Timing and Frequency Synchronization in OFDM Broadband Wireless Systems

by

Veeral S. Shah

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

Author ...

Department of Electrical Engineering and Computer Science

February 6, 2001

Certified by .....

George W. Pratt

Professor Emeritus of Electrical Engineering, M.I.T.

Thesis Supervisor

Certified by .....

David B. Ribner

6A Company Thesis Supervisor, Analog Devices, Inc.

Thesis Supervisor

Accepted by .....

Arthur C. Smith

Chairman, Department Committee on Graduate Theses

Acknowledgments

I would like to thank Dave Ribner and Sunder Kidambi of Analog Devices, who were tremendous sources of knowledge and a pleasure to work with. I am also very grateful to Prof. George Pratt, who offered me encouragement and guidance in completing this thesis. I would also like to thank Analog Devices for its participation in the 6A program, as it has afforded me the opportunity to work in the cutting-edge field of broadband communications. ADI is an excellent company, and I am grateful for having had the opportunity to work particularly with the members of the Broadband Wireless group.

Contents

Terms and Notation

Terms	Meaning
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
DDS	Direct Digital Synthesizer
DFT	Discrete Fourier Transform
DSP	Digital Signal Processor
FDM	Frequency Division Multiplexing
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FM	Frequency Modulation
FT	Fourier Transform
ICI	Intercarrier Interference
ISI	Intersymbol Interference
LAN	Local Area Network
LO	Local Oscillator
MAC	Media Access Control
OFDM	Orthogonal Frequency Division Multiplexing
PAM	Pulse Amplitude Modulation
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
SRRCF	Square Root-Raised Cosine Filter
TCM	Trellis coded Modulation
TPS	Transmitter Parameter Signalling
VSB	Vestigial Side Band

Terms and Notation

Notation	Meaning
N	Length of the FFT
v	Length of the cyclic prefix/extension
f	Frequency
T	Period
$r(t)$	Received signal
$s(t)$	Baseband Transmitted Signal
N_p	Number of Paths (in multipath model)
$\rho_n(t)$	Complex Attenuation of nth Path
$\tau_n(t)$	Delay of nth Path

Chapter 1

Introduction

Fixed broadband wireless networks are rapidly being developed and touted as a suitable alternative to wireline access methods such as DSL and cable modem. The Multichannel Multipoint Distribution System (MMDS) version of fixed wireless broadband operates in the 2.5 GHz to 2.686 GHz range and offers connections of up to 15 miles, whereas Local Multipoint Distribution Services (LMDS) operates in the 28 GHz range and operates over small ranges (1-4 miles). Data rates on LMDS (in the OC-1 to OC-12 range) are comparable to fiber, and will hence be used for connectivity to businesses, while MMDS rates are comparable to DSL and cable internet services for home broadband users.

Broadband wireless industry analysts at the Strategis group predict that the market will grow at a 418% pace over the next five years, with revenues reaching $3.4 billon in 2003 (compared to $11.2 million in 1999).

The MMDS band has emerged as the frequency band of choice for broadband access. In the presence of rain, the higher-frequency LMDS band experiences great signal attenuation (during medium-intensity rains, fades of greater than 20 dB are common over short distances [17]), and thus will not cover a large enough range to make it economically viable. Furthermore, the demand among broadband users for such ultra-high rate access is not robust enough. MMDS boasts both long-distance signal strength and suitable bandwidth for the broadband user.

1.0.1 Thesis Overview

This thesis aims to explore techniques in frequency and timing synchronization in an OFDM (orthogonal frequency division multiplexing) MMDS fixed broadband wireless system. Specifically, such systems transmit and receive over non-LOS (line-of-sight) channels, where there exist nonidealities

due to additive white Gaussian noise, phase noise due to imperfect synchronization between the transmitter and receiver oscillators, and multipath effects due to reflections in the channel.

Chapter two introduces the principles of OFDM and gives an overview of the features and important developments in communications that have made OFDM systems attractive from an implementation and performance standpoint. Beginning with an understanding of OFDM modulation and its precursors, the section progresses to cover the method by which OFDM signals are transmitted in wireless systems.

Chapter three introduces the channel and system defects that can reduce performance in such communications systems. Problems such as offset, multipath propagation, and various types of noise each are detrimental to the system performance in different ways, and this is explored.

Chapter four details the development of a simulation platform that helps gain visibility into the effects of these nonidealities. Using noise models and a simulation framework, the system allows for examination of channel effects vs. implementation parameters, followed by a discussion of the results.

Chapter five is concerned with the testing and comparison of numerous carrier and timing recovery algorithms. Many have focused on the problem of synchronization in OFDM systems, and this section of the thesis looks at the different classes of receivers, their performance, and their hardware implementation costs.

In order for a wireless modem to receive data and decode it correctly, that data has to be acquired properly. The tools necessary for this acquisition are a) a suitable channel estimation algorithm and b) suitable synchonization algorithms. The latter is the subject of the later portions of this thesis, which examimes the ability of various algorithms to perform the following two synchronizing tasks: the synchronization of framing/timing, and the synchronization of carrier/frequency in a broadband OFDM wireless modem.

Two particular topics important to the development of a broadband wireless modem will not be studied in this thesis:

1. *Antenna diversity* is the transmission and reception of the communication signal on more than one antenna, thus involving multiple datapaths to decode the signal. This availability of multiple copies of the signal serves to boost SNR, through techniques such as beamforming. It complicates the design of the receiver without greatly affecting the principles of the synchronization problem, and will thus remain outside the scope of this work, though useful in practice.

2. *Burst synchronization* is the problem of synchronization over a bursty channel, meaning that one cannot assume that data will be streaming into the receiver continuously. This requires either different algorithms or modifications to the ideal parameters for *continuous* or *back-to-back* synchronization, which is analyzed in this work.

Since most OFDM systems employ back-to back transmission in the downstream from the base station to the user and burst transmission in the upstream[23], this work primarily deals with synchronization in the downstream.

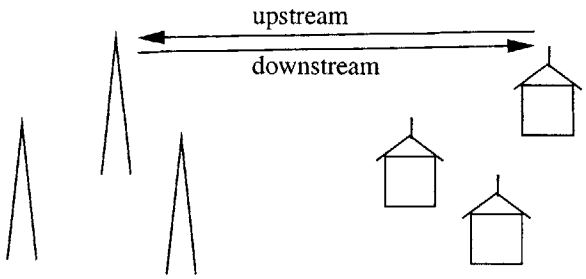


Figure 1.1: Upstream vs. Downstream Data Transmission in an MMDS Network

1.1 Opportunity

Analog Devices has decided to pursue the MMDS market. There are currently a number of bodies proposing different standards as to the parameters of broadband wireless system design. As a result, this work does not focus on a single standard, but rather looks at some of the most suitable and common implementations. Analog Devices is currently developing a physical layer chip that performs the forward error correction (FEC) and modulation/demodulation functions, as well as the medium access control (MAC) layer above it. MAC is a convergence layer— part of the data link control layer above the physical layer.

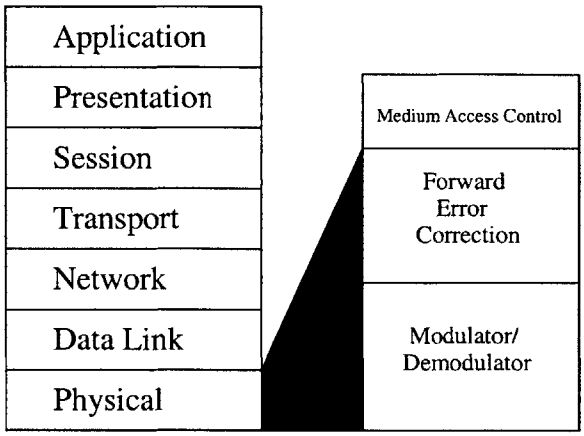


Figure 1.2: The OSI 7-Layer Model

1.2 Thesis Work

The project started out as being the development of a C platform, that was going to primarily be a systemic description of the entire physical layer broadband wireless modem operation. As time progressed, this model was to be ported into a C++ machine class system that could be ported directly into hardware description language. Along the way, it would incorporate numerous modules for timing recovery, carrier recovery, and channel estimation, and hence the entire system could be simulated, with swappable modular algorithms.

For a number of reasons, the project development diverged so that this idea of a singular platform diverged into three separate projects:

1. A C simulational model of channel studies with varying paramters and channel conditions
2. Numerous MATLAB simulations that allowed undstanding of different standards, timing recovery, carrier recovery, and channel estimation schemes
3. Fixed-point, C++ based, machine code with register, clock, and other hardware classes.

There were many reasons that made this arrangement more useful. The use of the C model allowed for a robust platform for testing carrier phase noise, white Gaussian noise, multipath, AM hum, and many other defects, as they exist before any algorithms to correct them. This helped the development team understand, for example, that phase noise could be modelled as simple additive white Gaussian noise due to the severe intercarrier interference it causes.

Separate MATLAB models were created because as the development effort proceeded, the method of modulation and the standards bodies influencing it changed many times— each time altering the class of possible solutions. MATLAB, with its in-depth signal processing, visualization, and data tools, proved to be the platform of choice for quickly understanding the different algorithms and their performance. Another engineer, Sunder Kidambi, led this effort while I developed the C model, and later I developed MATLAB models as well.

The C++ model, with its hardware-like declarations and code, was best for porting the model directly into a hardware design language (such as Verilog)— but because of its cycle-accurate, fixed-point, bit-true nature, it was too unwieldy to provide a good platform for quick initial testing of algorithms. Furthermore, since the algorithms would be mostly implemented in the DSP rather than in the datapath, there was no true benefit to coding them in a finalized hardware-like form so early on during the standardization and development process..

My work at Analog Devices constisted of all three of these projects; however, the work that will be described in this document concerns only the first two groupings. The hardware design of the demodulator does not fit into the scope of this work, and for purposes of intellectual property, will not be discussed here. In addition, details regarding the specific standard being used for the actual physical layer design will not be discussed, as the standard is confidential between ADI and the founding members of the consortium— Cisco, Broadcom, and Texas Instruments.

Below is the design flow of the project, for which the end goal was to evaluate channel characteristics, nonidealities, and suitable algorithms for carrier and timing synchronization of the broadband wireless chipset.

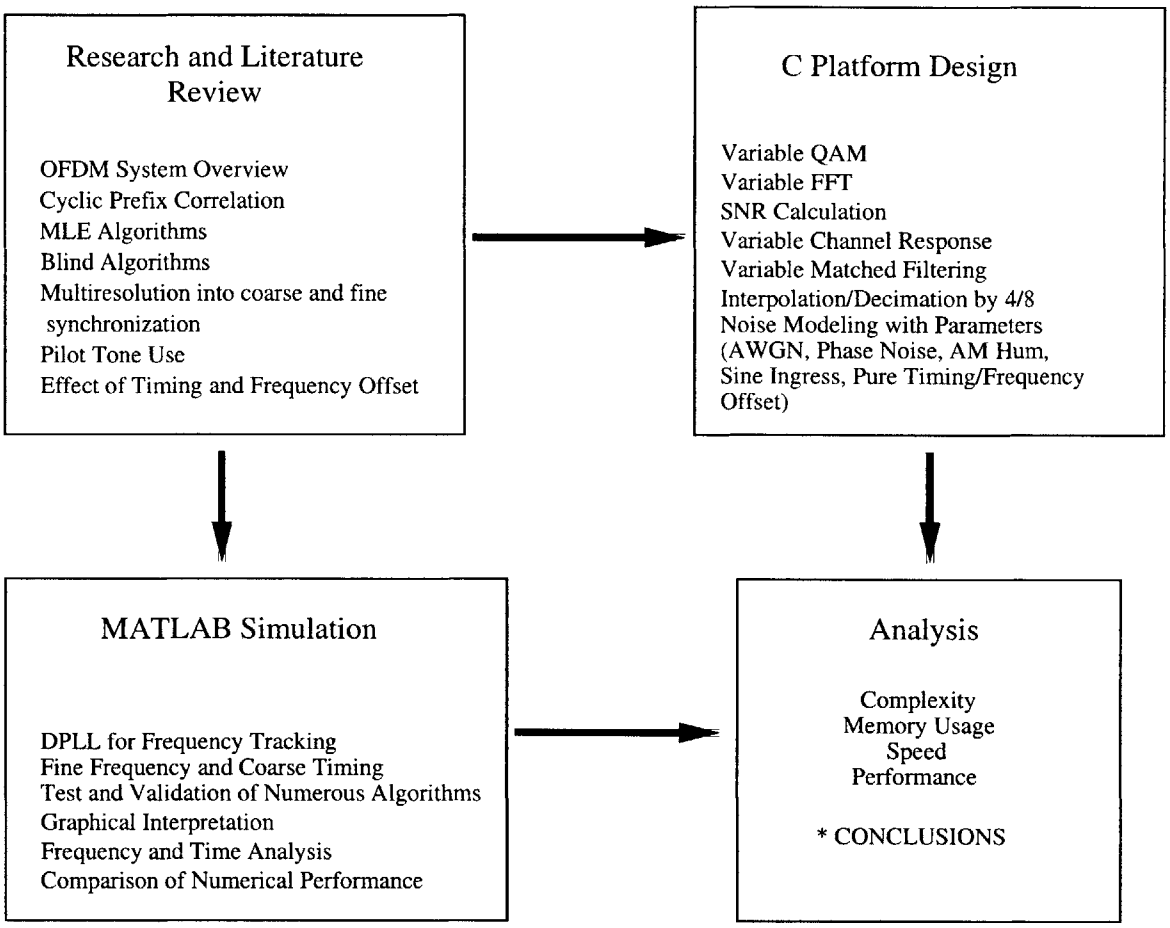


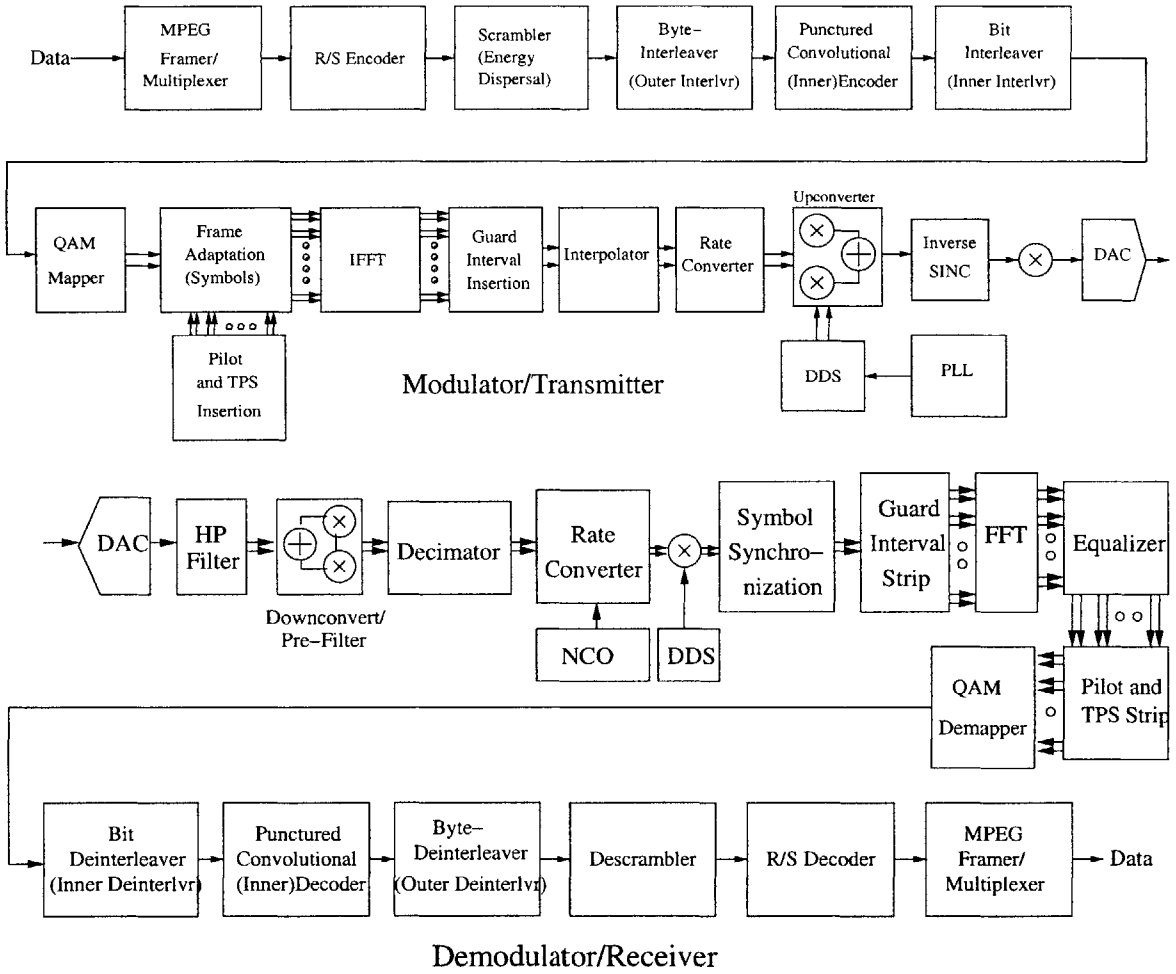
Figure 1.3: Project Design Flow

The OFDM system physical layer can be described broadly in two layers, which are pictured and mentioned below:

1. **FEC(forward error correction) encoding and decoding layer** - This includes, on the transmit side, the MPEG framer/multiplexer, scrambler (energy dispersal), R/S encoder (outer

coder), byte-interleaver (outer interleaver), punctured convolutional (inner) encoder, and the bit and symbol (inner) interleavers). The reverse of these blocks, along with error recovery algorithms, exists on the decoder (receiver) side. This part of the physical layer improves performance of the modem by decreasing the SNR necessary to reach a particular BER(bit error rate).

2. **Signal modulation and demodulation layer** - QAM mapping, frame adaptation(including TPS and pilot insertion), an IFFT block, cyclic prefix insertion, interpolation, rate conversion, upconversion, inverse sinc, and scaling on the modulation side. The demod includes the inverse blocks, but also some crucial algorithmic blocks that perform synchronization(both timing and frequency) and equalization.



Figures 1.4 and 1.5: Physical Layer (FEC and Mod/Demod) of Broadband Wireless Modem

The division of the physical layer into the FEC and modem layers is important for the study of sycnchronization. The QAM mapper in the structure above does not assume anything about the format of the serial bit stream that it is mapping, and likewise for the stream that is being demapped

by the QAM demapper in the receiver.

Hence none of the FEC functions have to be implemented to do signal-level timing recovery, carrier recovery, or channel estimation– a random bitstream is all that is needed to investigate channel phenomena and synchronization.

Chapter 2

Orthogonal Frequency Division Multiplexing: Principles

2.1 Types of Modulation: Motivation for OFDM

This section discusses some of the important digital transmission methods preceding OFDM and the reasons by which OFDM became the chosen modulation method for many communications networks.

