagram of System with datastreams.....	59
Figure 4.3: Ideal BER curves with lowest BER expected	67
Figure 4.4: Ideal BER curves with highest BER expected	67
Figure 4.5: TEST: Noise, no MP, no RFO	69
Figure 4.6: TEST: no Noise, MP, no RFO	69
Figure 4.7: TEST: Noise, MP, no RFO	70
Figure 4.8: a) TEST: Noise, no MP, RFO b) TEST: Noise, MP, RFO	71
Figure 4.9: Standard Deviation of the RFO estimation error versus the applied RFO....	73
Figure 4.10: BER curves	75
Figure 4.11: Average RFO Est Error	80
Figure 4.12: Std Dev of RFO est error for duration of run	82
Figure 4.13: Average Sampling Offset est error for duration of run	84
Figure 4.14: Std Dev of Sampling Offset est error for duration of run	86
Figure 4.15: Scatterplot of transmitted data.....	89
Figure 4.16: Scatterplots of results with noise and MP, but no offsets	90
Figure 4.17: Scatterplots of results with no Noise, no MP and a residual FO of 1000 Hz	92
Figure 4.18: Scatterplots of results with no Noise, MP and a residual FO of 1000 Hz ..	93
Figure 4.19: Scatterplots of results with Noise, no MP and a residual FO of 1000 Hz ..	94

Figure 4.20: Scatterplots of results with Noise, MP and a residual FO of 1000 Hz	97
Figure 4.21: Scatterplots of results with Noise, MP and a residual FO of 7000 Hz	99
Figure 4.22: (a) Noise and MP (b) MP (c) Noise (d) nothing	102
Figure 4.23: PER curves for each method and 4 data rates	104

List of Tables

Table 1: Corresponding modulation schemes, data rates and coding rates	24
Table 2: IEEE 802.11a Specifications Summary.....	25
Table 3: Comparison of Complexities of Methods.....	52
Table 4: SNR range corresponding to Data Rates and Eb/No	64
Table 5: Average RFO estimation error and Average value of Standard Deviation of RFO est err	72
Table 6: Summary of BER values at SNR = 25dB for each test and each data stream...	73
Table 7: BER for 2 tests, 3 methods, 4 data rates	77
Table 8: Summary of 7 Tests, 4 Data Rates with Average and Std ACQFO and RFO results.....	78
Table 9: Maximum tolerable ACQFO for each data rate	87
Table 10: Average Acquisition Estimation Error and Standard Deviation, MP applied	109
Table 11: Average Acquisition Estimation Error and Standard Deviation, no MP applied	110

List of Acronyms

ACQFO: Acquisition Frequency Offset

AGC: Automatic Gain Control

AWGN: Additive White Gaussian Noise

BER: Bit Error Rate

BW: Bandwidth

CFO: Carrier Frequency Offset

CP: Cyclic Prefix

FFT: Fast Fourier Transform

FO: Frequency Offset

ICI: Intercarrier Interference

IFFT: Inverse Fast Fourier Transform

ISI: Intersymbol Interference

MP: Multipath

OFDM: Orthogonal Frequency Division Multiplexing

PER: Packet Error Rate

RFO: Residual Frequency Offset (see TFO)

SC: Single Carrier

SER: Symbol Error Rate

SNR: Signal-to-Noise Ratio

TFO: Tracking Frequency Offset (see RFO)

TO: Timing Offset

WLAN: Wireless Local Area Network

1 Introduction

1.1 Motivation

In a day and age when people are expecting more from their electronic devices, wireless communication is moving to the forefront of the technology wave. Countless advantages over the tried and true wired communication include mobility and simplicity with a lack of necessity for cables or hookups. However, until wireless data rates reach or surpass wired rates, people will not be willing to accept the advantages of converting to wireless local area networks (WLAN's).

The indoor transmission environment has many obstacles, creating a complicated channel. Multipath and frequency selective fading restrict wireless transmission speeds. The movement of people and the close proximity of objects also contribute to a channel that is always changing.

IEEE 802.11a is a standard that has been set to reliably increase WLAN indoor data speeds. Occurring in the unlicensed 5 GHz spectrum, the standard draws on a large bandwidth to allow higher data rates. IEEE 802.11a uses Orthogonal Frequency Division Multiplexing (OFDM) as the modulation scheme. Due to the inherent characteristics of this scheme, multipath has much less of an effect on transmission, and rates of up to 54Mbps with 64-QAM data modulation are supported.

The strengths of an OFDM system also incur new weaknesses. OFDM uses an IFFT (Inverse Fast Fourier Transform) to modulate the data to be transmitted. However, near-perfect synchronization is required in the receiver for accurate demodulation with a Fast Fourier Transform (FFT). Timing and Frequency offsets wreck havoc on the results and must be detected and corrected. In particular, the higher the data rate, the larger the modulation constellation and the more susceptible it is to offsets.

1.2 Objective

An initial frequency offset (FO) estimate (acquisition) occurs at the start of every 802.11a packet. If this estimate is incorrect, a residual FO will exist throughout the remainder of the packet. This frequency offset could be caused by a difference between the transmitter and receiver crystals. The objective of this thesis is to determine the optimum architecture for frequency offset synchronization in an OFDM receiver conforming to IEEE 802.11a standards. The acquisition architecture was selected through a process of

elimination based on existing research. As well, three different architectures for detection and correction of the residual FO were analyzed and compared through extensive simulation. The performance criterion compares Bit Error Rate (BER) and Packet Error Rate test results and the ability of each method to track the residual FO in the presence of channel effects without overcompensation.

The remainder of this thesis progresses as follows; Chapter 2 describes the background knowledge required to understand the tradeoffs resulting from different architectures. The design of the system and the different architectures is described in Chapter 3. Chapter 4 presents testing and results. Finally, Chapter 5 provides a conclusion and recommendations for future research.

2 Background

2.1 Channel

Before a description of OFDM and IEEE 802.11a specifications can commence, an understanding of the anticipated channel characteristics must be grasped. An indoor wireless environment has inanimate items such as desks and walls that combine with moving people to create a complicated, dynamic channel. Multipath occurs when delayed reflections of the transmitted signal interfere with the original signal at the receiver. As well, Additive White Gaussian Noise (AWGN) is added to the signal from the environment and the front end of the receiver. Finally, fading occurs when reflected paths destructively combine with the original signal to cancel it out in the frequency domain.

A multipath channel causes intersymbol interference (ISI), which is the spreading of one symbol into the next. Delay spread (T_{RMS}) is a parameter used to describe the delay of the multipath reflections for a given environment. In a typical home environment, the delay spread is usually less than 50 ns [1]. In an office environment, the delay spread is closer to 100 ns due to more objects and more movement [1]. A manufacturing

environment, such as a warehouse, can have a delay spread as high as 200 to 300 ns [1].

For the intended use of this system, an office environment will be assumed.

Many techniques can be used to combat channel influences. Equalizers can compensate for multipath delay spreads on the order of 100 ns [1]. Generally, the equalizer is a multi-tapped delay line [1]. If the data rate is low and the OFDM symbol duration is long compared to the delay spread of the channel, an equalizer might not even be necessary. However, as distance or data rate increases, ISI becomes more severe, requiring equalization [2]. Using forward error correction coding can reduce the effects of fading. Estimating the channel and applying a correction to the data can diminish signs of multipath and AWGN.

The basis of channel estimation is explained well by Hsu et al [3] and is based on the following equations,

$$R = HS + W \quad (1)$$

$$\hat{H} = \frac{R}{S} \quad (2)$$

$$\hat{R} = \frac{R}{\hat{H}} \quad (3)$$

Equation 1: R = Received, H =Channel Response, S = Transmitted, W = Noise

The above equations show how to correct the received signal with the estimated channel response. Dividing the received signal by the *a priori* known transmitted signal gives a channel estimate. By further dividing the received signal with the channel estimate,

amplitude and phase distortions are minimized. A frequency domain equalizer can be used to perform these operations.

2.2 OFDM

As technology advances, Wireless LAN's become less of a luxury and more of a demand. The potential is limited only by the harshness of the indoor channel. Inherent to its design, Orthogonal Frequency Division Multiplexing (OFDM) is able to overcome some of the difficulties of indoor wireless transmission, especially the effects of multipath.

2.2.1 General Description

OFDM subdivides a high-speed binary stream into N lower-rate subcarriers that are processed and then combined in an IFFT. In a single carrier (SC) system, as the data rate increases, the symbol period becomes shorter and more susceptible to multipath (MP). In an OFDM system, each subcarrier has a lower symbol rate relative to the delay spread of the channel, thus multipath and ISI have less of an effect. As well, a guard interval or cyclic prefix (CP) is added, as shown in Figure 2.1, to further reduce the effects of interference from neighbouring symbols. The CP can be viewed as an extension of the IFFT. As long as the CP is greater than the delay spread, there is no ISI [4].

Figure 2.1: OFDM Signal in Time Domain

The OFDM system bandwidth is shared between subcarriers. The subcarrier spacing equals the symbol rate on each subcarrier, lending orthogonality to the subcarriers [5].

Figure 2.2 shows an example of three subcarriers. T_s is the symbol rate of the system.

At the peak of each subcarrier, there is no interference from the other subcarriers.

Figure 2.2: OFDM signal definition – Frequency Domain [5]

Since each subcarrier is essentially a narrow-band channel, frequency-selective fading is a problem. In particular, if the channel does not have a line-of-sight path, some subcarriers will be permanently flat-fading [2]. The bit error rate of the system is only as

good as the poorest subcarrier, requiring forward error correction (FEC) and bit-interleaving to combat fading channels and to increase the Signal to Noise Ratio (SNR) [5].

2.2.2 OFDM Architecture

A typical OFDM system is shown below. In the transmitter, the bits are processed to provide resilience to frequency selective fading. Each lower rate bit stream is then transformed into symbols by mapping m bits to one of $M=2^m$ complex symbols of the chosen data modulation scheme [6]. Subsequently, each subcarrier is modulated by the corresponding complex symbol and the subcarriers are combined with an IFFT. A cyclic prefix is added to the start of the symbol and the front end prepares the symbols for transmission over the channel.

Figure 2.3: General OFDM transmit/receive chain

In the receiver, the signal is downconverted and the cyclic prefix is removed. The FFT performs the opposite operation of the IFFT and separates out the lower-rate subcarriers from the continuous stream of received data. The FFT essentially acts as a demodulator followed by a complex integrate and dump [7]. The synchronization block estimates initial and residual offset values. Finally, after the equalizer, the signal is demapped and the transmitter processing is reversed.

A frequency domain equalizer is required to further reduce the effects of multipath and to compensate for the channel. Essentially, the equalizer requires one tap per FFT output. Dividing the received signal by the expected signal performs a channel estimate. The estimate is inverted and stored. The assumption is made that the channel does not change for the duration of the packet. When the next OFDM symbol enters, it goes through the

FFT. Each of the N FFT output streams is sent to a complex multiplier where the equalizer applies the channel estimate for that particular tap. In a single carrier system the equalizer would be much more complicated since it would be time-domain based.