2.1.1 Vestigial Sideband (VSB)

Vestigial sideband modulation is a technique still being used in many digital broadcasting systems, despite the disadvantages mentioned below. In VSB, the information to be transmitted is duplicated, residing in both sidebands. For one of the sidebands, some of the spectral components are greatly attenuated. The information transmitted by a modulated carrier wave lies in both the sidebands, and each sideband carries the same information. Since they are mirror images of each other, a duplication is present and more bandwidth than necessary is used.

It is possible to send a complete signal using one sideband only. However, it is not possible to filter one sideband cleanly off. Hence, suppression usually takes the form of a gradual roll-off of the unwanted portion. The choice is to have this occurring either in the wanted sideband or in the unwanted one. Neither of these alternatives is truly satisfactory. In the first case, LF frequency energy is lost if the roll-off occurs in the wanted sideband, while in the second case, incomplete suppression takes place and the bandwidth of the signal as a whole is still fairly large. In television transmission a compromise is sought, where the lower sideband is partially suppressed and because there is a vestige of it remaining, the scheme is known as vestigial sideband transmission.

2.1.2 QAM Transmission

QAM transmission, without multiplexing into different subcarriers, is the predominant form of single-carrier transmission. The serial data stream is mapped into 'constellation points' by the QAM rule in effect. One of the simplest such constellations is the QPSK constellation, depicted below:

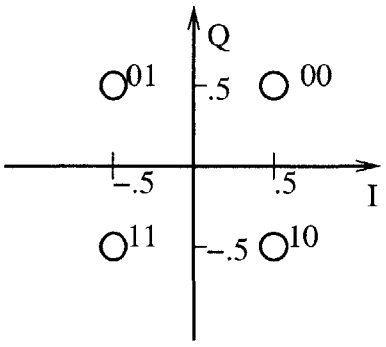


Figure 2.1: QPSK Constellation

There are a few major disadvantages to single-carrier modulation in communications systems. Primarily, such systems are prone to degrade in the presence of the time-domain interference, as momentary jumps in channel noise or other effects will destroy samples of the signal. In addition, such schemes are also prone to time dispersion in the form of channel delay spread and intersymbol interference. To equalize these effects means to increase the noise , which would require increasing the power of the signal to counteract it, or suffering the interference.

2.1.3 Frequency Division Multiplexing (FDM)

The use of frequency division multiplexing goes back over a century, where more than one low rate signal, such as telegraph, was carried over a relatively wide bandwidth channel using a separate carrier frequency for each signal. To facilitate separation of the signals at the receiver, the carrier frequencies were spaced sufficiently far apart so that the signal spectra did not overlap. This, of course, meant that empty spectral regions between the signals would be needed to assure that the different carriers could be separated with readily realizable filters. If a large spacing was not created, either a very difficult (sharp cutoff) filter would have to be implemented or there would be interchannel interference with the other data.

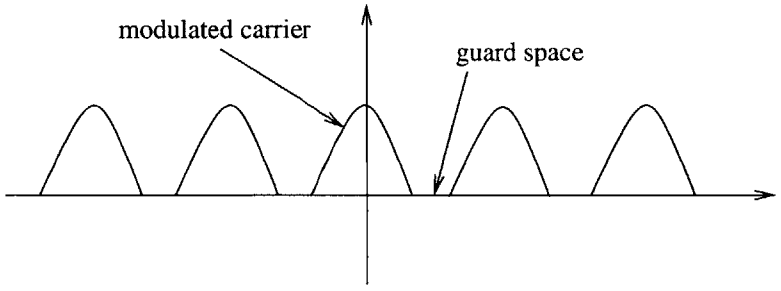


Figure 2.2: Ideal FDM Frequency Response

2.1.4 Modulation by Discrete Fourier Transform → FFT

In response to this inefficiency of bandwidth, it was noticed by R.W. Chang in 1966, among others, that utilizing bandlimited orthogonal channels would result in a fuller utilization of the frequency spectrum. Such channels could be created through the use of cosine filter banks, as well as many other schemes.

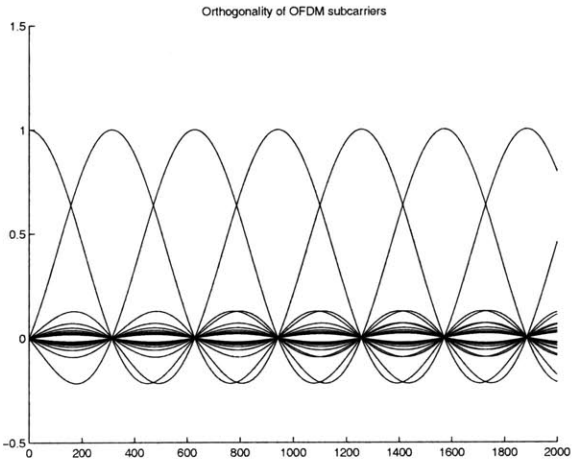


Figure 2.3: An OFDM signal. Note that neighboring carriers have zero crossings at the maximum point of the carrier.

However, this sceme was costly in terms of hardware implementation, requiring a large number of subchannel modems to do the frequency multiplexing.

Fortunately, it was shown mathematically that taking the discrete Fourier transform (DFT) of the original block of N QAM symbols and then transmitting the DFT coefficients serially is exactly equivalent to the operations required by an FDM transmitter. Substantial hardware simplifications can be made with FDM transmissions if the bank of subchannel modulators/demodulators is implemented using the computationally efficient pair of inverse Fast Fourier Transform and Fast Fourier Transform (IFFT/FFT).

2.1.5 OFDM Fundamentals

OFDM stands for orthogonal frequency division multiplexing. Since the OFDM modulator is an IFFT process, the physical meaning is to convert its data from a frequency representation into a time representation. If the modulated data is fed into a spectrum analyzer, what is displayed is the original data. In essence, the use of OFDM transforms a highly selective wideband channel into a

large number of nonselective narrowband channels which are frequency multiplexed.

2.1.6 The OFDM signal model

The OFDM signal is generated at baseband by taking the Inverse Fast Fourier Transform (IFFT) of quadrature amplitude modulated (QAM) or phase-shift keyed (PSK) subsymbols $c_k = a_k + jb_k$. An OFDM symbol has a useful period T and preceding each symbol is a cyclic prefix of length v, which is longer than the channel impulse response and multipath spread so that there will be no intersymbol interference (ISI). These topics will be explained below. The QAM symbols are modulated by the IFFT process, and with the addition of a cyclic prefix consisting of the last v symbols, the OFDM symbol is formed:

$$s(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} c_k e^{j2\pi \frac{kn}{N}}, \quad -v \leq n \leq N-1$$

The baseband signal is quadrature modulated, up-converted to the radio frequency (RF) at the carrier frequency f_0 and transmitted through the channel.

At the receiver, the signal is down-converted to an intermediate frequency (IF) by a bandpass filter of bandwidth N/T, and the first v QAM subsymbols of each OFDM symbol are removed and quadrature demodulated.

Hence, a carrier offset of Δf causes a phase rotation of $2\pi t \Delta f$. If uncorrected this causes both a rotation of the constellation and a spread of the constellation points similar to additive white Gaussian noise (AWGN). On the other hand, a symbol timing error will have little effect as long as all the samples taken are within the length of the cyclically extended OFDM symbol: in this case, each of the received signals will be multiplied by an exponential in the FFT operation, and this level difference is normalized at the equalizer.

2.1.7 Choice of QAM Constellation for OFDM Modulation

QAM (Quadrature Amplitude Modulation), explained very simply, is a mapping by which serial data can be mapped into a two-dimensional grid, in which the horizontal (in-phase) dimension is the real part and the vertical (quadrature) dimension is the imaginary part. The mapping is called a **constellation**.

According to [5], there are three factors to consider when choosing a constellation for QAM transmission:

1. **The minimum Euclidean distance among phasors**, which determines the noise immunity. The smaller the distance between possible points, the more likely that some noise will offset a particular transmitted point into the neighborhood of another, causing an error.

2. **The minimum phase rotation among constellation points**, which determines the phase jitter immunity and hence resilience against clock recovery imperfections and channel phase rotations.

3. **The ratio of peak-to-average phasor power**, which is a measure of robustness against non-linear distortions introduced by the power amplifier. A low peak-to-average ratio means that points will not be mapped to the neighborhood of other points as easily through a magnification by the power amplifier.

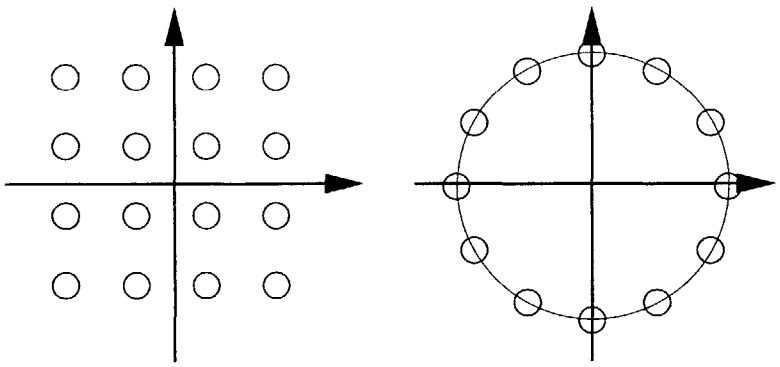


Figure 2.4: Different QAM Types

Given these desired characteristics, one would be choosing between circular and square constellations. In particular, the PSK (phase shift keying) constellation looks particularly attractive given its peak-to-average ratio of 1, as each phasor is the same length. Furthermore, PSK constellations are characterized by equal phase intevals between neighboring constellation points.

However, PSK constellations are very susceptible to problems regarding phase rotation. A significant amount of phase rotation can offset each constellation member into its neighboring point (as an example, in the PSK constellation above, a phase offset of $\pi/12$ puts each symbol into the location of its neighboring symbol). In contrast, square constellations, if affected by any rotation other than $\frac{\pi}{4}k$ (k is an integer), will have a more easily detectable frequency offset.

As a general principle, communications systems that can adjust the level of SNR incrementally in response to varying channel are best, as the system can more robustly handle different channel conditions by modifying parameters such as QAM modulation. For this reason (as well as easier recovery from rotation), square constellations are used in the DVB-T specification. To control the SNR of the system to some extent, constellation sizes of 4, 16, and 64 symbols are commonly used.

2.1.8 Error Probability for (M x M)-QAM (*analysis from* [27])

In uncoded M-PAM, the signal set A consists of M d-spaced signal points symmetrically arranged about the origin. If $M = 2^b$, then the nominal spectral efficiency is

$$\rho = 2b = \log M^2$$

bits per two dimensions. The average signal energy is

$$E_s = \frac{d^2(M^2 - 1)}{6}$$

With QAM, the standard M x M signal set is simply the Cartesian product of two M-PAM signal sets as above, since a signal from the M-PAM signal set is sent in each dimension.

In a constellation, in the presence of error, we will use the *minimum-distance* (MD) rule to determine the mapping between the received symbols and the intended symbols. By the MD rule, we choose the signal point $\hat{a}$ that minimizes the squared distance $(r_k - a_k)^2$ among the constellation points a_k- in other words, quantize the received r_k to the closest valid constellation point, when converting to QAM symbols.

By this methodology, the probability of a decision error given that a_k is sent by the transmitter is

$$P_s(E) = 1 - (1 - Pr(E))^2 = 2Pr(E) - (Pr(E))^2 \approx 2Pr(E).$$

Substituting $Pr(E) = 2(\frac{M-1}{M})Q(\frac{d}{2\sigma})$, where $\sigma^2 = \frac{N_0}{2}$ and $Q(x)$ is the Gaussian probability of error function, as defined in Appendix I, we obtain

$$P_s(E) \approx 2Pr(E) = 4\left(\frac{M-1}{M}\right)Q\left(\frac{d}{2\sigma}\right) \approx 4Q\left(\frac{d}{2\sigma}\right),$$

an approximation that holds for large M.

Substituting the energy per two-dimensional symbol $E_s = \frac{d^2(M^2-1)}{6}$, the noise variance $\sigma^2 = \frac{N_0}{2}$, and the normalized SNR in [27],

$$P_s(E) \approx 4Q(\sqrt{3SNR_{norm}}).$$

As the Q-function is a distribution similar to the Gaussian distribution, clearly the probability

of error decreases with increasing SNR.

This relation between $P_s(E)$ and SNR_{norm} allows the ability to design the system according to the SNR requirements. By the example given in [27], to achieve $P_s \approx 10^{-5}$ with an uncoded square constellation, it can be seen that an SNR_{norm} of roughly 8.4 dB is needed. Of course, when coding is applied at the FEC layer, further gains can be achieved.

2.1.9 OFDM Scheme Benefits

The tradeoff of sharp cutoff and steep filters vs. wasting bandwidth in FDM systems is avoided in OFDM.

Since now the symbol period is increased from a single carrier to an N-carrier symbol, the channel delay spread is a much smaller fraction of a symbol period than in the serial system, potentially rendering the system less sensitive to ISI than the conventional serial system.

One of the most attractive features of this scheme is the bandwidths of the subchannels is very narrow when compared to the communications channel's bandwidth. Therefore, flat-fading propagation conditions apply. The subchannel modems can use almost any modulation scheme, and QAM is the most common choice.

2.1.10 Stages of OFDM data Transmission and Reception

1. Serial data is fed into a QAM mapper, that, according to the type of QAM, maps the signal into a constellation of 4, 16, or 64 symbols.

2. This data (complex pairs) is then sent into an Inverse Fast Fourier Transform. The physical interpretation of this is that the data is now in the time, rather than frequency, domain. FFT sizes for most multicarrier communications algorithms vary between 64 and 8192 points. The IFFT is filled with null carriers as well after the modulation, so that the design of

3. Cyclic prefix addition: the last v data points of the N-point IFFT are prefixed to the data.

4. The data is interpolated, at either 4x or 8x, so that the sampling is done sufficiently above the Nyquist criterion.

5. The data is then filtered by a square root raised cosine filter, which assures that rolloff is within a proper bound.

6. The symbols are then converted to analog for RF upconversion.

2.1.11 How to deal with multipath delay in the channel response?

1. Increase the OFDM symbol duration: one can increase the duration of the OFDM symbol until it is much larger than the delay spread due to multipath. However, this may be difficult to implement, as this means that for large delay, a very large IFFT and FFT pair is required in hardware. Given the varying conditions that a broadband wireless channel is provisioned under, the hardware implementations of the FFT and IFFT pair would be forced to change a great deal, which is costly and infeasible.

2. Add a cyclic prefix to the symbol: this involves cyclically extending the N-length symbol with a v-length periodic extension of its data. When the cyclic prefix is longer then the channel impulse response, ISI is eliminated. However, in-band fading due to the frequency characteristics of the channel will still exist.

   Since the OFDM symbol duration is quite long, usually several hundred microseconds, and channel impulse responses are on the order of 10-30 microseconds, inserting a cyclic prefix of that range will not significantly reduce the data throughput.

   Even in the case that the impulse response is slightly longer than the guard interval, i.e. a few percentage points of it, the impact on system performance is limited. On the other hand, as the symbol duration for a single carrier modulation system, such as QAM (quadrature amplitude modulation) or VSB (vestigial sideband) is only about .1-.2 microseconds, it is impossible to insert a guard interval comparable to OFDM in terms of ISI elimination performance. Other techniques, such as adaptive equalization, must be used.

2.1.12 Flat Fading

Since each OFDM subcarrier occupies a very narrow spectrum, in the order of a few kHz, even severe multipath distortion will cause only flat fading (to an approximation) in a particular carrier. Hence, a channel that is wideband frequency-selective fading becomes a series of narrowband frequency non-selective fading subchannels by using the parallel multi-carrier scheme.

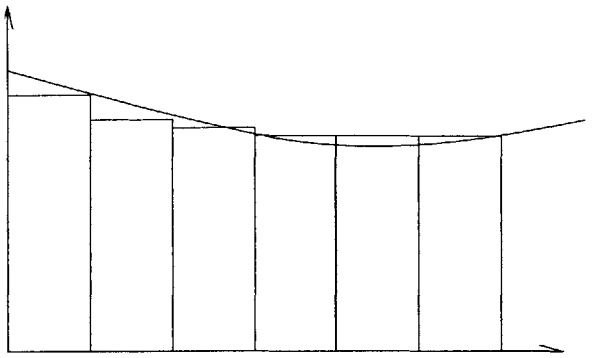


Figure 2.5: Narrowband flat fading

2.1.13 Cyclic Extension

Transmission of data in the frequency domain using the IFFT as a computationally efficient orthogonal linear transformation results in robustness against ISI in the time domain, through the use of the cyclic prefix. Unlike the Fourier transform, the FFT of the circular convolution of two signals is equal to the product of their FFT's:

$$\begin{aligned} FT(d_n * h_n) &= FT(d_N) \times FT(h_n) \\ DFT(d_n \otimes h_n) &= DFT(d_N) \times FFT(h_n) \end{aligned}$$

The signal and the channel, however, are linearly convolved. After adding a prefix to each block as shown in the figure below, linear convolution is equivalent to a circular convolution:

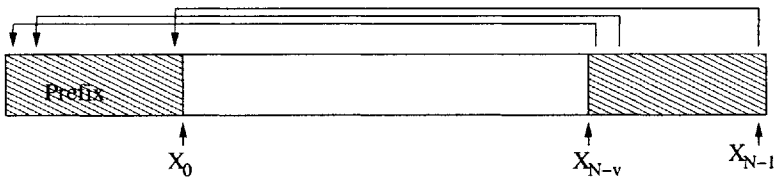


Figure 2.6: Cyclic Extension of the OFDM Frame

Using this technique, an otherwise aliased signal appears infinitely periodic to the channel.

After linear convolution of the signal and channel impulse response, the received sequence is of length $N + M$. The sequence is truncated to N samples and transformed to the frequency domain, which is equivalent to convolution and truncation. However, in the case of cyclic prefix extension, the linear convolution is the same as the circular convolution as long as the channel response is shorter than the guard interval. After truncation, the FFT can be applied, resulting in a sequence of length N, since the circular convolution of the two sequences has a period of N.

Intuitively, a N-point FFT of a sequence corresponds to a Fourier series of the periodic extension of the sequence with a period of N. So in the case of no cyclic extension we have

$$\sum_{i=-\infty}^{\infty} \sum_{k=0}^{N-1} d(k)h(n + iN - m)$$

which is equivalent to repeating a block of length $N + M - 1$ with period N. This results in aliasing or inter-symbol interference between adjacent OFDM symbols. In other words, the samples close to the boundaries of each symbol experience considerable distortion, and with longer delay spread, more samples will be affected. Using cyclic extension, the convolution changes to a circular operation. Circular convolution of two signals of length N is a sequence of length N so the inter-block interference issue is resolved.

The relative length of cyclic extension depends on the ratio of channel delay spread to the OFDM symbol duration.

2.1.14 Windowing of OFDM symbols

Proper windowing of OFDM blocks, as shown later, is important to mitigate the effect of frequency offset and to control transmitted signal spectrum. However, windowing should be implemented after cyclic extension of the frame, so that the windowed frame is not cyclically extended. A solution to this problem is to add null carriers to the block. For example, a block containing 2048 carriers may typically have 344 null carriers and 1704 with data.

N (fft size)	**data carriers**	**null carriers**
256	213	43
512	426	86
1024	852	172
2048	1704	344

Table 2.1: Typical number of data and null carriers in OFDM for different FFT sizes

2.1.15 Variable Bit-Loading

A broadband wireless channel in the MMDS band is quite different from a DSL (digital subscriber line) channel in terms of fading. The wireline SNR from 0 to 8 MHz in DSL has a sharply downward curve, whereas from 2.5 to 2.686 GHz in MMDS, while th channel characteristics vary in time, they

do not have as steep a downward slope as xDSL channels.

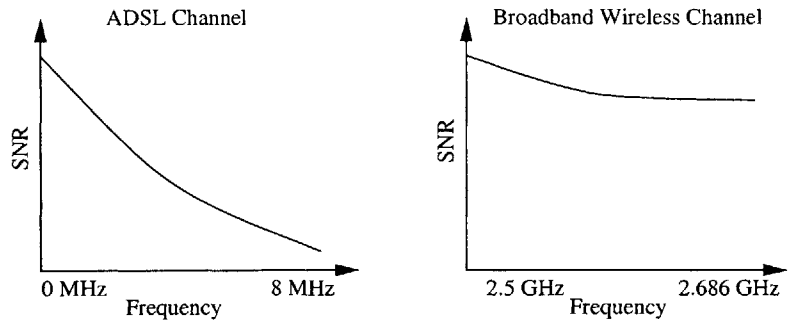


Figure 2.7: Comparison or Relative SNR in ADSL and Broadband Wireless Channels

As a result, variable bit loading, which is the technique of increasing or decreasing the number of bits transmitted in a particular subchannel in proportion to that subchannel's SNR, is not particularly useful in MMDS. The technique allows each subcarrier to encode its signal with a different QAM constellation and FEC method, which would greatly complexify the structure of both the transmitter and reciever.

2.1.16 Summary

As we have seen in this section, OFDM systems easily prevent ISI by inserting a cyclic prefix before each transmitted block. The use of multiple carriers in transmission is the most important difference between

Another distinction of OFDM which makes commonly implemented is that the receiver has simple frequency-domain equalization, composed of one-tap filters, that makes it very attractive to implement.

The next section discusses the behavior of OFDM systems when influenced by time domain and frequency domain errors and offsets.

Chapter 3

Effects of Offset and Defect

3.1 OFDM orthogonality and sensitivity

The orthogonality of subcarriers in OFDM can be maintained and individual subcarriers can be completely separated and demodulated by FFT at the receiver when there is no intersymbol interference (ISI) introduced by transmission channel distortion.
OFDM is a particularly sensitive medium of transmission: linear distortions such as multipath delay and reflections off surfaces cause ISI between OFDM symbols, resulting in loss of orthogonality and an effect that is similar to cochannel interference. However, when the delay spread is a very small percentage of the OFDM symbol length, the impact of ISI is insignificant.

3.2 Moose's discussion of freq offset

This seminal paper discusses the effects of frequency offset on the performance of OFDM digital communications. The main problem with frequency offset is that it introduces interference among the multiplicity of carriers in the OFDM signal. It is shown that to maintain SNR ratios of 20dB or greater for the OFDM carriers, offset is limited to 4% or less of the intercarrier spacing.

3.3 Frequency Domain Interference

Unlike minor time domain interference, frequency interference is quite detrimental to OFDM systems. For a single carrier system, where one carrier occupies the entire channel, a tone interference will not cause transmission error as long as its level is sufficiently lower than that of the carrier. In contrast, for an OFDM system, the transmission power is divided among many subcarriers-hence,

even low-level tone interference could destroy the corresponding sub-carriers and cause errors.

To mitigate this type of interference, one can employ spectrum shaping. Since the complex input from the QAM responds to frequency domain subcarriers, simply assigning them a zero value will create spectrum notches. When the notches are co-located with interference tones, there will be no impact on the OFDM signal. The major disadvantage of creating spectrum notches is the reduction of the data throughput.

3.4 Effect of Frequency Offset on OFDM systems

When the receiver is perfectly synchronized with the transmitter, and the discrete channel is ISI-free, the IDFT and DFT operators in OFDM systems appear cascaded and give the identity operator. In the presence of carrier asynchronism, orthogonality of the multiplexed signals is destroyed and interference is created between the data symbols in a DFT block. This phenomenon is easily visualized as follows: let us denote the DFT operator by an NxN dimensional matrix D. The effect of a radian frequency error ω between transmitter and receiver is a block transformation that can be represented using the diagonal matrix of exponentials. For the kth block, the receiver output is related to the transmitter input through the matrix transformation

$$D^{'} = e^{ikN\omega T} D^{-1} E D$$

which is obviously not proportional to the identity matrix E as long as $\omega \neq 0$. The implication of this in OFDM is that carrier frequency offsets lead to intercarrierinterference between the subcarriers of a DFT block. The nth output sample of the kth DFT block is given by

$$\begin{aligned} y_n(k) &= \frac{1}{T} e^{jkN\omega T} \sum_{m=o}^{N-1} a_m(k) \\ &= \sum_{l=o}^{N-1} e^{jl\omega T} e^{j2\pi l(m-n)/N} \end{aligned}$$

and this expression clearly shows that not only the data symbols are rotated, but they also interfere with each other. All terms corresponding to an index $m \neq n$ in the first sum on the right-hand side are indeed ISI terms. Note that this phenomenon is non-existent in single-carrier systems.

In fixed wireless OFDM systems, frequency offsets are created by

1. Differences in oscillators in the transmitter and reciever
2. Phase noise introduced by nonlinear channels

This offset causes

1. Reduction of sinc amplitude because the sinc is no longer sampled at the peak
2. Introduction of intercarrier interference, because the neighboring carriers are no longer sampled at 0 at the sampling point of any particular subcarrier

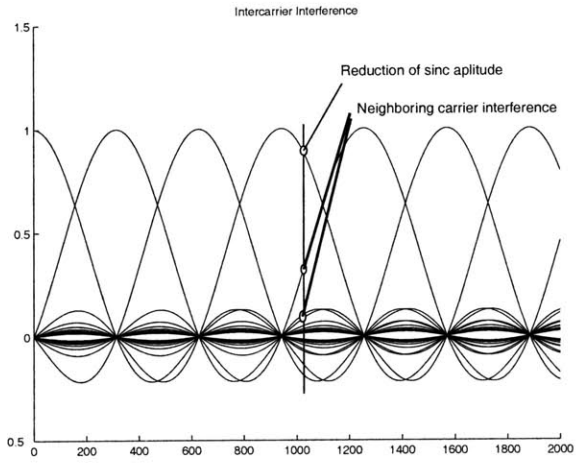


Figure 3.1: Effects of frequency offset

In [9] Pollet et. al. define the frequency offset per subcarrier $\Delta f = \frac{\Delta F}{W/N}$, where ΔF is the total frequency offset and N is the number of subcarriers. The degradation D in SNR (in dB) is

$$D(dB) \approx \frac{10}{3ln10}(\pi\Delta f)^2 \frac{E_s}{N_0} = \frac{10}{3ln10}\left(\pi\frac{N\cdot\Delta F}{W}\right)^2 \frac{E_s}{N_0}.$$

The equation shows that the degradation increases quadratically with respect to the number of subcarriers, if other factors are fixed.

To help deal with the problem of understanding degradation, Moose in [7] derived a signal-to-interference ratio (SIR) on a fading, dispersive channel. It is defined as the ratio of the power of the useful signal to the power of the interference signal, which is comprised of both ICI and additive noise. His derived degradation is

$$D(dB) \leq 10\log_{10}\left(\frac{1+0.5947\frac{E_s}{N_0}\sin^2\pi\Delta f}{sinc^2\Delta f}\right),$$

where $sinc(x) = \frac{\sin(\pi x)}{\pi x}$. The factor .5947 is found from a lower bound of the summation of all interfering neighboring subcarriers. Moose's conclusion from the data obtained by graphing this relation is that to avoid severe degradation, the frequency synchronization accuracy should be better than 2%.

3.5 Effect of Timing Offset in OFDM Systems

A timing offset gives rise to a phase rotation of the subcarriers, as the timing delay, once passed through the FFT operator, yields this rotation. This phase rotation is largest on the edges of the frequency band.

If a timing error is small enough to keep the channel response within the cyclic prefix, the orthogonality is maintained. In this case a symbol timing delay can be viewed as a phase shift introduced by the channel, and the phase rotations can be estimated by a channel estimator.

On the other hand, If a time shift is larger then the cyclic prefix, then the delay will 'leak' into the next symbol, causing intersymbol interference(ISI).

3.6 Time Domain Interference and Distortion

Since the duration of the entire OFDM symbol is many times longer than that of a single data point, any short term distortion or interference caused by time domain impulse interference, amplitude clipping, short term fading and instantaneous change of channel response will be averaged out by the FFT process in the receiver over the entire OFDM symbol period. If this distortion and interference occurs over a short portion of the OFDM period, their impact is negligible, since all of the data subcarriers would be only slightly affected (but still decodable). In a single carrier system, on the other hand, a few symbols could be destroyed completely, resulting in errors. Hence, OFDM is excellent protection against time domain interference.

3.7 Phase Noise

Phase noise is a zero mean random process of the deviation of the oscillator's phase from a purely periodic phase function at the average frequency. The impact of local oscillator phase noise on the performance of an OFDM system is an interesting and useful phenomenon to discuss.

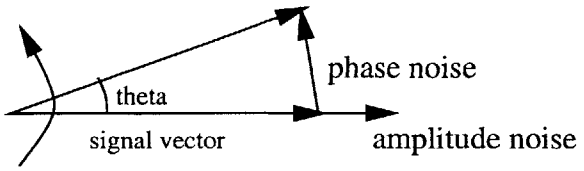


Figure 3.2: Phase Noise

The signal is represented by a vector with a length proportional to the signal amplitude rotating about the origin at the oscillator's frequency. At the tip of this vector is a randomly directed vector which represents the oscillator's noise.

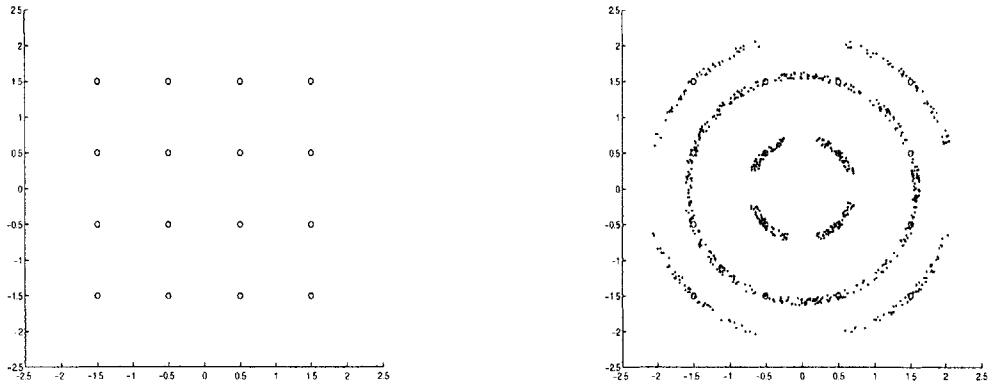
This noise vector may be represented by two orthogonal vectors, one in the direction of the signal vector and one in the direction of the rotation. The amplitude vector is the amplitude noise and the other vector is the phase noise vector. Clearly when the amplitude vector changes, the oscillator's overall amplitude changes and when the phase noise vector changes, the oscillator's phase changes.

Although it is intuitive to compare amplitude jitter to the overall amplitude, it may seem arbitrary to compare the phase jitter in radians to the amplitude of the carrier - the two quantities seem unrelated. The confusion comes from the small angle assumption, which states that the $\sin\theta = \theta$ for small angles. When phase noise is measured, it is small variations in the phase angle that are measured and then the length of the phase noise vector is inferred. The small angle assumption states that the length of the phase noise vector is equal to the measured angle multiplied by the signal size. Or stated differently, the measured angle is the phase noise vector divided by the signal size. (For reasonably good oscillators the noise angle is quite small.)

Phase noise in OFDM can result in two effects:

1. A common subcarrier phase rotation of all of the subcarriers
2. A thermal noise-like subcarrier degradation, much like AWGN.

The latter effect comes from subcarriers interfering with each otherdue to destruction of orthogonality (ISI). In OFDM systems, no distinction can be made between the phase rotation introduced by a timing error and a carrier phase offset, when the ISI isn't present. Phase noise on an enlarged QAM16 constellation is plotted below:



Figures 3.3 and 3.4: 16QAM constellation before and after phase noise insertion

In [9] an analysis of the impact of carrier phase noise is undertaken. It is modelled as a Weiner process $\theta(t)$ with $E\{\theta(t)\} = 0$ and $E\{(\theta(t_0 + t) - \theta(t_0))^2\} = 4\pi\beta||t||$, where β (in Hz) denotes the one-sided 3 dB linewidth of the power density spectrum of the free-running carrier generator. The degradation in SNR, i.e. the increase in SNR needed to compensate for this error, can be approximated by

$$D(dB) \approx \frac{11}{6ln10}\left(4\pi N\frac{\beta}{W}\right)\frac{E_s}{N_0},$$

where W is the bandwidth and $\frac{E_s}{N}$ is the per-symbol SNR. From the formula, it can be seen that this degradation increases linearly with the number of subcarriers, since this forces the spacings to be smaller and smaller.

The common phase error, i.e., constellation rotation on all the demodulated subcarriers, is caused by the phase noise spectrum from DC up to the frequency of subcarrier spacing.

The main weakness of OFDM is its sensitivity to frequency offsets caused by oscillator instabilities. As the subcarriers are closely spaced over the channel bandwidth, the frequency offset must be kept within a small fraction of the subcarrier distance to avoid severe bit error rate degradations.

3.8 Sine Ingress

The origin of these phenomena is interference from from power lines, and hence it is common at 60 Hz, 120 Hz, and other such frequencies. Sine ingress does not factor in to broadband wireless to a great extent. It is more prevalent in wireline methods such as ADSL, cable modem, and HomePNA.

3.9 Channel/Multipath

In any RF system, there always exists the problem of multipath– propagation of the same signal on many paths, arriving at the receiver at different times with different magnitudes. This phenomenon is due to objects of different sizes (cars, mountains, buildings, trees) reflecting and dispersing signals at different strengths.

In a multipath model, the baseband received signal can be written as

$$r(t) = \sum_{n=1}^{N_p} \rho_n(t) s[t - \tau_n(t)]$$

where $s(t)$ is the baseband transmitted signal, N_p is the number of paths, $\rho_n(t)$ and $\tau_n(t)$ are respectively the complex attenuation and the delay associated with the nth path. Generally the attenuation and the delay of each ray are time-variant. For fixed wireless channels, the added problem of Doppler fading due to motion is avoided. Estimating the multipath response is the responsibility of the channel estimation algorithm. Hence, the multipath model will not be explored in great detail except when simulating the channel with a particular $h(n)$. Rayleigh fading assumptions can be found in many of the technical papers in the bibliography.

The convolution program used to emulate the channel is in Appendix II.

3.10 Additive White Gaussian Noise (AWGN)

In the time domain, the amplitude distribution of the white thermal noise has a normal or Gaussian distribution. Since it is inevitably additive to the signal, it is referred to as additive white Gaussian noise (AWGN). Note that AWGN is therefore the noise generated in the receiver. The probability density function is the Gaussian distribution:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-m)^2}{2\sigma^2}}$$

where m is the mean and σ^2 is the variance. Because the noise is additive, the effects of AWGN can be mitigated by increasing the transmitted signal power and thereby reducing the relative effects of noise.

3.11 AM Hum

AM hum and noise on the carrier is the ratio of the DC voltage detected from an unmodulated carrier to the detected peak AC voltage. Such a signal generally modulates around 10 Hz, and hence can basically be seen as a change in the power level of the signal (since it is so slow). This level

increase and decrease is removed by the equalization process. For this reason, though it is included in the C simulation, it is not tested actively.

3.12 Summary

As we have seen, time and frequency offset and distortion impairs the quality of the OFDM signal. This impairment differs in many ways from the impairment of traditional single-carrier systems.

Mainly, there is a decreased sensitivity to of time-based noise and interference, due to the fact that the FFT process reconstructs each symbol from many time-domain subcarriers. For timing offset, the presence of a cyclic prefix acts as a buffer— as long as the offset is within the length of the cyclic prefix, the channel estimation process will eliminate it.

OFDM's FFT process also results in increased susceptibility to frequency interference—particulary because interfering tones at the frequency of a subcarriers need only $1/N$ of the power of a similar interfering tone in a single-carrier system to impair that subcarrier's data transmission.

The next section describes the use of an OFDM datapath program, written in C, to simulate the channel and noise models in more detail.

Chapter 4

OFDM Datapath Simulation

4.1 System Description

4.1.1 Diagram

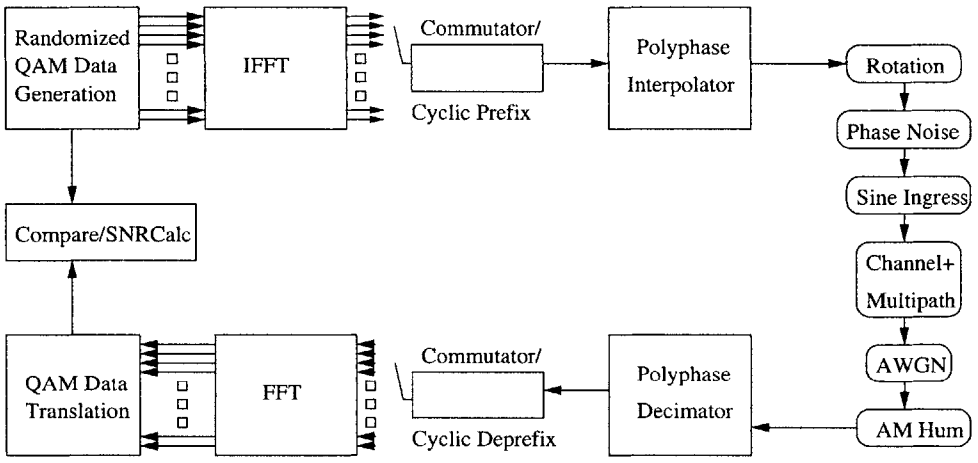


Figure 4.1: C Coded Modulation and Demodulation Testbench

4.1.2 Modulation and Demodulation Testbench `main_ofdm.c`

The first simulational system created was a test bench used to evaluate different carrier and recovery schemes. The system has the following datapath:

1. Random QAM symbols are generated at the modulator and passed into an IFFT (Inverse Fast Fourier Transform) block.

2. The data is transformed into the time domain, and a cyclic prefix is added to the data frame.

3. A parallel-to-serial converter passes data to an interpolator block, consisting of a square root raised cosine filter and an upsampler, implemented in polyphase form.