The data modulation schemes in this research are limited to M-PSK and M-QAM, and coherent demodulation in the receiver requires each subcarrier to be synchronized in amplitude and phase [2]. Time or frequency synchronization errors will destroy orthogonality and reduce the SNR [2].

Quantitatively, an OFDM system has N subcarriers separated by frequency spacing Δf . The subcarriers are mutually orthogonal within $T=1/\Delta f$. The period, T , is N times larger than a single carrier system of the same bandwidth [2]. The sampling time of the system is

$$T_s = \frac{1}{BW} = \frac{1}{N * \Delta f} \quad (4)$$

where BW is the bandwidth of the system. After the IFFT, the transmitted signal is [2]:

$$s_{n,i} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} S_{n,k} e^{j \frac{2\pi k i}{N}} \quad (5)$$

where

n is the symbol number,

k is the subcarrier number,

i is the sample number, varying within the range $0, 1 \dots N-1$

N is the total number of subcarriers

$S_{n,k}$ is the complex modulated symbol

The received signal after the FFT is [2]

$$R_{n,k} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N-1} r_{n,i} e^{-j \frac{2\pi i k}{N}} = \frac{\sqrt{N}}{T_s} \langle r_n(t), g_k^*(t - nT) \rangle \quad (6)$$

$$\text{where } g_k(t) = \begin{cases} e^{j2\pi k \Delta f t} & \forall t \in [-T_G, T_S] \\ 0 & \text{else} \end{cases}$$

The second term (in the triangular brackets) performs a correlation between r and g . T_s is the sampling period and r is the received signal.

By taking channel effects into account the overall received signal becomes [2]:

$$R_{n,k} = H_{n,k} S_{n,k} + W_{n,k} \quad (7)$$

$$r_{n,k} = h_{n,k} \otimes s_{n,k} + w_{n,k} \quad (8)$$

The first equation is in the frequency domain and the second is in the time domain.

2.3 Specifications for 802.11a

2.3.1 Packet Formation

IEEE 802.11a uses OFDM modulation and transmission at 5GHz for indoor wireless high-speed data communication [1]. Several thick binders are required to contain the complete design specifications for an 802.11a product. The focus of this research, however, is restricted to the Medium Access Control (MAC) and Physical (PHY) layer in

the OSI model. The reader is encouraged to look at [4] for direction and information on other transportation layers.

Figure 2.4 depicts an 802.11a packet. The preamble is made up of 10 short symbols and 2 long symbols. A cyclic prefix consisting of the last half of a long symbol separates the short and long preambles. The short symbols are used for AGC (Automatic Gain Control), antenna diversity and coarse carrier frequency offset estimation. The long symbols are used for fine carrier frequency offset estimation and channel estimation. The short and long symbol estimates are combined to produce an initial frequency offset estimate. Following the preamble is a 24-bit field called the SIGNAL, which contains information such as the data rate and the length of the packet. Finally, the OFDM data symbols occur, each preceded by a CP consisting of the last quarter of that particular data symbol. These data symbols will have a residual frequency offset if the initial frequency offset estimate is incorrect. The maximum number of octets that can be accepted for transmission in a single packet is 4095.

Figure 2.4: 802.11a Packet structure

The data is divided into 48 low-speed bit streams and each is mapped into a complex number. The bit streams modulate subcarriers corresponding to $2\pi f_1 kt$, where k is the subcarrier number, ranging from $[-26 \text{ to } 26]$ and f_1 is 312500Hz. Four pilots with a known phase and magnitude exist on subcarrier numbers 7, 21, -7 and -21. Figure 2.5 shows 12 nulls on internal taps 0 and 27 to 37 [4]. The null subcarriers carry no information and are used as a guard band between subcarriers [6]. The IFFT combines the 48 data subcarriers with the 4 pilots and 12 nulls for a total of 64 inputs to the IFFT. According to equation 4, this will give a total bandwidth of

$$(48+4) \times 312.5k = 16.25\text{MHz} \quad (9)$$

Figure 2.5: IFFT inputs and outputs. 'Subc' means subcarrier

2.3.2 Parameters

The following table outlines the possible data modulation schemes and the corresponding data and coding rates [4]:

Table 1: Corresponding modulation schemes, data rates and coding rates

Modulation	Data Rate (Mbps)	Coding Rate
64-QAM	54	3/4
64-QAM	48	2/3
16-QAM	36	3/4
16-QAM	24	1/2
QPSK	18	3/4
QPSK	12	1/2
BPSK	9	3/4
BPSK	6	1/2

As shown in Table 1, two data rates are possible with each modulation scheme due to different coding rates. In each case, the slower data rate uses a simpler coding scheme. Therefore, if a transmission system performs well at a set modulation scheme for the higher data rate, one can assume the transmission system will perform as well or better with the same modulation scheme and the lower data rate.

There are several other system specifications included in the IEEE 802.11a specifications and outlined in and included in Table 2 [4].

Table 2: IEEE 802.11a Specifications Summary

Parameter	Value
Total Bandwidth:	20 MHz
Useful Bandwidth:	16.6 MHz
Subcarrier Frequency Spacing (Δf):	312500 Hz
Sampling Spacing (T_s):	50 ns
Short symbol Length:	0.8 μ s
Long symbol length:	3.2 μ s
Long cyclic prefix length:	1.6 μ s
Data length:	3.2 μ s
Cyclic Prefix Length:	0.8 μ s
Symbol Length (T):	4 μ s
Number of samples in short symbols:	16
Number of samples in long symbols & data:	64
Number of subcarriers used in short:	12
Number of subcarriers used in long:	53
Number of subcarriers used in data:	48
Number of subcarriers used for pilots:	4

Notice that the short symbols use one in four subcarriers and are one quarter shorter than the regular OFDM symbols [6].

2.3.3 Transmitter Specifications

The IEEE 802.11a specifications include a description of the processing and deprocessing blocks shown in Figure 2.3. This section will give an overview of the important aspects of these blocks, but the reader is encouraged to look at [4] for more details.

The scrambler reduces peak to average power by scrambling the input with a sequence. The output with the lowest peak to average power ratio is transmitted [4]. The convolutional encoder is a standard unit with a rate of $1/2$ [4]. Puncturing must be used to omit bits in order to achieve the coding rates of $3/4$ and $2/3$ [4]. Since deep fades can cover several subcarriers and cause bursts of bit errors, bit interleaving is performed to randomize the occurrence of bit errors.

In the receiver, the processing is undone. Deinterleaving swaps the bits back into position. Depuncturing replaces the bits that were omitted with nulls or placeholders. A Viterbi Decoder is suggested for decoding and soft decision with 3 bits was chosen. Finally, unscrambling removes the scrambling sequence applied in the transmitter.

2.3.4 Channel Model

Figure 2.6: Channel Impulse Response for 100ns delay spread [1]

Figure 2.6 shows a typical channel impulse response for an office environment. The energy on the leading edge is caused by the reflections of the transmitted signal. Notice

that the peak occurs around 100ns, which is the typical delay spread for an office environment. Since the channel is dynamic, the exact impulse response is always changing. Therefore, to provide constancy in simulation, IEEE 802.11a describes a channel model, as outlined below.

Figure 2.7: Channel Model for IEEE 802.11a system [1]

The average power decays exponentially and the impulse response is complex with a random uniformly distributed phase and a Rayleigh distributed magnitude [1]. The equation for the impulse response is:

$$h_k = N\left(0, \frac{1}{2}\sigma_k^2\right) + jN\left(0, \frac{1}{2}\sigma_k^2\right) \quad (10)$$

where

$$\sigma_k^2 = \sigma_o^2 e^{-k \frac{T_s}{T_{RMS}}} \quad (11)$$

$$\sigma_o^2 = 1 - e^{-\frac{T_s}{T_{RMS}}} \quad (12)$$

$N\left(0, \frac{1}{2}\sigma_k^2\right)$ is a zero mean Gaussian random variable that is created by multiplying an $N(0,1)$ random variable by $\frac{\sigma_k}{\sqrt{2}}$ [1]. The sampling period is T_s and T_{RMS} is the delay spread of the channel. According to IEEE specifications, a performance assessment of the system must demonstrate operation for a multipath delay spread of the smallest value of $1/BW$ or $T_{RMS}/2$ [1]. As well, the IEEE working group allows the assumption that the channel changes very slowly and is constant over the length of the packet [6].

2.4 Effects of Timing and Frequency Offsets

2.4.1 Frequency Offsets and Carrier Phase Offsets

Carrier frequency offsets and carrier phase offsets have different effects on the phase of a signal. Figure 2.8a shows that frequency offsets are constant within an OFDM symbol, i.e. all subcarrier frequencies experience the same offset. However, as shown in Figure 2.8b, the constellation will rotate from one OFDM symbol to the next.

Figure 2.8: (a) Frequency offset (f_o) is constant for all frequencies (f_n) within one OFDM symbol

(b) Frequency offset ($w_o t$) causes rotation of constellation from one OFDM symbol to the next

If the expected phase of each pilot can be normalized to zero, then a deviation from zero would indicate the phase of the residual frequency offset. Therefore, an estimate, ϕ_f , could be found by averaging the phase of the 4 pilots within one OFDM symbol. In the following equation f_o is the frequency offset in Hz and θ_n is the measured phase of subcarrier number n .

$$\phi_f = \frac{\theta_{-21} + \theta_{-7} + \theta_7 + \theta_{21}}{4} \quad (13)$$

$$f_o = \frac{\phi_f}{2\pi(4\mu s)} \quad (14)$$

Technically, the largest frequency offset that can be detected is $312500/2 = 156250\text{Hz}$, since the subcarrier spacing is 312500Hz . If this offset is not completely detected in the preamble, the remaining residual frequency offset must be determined from the pilots in the OFDM symbols.

A carrier phase offset is different than a carrier frequency offset. While a carrier frequency offset causes a rotation of the constellation from data symbol to data symbol, a

carrier phase offset results in a static constellation rotation for all symbols in the packet.

This is similar to applying $e^{j\theta_{CPO}}$ to the received packet.

2.4.2 Sampling Offsets

A sampling offset creates a linear phase ramp across all frequencies within one OFDM symbol, the slope of which depends on the value of the timing offset, as shown in Figure 2.9. This can occur if the framing is incorrect.