4. A rate converter adjusts the rate of the data for transmission.

5. The data is then convolved through a channel model (a finite-length filter effecting a typical multipath response), and additive noise and other defects are added to the signal. Channel imperfections introduced include rotation, shaped phase noise, sine ingress, frequency shift, AWGN (additive white Gaussian noise), and AM hum.

6. At the demodulator, the rate converter receives the signal, shifts the frequency, and sends samples to the decimator, which implements the interpolator in reverse, using the same matched filter.

7. A serial-to-parallel converter removes the cyclic prefix from the received samples and indexes them into an FFT(Fast Fourier Transform) block.

8. The data is then converted back to the frequency domain, and the QAM input and output are compared, to determine the error rate and SNR (after adjusting for delays incurred by the matched filters and normalizing).

This simplified platform allows for visibility into the effects of various channel imperfections and noise elements independently. In addition, this simulation test bench permits data generation with channel imperfections, which can then be read into the MATLAB code containing the algorithms.

4.1.3 System Methodology

The C simulation system was built using `main_hpna.c` as a template. `main_hpna.c` was a program written by D. Ribner to simulate the performance of a HomePNA chipset— a LAN protocol designed to utilize copper telephony in the home. As HPNA is almost entirely different from an OFDM system, most of the submodules and the entire simulation shell had to be redesigned— but the code served as an invaluable tool in developing a style for submodule and shell creation.

As can be seen in the simulation shell `main_ofdm.c` in Appendix II, the simulation design involves many submodules that carry on specific tasks in the datapath. All operate at the same time, and the submodules have (mostly) generic charactersistics:

- A `strobe` variable, which functions as a normal strobe: when the module is called and the strobe is 1, the module (whether it be an FFT, interpolator, or other device) will operate on the data.

- An `init` variable, which when set, resets the internal counters of the module and sets the output to 0. It may also load internal registers with predefined values or create some other

initialization behavior that is distinct from the normal operational performance of the module. Depending on the module, `init` will either function or not function when strobe = 0.

- A `ready` output parameter, which the module sets to 1 or 0 for the purpose of strobing the following module.
- One or more `parameters`, which determine the mode of operation for the module. For example, N is a parameter for the FFT block that determines the size of the FFT.
- One or more `input_data` parameters, passed by reference to the module
- One or more `output_data` parameters, in which the output of the module is placed.

For example, in the call

```
awgn(cc1r, cc1i, &opr, &opi, sigma, nseed, passthrunz, 0, &ready_awgn, init);
```

`cc1r, cc1i` are the input data, `&opr, &opi` are the output data, `sigma, nseed, passthrunz` are various parameters `&ready_awgn` is the output ready, and `init` is the initializer. There is no strobe, signifying that the code executes each time that it is called.

The input `strobe` and the output `ready` of each submodule forms a 'chain' by which the modules interoperate and are able to input and output the data at the right times. In some locations where there were mismatches, a module that I created called `delay.c` aided in aligning the samples. Below is the method by which submodules and the main shell operate:

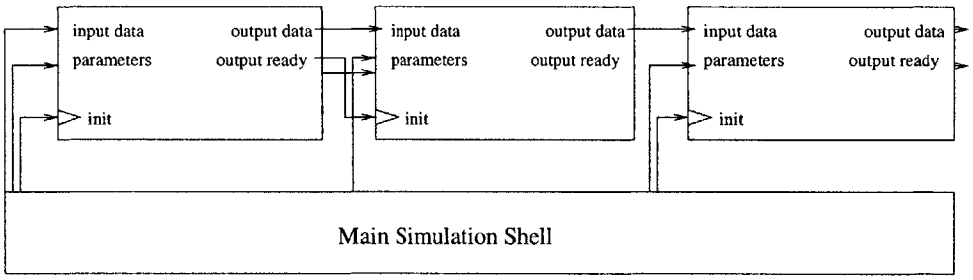


Figure 4.2: Simplified Structure of Data Interfacing in C Simulation

The simulation shell contains the main clocks, most of the system parameters, and data arrays that input and output the signal at different stages of its transmission.

4.1.4 QAM modulator and demodulator

The QAM modulator designed for this system takes in as input a pseudorandom binary sequence (PRBS) and outputs a square QAM constellation, of any size 2^{2n} from 4 to 4096, as determined by the parameter `con_size`. The code is in Appendix II.

4.1.5 IFFT and FFT pair

OFDM requires an orthogonal transform– meaning, the neighboring sub-bands have zero crossings at the location of data modulation of each particular band. What has made OFDM such a popular method of transmission is that a fairly straightforward operation– the IFFT– creates the orthogonal bands for modulation. Its $N \log N$ running time makes it suitable for a low-latency communications datapath, even on a system with as many as 2048 carriers (common to OFDM schemes).

The IFFT and FFT code is in Appendix B. It is adapted from *Numerical Methods in C* [2]. It replaces the input array `fft_data` with its output (if performing an IFFT, the output is in the time domain; if FFT, the output is in the frequency domain). `isign` determines which of the two is being performed. It performs a decimation in time algorithm. Note that in the actual hardware design of the FFT, a radix-4 based FFT is used to decrease the number of stages.

4.1.6 Commutator and Cyclic Prefix Addition

The commutator and CP addition block is responsible for converting the parallel output of the IFFT into a serial format. Since the IFFT output is all available at the same time, the commutator starts sending samples by sending s_{N-v} through s_{N-1} (the cyclic prefix), and then continues by transmitting s_0 through s_N. Hence, the code calculates simple address translation to index the proper data into each symbol.

4.1.7 Polyphase Interpolation and Decimation

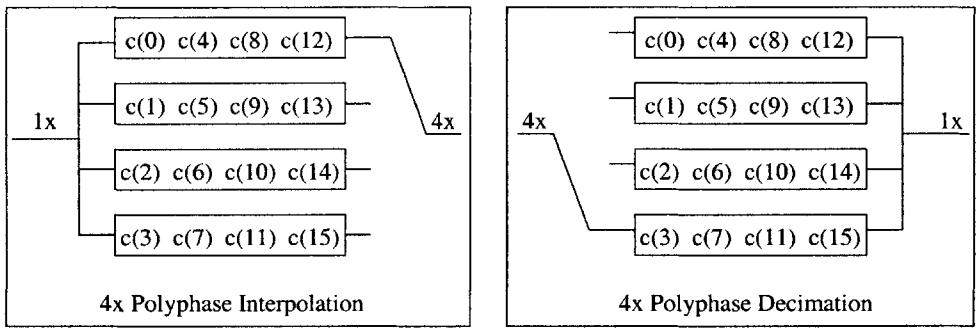


Figure 4.3: Polyphase Interpolation and Decimation Filter Banks

Decimation and Interpolation

In the *decimation* process, a sequence $x(nT_s)$, sampled at the rate $f_s = 1/T_s$, has its sampling period increased to a new value $T_s' = MT_s$, where M is an integer. Let $y(nT_s')$ denote the new sequence with a sampling period T_s'.

In the *interpolation* process, which is the dual of the decimation process, we increase the sampling rate by an integer ratio. Consider a sequence $x(nT_s)$, the sampling period of which is to be reduced to a new value, $T_s^{'} = T_s/L$, where L is an integer. This requirement is achieved by first inserting $L-1$ zero-valued samples between each sample of $x(nT_s)$, and then pass the resulting sequence $v(nT_s^{'})$ through a low-pass filter with a periodic frequency response.

For this system, the filters used are symmetric FIR filters, i.e., the first $n/2$ and last $n/2$ samples are the same, in reversed order. The benefit of symmetric filters in hardware is that only half the coefficients must be stored, and resource sharing can be employed([28]) if required.

In essence, **polyphase filter banks** are systems in which the interpolation/decimation is tied to a filter. The filter banks upsample or downsample the data and convolve it with the filter, as can be seen in the figure above. It shows an interpolator/decimator pair with a factor of 4 and matched filters with 16 coefficients.

In radio systems, transmitters must have bandpass filters to save as much spectrum as possible. In OFDM, the problem with blindly implementing such a filter with a good cutoff is that it can cause intercarrier interference between the subcarriers of the symbol. Hence, it must have zero crossings periodically at the peaks of the subcarriers and a magnitude of 1 at $t = 0$.

A raised cosine Nyquist filter fits these specifications; it is a low-pass filter which is commonly used for pulse shaping in data transmission systems (e.g. modems). The frequency response $|H(f)|$ of a perfect raised cosine filter is symmetrical about 0 Hz, and is divided into three parts: flat (constant) in the passband, a cosine curve to zero through the transition region; and zero outside the passband. The response of a real filter is an approximation to this behavior.

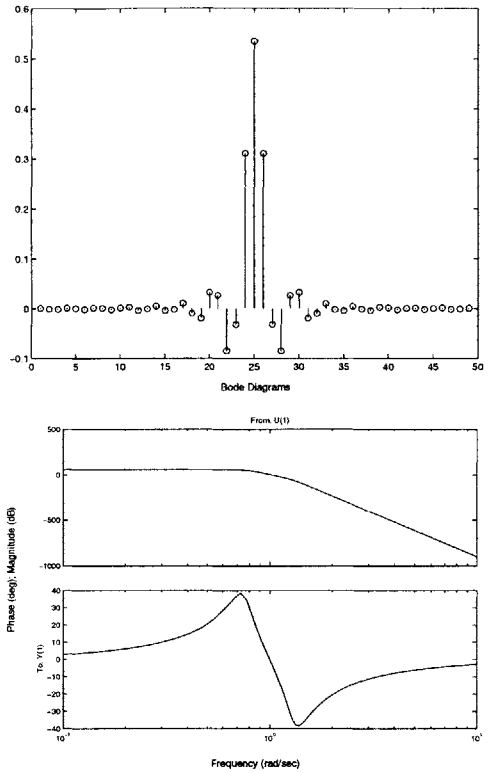
Using matched filtering, the filter on the interpolator and the filter on the receiver combine to form the desired characteristic Nyquist filter. Hence, each of the filters is designed as a square root-raised cosine filter.

The equations which defined the filter contain a parameter α, which is known as the roll-off factor or the excess bandwidth. α lies between 0 and 1, and is generally less than .5 in communications applications (a roll-off of more than .5 introduces too much excess bandwidth at the edges): To design this filter, I used MATLAB's `firrcos` function:

```
b = firrcos(n,F0,r,fs,'rolloff','type')
```

`n` is the filter order, `F0` is the cutoff frequency, `r` is the α value, `fs` is the sampling frequency,

'rolloff' tells MATLAB to interpret r as the rolloff, and 'type' indicates the type of the filter. Since we are using a root-raised cosine filter, the parameter is 'root'.



Figures 4.4 and 4.5: Square-root raised cosine filter for 48 taps, $\alpha = .25$: $h(n)$ and $H(z)$

4.1.8 SNR Calculation

The system calculates the signal-to-noise ration for each of the subcarriers separately. It does this by first aggregating the signal power $x_{real}^2 + x_{imag}^2$ for each transmitted subcarrier-modulated tone, as well as the noise power $(x_{real} - y_{real})^2 + (x_{imag} - y_{imag})^2$, based on the error between the transmitted x and received y. Next, this data is averaged by subcarrier. Finally, the SNR array is calculated by

$$SNR_{db}[j] = 10 * log_{10}\left(\frac{avgsig[j]}{avgerr[j]}\right).$$

```
    /* take error and signal information for
       QAM input and output */

    for (j=0; j<num_blocks*fft_size; j++)
      {
sig[j] = QAMArray[2*j]*QAMArray[2*j] +
  QAMArray[2*j+1]*QAMArray[2*j+1];
err[j] = (QAMArray[2*j]-QAMArray2[2*j])
```

```
  *(QAMArray[2*j]-QAMArray2[2*j]) +
  (QAMArray[2*j+1]-QAMArray2[2*j+1])
  *(QAMArray[2*j+1]-QAMArray2[2*j+1]);
/*
printf("sig[%d]=%f  ",j,sig[j]);
printf("err[%d]=%f  ",j,err[j]);
printf("QAMArray[%d]=%f  ",2*j,QAMArray[j]);
printf("QAMArray2[%d]=%f\n",2*j,QAMArray2[j]);
*/
fprintf(f_qam_in, "%f  %f\n",
QAMArray[2*j], QAMArray[2*j+1]);

fprintf(f_qam_out, "%f  %f\n",
QAMArray2[2*j], QAMArray2[2*j+1]);

      }

    /* use sig and err information to get
       average signal and error */

    for (j=0; j<fft_size; j++)
      {
for (i=0; i<num_blocks; i++)
  {
    avgsig[j] += sig[j+i*fft_size]/num_blocks;
    avgerr[j] += err[j+i*fft_size]/num_blocks;
  }
      }


    /* use average signal and error data to
       generate SNR */

    for (j=0; j<fft_size; j++)
      {
SNRdb[j] = 10 * log10(avgsig[j]/avgerr[j]);
```

```
/*printf("SNRdb[%d]=%f\n",j,SNRdb[j]);*/
      }

    for (j=0; j<fft_size; j++)
      fprintf(f_snr,"%f\n", SNRdb[j]);
```

4.2 Simulations and Interpretation

This section contains some the results of numerous simulations carried out on the platform described above. The SNR at the receiver's output provides the best measure of the received signal quality, as the work deals purely with the modulation and demodulation of the signal, without discussing coding gains.

4.2.1 Single subcarrier missing (phase noise and AWGN)

Below is the constellation output of four different subcarriers under −80dBc conditions, with the first and second carriers containing 0 and the third and fourth transmitting normal QPSK data.

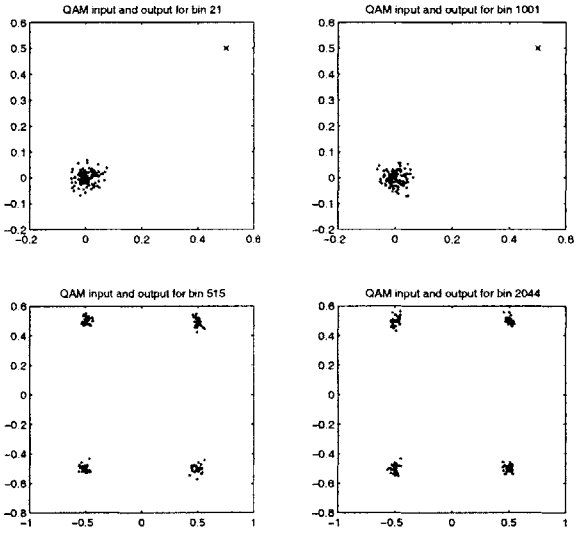


Figure 4.6: Effect of -80 dBc phase noise on null and data carriers

For bins 21 and 1001 of this QPSK OFDM signal, the transmitted data is 0, which gives us an opportunity to see the effects of phase noise given no data. As developed in the previous section, the phase noise is understood to be mostly a rotation of the data phasor– but the presence of the clusterings around the origin for these two subcarriers indicate that the effects due to ICI are prevalent. Upon further investigation, the noise power of the clusters about the null carriers were 94% as strong as the clusters of the data-modulated subcarriers 515 and 2044, indicating that the effects of rotation only represented 6% of the noise power. Simulations below that highlight the

change in the nature of the noise as the number of subcarriers change help explain why this is the case.

4.2.2 −70 dBc phase noise; one OFDM symbol

Below is the constellation output of one entire $N = 2048$ symbol, showing the data from each subcarrier.

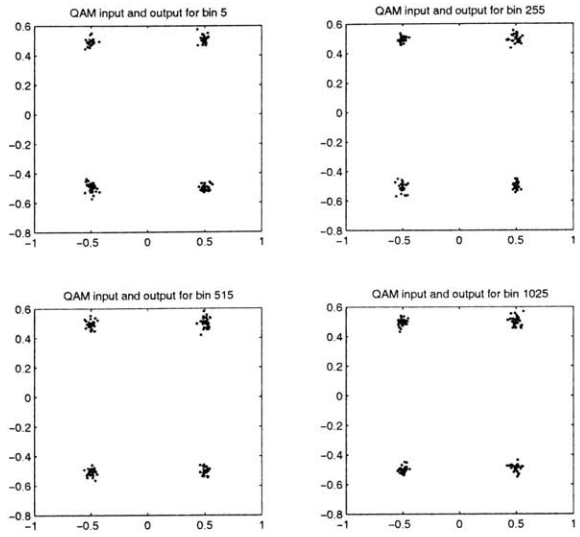


Figure 4.7: −70 dBc phase noise; one OFDM symbol

This simulation was repeated numerous times for different subcarriers, for the purpose of understanding the effect of the subcarrier index on the SNR. The next simulation highlights this finding.

4.2.3 SNR per bin for phase noise at -80 dBc

Below is the signal to noise ratio per carrier for OFDM data transmitted at a phase noise of −80 dBc. The horizontal axis is the carrier number, and the vertical axis is the SNR. The average SNR for the signal was 25.7.

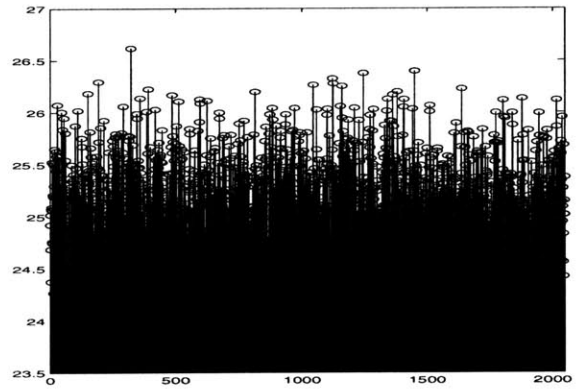


Figure 4.8: SNR per subcarrier

This result agrees with the previous simulation, in that the subcarrier number has little to do with the level of SNR in the signal. This contrasts the intuition that subcarriers in the middle of the symbol should have the lowest SNR and the ones at the edges should have higher SNR's, as only the central subcarriers would interfere with 'tails' of subcarriers from both sides.

4.2.4 Additive White Gaussian Noise, Varying N

AWGN is added to a QAM16 OFDM signal to observe the effects on SNR:

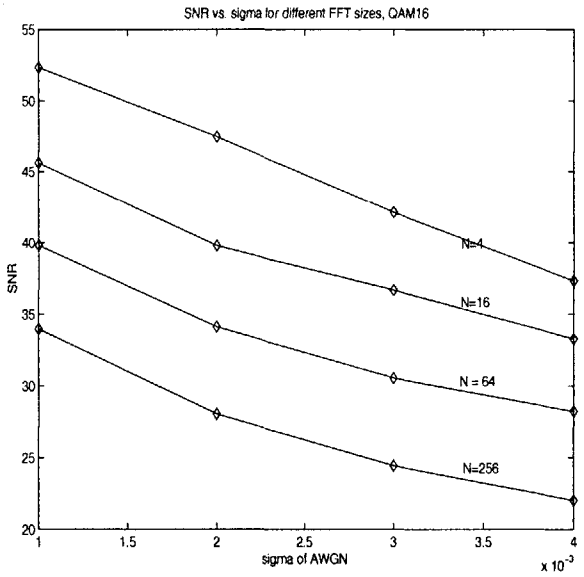


Figure 4.9: SNR vs. Phase Noise for $N = 4, 16, 64, 256$

From the graph, it is clear that the larger the FFT size, the lower the SNR.

4.2.5 SNR Calculations for Varying Phase Noise, Varying N

The deleterious effect of phase noise on SNR varies little with FFT size. Below is the relation of SNR vs. phase noise. From the graph, it appears that the FFT size does not affect the behavior of phase noise at all.

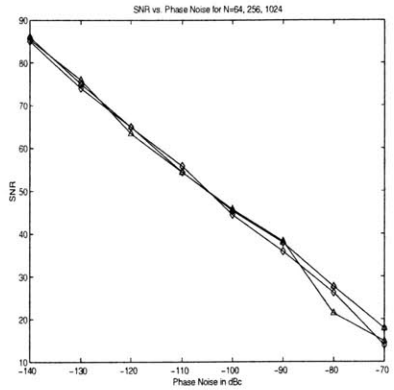
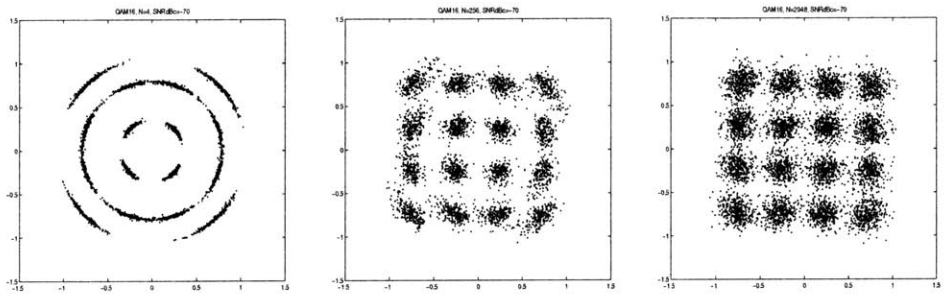


Figure 4.10: SNR vs. Phase Noise for $N = 64, 256, 1024$

The next set of MATLAB simulations showed that this was not quite the case.

4.2.6 Constellation Output for Phase Noise at -70dBc, varying N

Receiving the same SNR readings for each FFT size, and having agreed that phase noise closely resembled AWGN in simulations with $N = 2048$, I decided to look at the constellation outputs of the different FFT sizes, given the same phase noise and QAM16 modulation. The QAM outputs are displayed below for $N = 4, 256, 2048$:



Figures 4.11, 4.12, 4.13: QAM16 Output for $N = 4, 256, 2048$, phase noise $= -70$dBc

The results showed that the assumption of phase noise to be like AWGN only holds under certain conditions– namely, when the FFT size is large (> 2048) and phase noise is small. Otherwise, the effect of phase rotation becomes clearly the dominant term in the distortion of the signal, as the $N = 4$ output clearly shows. The more orthogonal subbands, the more the rotation gives way to Gaussian "smearing."

The most important results from the simulations showed that the phase noise from the oscillator could be treated as a Gaussian noise, rather than as a phase offset plus AWGN for large N, which is ubiquitous in OFDM broadband wireless systems.

This result simplifies the receiver design, as the receiver signal model can be fairly accurately modelled as consisting of the carrier offset, timing offset, channel, and AWGN. However, from the AWGN simulations, we can see that in the absence of any correction, a larger FFT size also indicates that AWGN will decrease SNR to a greater extent. The tradeoff between receiver simplicity and maximum performance should be decided based on the system requirements.

Chapter 5

Analysis of Synchronization Algorithms

5.1 Channel Estimation Assumptions

Channel estimation is the procedure by which the multipath channel response is measured. Once this estimate $\hat{H}(k)$ is determined, it can be used to deconvolve the received signal so that synchronization, demodulation, and decoding of the signal can occur.

Usually, OFDM systems provide pilot signals for channel estimation. These pilot signals are repeated frequently int he OFDM stream, as the channel conditions also vary rapidly. The algorithm below explains the usage of these known pilot symbols to determine the channel response. It is the most common way to do channel estimation in multicarrier systems:

1. The received data is passed through a Fast Fourier Transform
2. The v training tones are removed from the FFT data
3. A v-point IFFT is performed on the training tones to determine the time domain data
4. The tones are then normalized and time averaged to determine an estimate $\hat{h}(n)$ of the channel response
5. A prune FFT (v→N) determines the N-point frequency response, which can then be used to deconvolve the data.

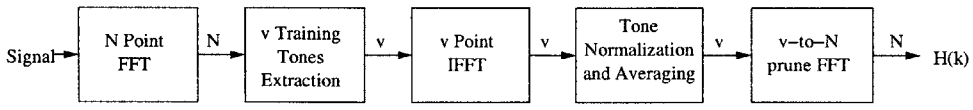


Figure 5.1: Channel Estimation Algorithm

Note that with this N-point frequency response, the channel response can be deconvolved from the signal by dividing by the $\hat{H}(k)$ coefficient corresponding to that term.

In this thesis, is is assumed that a suitable channel estimation algorithm, along the lines of the one above, exists and will be used to remove the effects of the channel. This assumption allows us to concentrate on synchronization.

5.2 Introduction to Synchronization

In OFDM systems, synchronization at the receiver is an important task that must be performed. The receiver must continuously scan for incoming data, and rapid acquisition is needed. The ratio of the number of sync bits vs. the number of data bits must be kept low, and the complexity of acquisition algorithm must also be kept low. Sync in OFDM requires finding the symbol timing and the carrier frequency offset.

It is different than single carrier because no there is no 'eye opening', where the best sampling time can be found– rather, there are thousands of samples per OFDM symbol since the number of samples necessary is proportional to the number of subcarriers. Finding the symbol timing for OFDM means finding an estimate of where the symbol starts– an additional level of complexity with which single-carrier systems need not concern themselves.

There is usually some tolerance for symbol timing errors when a cyclic prefix is used to extend the symbol. Synchronization of the carrier frequency at the receiver must be performed very accurately, or there will be loss of orthogonality between the subsymbols. As explained in chapter three, OFDM systems are very sensitive to carrier frequency offsets since they can only tolerate offsets which are a fraction of the spacing between the subcarriers without a large degradation in system performance. Clearly, being within $x\%$ frequency offset for a particular subcarrier is more difficult than achieving $x\%$ offset for the entire symbol.

However, OFDM's unique structure leads to increased opportunities to synchronize the data: the pilot tones, cyclic prefix, and even the simple **cyclostationarity** (see Appendix A) of the signal

There is a great number of methods by which synchronization can be achieved. In general, all involve some kind of knowledge of either data or the structure of the OFDM symbol to use a frame of reference in establishing the offset. Some methods are functional over the entire possible range of offset, while others only operate within very small margins. Other methods work in tandem to solve

both timing and frequency synchronization together, either through a single algorithm or tightly grouped sets of algorithms.

If there is no interest in saving space on the chipset and the most important concern is performance, then dedicated multiresolution methods– i.e., two or more algorithms with different ranges of optimality solving the same (either timing or carrier frequency) problem make the most sense. Otherwise, single algorithms that operate over the entire range are best.

Most of the literature classifies these methods ase being either data aided (DA) or non-data aided (NDA), but in a sense, even the algorithms that do not use specifically known data points take advantage of the known nature of OFDM modulation, or the known coefficients of the matched filters in the interpolator and decimator.

5.3 Separation of Symbol and Carrier Synchronization

The coherent reception of a digitally modulated signal requires that the receiver be synchronous to the transmitter. In such a system, there are two basic tasks of synchronization:

1. When coherent detection is used, as in OFDM systems, knowledge of both the frequency and phase of the carrier is necessary. The estimation of carrier phase and frequency is called **carrier recovery**, **carrier synchronization**, or **frequency synchronization.**

2. To perform demodulation, the receiver must know the instants of time at which the modulation can change it state, i.e., the end of one transmitted signal and the beginning of another. Knowing the symbol start and finish times is necessary to sample the signal in the right places and to correctly decode the data into the proper format. The estimation of these times is called **timing recovery**, **symbol synchronization**, or **frame synchronization**.

The two modes of synchronization either occur simultaneously or sequentially, depending on the scheme being employed.

The relationship between timing and frequency errors in OFDM is strong– manipulations to diminish the effects of timing errors lead to magnification of frequency errors, and vice-versa. For example, if one were to lower the number of subcarriers (thus increasing subcarrier spacing) for the purpose of reducing frequency offset problems, it will increase the demands on timing synchronization (for relatively smaller OFDM symbols, the same timing offset is a relatively larger portion of the total bandwidth).

5.4 Fine Frequency Method Using training OFDM symbol

[7] describes a technique to estimate frequency offset using a related data symbol. A maximum likelihood estimator (MLE) algorithm is derived and its performance computed and compared with simulation results. Since the intercarrier energy and signal energy both contribute coherently to the estimate, the algorithm generates very accurate estimates even when the offset is far too great to demodulate the data values. Also, the estimation error depends only on total symbol energy so it is insensitive to channel spreading and frequency selective fading. The limit of this algorithm is $\frac{1}{2}$ of the subcarrier spacing.

As detailed in Chapter 3, frequency offset in the channel causes

1. reduction of signal amplitude in the output of the filters matched to each of the carriers
2. introduction of ICI (intercarrier interference) from the other carriers which are now no longer orthogonal to the filter.

The tolerable frequency offset in OFDM systems is a very small fraction of the channel bandwidth.

5.5 Frequency Detector Utilizing CP structure

One of the fundamental functions of an OFDM receiver is the carrier frequency synchronization. A residual frequency error between the transmitter and receiver local oscillators leads to a loss of orthogonality between the different subcarriers making up the OFDM signal, giving rise to a degradation on the overall system performance.
The traditional way to to correct frequency error has been using tracking loops in which a frequency detector (FD) provides an estimate of the frequency error to be compensated for. Normally based on the analysis of the signal at the output of the FFT at the receiver side, they are one of two types:

1. Algorithms based on the analysis of special synchronization blocks (pilots) embedded in the OFDM temporal frame
2. Algorithms based on the analysis of the received data at the output of the FFT

The first category presents many drawbacks: The acquisition time is long due to the infrequency of synchronization blocks in the data stream, and high computational complexity is needed to remove nonlinearity problems and multipath channel effects. The second category needs no special blocks, but performs poorly in the presence of multipath propagation.

[10] algorithm introduces a carrier frequency detector that overcomes those drawbacks. It uses the inherent redundancy created by the cyclic prefixing in OFDM systems to estimate the offset.

The lth block of the received baseband signal will consist of $N+v$ samples, corresponding to the guard interval and the useful block. To see how this carrier recovery method works, consider the case of a channel with AWGN. When the frequency error Δf is null, we have $r_{N-i} = r_{-i}(i = 1, 2, \ldots, v)$ and therefore the product $r_{N-i}r^*_{-i}$ is a real number.

On the other hand, in the presence of frequency error the two samples r_{N-i} and r^*_{-i} are affected by a different rotation, but the imaginary part of their product contains some information about the frequency offset. The basic idea of the algorithm is to use this information. The error signal $\epsilon(l)$ corresponding to the lth block is given by

$$\epsilon(l) = \frac{1}{L}\sum_{i=1}^{L} Im[r_{N-i}r^*_{-i}] \; (1 \leq L \leq v)$$

where L is the number of guard interval samples taken into account. Over here I will be putting results in that compares this to other fine frequency detectors.

5.6 Dual Frequency and Timing Method Using Data Symbols

[18] describes a synchronization method using either a continuous transmission or a burst operation over a frequency-selective channel. The presence of a signal can be detected upon the receipt of just one training sequence of two symbols. The start of the frame and the beginning of the symbol can be found, and carrier frequency offsets of of many subchannels spacings can be corrected. The algorithms operate near the Cramér-Rao lower bound for the variance of the frequency offset estimate, and the inherent averaging over many subcarriers allows acquisition at very low SNR levels.

5.6.1 Estimation of Symbol Timing

The symbol timing recovery relies on searching for a training symbol with two identical halves in the time domain, which will remain identical after passing through the channel, except that there will be a phase difference between them caused by the carrier frequency offset. The two halves of the training symbol are made identical (in time order) by transmitting a pseudonoise (PN) sequence on the even frequencies, while zeros are used on the odd frequencies. This means that at each even

frequency one of the points of a QPSK constellation is transmitted.

In order to maintain an approximately contant signal energy for each symbol the frequency components of this training symbol are multiplied by $\sqrt{2}$ at the transmitter, or the four points of the QPSK constellation are selected from a larger constellation, such as 64-QAM, so that points with higher energy can be used. Transmitted data will not be mistaken as the start of of the frame since any actual data must contain odd frequencies. Note that an equivalent method of generating this training symbol is to use an IFFT of half the normal size to generate the time domain samples. The repetition is not generated using the IFFT, so instead of just using the even frequencies, a PN sequence would be transmitted on all of the subcarriers to generate the time domain samples which are half a symbol in duration. These time-domain samples are repeated (and properly scaled) to form the first training symbol.

The second training symbol contains a PN sequence on the odd frequencies to measure these subchannels, and another PN sequence on the even frequencies to help determine frequency offset. The selection of a particular PN sequence should not have much effect on the performance of synchronization algorithms. Instead, the PN sequence can be chosen on the basis of being easy to implement or having a low peak-to-average power ratio so there is little distortion in the transmitter amplifier.

Complex samples r_m are taken by mixing the received signal down to IF, splitting the signal into two branches, multiplying both by the in-phase and quadrature local oscillators, and low-pass filtering and sampling to get baseband in-phase and quadrature components. This can be expressed mathematically by writing the IF local oscillator for the in-phase branch as

$$v_i(t) = 2cos(2\pi f_{IF} t)$$

and the IF local oscillator for the quadrature branch at the receiver as

$$v_q(t) = 2sin(2\pi f_{IF} t).$$

Let the output of the mixer after down-conversion be $s(t)$. The demodulated signal before the sampler can be expressed as

$$r(t) = LPF\{s(t)v_i(t)\} + jLPF\{s(t)v_q(t)\}$$

where LPF{} means to low-pass filter the terms in the argument. The output of the in-phase branch

is considered to be real and the output of the quadrature branch is considered to be imaginary. This is a mathematical convention to represent the in-phase and quadrature components as a complex number. After sampling, the complex samples are denoted as r_m.

Consider the first training symbol where the first half is identical to the second half (in time order), except for a phase shift caused by the carrier frequency offset. If the conjugate of a sample from the first half is multiplied by the corresponding sample from the second half($T/2$ seconds later), the effect of the channel should cancel, and the result will have a phase of approximately $\Phi = \pi T \Delta f$. At the start of the frame, the products of each of these pairs of samples will have approximately the same phase, so the magnitude of the sum will be a large value.

Let there be L complex samples in one-half of the first training symbol (excluding the cyclic prefix), and let the sum of the pairs of products be

$$P(d) = \sum_{m=0}^{L-1} (r^*_{d+m} r_{d+m+L})$$

which can be implemented with the iterative formula

$$P(d+1) = P(d) + (r^*_{d+L} r_{d+2L}) - (r^*_d r_{d+L}).$$

Note that d is a time index corresponding to the first sample in a window of $2L$ samples. This window slides along in time as the receiver searches for the first training symbol. The received energy for the second half-symbol is defined by

$$R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2$$

which can be calculated iteratively. $R(d)$ may be used as part of an automatic gain control (AGC) loop. A timing metric can be defined as

$$M(d) = \frac{|P(d)|^2}{(R(d))^2}.$$

Schmidl and Cox calculate the timing metric. It reached a plateau which has a length equal to the length of the guard interval minus the length of the channel impulse response (since there is no ISI within this plateau to distort the signal). For the AWGN channel, there is a window with a length of the guard interval where the metric reaches a maximum, and the start of the frame can be taken to be anywhere within this window without a loss in the received SNR. For the frequency selective channels, the length of the impulse response of the channel is shorter than the guard interval by

design choice of the guard interval, so the plateau in the maximum of the timing metric is shorter than for the AWGN channel.

This plateau leads to some uncertainty as to the start of the frame. Schmidl and Cox generated OFDM symbols with 1000 frequencies, -500 to 499. They are slightly oversampled at a rate of 1024 samples for the useful part of each symbol. In an actual hardware implementation, the ratio of the sampling rate to the number of frequencies would be higher, to ease filtering requirements. The guard interval is set to about 10% of the useful part, which is 102 samples.

5.7 A Data-Aided Improved Frequency Offset Estimator

[21] extends the use of Schmidl and Cox's algorithm for carrier frequency estimation in OFDM systems, which basically required the transmission of a training symbol composed of two identical halves in the time domain. Morelli and Mengali extend the alorithm by considering the use of a symbol with $L > 2$ identical parts: this way, better accuracy is achieved, with a small increase in computational complexity. The scheme also obviates the need for a second training symbol, which was required in Schmidl and Cox's original method.

Morelli and Mengali extend SCA (the Schmidl and Cox algorithm) by dividing the first training symbol into $L > 2$ identical parts. This way, the estimation range can be made equal to $\pm\frac{L}{2}$ the subcarrier subspacing with no loss in accuracy. Thus, by making L sufficiently large it is possible to achieve the desired estimation range with an overhead reduction of 50% with respect to the SCA (because one, rather than two, pilot training symbols is used).

The estimation of v is based on the observation of a symbol consiting of L identical parts. They are generated by transmitting a pseudonoise seqence on the frequencies multiple of $\frac{L}{T}$ and setting zero on the remaining frequencies.
Advantages over Schmidl and Cox: slightly superior estimation range; needs just one training symbol. However, the problem still remaining with this system is that, like its predecessor, it can only be used for PSK constellations, which are inherently disadvantageous due to their susceptibility to errors resulting from phase offset.

5.8 High-Efficiency Carrier Estimator Using Subspace Methods

While multipath induced phase rotations can be dealt with using differential encoding in OFDM communications, the loss of orthogonality due to the carrier offset must be compensated before DFT-based demodulation can be performed. In [20], high-perfomance/low-complexity blind carrier offset estimation algorithm is presented, which exploits intrinsic structure information of OFDM signals. The algorithm offers the accuracy of a super resolution subspace method without involving computationally intensive subspace decompositions.

Carrier estimation in OFDM techniques previous to this relied on periodic transmission of reference symbols, which inevitably slightly reduces the bandwidth efficiency. [18] introduced a blind approach which is capable of estimating the carrier offset toward its closest subcarrier. In addition to bandwidth saving, the technique also provides the system certain robustness or self-starting capability against channel breakdowns and rapid environmental changes. However, the algorithm requires the constellation on each subcarrier to have points equally spaced in phase— i.e., a PSK constellation. Moreover, the length of the guard interval must be chosen from a subset of allowed values.