Figure 2.9: Sampling Offset causes linear phase variation over frequency within one OFDM symbol

Figure 2.9 suggests that $\theta_{21} = -\theta_{-21}$ and $\theta_7 = -\theta_{-7}$ when a sampling offset occurs. Also, $\theta_{21} = 21 * \theta_1$ and $\theta_7 = 7 * \theta_1$, which leads to an averaging of the pilots by referencing them back to subcarrier number one and using equation 13, as shown below.

$$\phi_t = \frac{1}{4} \left[\frac{\theta_{21}}{21} + \frac{\theta_7}{7} + \frac{\theta_{-21}}{-21} + \frac{\theta_{-7}}{-7} \right] \quad (15)$$

$$\begin{aligned} \tau &= \frac{1}{4(2\pi f_1)} \left[\frac{\theta_{21}}{21} + \frac{\theta_7}{7} + \frac{\theta_{-21}}{-21} + \frac{\theta_{-7}}{-7} \right] \\ &= \frac{1}{28(2\pi f_1)} \left[\frac{\theta_{21}}{3} + \theta_7 - \frac{\theta_{-21}}{3} - \theta_{-7} \right] \\ &= \frac{1}{28(2\pi f_1)} \left[\frac{1}{3}(\theta_{21} - \theta_{-21}) + (\theta_7 - \theta_{-7}) \right] \end{aligned} \quad (16)$$

Where, ϕ_t is the weighted average phase of the 4 pilots and τ is the timing offset value.

Since τ is the slope of the line in Figure 2.9, the maximum value of τ that can be detected

depends on the equation of the line: $\theta_n = \tau \cdot f_n$, which leads to:

$$\begin{aligned} \tau &= \frac{\Delta\theta}{2\pi\Delta f} = \frac{\theta_{31} - \theta_0}{2\pi(f_{31} - f_0)} \\ &= \frac{\theta_{31}}{2\pi f_{31}} \\ &= \frac{\pi}{2\pi \cdot 31 \cdot 312500} \\ &= 51.6ns \end{aligned} \quad (17)$$

Therefore, the maximum sampling offset value that can be detected is 51.6ns, which is approximately one sample if the sampling period is 50ns. As shown in Figure 2.8, the phase relationship wraps around and repeats. If a sampling offset occurs that is greater than 51.6ns, the wraparound will take place within the frequency range of the subcarriers and will distort the results.

2.5 Tracking the Frequency Offsets

Tracking refers to the detection and correction of frequency offsets. The following equation corresponds to a perfect received signal after the FFT.

$$R_{n,k} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N-1} r_{n,i} e^{-j \frac{2\pi i k}{N}} \quad (18)$$

When a residual frequency offset exists, the received signal becomes

$$R_{n,k} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N-1} r_{n,i} e^{-j \frac{2\pi i k}{N}} e^{j 2\pi f_o T} \quad (19)$$

To remove the frequency offset, $e^{-j 2\pi f_o T}$ must be applied in either the time or frequency domain. In the time domain, a Numerically Controlled Oscillator (NCO) can create $\cos(2\pi f_o n) - j \sin(2\pi f_o n)$ which can be used to correct the data with a complex multiplier. The drawback of an NCO is that any difference in phase between the actual FO and the NCO waveform will rotate the constellation. In the frequency domain, $e^{-j 2\pi f_o T}$ can be generated and multiplied with each of the 48 data streams, without accidental constellation rotation.

The expected phase of the normalized pilots after the FFT is zero. If the pilots in the first symbol have a phase other than zero, this phase is the rotation of the constellation, which corresponds to a residual frequency offset or a static carrier phase offset. In the equation below, ϕ_n is the expected phase of the pilots, γ_{static} is the initial phase offset and γ_{CFO} is

the residual carrier frequency offset phase that causes the symbols to rotate around the constellation.

$$\begin{aligned}
 \phi_0 &= \gamma_{static} + \gamma_{CFO} \\
 \phi_1 &= \phi_0 + \gamma_{CFO} = \gamma_{static} + \gamma_{CFO} + \gamma_{CFO} \\
 \phi_2 &= \phi_1 + \gamma_{CFO} = \gamma_{static} + \gamma_{CFO} + \gamma_{CFO} + \gamma_{CFO}, etc
 \end{aligned} \tag{20}$$

A static phase offset will distort the frequency offset estimated from the first symbol.

Therefore, a more robust method is to take the difference in phase of subsequent symbols, as shown below. The constellation will rotate by the full frequency offset value each symbol, regardless of phase offsets.

$$\begin{aligned}
 \Delta\phi_{10} &= \phi_1 - \phi_0 = \gamma_{CFO} \\
 \Delta\phi_{21} &= \phi_2 - \phi_1 = \gamma_{CFO}
 \end{aligned} \tag{21}$$

A discrepancy occurs between the conversion from the time to frequency domains

because the FFT uses only 3.2us of a 4us symbol due to removal of the cyclic prefix.

Therefore, if the FO estimate found in the frequency domain is used in the time domain, a correction factor of 4/3.2 must be applied to the estimate and to the NCO phase.

3 System Description

3.1 Top Level

The following figure is a top level diagram of the system simulated for this thesis.

Figure 3.1: Top level block diagram

The Transmitter and the Determine Bits blocks have standard components regulated by the IEEE 802.11a specifications and described in section 2.3.3. This thesis will focus on the synchronization blocks.

Upon receiving the signal, two types of synchronization are required. Acquisition uses the preamble for packet detection, framing and initial frequency offset estimation. The tracking block uses the pilots in the OFDM data symbols to detect residual offsets remaining after the acquisition block. Figure 3.2 shows a more detailed description of the synchronization blocks.

Figure 3.2: Acquisition and Tracking Blocks

FO = Frequency Offset

RFO = Residual Frequency Offset (Tracking)

TO = Timing Offset = Sampling Offset

Stages I, II and III will be explained further in the following sections.

3.2 Channel

Figure 3.3: Channel model

Figure 3.3 shows how the offsets and channel effects are applied to the data in the simulations. The actual offsets applied depend on the type of test being run and are not necessarily applied concurrently. As well, the sequence of the offsets is arbitrary. Multipath is the first channel effect applied and is generated according to the algorithm outlined in Section 2.3 with a delay spread of 50ns. Additive White Gaussian Noise is then generated dynamically and added to the data, after which the applied frequency offset is shown as f_o . The first few incoming samples can be removed to simulate an incomplete first symbol. Finally, the incoming signal is oversampled (os) by 4, creating an 80MHz sampling rate to improve the resolution of the acquisition techniques. Downsampling to 20MHz is performed before the FFT is reached.

3.3 Stage I

Two possible methods for packet detection are power estimation (or received signal strength indicator - RSSI) and correlation. As described in [8], a threshold detector can be set to monitor the level of the incoming signal power (RSSI). When the RSSI measurement passes a given threshold, activity is detected on the channel, but this does not guarantee the activity is an 802.11a packet. A second option is to perform an auto-correlation of the incoming signal with itself. The repetition of the short symbols can be used to detect activity on the channel and confirm it is a packet, as suggested in [3]. Similarly, as a final alternative, a cross-correlation can be performed between the incoming signal and the expected value of the signal, using the short symbol values [9].

Although more accurate, correlation would require additional hardware and would also be power intensive. Therefore, an RSSI measurement is used to detect activity on the channel and send a *Start ST II* signal that initiates Stage II. In Stage II a correlation is performed to confirm the activity is a packet. Figure 3.4 shows a schematic of the RSSI measurement.

Figure 3.4: Packet Detect Architecture

3.4 Stage II

In the literature there are a variety of methods suggested for acquisition. Some references, such as [9], [10] and [11] propose exploiting the short symbol repetition by auto-correlating the incoming signal with a delayed version of itself and then using a sliding window to average the results. This auto-correlation is also known as a Maximum Likelihood Estimation (MLE). Hsu et al [3] quote the following equation as their MLE algorithm.

$$\alpha = \frac{1}{L} \sum_{l=0}^{L-1} \text{angle} \left(\sum_{n=0}^{N-1} z_{i+n+IN} z_{i+n+(l+1)N}^* \right) \quad (22)$$

where, L = number of symbols

N = number of samples

This technique creates one large flat peak spanning from the end of the first short symbol to the end of the last short symbol. Other references, such as Warner [8] and Tufvesson [9] suggest performing a cross-correlation between incoming data and expected short symbol values. This will create a peak at the end of each short symbol, which is more

useful than one large peak. Although auto-correlation requires less hardware, cross-correlation is the chosen method because the sequence of peaks can be used for both framing and frequency offset estimation.

Stage II is run, with slightly different variations, for long and short symbol processing.

The incoming preamble is cross-correlated with the expected values, as per the following equation,

$$v_i = \sum_{k=0}^{N-1} c_k^* r_{k+i} \quad (23)$$

$$i = 0, 1, 2, \dots$$

c = known symbol

r = received symbol

N = number of samples = 64

This is similar to using a maximum likelihood detector with a bank of matched filters.

The signal after the cross-correlation summation, v , will have peaks at the end of every matched short symbol. The location of the peaks can be used to determine framing and the difference in phase between peaks gives the phase rotation of the short symbols, i.e. the coarse frequency offset (CFO). The long symbols are downsampled to 20MHz and corrected for the CFO before being sent through the cross-correlator. The taps of the cross-correlator are changed to the expected long symbol values and 2 peaks are produced, one at the end of each long symbol. The difference in phase values found at the two peaks is the fine frequency offset (FFO) estimation. An improved framing estimation can also be performed, based on the location of the long symbol peaks. The

value of the FFO estimation is added to the CFO estimation to generate the total acquisition frequency offset (ACQ FO) estimation.

Figure 3.5 shows the acquisition block diagram. The magnitude of the signal after the cross-correlation is sent through a peak detector, which sends a signal every time a peak is detected. At each peak, the phase is sampled and subsequent phase samples are subtracted and averaged to give the frequency offset (FO) estimation. The peaks must also be counted in case less than 10 complete short symbols are received. The FO estimation is ignored if less than 1000Hz because the accuracy of the system under noise is limited to 1000Hz. However, a 1000Hz offset is easily detected and corrected in Stage III. Finally, the Index designates the end of the short symbols and the start of the long.

Ten short symbols are used for the coarse frequency offset estimate. In reality, fewer short symbols will be available for the coarse FO estimate since some short symbols will be dedicated to antenna diversity. If fewer short symbols are used for the estimate, the accuracy of the estimate will decrease slightly since less short symbols will be available for the average.

Figure 3.5: Preamble Processing

Oversampling the short symbols delivers an improved estimate of the short symbol peak locations and phase values. If the short symbols are oversampled by 4, then 64 samples are taken per short symbol at a sampling rate of 80MHz. Since the long symbols are downsampled *before* entering the cross-correlator, there will be 64 long symbol samples at 20MHz, allowing the hardware to be reused.

Figure 3.6 shows the chain of correction between the short and long symbols. The short symbol estimate (CFO) is used to correct the long symbols with a Numerically Controlled Oscillator (NCO). Then the long symbol fine frequency offset (FFO) estimate is added to the CFO to give the total acquisition frequency offset estimate (ACQ FO). Notice the short symbols are oversampled at a rate of 80MHz and the long symbols are sampled at a rate of 20MHz.