Liu and Turelli's solution does not use any pilots. It provides a high-accuracy carrier estimate by taking advantage of the inherent orthogonality among OFDM subchannels. Even when the OFDM signal is distorted by an unknown carrier offset, the received signal possesses an algebraic structure which is a direct function of the carrier offset. From this, a closed-form estimate of the carrier offset is obtained.

This system compared favorably with other frequency estimators.

5.9 Frequency Ambiguity Resolution in OFDM Systems

In OFDM systems, the carrier-frequency offset can be divided into two parts: 1) an integer one—multiple of the subcarrier spacing $\frac{1}{T}$ and 2) a fractional one—less than $\frac{1}{2T}$ in amplitude. Some schemes proposed in the literature can only recover the fractional part. [25] proposes two algorithms for estimating the integer part. They are based on the observation of two consecutive OFDM symbols and are based on maximum-likelihood techniques. The first algorithm exploits pilot symbols multiplexed with the data, the second is blind.

OFDM's main weakness is its sensitivity to oscillator instabilities. Depending on the application,

the offset can amount to several multiples of the subcarrier spacing $1/T$ and is usually divided into an *integer part*, multiple of $1/T$, and a *fractional part*, less than $1/2T$ in amplitude. If not properly compensated, the former results in a shift of the subcarrier indices while the latter produces inter-carrier interference due to loss of orthogonality between subcarriers.

Several schemes proposed in the literature can only recover the fractional offset. Hence, their estimates are ambiguous by multiples of $1/T$. The previous method proposed by [7] to overcome this limitation involves a considerable degradation in estimation accuracy. The feedforward scheme discussed in [18] makes use of pilot symbols and provides the overall offset (integer plus fractional part).

5.10 Blind Dual Carrier and Timing Method Employing Pulse Estimation

[26] has a blind algorithm for the estimation of symbol timing and carrier frequency offset: if the OFDM system in question employs pulse shaping, then there is an class of estimators that can be used for carrier frequency estimation [Boelcskei]. This class is able to do carrier frequency acquisition over the entire bandwidth of the OFDM signal. It can be applied even in the absence of a cyclic prefix. The estimator makes use of 1) knowledge of the transmitter pulse shaping filter and 2) the cyclostationarity of received OFDM signal.

5.11 Pilot-Based Timing Offset Detection Scheme

[4] cites a pilot-based algorithm, in which the OFDM signal is transmitted by FM. The transmitter encodes a number of reserved subchannels with known phases and amplitudes. Such an algorithm would consist of coarse synchronization followed by fine synchronization.

The coarse synchronization acquires synchronization alignment to within ± 0.5 samples: not acceptable for demodulation, but suitable enough to simplify the tracking algorithm. The coarse synchronization is done by correlating the received signal to a copy of the transmitted synchronization signal. To find the peak of this correlation with enough accuracy, a digital filter is used to provide the interpolated values at four times the original data rate.

In the second phase (fine synchronization), the subchannels with pilots are equalized with the estimated channel obtained from pilots. Since the coarse synchronization guarantees that the timing

error is less than ± 0.5, the subchannels are due to timing error and can be estimated using linear regression.

5.12 Frequency Estimators

Several carrier synchronization schemes have been suggested in the literature. As with symbol synchronization, they can be divided into two categories– those based on the pilots and those based on the cyclic prefix.

Pilot-aided algorithms have been addressed in [6]. In that work some subcarriers are used for the transmission of pilots, which are usually some pseudonoise sequence. Using these known symbols, the phase rotations caused by the frequency offset can be estimated.

If we can assume that the frequency offset is less than half the subcarrier spacing, there is a $1-1$ mapping between the phase rotations and frequency offset. Hence, we need an algorithm to achieve this.

Such an algorithm can be constructed by forming a function which is sinc-shaped and has a peak for $f - \hat{f} = 0$ ([6]). Evaluating this function at points $\frac{0.1}{T}$ apart, an acquisition could be obtained by maximizing that function.

A related technique is to use the cyclic prefix. If one thinks of the pilots as basically being the transmission of known symbols so that frequency and timing correction can be done, then in effect, the cyclic prefix is pilotlike itself, as it gives an estimate of the signals to follow N points forward (i.e., the data from which the cyclic prefix was copied).

There are different ways to use this redundancy:

1. Create a function that peaks at the zero offset location, and find its value.
2. Calculate a maximum likelihood estimation: this approach, in [11] derives a likelihood function for both timing an frequency offsets, by assuming a non-dispersive channel and by considering the transmitted data symbols x_k uncorrelated. By maximizing this function, a simultaneous estimate of timing and frequency offsets can be obtained. If the frequency error is slowly varying compared to the OFDM symbol rate, a PLL can be used to reduce the error further.

5.13 Symbol Timing Offset Estimation in Coherent OFDM Systems - van de Beek, Borgesson

The authors of [16] came up with a method to perform timing offset estimation in coherent OFDM systems. The method uses both the cyclic prefix redundancy and pilots in the signal (which must be equi-spaced) to perform the algorithm. Previous to this, they developed an ML estimator based on these that didn't take into account dispersion, frequency offset, or signal correlation. But in a second attempt, they created an ad hoc estimator out of it that is robust against frequency offsets as well– putting this estimator in the class of simultaneous time/frequency methods.

The $N - N_p$ transmitted data subcarriers can be denoted by

$$s(k) = \frac{1}{\sqrt{(N)}} \sum_{n=0, n\neg\Pi}^{N-1} x_n e^{\frac{-j2\pi kn}{N}}, \tag{5.1}$$

where x_i is the data being modulated by the ith subcarrier. Π is the set of pilot tones.

The N_p pilot subcarriers are

$$m(k) = \frac{1}{\sqrt{(N)}} \sum_{n\in\Pi}^{N-1} p_n e^{\frac{-j2\pi kn}{N}}, \tag{5.2}$$

where p_i is the pilot symbol transmitted on the ith subcarrier. Together, these two make up the transmitted OFDM symbol. The average energy of the symbol is $\sigma_x^2 = E(|x_n|^2)$.

In the work, they computed the autocorrelation function for $N = 128$, $v = 16$, and 1 pilot every 5 subcarriers. The autocorrelation of $m(k)$ has prominent peaks representing locations of possible offset amounts.

If we make the AWGN assumption on the data for modelling purposes, then we can model the received signal as $r(k) = s(k - \theta) + m(k - \theta) + n(k)$, where θ is the integer time offset and $n(k)$ is AWGN with variance of σ_n^2.

We can use the statistical properties of $s(k)$ and the knowledge of the pilots $m(k)$ to derive an estimator:

For $N_p \ll N$, $s(k)$ statistically is similar to a discrete time Gaussian process, and we can as-

sume that $s(k)$ has a variance $\alpha\sigma_x^2$ where $\alpha = \frac{N-N_p}{N}$. The addition of the cyclic prefix makes $s(k) = s(k+N)$ and $m(k) = m(k+N)$ for $k \in (0 \ldots L-1)$.

Since n is zero-mean Gaussian and $m(k)$ is known at the receiver, $r(k)$ is Gaussian with a time-varying mean $m(k)$ and variance $\alpha\sigma_x^2$.

The assumption models the system as one in which the only two defects present are AWGN and the defect being studied, namely integer timing offset.

Define the autocorrelation function

$$r(k) = c_r(k,l) = \left\{ \begin{array}{l} 1, k-l = 0 \\ \rho, k-l = -N, k \in [\theta \ldots \theta+v-1] \\ \rho, k-l = N, l \in [\theta \ldots \theta+v-1] \\ 0 \quad otherwise \end{array} \right\}$$

SNR is defined as the ratio signal power to noise power, or $\frac{\sigma_x^2}{\sigma_n^2}$. ρ is $\frac{\alpha\sigma_x^2}{\alpha\sigma_x^2+\sigma_n^2} = \frac{\alpha SNR}{\alpha SNR+1}$.

Now we will develop the ML estimator. It involves investigating the log-likelihood of θ, which is the joint probability of received samples $r(\cdot)$ given θ.

$$\Lambda(\theta) = Pr\{r(\cdot)|\theta\}$$

$$\hat{\theta}_{ML} = \arg\ \max(\theta)\{\Lambda(\theta)\}, \Lambda(\theta) = \rho\Lambda_{cp}(\theta) + (1-\rho)\Lambda_p(\theta)$$

The cyclic prefix MLE correlates samples spaces N samples apart:

$$\Lambda_{cp}(\theta) = \mathrm{Re}\left\{ \sum_{k=\theta}^{\theta+v+1} r^*(k)r(k+N) \right\} - \frac{\rho}{2} \sum_{k=\theta}^{\theta+v-1} (|r(k)|^2 + |r(k+N)|^2)$$

The pilot-based MLE contains a filter matched to the pilots:

$$\Lambda_p(\theta) = (1+\rho)\mathrm{Re}\left\{ \sum_k r^*(k)m(k-\theta) \right\} - \rho\mathrm{Re}\left\{ \sum_{k=\theta}^{\theta+v-1} (r(k)+r(k+N))^* m(k-\theta) \right\}$$

This system works in this way: the ML estimator is based mostly on the cyclic prefix when the SNR is high, and when the SNR is low, the ML estimator is based mainly on the pilots.

Now, what we can do with this information is use it to build a *robust estimator*. This will take into account the frequency offset ϵ. The authors of [16] simulate the system with $N = 128, L = 16, \frac{1}{32}$ pilot ratio, and an SNR of 10. They found that even when $\epsilon > .2\%$, the reference estimator using only Λ_{cp} works better.

This is because the frequency offset makes the peaks appear in a random manner. Hence, if you take the absolute value (rather than the real part) of the terms in the log-likelihood function, it preserves the constructive contributions to $\Lambda(\theta)$. In this way, it compensates for the unknown frequency offset.

Since the SNR is unknown at the receiver, we design a generic estimator that assumes a fixed SNR ($S\tilde{N}R$). We choose a design SNR lower than the typical SNR in the system, so that the low-end SNR's are satisfied. By the relation mentioned earlier, it effectively increases the contribution of the pilots. In the equations below, $\tilde{\rho} = \frac{\alpha \tilde{\text{SNR}}}{\alpha \tilde{\text{SNR}}+1}$:

$$\hat{\theta}_{ML} = \arg\max(\theta)\{\tilde{\rho}\tilde{\Lambda}_{cp}(\theta) + (1-\tilde{\rho})\tilde{\Lambda}_p(\theta)\}$$

$$\tilde{\Lambda}_{cp}(\theta) = \left|\sum_{k=\theta}^{\theta+v+1} r^*(k)r(k+N)\right| - \frac{\tilde{\rho}}{2}\sum_{k=\theta}^{\theta+v-1}(|r(k)|^2 + |r(k+N)|^2)$$

$$\tilde{\Lambda}_p(\theta) = (1+\tilde{\rho})\text{Re}\left|\sum_k r^*(k)m(k-\theta)\right| - \tilde{\rho}\text{Re}\left|\sum_{k=\theta}^{\theta+v-1}(r(k)+r(k+N))^*m(k-\theta)\right|$$

In this system, applying a 10 dB working SNR leads to a robust loss of .3 dB, an ML loss of 1.3 dB, with a reference dB of 1.7 dB, in a situation where the cyclic prefix and the channel length are equivalent.

With unevenly spaced pilots, it is possible to lower the peaks surrounding the symbol start, making one peak the obvious symbol start, even for the simple estimator $\Lambda_p(\theta)$.

5.14 Carrier Frequency Synchronization

OFDM carriers exhibit orthogonality on a symbol interval if they are synthesized such that they are spaced in frequency exactly at the reciprocal of the symbol interval, which the FFT effects correctly. However, the channel causes a frequency offset. It reduces the output of the filters matched to each of the carriers, and it also introduces ICI(intercarrier interference) from the other carriers, which are no longer orthogonal to the filter. Because in OFDM the carriers are inherently closely spaced in frequency compared to the channel bandwidth, the tolerable frequency offset is a very small fraction of the channel bandwidth.

The rest of this section summarized the key parameters of several carrier synchronization algorithms.

5.14.1 MLE algorithm (Moose)

In [7] Paul H. Moose developed an algorithm to estimate frequency offset from the demodulated data signals in the receiver. The technique involves repetition of a data symbol and comparison of the phases of each of the carriers between the successive symbols.

Since the modulation phase values are not changed, the phase shift of each of the carriers between successive repeated symbols is due to the frequency offset.

The frequency offset is estimated using a maximum likelihood estimate (MLE) algorithm. If the frequency offset exceeds $\pm\frac{1}{2}$ the intercarrier spacing, then a separate algorithm/strategy is required to achieve the initial acquisition. Hence, it can be classified as a *fine* frequency synchronization algorithm.

summary: limited range ($\pm\frac{1}{2}$ the subchannel width), overhead of symbol repetition.

5.14.2 Algorithm Using Cyclic Correlation (Sandell, Van de Beek, Borjesson)

This algorithm, which makes use of of the redundancies inherent in the cyclic prefix, was written about in 1995. It performs a correlation with the cyclic prefix to find the symbol timing and an estimate of the carrier frequency offset if it is less than half a subcarrier spacing.

summary: limited range ($\pm\frac{1}{2}$ the subchannel width).

5.14.3 Blind Algorithm (Schmidl and Cox)

T.M. Schmidl and D.C. Cox, in 1997, wrote about an algorithm to perform blind synchronization over a wireless frequency selective channel.

The algorithm has a larger acquisition range than previous algorithms, such as [Mo94], if the number of subchannels is greater than 4, which almost always the case. No training data is required for this method.

The algorithm is hybrid: It uses [SaVaBo95] to find the fractional frequency offset. After modulating by that amount (i.e. after multiplying by $e^{\frac{-j2\pi t\mu}{T}}$, where μ is the fractional offset), only the integer frequency offset remains.

Next, as in [Mo94], an MLE can determine the phase difference (by first multiplying the phase difference between carriers so that it is a multiple of 2π, and then applying the MLE).

The paper shows that the synchronization information can be derived by taking samples from any window with a length of about three symbols– so it is guaranteed that two full symbols are contained. Thus there is no delay incurred by having to wait for the beginning of a symbol.

Another advantage is that since there is no training data, the amount of computation needed for synchronization is lower because the frequency and timing acquisition algorithms can be performed once instead of continuously performing opeerations to look for training data.

The problem here is that the constellation must be some variety of PSK– with equal spacing in phase on the constellation. This means that it cannot be used for a BWIF-compliant chipset. However, this problem can be overcome by the use of a synchronization symbol, with points corresponding to the subset of square constellation points falling on a PSK constellation, as shown below:

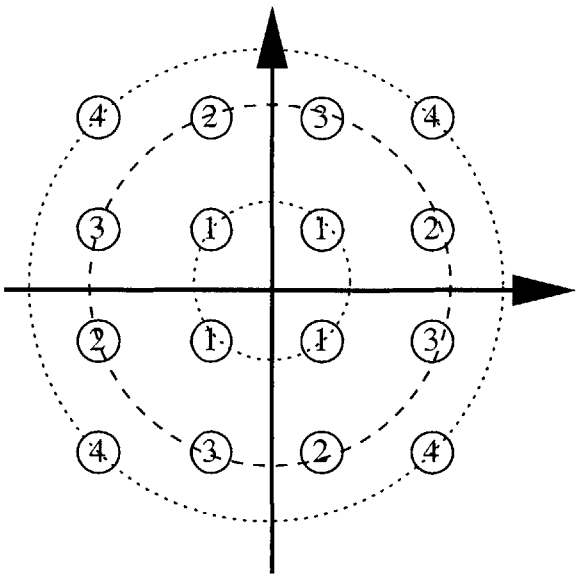


Figure 5.2: PSK Subconstellations of QAM 16

This, of course, reintroduces training data in the form of this synch symbol, and would hence eliminate the ability to begin synchronization immediately.

summary: range is $(\pm\frac{1}{8})$ of the total channel bandwidth, constellation (for acquisition) must be points equally spaced in phase, length of guard interval must be from a specified set, no training data is needed.

5.15 Timing Synchronization

5.15.1 Algorithm Using Cyclic Correlation/Pilots (Landstrom, van de Beek, Borjesson)

This algorithm exploits both cyclic redundancy and the presence of pilot tones in the OFDM stream to obtain timing synchronization.

The timing offset estimator's requirements are determined by the difference between the length of the cyclic prefix and the length of the channel response. As long as the offset is less than this difference, the orthogonality of the carriers is preserved, and a time offset within this interval only results in a phase rotation of the subcarrier constellations.

If the system is a coherent, pilot-based system, the phase shift will be compensated with a frequency domain channel equalizer.

The paper derives an ML estimator of the time offset θ by investigating the log-likelihood function of θ, i.e. the joint probability of the received samples $r(\cdot)$ given θ (assuming an AWGN channel):

$$\Lambda(\theta) = Pr\{r(\cdot)|\theta\}$$

The algorithm proposed here takes advantage of both the MLE for the pilots and the MLE for the cyclic prefix correlation. The former is a group of spikes with similar amplitude (including an exact spike at the true timing offset), and the latter has a rough peak near the true timing offset. Combining the coarse absolute definition of MLE_{CP} and the local fine resolution of MLE_P, we get a clear indicator of the timing offset.

summary: promising choice for implementation because it uses pilots and cyclic prefix, both of which are specified in the BWIF spec. No reliance on a particular constellation or FFT size is required.

5.16 An Integrated Scheme for Timing and Frequency Synchronization

The problem of synchronization can be solved in a multistage approach, so that different approaches can be integrated and can be used in their optimal ranges. The particular scheme discussed below was put together by Sunder Kidambi, and can also be found in the common literature.
Note that the tasks of frequency and timing sync are coupled, unlike Pollet and Peeters' scheme for DMT.

5.16.1 Coarse Timing/Frame Synchronization

In coarse timing recovery, the objective is to determine the starting point and ending point of a data frame (OFDM symbol) in the received data stream. This step is necessary for further demodulation, as any offset will result in errors in fine carrier frequency synchronization.

One method of coarse timing recovery involves taking advantage of the repetition of the cyclic prefix in the frame: In a transmitted frame with $N+v$ symbols, the first v and last v symbols are the same as a result of appending the cyclic prefix of length greater than the channel response to maintain orthogonality. The algorithm takes a symbol r_i and the N-delayed symbol r_{i-N}, multiplying the conjugate of the former by the latter. These products are taken on every cycle. The v-length moving average of the output of this multiplier is calculated, and its magnitude is stored in an array of length $2N+v$ (which is cleared each time it fills). When the array is filled, the argument of the

maximum element is stored in a second array, which keeps track of these maxima. After taking an arbitrary number of these maxima, the average is taken and rounded to the nearest integer: this value represents the coarse frame offset, or the number of symbols by which the input must be delayed to realign it with the frame boundary.

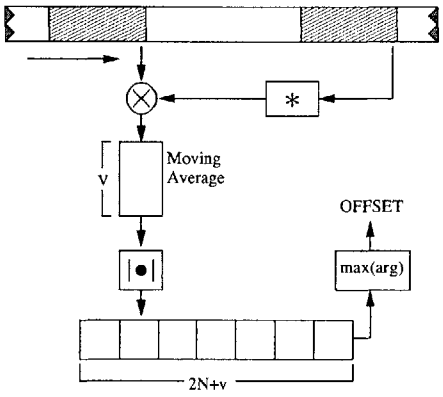


Figure 5.3: Coarse Timing Estimation

5.16.2 Fine Carrier Frequency Synchronization

This algorithm determines the fractional frequency offset of the received data. As in the previous algorithm, this one takes advantage of the repetitive nature of the stream due to the prepending of the cyclic prefix. Here, r_{N-i} and r^*_{-i} are multiplied, and the imaginary part is extracted (note that in the absence of noise, this imaginary part should be 0 if r_{-i} is part of the cyclic prefix). A moving average is taken for these imaginary parts, and passed through a first-order dpll (digital phase-locked loop). This output settles to the fractional frequency offset: once found, the remainder of the received input can be multiplied by $e^{-j2\pi t\Delta w}$ to remove this offset.

5.16.3 Coarse Carrier Frequency Synchronization

The coarse carrier recovery algorithm is only performed if it can be assumed that the frequency offset is an integer number of subcarriers. Hence, it is triggered only when the fine carrier offset error ϵ drops below a certain threshold ϵ_d.

In OFDM, data carriers occupy roughly $80 - 85\%$ of the modulation bandwidth, with the rest of the symbol composed of "null carriers," with a value of zero magnitude. The primary reason for this is so that they can be coupled with roll-off filters, to assure that there is no ISI.

This method of frequency synchronization makes additional use of these "null carriers" to detect the frequency shift as follows: it takes the sum of the square of the symbol magnitudes, over the

1704 symbols with data. It slides a window over the 2048 samples, calculating the magnitude of all the data in it. Clearly, where this "energy" is highest, the window covers the 1704 samples that are the non-null carriers. The offset of the window at that point is the coarse frequency offset.

The algorithm works for a shift of ± 100 samples, or $\pm 5\%$.

5.16.4 Fine Timing/Frame Synchronization

The coarse timing recovery algorithm is useful for finding the integer number of samples to delay by to recover the frame; however, estimating and recovering the fractional part is the job of the fine timing recovery mechanism. Below is the algorithm listed with simple linear interpolation; a more accurate result can be obtained using quadratic, cubic, or quartic interpolation.

The receiver will want to sample the data at a desired sampling point, but it will have to instead average its value from the neighboring samples x_R and x_{R-1}. If the distance between the samples is normalized to 1 and the distance to sample x_R is u, then the estimate $y_K = u_R x_R + 1 + (1 - u_R) x_R$.

(Continue with clock asynchrony, interpolation methods)

Because we are dealing with narrowband signals, it is appropriate to employ interpolation techniques to determine the fine timing offset and adjust for it.

When describingthis technique, it is important to note that a distinguishing characteristic of this broadband wireless system is that the transmitter and receiver clocks are not expected to be synchronized– rather, the receiver clock

On the transmitter side, the

5.17 ISI

If there is ISI in the channel, it is in general caused by dispersive channel. Previously transmitted symbols will cause non-zero contributions at subsequent sampling instants. The synchronization of an OFDM transmitter and receiver is important because OFDM systems are in general more sensitive to time and frequency offsets than single-carrier systems [PoBlMo95]. Not only may synchronization errors cause intersymbol interference (ISI), they can also cause a loss of the orthogonality between the subcarriers (ICI).

It appears that most systems that avoid the use of a cyclic prefix or pilot tones must resort to using pulse shaping and cyclostationarity for estimation.

5.18 Cyclic Prefix-Based Synchronization Algorithms

The main idea of this scheme has already been detailed in this work– namely, that the difference between samples r_i and r_{i-N} should be small if one of them belongs to the cyclic prefix and the other to the symbol from which it is copied. Otherwise, this difference would have twice the power, since it would be the difference of two uncorrelated random variables. Windowing this difference with a rectangular window of the same length as the cyclic prefix gives an output signal with a minimum value at the start of a new OFDM symbol.

To elaborate on this, one can look at the likelihood function given the observed signal $r(k)$ with a timing and frequency error. This function is maximized to simultaneously obtain estimates of both the timing and frequency offsets. Looking at a system with only timing offset (θ), it is

$$\Lambda(\theta) = \sum_{k=\theta}^{\theta+v-1} \frac{2}{SNR+1} \Re\{r_i r_{i-N}^*\} - \frac{SNR}{SNR+1} |r_i r_{i-N}|^2$$

For medium and high SNRs ($\gg 1$) an ML (maximum likelihood) estimator based on $\Lambda(\theta)$ applies a moving average to the term $|r_i r_{i-N}|^2$. But for small SNR values, the crosscorrelation term $r_i r_{i-N}^*$ must be taken into account as well.

Random Frequency Method

To cancel this source of interference, carrier frequency offsets must be compensated before the DFT at the receiver. A first possibility is to use a decision-directed carrier recovery technique. The phase detector employs the instantaneous receiver decision and the corresponding error signal. After passing through the loop filter, its output drives a voltage-controlled oscillator(VCO) which delivers the recovered carrier to the demodulator. The problem associated with this technique is that the excessive delay due to the presence of an N-point DFT in the loop will seriously affect its stability, increase its steady-state phase jitter, and reduce its acquisition range.

In practical applications, this difficulty is circumvented by resorting to a "pilot tone" carrier synchronization technique.

Such technique is easily implemented in OFDM systems by transmitting an unmodulated carrier at the channel center, with a number of virtual (zero-valued) carriers on both sides.

On the receiver side, the received signal is passed to a bandpass filter whose nominal center

frequency is equal to that of the pilot carrier, and the filter output drives a phase-locked loop (PLL) that supplies the recovered carrier to the demodulator. This solution alleviates carrier synchronization, but reduces the spectral efficiency and the power efficiency of the system. The loss in spectral efficiency is proportional to the number of virtual carriers, a parameter directly related to the frequency uncertainty of the oscillators used.

5.19 Implementation Costs

What are the costs of associating these different algorithms? Most of the research is concerned with the development of more reliable and quicker schemes, yet little has been done in the professional literature to address the actual implementation cost, in terms of hardware. For example, the presence of an FFT in the feedback loop can be prohibitively slow, given the need for its execution at every call for synchronization. Algorithms that require the usage of maximization of an aray work in O(n) time at best on the data set...to require this usage in more than one occassion, while requiring that they work serially, leads to a large increase in the time needed to perform the full algorithm.

These concerns are highlighted in non data-aided schemes, as the synchronization is often occurring on every symbol. Such a system often cannot wait for estimation results while receiving, and must instead obtain an estimate "on-the-fly".

Area is another important factor to discuss. Hardware FFT implementations are (include report).

5.20 Comparison of Various Timing and Carrier Offset Detection and Recovery Methods

The scientific literature includes a multitude of methods with regard to recovering offset in frame/timing and carrier/frequency in multicarrier communication. The algorithms differ among various factors, including computational complexity, time required to obtain a satisfactory solution, and area needed in silicon/program memory to implement the solution.

Chapter 6

Conclusions

There is great benefit to the use of a pilot signal–an advantage that purely blind estimation algorithms cannot overcome in data throughput. As we have seen, the pilot is used in both the timing/frequency estimation algorithms, as well as the purpose for which it was originally intended (channel estimation). Different sources have suggested that the cyclic prefix can be used in much the same manner as pilots for recovery, and that issue is a debate between synchronization performance requirements, channel conditions and requirements, and the desire for optimal bandwidth utilization.

It is interesting to note that virtually any data added to perform any function to the OFDM signal can be used to aid in synchronization. The synchronization problem, thus, only appears to be of secondary importance to other problems (such as estimation and equalization), due to the ingenuity of recovery algorithms to get extra benefit from the data vestiges of other data-aided algorithms.

As the results of this study show, there is a unique optimal receiver depending on the channel conditions, modulation bandwidth, and hardware cost requirements. Moreover, there is an optimal receiver for each standard—hence, the groups driving those standards forward have a great deal of control over the class of methods that can be explored.

So the design choices in the creation of an OFDM system are primarily the type of QAM modulation, the length of the cyclic prefix, the algorithms for synchronization and the algorithms for chanel estimation.

This thesis described, explored, and simulated the performance of OFDM broadband wireless systems, through an examination of system structure, channel conditions, defects and offsets, the modulation parameters, and synchronization algorithms.

Appendix A

Definitions

$$\text{autocorrelation: } \Phi t_1, t_2 = E(X_{t_1} X_{t_2}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_{t_1} x_{t_2} p(x_{t_1}, x_{t_2}) dx_{t_1} dx_{t_2}$$

$$\text{Gaussian Q-function: } Q(x) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} dy$$

cyclostationary process – one which has statistics which vary periodically with time: the mean, autocorrelation, 3rd, 4th,... order cumulants are all periodic function of time. An example of a first-order cyclostationary process is a sinusoid with added Gaussian noise: the noise clearly causes the signal to be random; however if the signal is averaged by matching up points from the same position in the cycle each time (synchronous averaging) then the underlying sinusoid will be recovered. If the variance of the signal is computed in the same way however it will be found to be constant.
fading – attenuation of a signal.