Figure 3.6: Correction of frequency offset

3.5 Tracking

The frequency offset tracking system includes Stage III and an Equalizer. The frequency domain equalizer was already described in section 2.2.2 and will not be discussed in much detail here. The equalizer simply performs a channel estimate and assumes the

channel does not change significantly during the packet. The pilots and nulls are removed before the data symbols are equalized; however, the pilots are equalized within Stage III.

Three different tracking detection/correction methods were designed and tested. In general, the distinctions are:

Method 1: Correct next symbol with previous symbol results (time domain correction)

Method 2: Correct current symbol with its own results (freq domain correction)

Method 3: Correct current symbol with its own results (time domain correction)

All three methods use the embedded pilots to estimate the residual frequency offset in the data symbol, as described in Section 2.4.3. As well, the sampling (timing) offset (TO) detection method described in Section 2.4.2 was tested for accuracy alongside the tracking frequency offset methods.

3.5.1 Method #1

This section will walk through Figure 3.7 to describe the flow of data through Tracking Method #1.

Data symbols are received into Stage III after being downsampled to 20MHz. The data symbols are corrected for the acquisition frequency offset estimate stored in FO_tot_1 by using an NCO to generate cos and sin waveforms. The phase of the NCO is stored in nco_1. The cyclic prefix is a repetition of the last quarter of the data symbol. Therefore,

the data symbol boundaries are “shifted” into the cyclic prefix to remove ISI on the tail end of the data symbol. In total, 3.2us of the 4us long OFDM symbol remains. The symbol goes through a 64-point FFT, after which the pilots are separated out and sent to the equalizer. The equalizer produces tap values to multiply the pilots by to reduce the channel effects on the pilots. According to IEEE transmission specifications, the polarity of the pilots during transmission is reversed under a predefined pattern. As a result, some pilots will have a phase of π for one symbol and a phase of zero the next symbol. This polarity reversal must be undone to properly average the pilot phases for frequency offset estimation, hence the block “Add π to pilots”.

After the “*Determine TFO/TO*” block, the TFO is the averaged phase of the pilots and is *not* the final frequency offset to be used for correction. TO is the detected timing offset value. The TFO is fed to the next code block for processing where the previous TFO is subtracted from the current TFO and the difference is added to the previous TFO_tot that was applied. When the difference goes to zero, the previous TFO_tot is correct and is continued to be applied.

The second symbol is corrected only for the pre-existing acq_FO_phase. Therefore, the NCO must be updated to reflect the delay in TFO estimation and the frequency-time conversion. As well, by combining the ACQ FO and TFO, the NCO causes a static phase offset, so $(\text{TFO}(2) - \text{TFO}(1)) * 4/3.2$ must be applied to all subsequent estimates.

Figure 3.7: Tracking Method #1

3.5.2 Method #2

This method is much simpler than #1 and is outlined in Figure 3.8. As with method #1, the acquisition FO estimate is sent through an NCO to generate a waveform to correct the incoming data. The cyclic prefix is removed and the OFDM symbol is sent through an FFT. Method #2 uses a feedforward correction to eliminate feedback complexities. The symbols are unwrapped from one to the next, to ensure consistency of rotation of consecutive symbols. According to Matlab help files, “unwrap(P) corrects the radian phase angles in array P by adding multiples of $+2\pi$ when absolute jumps between consecutive array elements are greater than π radians.” As an example, if the frequency offset phase is 0.1 and $TFO(x) = 3.10$, then the expected value of the next estimate should be $TFO(x+1) = 3.20$. However, the phase limits of the simulator are from $-\pi$ to π , so the recorded value of $TFO(x+1)$ will be -3.08 . The difference between these two numbers gives

$$\begin{aligned}
 &\text{Tracking FO Phase} \\
 &= TFO(x+1) - TFO(x) \\
 &= -3.08 - 3.1 \\
 &= -6.18
 \end{aligned} \tag{24}$$

which is not the correct FO. By unwrapping from symbol to symbol, $TFO(x+1)$ maintains its correct value of 3.20 and the true Tracking FO phase is found.

For correction in the frequency domain, all 48 data streams after the FFT are multiplied by the same value of $e^{(-j * TFO_{tot})}$. Note that method #2 is the only method that uses the

true value of the averaged pilots without alteration because the estimation and correction both occur in the frequency domain.

Figure 3.8: Tracking Method #2

3.5.3 Method #3

Method #3 is a combination of #1 and #2 and is shown in Figure 3.9. The current symbol is used to estimate the frequency offset value and the correction is applied to the same symbol in the time domain using a second NCO to allow parallel processing. Method #3 has the same processing as #2 up until the TFO has been determined. Similar to method #1, the difference in phases between consecutive symbols, $TFO(x) - TFO(x-1)$, is found and is added to the acquisition FO phase to create FO_tot_3 , the FO correction value. FO_tot_3 is fed to a second NCO to correct the current symbol in the time domain. The corrected data stream is sent to an FFT to be converted to the frequency domain for bit processing. The conversion from frequency domain estimation to time domain correction requires the NCO phase to be updated during the second data symbol. Two NCO's and FFT's are required for simultaneous processing, otherwise the detection and correction would take longer because each data symbol would have to go through the FFT twice.

This method has some obvious inefficiencies, such as a delay between receiving, buffering and correcting the symbol. As well, additional hardware is a drawback. However, for the sake of analysis, the results of this method can be compared to #1 and #2 since this method is a combination of both.

3.5.4 Complexity

An important issue in comparing architectures is to compare their complexities. This can be performed intuitively for the above three methods. All three methods have similar components (e.g. Equalizer followed by multiplier). By counting the additional operations that are required for each method, the complexity can be compared, as shown in Table 3. The numbers are based solely on Figure 3.7, 3.8 and 3.9. Method #3 has an additional level of complexity not shown in the table because it requires an extra NCO and FFT. Based on the table, Method #2 has the lowest complexity, followed by Method #3 and then Method #1.

Table 3: Comparison of Complexities of Methods

	Additions/ Subtractions	Multiplications/ Divisions	Buffers/ Delays
Method #1	6	7	1
Method #2	0	2	1
Method #3	3	5	3

3.6 Overall System

Figure 3.10, Figure 3.11 and Figure 3.12 give graphical representations of how each tracking method combines with acquisition. The purpose of these figures is to give the reader an idea of the additional level of complexity and hardware associated with each method and to depict in a general way how each correction method is different. The tracking blocks are shaded in each diagram. The unshaded blocks are the acquisition blocks and are the same for all three methods.

Figure 3.10: Method #1 with acquisition

Figure 3.11: Method #2 with acquisition

Figure 3.12: Method #3 with acquisition

4 Results and Testing

4.1 Simulation Setup

The system described in Section 3 was transformed into code for extensive simulations. The final code is rather large and bulky, approximately 4000 lines long, and will not be included here. Instead, a description of the platforms and tests will be examined. However, for the purpose of illustration of the code, the Main Code is included in Appendix B (see also section 4.1.2).

4.1.1 Platforms

The system was built and tested on a block-by-block basis. The “Transmitter” and “Determine bits” blocks from Figure 3.1 were provided by Jeff Lariviere and Trevor Yensen from AvalonDSP, since the blocks use standard components. This offered more time to focus on designing the synchronization blocks. Initially, SystemView, by Elanix, was used to build the simulations. SystemView uses a GUI interface that allows simple communication blocks to be added and manipulated. Adding components is easy and output can be extracted at any stage in the flow of data, primarily for graphical analysis. Most of the Acquisition design testing was performed in SystemView. However,

SystemView processes data on a serial basis, and has difficulty performing a 64-point FFT in parallel. Attempting the feat for Stage III was both processor intensive and time consuming.

Therefore, alternative platforms were investigated and Matlab was selected as a possibility. Matlab code is easy to write and easy to change. Output can be extracted at any point in the flow of data and saved to a file for future use or plotted in a graph. As well, Matlab can process data serially or in parallel with the use of matrices. Matlab was accepted as the new platform and the acquisition synchronization block was recoded and retested. After the Matlab Acquisition results matched the SystemView Acquisition results, the tracking blocks were added to the Matlab code one at a time, which was tricky and time consuming. The longest, most difficult part was confirming the tracking methods were working properly and performing debugging when they were not. Data had to be extracted and compared to what was expected. The feedback loop in Method #1 made this even more difficult to do.

The final amount of code for end-to-end simulations is quite large. The acquisition block runs once per packet to operate on the preamble. However, the tracking block is run for each OFDM symbol in the packet. As well, each OFDM symbol must go through the three tracking methods within the tracking block. Loops are not Matlab's forte and the use of Matlab for long simulations became questionable. The thought was that perhaps C code would be more efficient. The Matlab C code generator was used to convert the Matlab code into C code files. The C code was loaded onto a Unix station and run with a

C compiler. A competition between Matlab and C proved that the Matlab code actually ran faster than the generated C code. As well, by converting to C, twice as many files were generated. The benefits of continuing with Matlab far outweighed the trouble of trying to alter C code if a code change had to be made. So, Matlab remained the testing platform.

4.1.2 Testbench Generation

The Main code was generated in such a way as to accept several different parameters as inputs, see Figure 4.1. A driver file or testbench can wrap around Main and send inputs to Main on a packet-by-packet basis. Automatic runs can occur over thousands of iterations while controlling the length of the packets, the data rate and the offsets applied. The testbench can then extract and process the outputs and save the performance results to a file.

Figure 4.1: Inputs and Outputs of Main Code

Internally, the system controls 5 data streams after the TX block, as shown in Figure 4.2.

The Perfect data stream is ideal; the data is sent directly to the FFT from the transmitter without noise or offsets or synchronization. The Genie data stream is sent through the channel and then through the FFT. Finally, three data streams (#1, 2, 3) are created by the three Tracking methods. All data streams are sent to the *Rx Determine bits* block to generate bit streams for comparison. Notice that although the Perfect data stream is shown going to the “Rx Determine Bits” block, it does not go through the Equalizer within that block. The Perfect data stream does not go through the channel and, thus, does not require equalization.

Figure 4.2: Top level diagram of System with datastreams

The Genie and bit streams #1, 2 and 3 are compared to the Perfect bit stream to count the number of bit errors in the packet. The Genie is a baseline for comparison of Methods #1, 2 and 3 and should generate textbook BER curves when no offsets are applied. When frequency offsets occur, Methods #1, 2 and 3 are expected to perform better than the Genie, since the Methods incorporate Acquisition and Tracking Frequency Offset correction, while the Genie does not. In this case, the Genie shows how poor the results would be without tracking the residual FO.

4.2 Testing

There are several tests that can be run to confirm the operation of the code and to determine the performance of each Tracking correction/detection method. This section will begin with a general overview of types of tests that could be run before leading into system specific tests.

4.2.1 Description of General Tests

4.2.1.1 Ideal

This test is run without channel effects (noise, multipath and offsets). The results indicate the highest possible level of performance of the system. As well, if the results are not as expected (e.g. very low BER) then suspicion is raised about the code and the code must be reviewed for correctness.

4.2.1.2 General tests for all data modulation schemes

For each modulation scheme, two data rates exist, due to two different coding rates, as explained in section 2.3. In each case, the higher data rate is tested and the assumption is made that the lower data rate will have a slightly improved performance over the higher data rate. As well, if the higher data rate satisfies all requirements, then the lower data rate will probably also satisfy all requirements.

4.2.1.3 Apply Offsets

A combination of the offsets described in Sections 2.4.1 and 2.4.2 can be applied.

The Main code allows the offsets to be applied alone or together, on a packet-by-packet basis. BER curves can be generated and tracking performance can be analyzed.

4.2.2 Specific Tests

The extent of the tests resulting from a combination of the general test descriptions can be almost limitless and can run for eons. So, a few were selected based on the characteristics of the system being tested. Below is an outline of the tests that were run, with pros and cons of running each. Note that if Multipath and AWGN are not added, the test becomes a confirmation of operation and not a system performance test. As well, each of the tests must be run over a range of SNR values and data rates and with a variable packet length.

1. Apply an acquisition frequency offset to test if system can recover from residual FO estimation errors. Also, test if unwarranted Residual Frequency Offset (RFO) estimation errors are incurred due to ACQ FO application.
2. Apply a forced RFO and determine performance of methods. This forced RFO can be applied to the data symbols after the acquisition has occurred but

before the tracking begins. The application method is the same as for the acquisition FO in Figure 3.3.

3. Apply ACQ FO and RFO together.
4. Determine limits of system. Apply offsets such as, Incorrect Sampling Instant, Incomplete First Data Symbol and severe carrier frequency offsets, independently, until the System breaks (i.e. the bit error rate is 0.5). The System can detect a sampling offset, but cannot tolerate it because no correction exists.
5. Stemming from #4, the offsets can be applied together to determine system performance. In light of the fact that all offsets *could* occur simultaneously in real life, this test might be important, but might have disastrous results since some offsets are not currently being corrected or detected.

4.2.3 Performance Measurements

The purpose of the tests is to determine which tracking method detects and corrects a residual frequency offset the best. Performance is based on 2 parameters: BER and severity of RFO estimation errors. A RFO estimation error occurs when the method doesn't completely detect/correct the ACQ FO estimation error or the forced RFO. Tolerance of RFO estimation errors will depend on how severely the errors affect the BER.

4.2.3.1 BER tests

The expected BER curves correspond to the following equation:

$$SNR = 10 \log_{10} \left(\frac{E_b}{N_o} \frac{R_T}{B} \right) dB \quad (25)$$

$$R_T = \text{TransmissionRate}$$

$$B = \text{Bandwidth} = \frac{48}{64} (20e6)$$

$$\frac{E_b}{N_o} = \text{BitEnergyToNoiseRatio}$$

From this equation, Table 4 shows the derived SNR ranges to be used in the BER tests. The E_b/N_o range used to calculate corresponding Signal to Noise Ratios varies for each data rate and the limits are taken from typical Symbol Error Rate (SER) curves, such as those found in [12]. The corresponding data symbol error rates for the selected E_b/N_o values are:

$$1e-01 \quad 1e-02 \quad 1e-03 \quad 1e-04 \quad 1e-05 \quad 1e-06$$

To find the expected BER values from the SER values, the following relationship holds true [14]:

$$\frac{P_e}{\log_2 M} \leq BER \leq P_e \quad (26)$$

P_e is the probability of a symbol error and M is the M -ary data mapping scheme. The 'Expected BER' column in Table 4 lists the BER range of values from the above equation.

Table 4: SNR range corresponding to Data Rates and Eb/No

Data Rate (Mbps)	Eb/No Values [dB]	Corresponding SNR Values [dB]	Expected BER	
			MIN	MAX
54 (64-QAM)	10	15	1.7e-02	1e-01
	14	19	1.7e-03	1e-02
	17	22	1.7e-04	1e-03
	18	23	1.7e-05	1e-04
	19	24	1.7e-06	1e-05
	20	25	1.7e-07	1e-06
48 (64-QAM)	10	15	1.7e-02	1e-01
	14	19	1.7e-03	1e-02
	17	22	1.7e-04	1e-03
	18	23	1.7e-05	1e-04
	19	24	1.7e-06	1e-05
	20	25	1.7e-07	1e-06
36 (16-QAM)	6	9	2.5e-02	1e-01
	10	13	2.5e-03	1e-02
	12	15	2.5e-04	1e-03
	14	17	2.5e-05	1e-04
	15	18	2.5e-06	1e-05
	16	20	2.5e-07	1e-06
24 (16-QAM)	6	8	2.5e-02	1e-01
	10	12	2.5e-03	1e-02
	12	14	2.5e-04	1e-03
	14	16	2.5e-05	1e-04
	15	17	2.5e-06	1e-05
	16	18	2.5e-07	1e-06
18 (QPSK)	0	0	5.0e-02	1e-01
	4.5	5	5.0e-03	1e-02
	7	7	5.0e-04	1e-03
	9	9	5.0e-05	1e-04
	10	10	5.0e-06	1e-05
	11	12	5.0e-07	1e-06
12 (QPSK)	0	-1	5.0e-02	1e-01
	4.5	3	5.0e-03	1e-02
	7	6	5.0e-04	1e-03
	9	8	5.0e-05	1e-04
	10	9	5.0e-06	1e-05
	11	10	5.0e-07	1e-06

Table 4 continued

Data Rate (Mbps)	Eb/No Values [dB]	Corresponding SNR Values [dB]	Expected BER	
			MIN	MAX
9 (BPSK)	0	-2	1e-01	1e-01
	4.5	2	1e-02	1e-02
	7	4	1e-03	1e-03
	9	6	1e-04	1e-04
	10	7	1e-05	1e-05
	11	9	1e-06	1e-06
6 (BPSK)	0	-4	1e-01	1e-01
	4.5	0	1e-02	1e-02
	7	3	1e-03	1e-03
	9	5	1e-04	1e-04
	10	6	1e-05	1e-05
	11	7	1e-06	1e-06

The testbench will send packets through the system until 20 Genie packet errors are counted, with a minimum of 100 packets received. Smaller BER values require more packets to be sent because the SNR is very good and fewer errors occur. When offsets are applied, the Genie is expected to perform worse than the Tracking Methods, in which case 100 packets will be sent through, regardless. This restriction was enforced because every Genie packet has an error when offsets are applied, so rather than cut out at 20 packets the test is forced to run until 100 packets have been received.

In the IEEE 802.11a specifications, a minimum of 20 packets are required to be sent through the system to count the errors in each packet [4]. Each packet is required to contain at least 16 OFDM symbols [4]. By running the system through 100 packets instead of only 20 packets, the confidence in the statistics is higher.

4.2.3.2 PER

According to IEEE 802.11a standards [4], the Packet Error Rate (PER) of the system must be less than 10% for 1000 bytes of data. This is the final performance test that must be checked. The expected results are different than for the BER tests since only one bit error will cause an entire packet to be in error. As well, the longer the packet, the more errors expected within the packet, which is justification for restricting the packet length to 1000 bytes.

4.3 Results

4.3.1 Expected BER Curves

The following curves in Figure 4.3 were generated from the data in Table 4, using the minimum value of BER expected. A perfect system without multipath and frequency offsets would produce these results. Figure 4.4 shows the ideal BER curves with the maximum BER expected values, which also matches the symbol error rate values.

Figure 4.3: Ideal BER curves with lowest BER expected

Figure 4.4: Ideal BER curves with highest BER expected

4.3.2 Simulation time

The length of time for simulation is based on the number of tests and the expected length of time taken to run each test. The tests were run on a PC with an 800MHz CPU and 262MB of RAM. 32 ACQFO tests, 80 RFO tests and 14 Noise/MP tests were run, giving a total of 126 tests. However, each test was run with 4 data rates, so 504 tests were run overall. Each test took approximately an hour and a half to complete, giving a total running time of 756 hours or 31.5 days straight. This does not include time for analysis or for system confirmation of operation and design.

4.3.3 54Mbps results

The following figures show test results at a data rate of 54 Mbps in the face of noise and/or multipath. In Figure 4.4, only noise is applied and all curves should match the ideal curves in Figure 4.3. However, only the Genie approximates the ideal curve.

Method #1, 2 and 3 are further away, indicating overcompensation – a correction without cause. Notice the caption at the top of the figure indicates the parameter settings for the results, 0 means the parameter was off, 1 indicates on. Figure 4.5 shows BER results with multipath, but no noise. Since there is no noise, the BER should be quite low.

However, with MP applied, the BER is worse than in the previous figure. This could be due to less accurate acquisition since the preamble does not have a cyclic prefix and will be affected more by ISI. Notice that whenever noise is *not* applied, the SNR will read 30dB - the internal notation for specifying no noise. Finally, by combining noise and MP, the results degrade even further, as shown in Figure 4.7.

Figure 4.5: TEST: Noise, no MP, no RFO

Figure 4.6: TEST: no Noise, MP, no RFO

Figure 4.7: TEST: Noise, MP, no RFO

Figure 4.8 shows BER results with a forced residual FO (or Tracking Frequency Offset - RFO) applied. The RFO values used for testing were: [-17 -13 -7 -4 -0.5 +0.5 +4 +7 +13 +17] kHz. 100 packets were sent through each of the RFO values. A comparison of the two figures indicates that adding multipath slightly degrades the BER.

(a)

(b)

Figure 4.8: a) TEST: Noise, no MP, RFO b) TEST: Noise, MP, RFO

Table 5 shows the average residual FO estimation error and the average standard deviation of the RFO estimation error for each method over approximately 200000 OFDM symbols. In each case, MP causes the average standard deviation to increase. As well, both Method #1 and Method #3 have a high average standard deviation, which could explain their inferior BER results and indicate poor correction capabilities for these two methods. Conversely, the universal low average estimation error proves that detection of RFO is good for all 3 methods.

Table 5: Average RFO estimation error and Average value of Standard Deviation of RFO est err

	Average Est Error [Hz]	Average Std [Hz]
RFO, Noise		
Method #1	46.76	1485
Method #2	13.48	254
Method #3	13.81	1687
RFO, Noise, MP		
Method #1	14.56	1875
Method #2	-4.93	316
Method #3	2.7	2097

Table 6 summarizes the BER results for each test at an SNR of 25dB. Overall, when multipath is added, the BER values get worse. When a RFO is applied with noise and multipath, Method #2 is the only result that remains on the same order of magnitude as without the RFO. As well, Method #2 has the best results in all cases.

Table 6: Summary of BER values at SNR = 25dB for each test and each data stream

Test	Genie	Method #1	Method #2	Method #3
Noise	5.4e-6	9.2e-3	9.4e-5	7.2e-4
Noise, MP	1.8e-2	5.3e-2	2.1e-2	2.5e-2
Noise, RFO	4.6e-1	9.6e-2	2.2e-3	3.9e-2
Noise, MP, RFO	4.7e-1	1.7e-1	4.9e-2	1.0e-1

Figure 4.9 is a plot of the standard deviation of the RFO estimation error for each packet versus the applied RFO. As the applied RFO increases, the standard deviation of the RFO estimation error increases as well. This change is significant for methods #1 and #3. The best tracking performance occurs when the applied residual FO is between +/- 4000Hz.

Figure 4.9: Standard Deviation of the RFO estimation error versus the applied RFO

Finally, Figure 4.10 shows that the BER curves are much better when the TFO is restricted to 500Hz and 4000Hz, as compared to Figure 4.8b where the BER curve was generated using RFO tests varying from -17Hz to 17kHz.

(a)

(b)

Figure 4.10: BER curves for (a) RFO = 500 and (b) 4000Hz with MP

Several conclusions can be drawn from the above results. When a RFO is *not* applied, the Genie becomes the baseline for what is expected, however, none of the methods reach the baseline, indicating all are overcompensating. When a RFO is applied, the Genie has a BER of 0.5, showing that any method is better than nothing. Finally, multipath still has an effect on the system. In each case, when MP was added the results got slightly worse. All of the above results indicate that Method #2 is best.

4.3.4 A Comparison of Different Data Rates

All tests were performed with data rates of 54Mbps, 36Mbps, 18Mbps and 9Mbps. The higher data rates use a larger constellation (64-QAM) and are more sensitive to frequency offsets and constellation rotations. The lower data rates are tested with a lower SNR and have more noise to overcome. Each situation has pros and cons that can be seen by comparing the different data rates.

Overall, as the data rate DECREASES:

- The BER curves remain approximately the same except when both multipath and noise occur, in which case, the BER values get worse as the data rate decreases (Table 7).
- The average and standard deviation of the Acquisition FO (ACQFO) estimation error gets worse (the error increases, causing the residual FO to increase) (Table 8).
- The standard deviation of the RFO estimation error increases for all three Tracking Frequency Offset (TFO) methods (Table 8).
- The spread of the standard deviation and average RFO estimation error decreases (Figure 4.11 and Figure 4.12). These figures also show the best results are for $n = 400$ to 600 , which corresponds to an applied RFO of ± 500 Hz.
- The tracking timing offset average error, standard deviation and spread increases (Figure 4.13 and Figure 4.14).

Table 7: BER for 2 tests, 3 methods, 4 data rates

Data Rate	SNR (dB)	1 st Test: Noise			2 nd Test: Noise, MP		
		Method #1	Method #2	Method #3	Method #1	Method #2	Method #3
54 Mbps	25	9.2e-3	9.4e-5	7.2e-4	5.3e-2	2.1e-2	2.5e-2
36 Mbps	19	1.8e-2	2.6e-4	1.7e-3	8.4e-2	4.1e-2	3.4e-2
18 Mbps	12	5.6e-2	5.3e-3	1.0e-2	1.5e-1	1.1e-1	1.0e-1
9 Mbps	10	6.4e-3	4.7e-4	6.1e-4	7.8e-2	1.5e-1	1.5e-1

Table 8: Summary of 7 Tests, 4 Data Rates with Average and Std ACQFO and RFO results

Test Number	Test	Data Rate	SNR	Avg ACQFO Est Err	Std of ACQFO Est Err	Avg RFO Est Err #1	Avg RFO Est Err #2	Avg RFO Est Err #3	Avg STD of RFO Est Err #1	Avg of STD of RFO Est Err #2	Avg of STD of RFO Est Err #3
1	Noise	54	25	-1.9	146.3	2.8	3.8	3.9	1368.0	207.5	1377.5
2		36	19	11.9	857.6	-11.1	5.6	-2.6	2721.6	351.7	2741.4
3		18	12	-236.0	1955.2	14.6	25.5	14.8	6294.9	601.5	6381.4
4		9	10	-32.6	2810.1	-10.4	-5.6	-14.4	8152.7	602.5	8214.9
5	MP	54	N/A	0.19	4.7	-1.3	9.20E-04	-0.0012	29.02	0.0083	0.098
6		36	N/A	-0.075	6.8	-0.094	0.0013	-0.0024	14.4	0.0093	0.11
7		18	N/A	0.094	3.4	-0.13	0.015	-0.71	8.4	0.044	1.3
8		9	N/A	-1.1	39.9	0.066	-7.68E-04	-0.0022	10.6	0.0083	0.28
9	RFO	54	N/A	0	0	-0.016	-5.6	-4.4	375.7	138.9	947.1
10		36	N/A	0	0	-7.5	0.82	3.6	278.9	130.7	952.9
11		18	N/A	0	0	0.56	2.1	1.2	255.3	85.5	952.2
12		9	N/A	0	0	-5.6	-2.0	-0.23	251.7	86.4	980.4
13	Noise, MP	54	25	13.6	110.3	-101.1	-109.4	-107.1	1710.4	221.7	1729.5
14		36	19	-43.6	1045.0	-46.1	-100.2	-12.2	3681.8	451.6	3627.5
15		18	12	335.4	2147.2	27.2	-47.0	-49.7	8103.2	856.3	8016.6
16		9	10	411.3	2883.4	75.6	-1.9	30.4	10582.0	816.5	10345.0
17	Noise, RFO	54	25	0.56	170.5	46.7	13.5	13.8	1485.5	253.8	1686.9
18		36	19	11.8	872.0	43.0	4.0	0.17	2902.0	399.0	2991.0
19		18	12	-113.7	2097.7	0.62	13.4	2.6	6401.1	606.2	6456.9
20		9	10	-113.3	2619.3	16.4	16.1	9.7	8163.9	581.3	8228.0
21	MP, RFO	54	N/A	0.36	8.4	0.19	3.3	7.5	403.9	140.6	955.2
22		36	N/A	0.37	8.3	-14.2	3.1	5.0	333.1	132.1	969.0
23		18	N/A	0.12	4.9	9.5	-0.24	0.71	263.5	87.1	969.4
24		9	N/A	-0.47	34.1	10.6	-0.76	0.69	252.5	82.4	1001.3
25	Noise, MP, RFO	54	25	0.85	257.2	14.5	-4.9	2.7	1875.5	315.9	2097.1
26		36	19	10.7	929.9	-14.2	-13.7	1.3	3658.0	490.4	3726.9
27		18	12	-11.8	2379.8	-11.7	-20.1	-19	8060.9	840.4	8021.5
28		9	10	11.8	2919.7	101.1	27.2	20.5	10313.0	724.8	10136.0

Figure 4.11 (a)

Figure 4.11 (b)

Figure 4.11 (c)

(d)

Figure 4.11: Average RFO Est Error (a) 54Mbps (b) 36Mbps (c) 18bps (d) 9Mbps

Figure 4.12 (a)

Figure 4.12 (b)

Figure 4.12 (c)

(d)

Figure 4.12: Std Dev of RFO est error for duration of run (a) 54Mbps (b) 36Mbps (c) 18Mbps

(d) 9Mbps

Figure 4.13 (a)

Figure 4.13 (b)

Figure 4.13 (c)

(d)

Figure 4.13: Average Sampling Offset est error for duration of run (a) 54Mbps (b) 36Mbps
(c) 18Mbps (d) 9Mbps

Figure 4.14 (a)

Figure 4.14 (b)

Figure 4.14 (c)

(d)

**Figure 4.14: Std Dev of Sampling Offset est error for duration of run (a) 54Mbps (b) 36Mbps
(c) 18Mbps (d) 9Mbps**

4.3.5 Limitations

The limitations of the system were found by applying increasingly higher offsets until the decoder could no longer accurately determine the packet length and data rate used.

Applying multipath and noise at the best SNR test value was used to generate the results for each data rate. Packets of 1000 bytes were sent until the system generated a BER of 0.5.

The first offset tested was the acquisition frequency offset. Table 9 shows the maximum ACQFO value that the system could handle before a BER of 0.5 was produced.

According to Section 2.4.1, the maximum frequency offset that should be detectable is 156250 Hz. The numbers in Table 9 fall short of this figure because the channel includes noise and multipath. Without channel effects, an offset of 156250 Hz *is* detectable.

Also, 9Mbps has an extremely low tolerance to frequency offsets. This low tolerance indicates the acquisition synchronization begins to fail when the noise increases above a certain level.

Table 9: Maximum tolerable ACQFO for each data rate

Data Rate	Max ACQFO for accurate decoding
54 Mbps	114 kHz
36 Mbps	119 kHz
18 Mbps	119 kHz
9 Mbps	22 kHz

The next offset tested was the residual frequency offset. All four data rates could reliably handle a residual frequency offset up to 30kHz. Above 30kHz, the decoding became less reliable and the results significantly poorer.

Finally, the system was tested with sampling offsets of 12.5ns, 25ns, 37.5ns and 50ns.

According to Section 2.4.2, a sampling offset of 51.6ns should be detectable. For applied offsets of 12.5ns and 25ns the results were consistently reliable – these offsets are detectable. For lower data rates and larger applied offsets, the sampling offset estimates became less accurate.

4.3.6 Scatterplot Results

Scatterplots can be used to show the rotation or smearing of a constellation. The following plots were generated with a data rate of 54Mbps. The tests applied different combinations of noise, multipath and frequency offsets. The most interesting results are shown below.

Figure 4.15: Scatterplot of transmitted data

Figure 4.15 shows the scatterplot of the data that was transmitted. The following figures show what happens when different channel effects are applied.

(a)

(b)

Figure 4.16: Scatterplots of results with noise and MP, but no offsets

(a) Genie (b) Method #1,2,3

Figure 4.16 shows that with noise and multipath, the Genie looks about the same as the other 3 methods. As well, the 3 methods are so similar they can be shown on one plot, Figure 4.15b. However, once a residual frequency offset (or TFO) is applied, the constellation of the Genie rotates, as seen below.

Figure 4.17 (a)

(b)

Figure 4.17: Scatterplots of results with no Noise, no MP and a residual FO of 1000 Hz

(a) Genie (b) Method #1,2,3

With a RFO and MP, all 4 constellations get a bit fuzzier, as shown in Figure 4.18. As well, the three methods show small signs of constellation rotations.

(a)

(b)

Figure 4.18: Scatterplots of results with no Noise, MP and a residual FO of 1000 Hz

(a) Genie (b) Method #1,2,3

(a)

(b)

Figure 4.19: Scatterplots of results with Noise, no MP and a residual FO of 1000 Hz

(a) Genie (b) Method #1,2,3

With noise and RFO, the Genie becomes very messy, as shown in Figure 4.19, while the three methods seem to have a constellation rotation.

Finally, with noise, MP and RFO applied (Figure 4.20), the Genie is completely indiscernible, while Methods #1, #2 and #3 have similar appearances to each other and the constellation can still be seen.

Figure 4.20 (a)

Figure 4.20 (b)

Figure 4.20 (c)

(d)

Figure 4.20: Scatterplots of results with Noise, MP and a residual FO of 1000 Hz

(a) Genie (b) Method #1 (c) Method #2 (d) Method #3

Up until this point, the three methods have had very similar scatterplots and could be combined into one. An interesting thing happens when a higher RFO is applied. Figure 4.21 is the result of a 7000 Hz RFO. Methods #1 and #3 have a visible constellation rotation. This explains why, as the frequency increases, Method #1 and #3 deteriorate quickly while Method #2 remains stable and reliable longer.

Figure 4.21 (a)

Figure 4.21 (b)

Figure 4.21 (c)

(d)

Figure 4.21: Scatterplots of results with Noise, MP and a residual FO of 7000 Hz

(a) Genie (b) Method #1 (c) Method #2 (d) Method #3

4.3.7 ACQFO Results

This section describes the results that came from applying an acquisition frequency offset without a forced residual offset. Overall, the acquisition block provides a very good initial estimate of the frequency offset. Table 10 and Table 11 in Appendix A show that the average RFO doesn't exceed 100 hertz. Taking into account the standard deviation, the maximum residual frequency offset should never be more than a few thousand hertz, which is easily manageable by the tracking block. Plots of the Standard deviation proved to be the most useful and are shown below.

Figure 4.22 (a)

Figure 4.22 (b)

Figure 4.22 (c)

(d)

Figure 4.22: (a) Noise and MP (b) MP (c) Noise (d) nothing

Interestingly, there is no relationship between a higher acquisition FO and more errors.

As well, applying multipath does not affect the results significantly, although multipath seemed to have an effect on every other test.

4.3.8 Packet Error Rate Results

According to IEEE 802.11a standards, the system must have a PER of 10% for a 1000 byte packet at a given minimum sensitivity. The corresponding signal to noise ratios tested for each data rate are:

Data Rate	54Mbps	36Mbps	18Mbps	9Mbps
SNR	26dB	20dB	12dB	9dB

The three tracking methods were tested with and without noise and multipath. As well, an ACQFO of 22kHz and a forced RFO of 1000Hz were applied separately to determine the effects on the systems.

The results were not surprising. When noise and multipath were applied, the PER was above 10% for all of the methods, regardless of frequency offsets applied. Without multipath the following three plots are generated:

Figure 4.23 (a)

Figure 4.23 (b)

(c)

Figure 4.23: PER curves for each method and 4 data rates

(a) No offsets (b) RFO = 1kHz (c) ACQFO = 22kHz

These curves show that Method #2 satisfies the PER requirements for 54Mbps and 36Mbps under all conditions. Method #3 sometimes satisfies the requirements and Method #1 never satisfies the requirements. The lower signal to noise ratios also have more difficulty satisfying the requirement.

5 Conclusion

5.1 Discussion of Results

Overall, Method #2 provided the best results. The difference between the architecture of the methods can be used to explain the difference in results. Method #1 has a complicated feedback routine, which delays and degrades correction abilities. Also, Method #1 corrects the results, and then uses the corrected data to detect an offset. This has potential for the estimation results to oscillate above and below the correct value without settling quickly. Method #3 uses the best concepts from both architectures - the feedforward of Method #2 and the NCO of Method #1, yet Method #3 still has problems. Since the feedforward design of Method #2 works very well, the undesirable results must stem from the feedback to the time domain and the NCO. When the NCO corrects the data symbols, the phase of the NCO adds another offset unless the phase is perfectly synchronized with the existing carrier frequency offset.

All methods were equally capable of detecting frequency offsets, as shown by the low average residual frequency offset estimation error. The difference between methods lies in their varying abilities to correct the offset. As well, the standard deviation of the RFO estimation error for Method #2 was always lower than for the other methods. A high standard deviation indicates that per packet, Method #1 and #3 will have more extreme negative and positive estimation errors. However, the negative and positive estimation errors balance out to give a low average estimation error.

Although even Method #2 does not provide stellar results, it is still better than nothing. Without any correction, the Genie has a BER value of 0.5 in the presence of frequency offsets, while Method #2 improves on this dramatically. Alternatively, without frequency offsets, Method #2 and the Genie should have the same BER curves, yet the Genie was consistently better. Even though Method #2 is much better in the face of a frequency offset, Method #2 also overcompensates when offsets are not applied.

Multipath appears to be detrimental in almost every test case. Although OFDM is supposed to be better than a single carrier system with Multipath, the OFDM system is still slightly affected.

Finally, Method #2 is capable of accurately detecting and correcting residual frequency offsets up to 4000Hz, which is more than sufficient. With an acquisition frequency offset

of 112kHz, the average residual frequency offsets were around 1000Hz, plus or minus a few hundred and were easily detected and corrected by Method #2.

5.2 Future Work

There are many suggestions for future work in this area. For starters, a comparison could be made between the OFDM architecture and a single carrier system bearing the same channel conditions. A further architecture enhancement could involve determining an optimum sampling offset correction method, using the sampling offset detection method suggested in this thesis. As well, the location of the equalizer could be analysed.

Currently, the equalizer operates on data that has been corrected for both acquisition and tracking frequency offsets. However, the equalizer could be moved before the tracking block (Stage III) so that the data is equalized before correction. Whether or not the order is important is not clear and has not been tested.

Other possible system tests include extracting bit errors before the Viterbi. This allows for a hard-decision detection to occur and eliminates any possibility of covering up errors in the tracking methods. Currently the system is floating point. A fixed point simulation could be performed, but this requires more code modification. To turn a floating point simulation into a fixed point simulation requires altering all additions, multiplications and divisions for buswidths and precision.

6 Appendix A

Table 10: Average Acquisition Estimation Error and Standard Deviation, MP applied

*Rate	*SNR	*ACQFO	*MP	*Noise	*ACQFO Est Err	*STD of ACQFO Est Err
54	25	-22000	1	1	-66.3281	459.549
				0	0.007049	0.006243
54	25	7000	1	1	-67.9374	478.261
				0	2.76547	26.7348
54	25	22000	1	1	-58.9405	379.177
				0	1.29678	10.6913
54	25	75000	1	1	12.004	377.244
				0	2.6843	39.8103
54	25	112000	1	1	-7.04242	417.791
				0	-15.2726	134.232
36	19	-22000	1	1	-121.689	819.414
				0	0.007049	0.006243
36	19	7000	1	1	-236.727	1169
				0	2.44214	26.8785
36	19	22000	1	1	-2.25152	1022.96
				0	1.16547	9.42142
36	19	75000	1	1	-16.2353	1042
				0	0.227666	5.49268
36	19	112000	1	1	-16.9817	924.575
				0	0.007791	0.007204
18	12	-22000	1	1	-778.722	2285.26
				0	0.007049	0.006243
18	12	7000	1	1	249.228	2087.93
				0	2.48154	26.8404
18	12	22000	1	1	412.517	2384.2
				0	0.246335	2.36094
18	12	75000	1	1	-151.814	2339.53
				0	0.520266	9.80832
18	12	112000	1	1	322.919	2306.87
				0	0.557851	5.45641
9	10	-22000	1	1	-220.231	2734.94
				0	0.007204	0.006075
9	10	7000	1	1	103.068	2581.92
				0	2.51589	26.8119
9	10	22000	1	1	440.759	2734.18
				0	1.59977	13.6482

Table 11: Average Acquisition Estimation Error and Standard Deviation, no MP applied

*Rate	*SNR	*ACQFO	*MP	*Noise	*ACQFO Est Err	*STD of ACQFO Est Err
54	25	-22000	0	1	-61.7049	327.069
				0	0.00876	8.72E-18
54	25	7000	0	1	-12.5598	401.916
				0	0.00886	1.22E-17
54	25	22000	0	1	2.26889	396.874
				0	0.00891	1.74E-17
54	25	75000	0	1	51.8165	303.631
				0	0.009	6.97E-18
54	25	112000	0	1	-30.0255	306.125
				0	0.00901	1.39E-17
36	19	-22000	0	1	-177.317	906.149
				0	0.00876	8.72E-18
36	19	7000	0	1	159.71	842.99
				0	0.00886	1.22E-17
36	19	22000	0	1	-57.0905	895.108
				0	0.00891	1.74E-17
36	19	75000	0	1	-137.157	1020.04
				0	0.009	6.97E-18
36	19	112000	0	1	30.9574	913.956
				0	0.00901	1.39E-17
18	12	-22000	0	1	-5.44717	1883.82
				0	0.00876	8.72E-18
18	12	7000	0	1	-53.4263	2070.21
				0	0.00886	1.22E-17
18	12	22000	0	1	419.091	2137.56
				0	0.00891	1.74E-17
18	12	75000	0	1	-542.629	2223.72
				0	0.009	6.97E-18
18	12	112000	0	1	-50.9394	1933.29
				0	0.00901	1.39E-17
9	10	-22000	0	1	-133.689	2527.46
				0	0.00876	8.72E-18
9	10	7000	0	1	38.0636	2523.46
				0	0.00886	1.22E-17
9	10	22000	0	1	107.357	2876.12
				0	0.00891	1.74E-17

7 Appendix B

```
% Main.m
```

```
% This function pulls together all the blocks of code from transmitter to receiver including acquisition and
% tracking and channel application.
```

```
% Inputs: param = struct containing following fields:
```

```
%     SNR = desired SNR
%     offset_f = frequency offset applied to incoming data
%     offset_t = number of 20MHz samples to shorten first symbol by
%     N_shift = number of 80MHz samples to shift entire packet by to test tracking ...
%             therefore, one sample is equivalent to 1/4 20MHz sample or 1 80MHz sample
%     offset_packet = number of 20MHz samples to shift entire packet over by so it doesn't
%                   begin at t=0
%     offset_track_f = frequency offset to apply to data before tracking - will occur after
%                   acquisition has determined frequency offset
%     os = oversample: 1 if already done, 0 if must do in here
%     rem_noise = 1 if noise is to be removed (ie not added)
%     mp = 1 if Multipath is to be added
%     trueTXRATE = desired transmission rate (54, 48, 36, 24, 18, 12, 9 or 6)Mbps
%     sfo = 1 if sampling frequency offset is to be added
%     cpo = value of carrier phase offset to apply
%     GENERATEDBYTES = number of random bytes to generate
```

```
% Outputs: out = struct containing:
```

```
%     stats = struct with acquisition statistics
%     stats_tr = struct with tracking statistics
%     nBER = struct with BER values
```

```
% Written by: Shauna Rauhala, Oct 2001. Last Modified: June 2002
```

```
function out = main_rx(param)
```

```
% start timer - to see how long it takes to run through code
```

```
tic
```

```
% set up file names
```

```
Run = [num2str(param.SNR) '_' num2str(param.trueTXRATE) '_' num2str(param.offset_f) '_'
num2str(param.offset_t) ...
      '_' num2str(param.N_shift) '_' num2str(param.offset_track_f) '_' ...
      num2str(param.sfo) '_' num2str(param.cpo) '_' num2str(param.mp) '.txt'];
```

```
% determine location of stored results: memory_file = data that generated stats
```

```
memory_file = ['results_memory_' Run];
```

```
dataDir = [pwd '\'];
```

```
param.fid_memory=fopen([dataDir memory_file], 'wt');
```

```
% store initial parameters
```

```
StoreParam(param);
```

```
% ***** Initialize testing variables
```

```
% to suppress all plots (especially for large number of symbols), set fig=0
```

```
t.fig = 0;
```



```

% to test a hiperlan input for detection
t.test_hip = 0; % set to 1 to test this

% ***** Initialize internal variables
% set up matlab quantization
global FIXED_POINT
FIXED_POINT = 0; % FP = 0 to turn it off

% Initialize quantizer - no integer portion
global Q
if FIXED_POINT
    Q=quantizer('fixed',[12 11],'saturate','ceil');
end

% this can be either hipA, hipB or ieee .... must set when setting data_file
param.preamble_type = 'ieee';

% determine how many oversampled samples the packet must be shifted over by
param.offset_packet_os=param.offset_packet*4;

% initialize amount to interpolate - this is # of samples bt points.
param.interpolate = 3;

% ***** extract some data if necessary
% hiperlan input
if t.test_hip
    data_file_hip = input_hiperlan_BCCH;
    data_file2 = input_long_4data_GI;
    data_file =[data_file_hip data_file2];
else
    % extract data once, unless using Jeff_randdata and running testbenchSIM3
    % each time because then data will be different
    data_file = input_create_randdata(param.trueTXRATE, param.GENERATEDBYTES);
end
param.data_file = FP(data_file);

% ***** Apply noise, offsets, etc.
[process_data, t.fig, perfect_data_in] = DataSet(param,t);

% ***** Stage I
[param.start_input, t.fig, buffer] = Stage_I(process_data, t.fig);

% ***** Stage II
[o,param] = Stage_II(buffer, param, t);

% warning will be generated if less than 2 long peaks were detected - discard packet
% if that occurs and go onto next loop
if o.warning
    out.warning = 1;
    return;
else
    out.warning = 0;
end

% if hiperlan was detected continue ----> skip rest of processing and
% go onto next loop of "for i = 1:Nruns"

```

```

if o.hiperlan
    temp_msg = ['Hiperlan preamble detected in packet ' int2str(i)];
    disp(temp_msg)
    out.warning = 1;
    return;
end
t.fig = o.fig;

% ***** Stage II - save results to files/matrices
s = SaveStats(o,param,t)

% ***** prepare for Tracking
[param, data.symbols, data.ideal, cmprdata] = TrackPrep(o,param,t,buffer,perfect_data_in);
t.fig = cmprdata.fig;

% ***** Tracking Loop
% Prepare for use of equalizer on pilots
data.long_fft = fft(o.long,64);
data = Prep_equalize(data, 1);
cmprdata.method = data.method;
cmprdata.mu = 0;
cmprdata.bpsk = data.bpsk;
cmprdata.initial = data.initial;

% ***** process each symbol individually.
for x = 1:size(data.symbols, 2)

    % ***** process ideal symbols
    ideal_data(:,x) = Process_ideal(data.ideal(:,x).');
    no_offset_data(:,x) = Process_ideal(data.symbols(:,x).');
    % *****

    % TEST: 3 different types of FO correction methods:
    % 1. Correct next symbol with previous sym results (time domain corr)
    % 2. Correct current sym with it's own results (freq domain corr)
    % 3. Correct current symbol with it's own results (time domain corr)
    cmprdata.x = x;
    cmprdata.symbols = data.symbols(:,x).';

    cmprdata = TFO_corr1(cmprdata, param);
    % initial structure of method 2 and 3 is same ... call initial structure first
    % and pass parameters to corr2 and corr3
    cmprdata = TFO_2_and_3(cmprdata, param);

    cmprdata = TFO_corr2(cmprdata, param);
    cmprdata = TFO_corr3(cmprdata, param);
end % end tracking

if length(cmprdata.angle_pilots_1) > 10
    disp('cmprdata.angle_pilots_1')
    cmprdata.angle_pilots_1(1:10,:)
    disp('cmprdata.angle_pilots_2_3')
    cmprdata.angle_pilots_2_3(1:10,:)
else

```

```

disp('cmprdata.angle_pilots_1')
cmprdata.angle_pilots_1(1:end,:);
disp('cmprdata.angle_pilots_2_3')
cmprdata.angle_pilots_2_3(1:end,:);
end

% ***** call equalizer ... if not doing acquisition testing
data.mu = 0;%0.0001%cmprdata.mu;
% Genie Aid
data.in = no_offset_data;
data = Prep_equalize(data,0);
y_genie = clms(data); % equalizer
cmprdata.genie = y_genie.eq;

cmprdata.ideal = ideal_data;
zero_pilots = zeros(1,size(cmprdata.ideal,2));
zero_null = zeros(11,size(cmprdata.ideal,2));
cmprdata.ideal = [zero_pilots; cmprdata.ideal(1:6,:); zero_pilots; cmprdata.ideal(7:19,:); zero_pilots;
cmprdata.ideal(20:24,:); ...
zero_null; cmprdata.ideal(25:29,:); zero_pilots; cmprdata.ideal(30:42,:); zero_pilots;
cmprdata.ideal(43:48,:)].';

cmprdata.genie = [zero_pilots; cmprdata.genie(1:6,:); zero_pilots; cmprdata.genie(7:19,:); zero_pilots;
cmprdata.genie(20:24,:); ...
zero_null; cmprdata.genie(25:29,:); zero_pilots; cmprdata.genie(30:42,:); zero_pilots;
cmprdata.genie(43:48,:)].';

% Method 1
data.in = cmprdata.fft_corr_1.';
data = Prep_equalize(data,0);
y1 = clms(data); % equalizer
zero_pilots = zeros(1,size(y1.eq,2));
zero_null = zeros(11,size(y1.eq,2));
y1.eq = [zero_pilots; y1.eq(1:6,:); zero_pilots; y1.eq(7:19,:); zero_pilots; y1.eq(20:24,:); zero_null;
y1.eq(25:29,:); ...
zero_pilots; y1.eq(30:42,:); zero_pilots; y1.eq(43:48,:)].';

% Method 2
data.in = cmprdata.fft_corr_2.';
data = Prep_equalize(data,0);
y2 = clms(data); % equalizer
zero_pilots = zeros(1,size(y2.eq,2));
zero_null = zeros(11,size(y2.eq,2));
y2.eq = [zero_pilots; y2.eq(1:6,:); zero_pilots; y2.eq(7:19,:); zero_pilots; y2.eq(20:24,:); zero_null;
y2.eq(25:29,:); ...
zero_pilots; y2.eq(30:42,:); zero_pilots; y2.eq(43:48,:)].';

% Method 3
data.in = cmprdata.fft_corr_3.';
data = Prep_equalize(data,0);
y3 = clms(data); % equalizer
zero_pilots = zeros(1,size(y3.eq,2));
zero_null = zeros(11,size(y3.eq,2));

```

```

y3.eq = [zero_pilots; y3.eq(1:6,:); zero_pilots; y3.eq(7:19,:); zero_pilots; y3.eq(20:24,:); zero_null;
y3.eq(25:29,:);...
        zero_pilots; y3.eq(30:42,:); zero_pilots; y3.eq(43:48,:)].';

```

```

cmprdata.fft_corr_1 = y1.eq;
cmprdata.fft_corr_2 = y2.eq;
cmprdata.fft_corr_3 = y3.eq;

```

```

out.nBER = BER(cmprdata,param.trueTXRATE);
if out.nBER.warning
    out.warning = 1;
    return;
end

```

```

fclose(param.fid_memory);

```

```

% Determine statistics
[out.stats out.stats_tr] = FinalStats(cmprdata, s, o, t, param, out.nBER);

```

```

% end timer
toc
return;

```

8 References

1. Al Petrick and Bob O'Hara, *IEEE 802.11 Handbook – A Designer's Companion*, New York: IEEE Press, 1999.
2. Andreas F. Molisch, *Wideband Wireless Digital Communications*, New Jersey: Prentice Hall, 2001.
3. Chien-Fang Hsu, Y.H. Huang and T. D. Chiueh, "Design of an OFDM receiver for High-speed Wireless LAN," *ISCAS 2001*, IEEE, 2001.
4. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band," *IEEE Std 802.11a-1999*, IEEE, 1999.
5. Alister Burr, *Modulation and Coding for Wireless Communications*; England: Prentice Hall, 2001.
6. Vafa Ghazi-Moghadam, "Modeling OFDM Effects in Baseband WLAN Systems," *Cadence Wireless Technical Webcast*, Created: Mar 28, 2002, Accessed: May 27th, 2002.
<http://seminar2.techonline.com/~cadence22/mar2802/>

7. John A. Bingham, *ADSL, VDSL, and Multicarrier Modulation*, New York: Wiley, 2000.
8. William Warner and C. Leung, "OFDM/FM Frame Synchronization for Mobile Radio data Communication," *IEEE Transactions on Vehicular Technology*, Vol. 42, No. 3; August 1993.
9. F. Tufvesson and O. Edfors, "Preamble-based Time and Frequency Sync for OFDM systems," *Design of Wireless Communication Systems - Issues on Synchronization, Channel Estimation and Multi-Carrier Systems*, Department of Applied Electronics, Lund University, Sweden, 2000.
10. Wolfgang Eberle et al, "A Digital 72Mb/s 64-QAM OFDM Transceiver for 5GHz Wireless LAN in 0.18um CMOS," *IEEE International Solid-State Circuits Conference, ISSCC 2001*.
11. J.J. Van de Beek, et al, "A time and Frequency Synchronization Scheme for Multiuser OFDM," *48th IEEE Vehicular Technology Conference, VTC'98*, May18-21, 1998.
12. John G. Proakis and Dimitris G. Manolakis, *Digital signal processing : principles, algorithms, and applications*, 3rd ed, New Jersey: Prentice Hall, 1996.

13. Ramjee Prasad and Richard van Nee, *OFDM for Wireless Multimedia Communication*, Boston: Artech House, 2000.
14. Simon Haykin, *Communications Systems*, 3rd edition, Canada: John Wiley and Sons, 1994.