Appendix B

Code

B.1 Main Simulation Shell (main_ofdm.c)

```
/*************************************************************************
(C) 2000 by ANALOG DEVICES all rights reserved.  ** ** main_ofdm.c **
** QAM OFDM Modulator and Demodulator Simulation ** ** ** ** Author :
Veeral Shah, June 30, 2000 ** ** **
*************************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "../equ/define.h"
#include "../equ/defineHPNA.h"
#include "../equ/globals.h"
#include "../equ/macros.h"

#define DRT  4000            /* timing offset array         */
#define DC   1000            /* filter compensation array   */
#define DRCT 4000            /* coarse timing delay array   */
#define MRCT 1000            /* coarse timing mov avg array */
#define DRFC 4000            /* fine carrier delay array    */
#define MRFC 1000            /* fine carrier mov avg array  */

extern void qam_up_filt();
extern void rateconv_quad();
extern void gainInputGen();
extern void freq_shift();
extern void prefilter();
extern void timing_recovery();
extern void matched_filter_2X();
extern void rotation();
extern void awgn();
extern void powlevel();
extern void std();
extern void phznoise();
extern void phztrack();
```

```
extern void amhum();
extern void sine_ingress();

extern double *cyc_pref();
extern double *decyc_pref();
extern void symbgen();
extern void fft_time2();
extern void upfilt();

extern COMPLEX delay1();
extern COMPLEX delay2();
extern COMPLEX delay3();
extern COMPLEX delay4();
extern COMPLEX delay5();
extern COMPLEX delay6();

extern COMPLEX moving_average1();
extern COMPLEX moving_average2();

extern COMPLEX dpll1();
extern void max_array();
extern double sum_array();
extern void carrier_offset1();
extern void carrier_offset2();

extern  double gauss();
extern  double f2f(), i2f();
extern TXFRAME_OUT hpnaTxFrame();
extern CORRELATOR_OUT correlator();
extern MF_DOUBLE_OUT matchFilt();
extern MF_DOUBLE_OUT preFilt();
extern CONV_OUT v_convolve();
extern DELAY_OUT blkDelay(), varDelay1();
extern DCORR_OUT delCorrelator();
extern BEQZ_OUT equalizer();
extern BAGC_OUT agc();
extern PK_OUT peakDetect();
FILE *ferr, *f_prefixed, *f_upfiltered, *f_convolved, *f_decimated,  *fopen();
FILE *fp, *f_snr, *f_temp;
FILE *f_qam_in, *f_qam_out;

main(argc, argv)
int argc;
char *argv[];
{
int nqam, nseed, i, j, k;
int strobe, init;
int ready, ready1, ready2, ready3, ready4, ready5, ready6;
int ready3b, ready3c, ready_awgn, ready7;
int errorcount, newsym, qampower;
int nfactor = 8, num_blocks;
int interpolator_index, deinterpolator_index;
int offsetcount = 0, offsetcount2 = 0, offsetcount3 = 0;
```

```
char *chan_filename, *interp_filename;
char *mfilt_filename, *txfilt_filename, *pfilt_filename;

double ratio_conv, rdel, gain1;
double a1r, a1i, b1r, b1i, c1r, c1i, d1r, e1r, e1i, opr, opi;
double f1r, f1i, cc1r, cc1i, c2r, c2i, g2r, g2i, out2r, out2i;
double p1r, amlevel, amlevop, gainsig, gainqam;
double g3r, g3i, tempdouble;
double awgnoutr, awgnouti;
double snr, fbaud = 6000000.0;
double cc2r, cc2i, sinesnr;
double dist, std1, avg, maxx, minn, pkrms;
double dcorrR, dcorrI, scale;
double pow(), refLevel, gainOutR, gainOutI, anglR, anglI, sqrt();
double symr, symi;
double ratio_timrec = 32.0/32.0;
double ratio_timrec_rx = 32.0/32.0;

/* noise impairments */
double phang=0.0, freq_off_rot=0.0, phz_off=0;
double phznz10k, fmfreq=0.0;
double sinefreq=0.0, sinelevel=0.0, freq_off=0.0;
double sigma = 0.0;
double snr_phang = 0.0, sum_phang_pwr = 0.0;
int passthrunz = 0, number_of_phangs = 0;

/* data arrays */
double *QAMArray, *IFFTArray, *PrefixArray;
double *QAMArray2, *FFTArray, *DePrefixArray;
double *QAMArrayRe, *QAMArrayIm;
double *QAMArray2Re, *QAMArray2Im;
double *sig, *err, *avgsig, *avgerr, *SNRdb;


double ber, qout, iout, outputr, outputi;
int fft_size, prefix_size;
int iter;
int rand32bits;
int qam_ec, qam_bc, qam_ec2, qam_bc2;
double tol;

MF_DOUBLE_OUT  mfOut;
CONV_OUT convOut;


/* timing offset variables */
COMPLEX delay_reg_to[DRT];
COMPLEX presamp, presamp1, samp;
int length_to;

/* delay to compensate for the halfband filters */
COMPLEX delay_compensate[DC];

/* carrier offset variables */
```

```
double offset_co;

/* fine carrier recovery variables */
COMPLEX input_fc, delay_reg_fc[DRFC], ma_reg_fc[MRFC];
double k_dpll;
int init_fc;

/* later:*/
/* introducing the carrier offset block */


if (argc != 9)
  {
    printf("Usage: %s chan_filename con_size nseed ", argv[0]);
    printf("fft_size prefix_size num_blocks\n");
    printf("char *chan_filename\tChannel coef file name\n");
    printf("int nqam\tconstellation size\n");
    printf("int nseed\trandom number seed\n");
    printf("int fft_size\tsize of FFT\n");
    printf("int prefix_size\tsize of prefix\n");
    printf("int num_blocks\tnumber of frames to transmit\n");
    printf("int phznz10k\tin dBc at 10Khz\n");
    printf("int sigma\tsigma for AWGN\n");
    printf("\n");

    exit(1);
  }

chan_filename = *(++argv);
nqam = atoi(*(++argv));
nseed = atoi(*(++argv));
fft_size = atoi(*(++argv));
prefix_size = atoi(*(++argv));
num_blocks = atoi(*(++argv));
phznz10k = atoi(*(++argv));
sigma = atof(*(++argv));

length_to = 5;

sinefreq = 0.5;
sinelevel = pow(10.0, -sinesnr / 20.0); /* assumes unit power signal */
freq_off = 0.0;
/* scale carrier offset */
freq_off_rot = freq_off / ((double) nfactor / ratio_conv);

tempdouble = log((double)nqam) / log(2.0);
qampower = (int)tempdouble;

/* defining tolerance as half the distance between the symbols
           on the constellation.  If the output symbol is different by
           less than tol, it can be recovered without coding */

tol = 1.0 / ((double)(sqrt((double)nqam)));
```

```
QAMArray=calloc(num_blocks*2*fft_size,sizeof(double));
IFFTArray=calloc(2*fft_size,sizeof(double));
PrefixArray=calloc(2*(fft_size+prefix_size),sizeof(double));
QAMArrayRe=calloc(num_blocks*fft_size,sizeof(double));
QAMArrayIm=calloc(num_blocks*fft_size,sizeof(double));

QAMArray2=calloc(num_blocks*2*fft_size,sizeof(double));
FFTArray=calloc(2*fft_size,sizeof(double));
DePrefixArray=calloc(2*(fft_size+prefix_size),sizeof(double));
QAMArray2Re=calloc(num_blocks*fft_size,sizeof(double));
QAMArray2Im=calloc(num_blocks*fft_size,sizeof(double));

sig = calloc(num_blocks*fft_size,sizeof(double));
err = calloc(num_blocks*fft_size,sizeof(double));
avgsig = calloc(fft_size,sizeof(double));
avgerr = calloc(fft_size,sizeof(double));
SNRdb = calloc(fft_size,sizeof(double));

printf("\nAllocation complete\n");

/* load the matched filters - null for brian */

        txfilt_filename = "../../channels/alpha25/txfilt_m4_null.dat";
mfilt_filename = "../../channels/alpha25/mf_m4_null.dat";

init = 1;
strobe = 0;

ferr = fopen("err.dat", "w");

f_prefixed   = fopen("prefixed.dat", "w");
f_upfiltered = fopen("upfiltered.dat", "w");
f_convolved  = fopen("convolved.dat", "w");
f_decimated  = fopen("decimated.dat", "w");
f_snr = fopen("snrdb.dat","w");
f_temp = fopen("temp.dat","w");

f_qam_in = fopen("qam_in.dat","w");
f_qam_out = fopen("qam_out.dat","w");

printf("File open complete\n");

ready = ready1 = ready2 = ready3 = ready3b = ready4 = ready5 = 0;
qam_ec = qam_bc = qam_ec2 = qam_bc2 = ready3c = ready_awgn = 0;
rand32bits = errorcount = ready7 = 0;
symr = symi = cc2r = cc2i = 0.0;

gainqam=0;

/* Initialization of all functions by calling with init = 1 */

convOut = v_convolve(cc2r, cc2i, strobe, chan_filename, init);
printf("Convolve init complete\n");
```

```
printf("init = %d\n",init);

upfilt(symr, symi, &a1r, &a1i, strobe,
       &ready1, init, qampower,nfactor, gainqam, txfilt_filename);

mfOut =  matchFilt(symr, symi,
   strobe, nfactor,
   mfilt_filename, init);

rateconv_quad(a1r, a1i, &opr, &opi, ratio_timrec, strobe,
      &ready, init);

rateconv_quad2(a1r, a1i, &opr, &opi, ratio_timrec_rx, strobe,
       &ready, init);

rotation(c1r, c1i, &cc1r, &cc1i, freq_off_rot, phz_off,
 0, &ready3b, 1);

printf("init = %d\n",init);

cc1r = cc1i = 0.0;

awgn(cc1r, cc1i, &opr, &opi, sigma, nseed, passthrunz, 0,
     &ready_awgn, init);

phznoise(cc1r, cc1i, &opr, &opi, &phang, phznz10k,
 fmfreq, fbaud, ready3c, &ready, init,
 nseed, nfactor);

sine_ingress(opr, opi, &cc2r, &cc2i, sinefreq, sinelevel,
     ready3b, &ready, init);

printf("Rate conv init complete\n");

symbgen(&iout, &qout, nqam, nseed, init, &ready, strobe);

printf("symbgen init complete\n");

printf("made it past timing init\n");

init = 0;
strobe = 1;
interpolator_index = -1;
deinterpolator_index = 0;

/** QAMArray[] **/
for (i=0; i<num_blocks*fft_size;i++)
  {
    for(j=1;j<=nseed; j++)
      rand32bits = mrand48();
    symbgen(&iout, &qout, nqam, rand32bits, init, &ready, strobe);

    /*
      QAMArray[2*i]= ((i%fft_size==0) ? 1.0 : 0.0);
```

```
      QAMArray[2*i+1]=0.0;
      */

    /*
    QAMArray[2*i]= (((i+2)%fft_size==0) ? 1.0 : 0.0);
    QAMArray[2*i+1]=0.0;
    */

    /*
      QAMArray[2*i]= sin(3.14159265359/4*i);
      QAMArray[2*i+1]=0.0;
      */

    QAMArray[2*i]=iout;
    QAMArray[2*i+1]=qout;
  }

/* bin 20 and bin 1000 are knocked out */
for (i=0; i<num_blocks; i++)
  {
    QAMArray[2*(i*fft_size) + 2*20]=0.0;    /* erase bin 20 I */
    QAMArray[2*(i*fft_size) + 2*20+1]=0.0;  /* erase bin 20 Q */

    QAMArray[2*(i*fft_size) + 2*1000]=0.0;    /* erase bin 1000 I */
    QAMArray[2*(i*fft_size) + 2*1000+1]=0.0;  /* erase bin 1000 Q */
  }

printf("QAM generation complete\n");

for (iter=0; ;iter++)
  {
    if (iter % (nfactor*2) == 0)
      {
if (((interpolator_index % fft_size) == 0) &
    (qam_bc<num_blocks))
  {
    printf("Block %d of %d generated.\n",qam_bc+1, num_blocks);
    /** IFFTArray[] **/
    for (i=0; i < 2*fft_size; i++)
      {
IFFTArray[i] = QAMArray[qam_ec+i];
      }
    fft_time2(IFFTArray-1,fft_size,-1,1); /* compute IFFT */
    for(i=0; i<2*fft_size; i++) /* normalize by 1/fft_size */
      {
IFFTArray[i] /= fft_size;
      }
    /** PrefixArray[] **/
    for(i=0; i< 2*prefix_size; i++)
      PrefixArray[i] = IFFTArray[i+2*fft_size-2*prefix_size];
    for (i = 0; i < 2*fft_size; i++)
      PrefixArray[i+2*prefix_size] = IFFTArray[i];
    for(i=0; i< fft_size+prefix_size; i++)
      {
```

```
fprintf(f_prefixed, " %f",PrefixArray[2*i]);
fprintf(f_prefixed, " %f\n",PrefixArray[2*i+1]);
      }
    qam_ec += 2*fft_size;
    qam_bc += 1;
  }
      newsym = 1;
      interpolator_index++;
      fprintf(f_temp,"interpolator index = %d\n", interpolator_index);
      }
    else
newsym = 0;

    /* commutator */

    presamp1.real = PrefixArray[2*((interpolator_index-1) %
  (fft_size+prefix_size))];
    presamp1.imag = PrefixArray[2*((interpolator_index-1) %
  (fft_size+prefix_size))+1];

    upfilt(presamp1.real, presamp1.imag,
   &cc1r, &cc1i, newsym,
   &ready3b, init, qampower, nfactor, gainqam,
   txfilt_filename);

    if (ready1==1)
      {
fprintf(f_upfiltered, " %f",outputr);
fprintf(f_upfiltered, " %f\n",outputi);
      }


    /*
      gainInputGen(b1r, b1i, &c1r, &c1i, gain1, ready2, &ready3);

      c1r=c1i=0.0;
      */

    /*

      rotation(c1r, c1i, &cc1r, &cc1i, freq_off_rot, phz_off,
      ready2, &ready3b, init);

           */

      phznoise(cc1r, cc1i, &opr, &opi, &phang, phznz10k,
      fmfreq, fbaud, ready3b, &ready, init,
      nseed, nfactor);

              sum_phang_pwr += 2*(1-cos(phang));
              number_of_phangs++;


    /*
```

```
      sine_ingress(opr, opi, &cc2r, &cc2i, sinefreq, sinelevel,
      ready3b, &ready, init);


      freq_shift(cc2r, cc2i, &d1r, ready3b, &ready4, init);
    */


    if (ready == 1)
      offsetcount++;

    if (ready == 1 && offsetcount > 6)
      ready7=1;
    else
      ready7=0;

    convOut = v_convolve(opr, opi,
 ready7,
 chan_filename, init);

    /*
      amhum(convOut.yR, &p1r, &amlevop, amlevel,
  8.0*fbaud/ratio_conv, nseed, convOut.ready, &ready4b,
  init);
  */

    if (convOut.ready /*ready4b*/ == 1)
      {
presamp.real = convOut.yR;
presamp.imag = convOut.yI;

samp = delay5(presamp, delay_reg_to, length_to);

fprintf(f_convolved, " %f",samp.real);
fprintf(f_convolved, " %f\n",samp.imag);

/* white gaussian noise */

awgn(samp.real, samp.imag, &awgnoutr, &awgnouti,
     sigma, nseed, passthrunz, strobe,
     &ready_awgn, init);

/*
  rateconv_quad2(awgnoutr, awgnouti, &opr, &opi,
  ratio_timrec_rx,
  strobe,
  &ready5, init);
  */

if (ready_awgn==1)
  {
    offsetcount2++;
  }
```

```
/*
  printf("mfin = %f, %f\t\t", awgnoutr, awgnouti);
  */

if (ready_awgn==1 && offsetcount2 > 2)
  {
    mfOut =  matchFilt(awgnoutr, awgnouti, strobe,
       nfactor, mfilt_filename, init);
    /*
    printf("mfout = %f, %f, ready = %d\n",
   mfOut.yR, mfOut.yI, mfOut.ready);
   */

    if (mfOut.ready == 1)
      {
fprintf(f_decimated, " %f",mfOut.yR);
fprintf(f_decimated, " %f\n",mfOut.yI);

/** DePrefixArray[] **/

DePrefixArray[2*(deinterpolator_index%
 (fft_size+prefix_size))] = mfOut.yR;
DePrefixArray[2*(deinterpolator_index%
 (fft_size+prefix_size))+1] = mfOut.yI;
deinterpolator_index++;

/* when the deinterpolator runs out of storage,
   reset to the top and give the data to FFTArray */

if ((deinterpolator_index %
     (fft_size+prefix_size)) == 0)
  {
    /** FFTArray[] **/
    for (i=0; i<2*fft_size; i++)
      FFTArray[i] = DePrefixArray[i+2*prefix_size];

    /* output in the frequency domain */
    fft_time2(FFTArray-1, fft_size,1,1);

    /** QAMArray2[] **/
    for (i=0; i < 2*fft_size; i++)
      QAMArray2[qam_ec2+i] = FFTArray[i];

    qam_bc2 += 1;
    qam_ec2 += 2*fft_size;
  }


if (qam_bc2 == num_blocks)
  {
    /* collect all error data and exit */

    /* take error and signal information for
```

```
       QAM input and output */

    for (j=0; j<num_blocks*fft_size; j++)
      {
sig[j] = QAMArray[2*j]*QAMArray[2*j] +
  QAMArray[2*j+1]*QAMArray[2*j+1];
err[j] = (QAMArray[2*j]-QAMArray2[2*j])
  *(QAMArray[2*j]-QAMArray2[2*j]) +
  (QAMArray[2*j+1]-QAMArray2[2*j+1])
  *(QAMArray[2*j+1]-QAMArray2[2*j+1]);
/*
printf("sig[%d]=%f  ",j,sig[j]);
printf("err[%d]=%f  ",j,err[j]);
printf("QAMArray[%d]=%f  ",2*j,QAMArray[j]);
printf("QAMArray2[%d]=%f\n",2*j,QAMArray2[j]);
*/
fprintf(f_qam_in, "%f  %f\n",
QAMArray[2*j], QAMArray[2*j+1]);

fprintf(f_qam_out, "%f  %f\n",
QAMArray2[2*j], QAMArray2[2*j+1]);

      }

    /* use sig and err information to get
       average signal and error */

    for (j=0; j<fft_size; j++)
      {
for (i=0; i<num_blocks; i++)
  {
    avgsig[j] += sig[j+i*fft_size]/num_blocks;
    avgerr[j] += err[j+i*fft_size]/num_blocks;
  }
      }


    /* use average signal and error data to
       generate SNR */

    for (j=0; j<fft_size; j++)
      {
SNRdb[j] = 10 * log10(avgsig[j]/avgerr[j]);
/*printf("SNRdb[%d]=%f\n",j,SNRdb[j]);*/
      }

    for (j=0; j<fft_size; j++)
      fprintf(f_snr,"%f\n", SNRdb[j]);

    for (i=0; i<num_blocks*fft_size; i++)
      {
QAMArrayRe[i] = QAMArray[2*i];
QAMArrayIm[i] = QAMArray[2*i+1];
QAMArray2Re[i] = QAMArray2[2*i];
```

```
QAMArray2Im[i] = QAMArray2[2*i+1];
      }

                    snr_phang = -10*log10(sum_phang_pwr/number_of_phangs);
                    printf("sum_phang_pwr = %f\n",sum_phang_pwr);
                    printf("number_of_phangs = %d\n",number_of_phangs);
                    printf("SNR calculated by phang power = %f\n",snr_phang);


    fprintf(ferr, "\n");

    for (i=0; i<num_blocks*fft_size; i++)
      fprintf(ferr, " %f",QAMArrayRe[i]);
    fprintf(ferr, "\n");

    for (i=0; i<num_blocks*fft_size; i++)
      fprintf(ferr, " %f",QAMArrayIm[i]);
    fprintf(ferr, "\n");

    for (i=0; i<num_blocks*fft_size; i++)
      fprintf(ferr, " %f",QAMArray2Re[i]);
    fprintf(ferr, "\n");

    for (i=0; i<num_blocks*fft_size; i++)
      fprintf(ferr, " %f",QAMArray2Im[i]);
    fprintf(ferr, "\n");

    /* free dynamically allocated memory */
    free(QAMArray);
    free(IFFTArray);
    free(PrefixArray);
    free(QAMArray2);
    free(FFTArray);
    free(DePrefixArray);
    free(QAMArrayRe);
    free(QAMArrayIm);
    free(QAMArray2Re);
    free(QAMArray2Im);

    free(sig);
    free(err);
    free(avgsig);
    free(avgerr);
    free(SNRdb);

    fclose(ferr);
    fclose(f_prefixed);
    fclose(f_upfiltered);
    fclose(f_convolved);
    fclose(f_decimated);

    fclose(f_qam_in);
    fclose(f_qam_out);
```

```
    exit(1);
  }
      }
  }
      }
  }
}
```

B.2 Convolution Code (v_convolve.c)

```
/*****************************************************************************
ANALOG DEVICES PROPRIETARY AND CONFIDENTIAL
Copyright (C) 1999 by ANALOG DEVICES all rights reserved.
Function to perform convolution between a stored real impulse response
and a real input.
         Author: Dave Ribner Oct. 12, 1999
 Modified by Veeral Shah on June 29, 2000 to do complex convolution
 (added (double yI) to CONV type for Im{} element)
*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "../define.h"
#include "../defineHPNA.h"
#include "../globals.h"
#include "../macros.h"

FILE *fimpulse, *fopen();

CONV_OUT v_convolve(xr, xi, strobe, coef_filename, reset)
double xr, xi;                  /* input real, imag symbols */
int strobe;                     /* input data ready signal if = 1 */
int reset;                      /* initialize registers if = 1 */
char *coef_filename;          /* transmit filter filename */
{
  int i, j;
  double tmpr, tmpi, val;
  CONV_OUT out;
  static double impulse_coef[1024], inputr[1024], inputi[1024];
  static int impulse_length;

  if (reset == 1)
    {
      if ((fimpulse = fopen(coef_filename, "r")) ==  NULL)
{
  printf("\n Couldn't open %s. Use correct filename. \n \n", coef_filename);
  exit(2);
}
      /* read in coefficients of transmit filter from file */
      i = 0;
      while (fscanf(fimpulse,"%lf", &val) != EOF) impulse_coef[i++] = val;
      fclose(fimpulse);
```

```
      impulse_length = i;
      for (i = 0; i< impulse_length; i++)
{
  inputr[i] = 0.0;
  inputi[i] = 0.0;
}
      out.yR = 0.0;
      out.yI = 0.0;
      out.ready = 0;

    }
  else if (strobe == 1)
    {

      tmpr = 0.0;
      tmpi = 0.0;
      for (i = 0; i < impulse_length; i++)
{
  tmpr += inputr[i] * impulse_coef[i];
  tmpi += inputi[i] * impulse_coef[i];
}
      out.yR = tmpr;
      out.yI = tmpi;
      out.ready = 1;
      /* Symbol Shift Register */
      for (i = impulse_length-1; i > 0; i--)
{
  inputr[i] = inputr[i-1];
  inputi[i] = inputi[i-1];
}
      inputr[0] = xr;
      inputi[0] = xi;
    }
  else
    {
      out.ready = 0;
    }
  return(out);
}
```

B.3 QAM Generation (symbgen.c)

```
/*****************************************************************************
**                                                                          **
**           Analog DEVICES PROPRIETARY AND CONFIDENTIAL                    **
**           Copyright (C) 2000 by ANALOG DEVICES all rights reserved.      **
**           Function to generate random symbols using QAM/QPSK             **
**             for different constellation sizes depending on               **
**             con_size parameter.                                          **
**           Author : Veeral Shah, June 14, 2000.                           **
**                                                                          **
**           con_size: constellation size (4,16,64,256,1024,4096)           **
```

```
**           seed: to randomize the generation                              **
**                                                                          **
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void symbgen(iout, qout, con_size, rand32Bits, init, ready, strobe)
int             con_size;
double         *iout, *qout; /* in-phase and quadrature output */
int             init, *ready, strobe;
int             rand32Bits;

{
double          edge;
int             ival, qval;


if (strobe == 1) {
edge = sqrt(con_size);
ival = rand32Bits & ((int) edge - 1);
rand32Bits >>= (int) (log(edge) / log(2.0));
qval = rand32Bits & ((int) edge - 1);

/* convert the value in (0,n-1) to their QAM levels */

*iout = (double) ival *2.0 / edge - ((edge - 1) / edge);
*qout = (double) qval *2.0 / edge - ((edge - 1) / edge);
}
/*
 * leave this check to the main program if ((con_size != 4) &&
 * (con_size != 16) && (con_size != 64) && (con_size != 256) &&
 * (con_size != 1024)) { printf("Acceptable constellation sizes are
 * 4, 16, 64, 256, and 1024"); printf("\n"); exit(1); }
 */


}
```

B.4 IFFT and FFT block (v_fft_time.c)

```
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr;

void fft_time2(fft_data,nn,isign,strobe)

int strobe;
double fft_data[];
int nn,isign;

{
  int n, mmax,m,j,istep,i;
  double wtemp,wr,wpr,wpi,wi,theta;
```

```
  double tempr,tempi;

  /* clear the array if starting to fill a new OFDM block */

  /*
  if (init == 1)
    {
      for (i=1; i< (2*nn); i++)
fft_data[i] = 0.0;
    }
    */

  if (strobe == 1)
    {
      n=nn << 1;
      j=1;
      for (i=1;i<n;i+=2) {
if (j > i) {
  SWAP(fft_data[j],fft_data[i]);
  SWAP(fft_data[j+1],fft_data[i+1]);
}
m=n >> 1;
while (m >= 2 && j > m) {
  j -= m;
  m >>= 1;
}
j += m;
      }
      mmax = 2;

      while (n > mmax)
{
  istep = 2*mmax;
  theta = 6.28318530717959/(isign*mmax);
  wtemp = sin(.5*theta);
  wpr = -2.0*wtemp*wtemp;
  wpi = sin(theta);
  wr = 1.0;
  wi = 0.0;
  for (m=1;m<mmax;m+=2)
    {
      for (i=m;i<=n;i+=istep)
{
  j = i+mmax;
  tempr=wr*fft_data[j]-wi*fft_data[j+1];
  tempi=wr*fft_data[j+1]+wi*fft_data[j];
  fft_data[j]=fft_data[i]-tempr;
  fft_data[j+1]=fft_data[i+1]-tempi;
  fft_data[i] += tempr;
  fft_data[i+1] += tempi;
}
      wr=(wtemp=wr)*wpr-wi*wpi+wr;
      wi=wi*wpr+wtemp*wpi+wi;
    }
```

```
  mmax=istep;
}
    }
}
```

Bibliography

[1] D. N. Goddard. "Passband Timing Recovery in an All-Digital Modem Receiver," *IEEE Transactions on Communications*, May 1978.

[2] William H. Press, et. al. *Numerical Recipes in C : The Art of Scientific Computing, Second Edition.* 1993: Cambridge Univ Press.

[3] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* 1989: Prentice-Hall.

[4] W.D. Warner, C. Leung. "OFDM/FM Frame Synchronization for Mobile Radio Data Communication", *IEEE Trans. on Vehicular Technology*, August 1993.

[5] W. T. Webb and L. Hanzo. *Modern Quadrature Amplitude Modulation: Principles and Applications for Fixed and Wireless Communications.* 1994: IEEE Press and Pentech Press.

[6] F. Classen, H. Meyr. "Frequency Synchronization Algorithms for OFDM Systems Suitable for Communication over Frequency-Selective Fading Channels," *Proc. IEEE Vehic. Tech. Conf.*, June 1994.

[7] P. H. Moose, "A Technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction," *IEEE Transactions on Communications*, October 1994.

[8] H. Sari, G. Karam, and I. Jeanclaude, "Transmission Techniques for Digital Terrestrial TV Broadcasting," *IEEE Communications Magazine*, February 1995.

[9] T. Pollet, M. V. Bladel, and M. Moeneclaey, "BER sensitivity of OFDM systems to carrier frequency offset and wiener phase noise," *IEEE Trans. Commun.*, Feb. 1995.

[10] F. Daffara and O. Adami. "A New Frequency Detector for Orthogonal Multicarrier Transmission Techniques," *Proc. IEEE VTC*, Vol. 2, pp. 804-809, Jul. 1995.

[11] J. van de Beek, M. Sandell, P.O. Borjesson, "Timing and Frequency Synchronization in OFDM Systems Using the Cyclic Prefix", *Proceedings of the 1995 IEEE International Symposium on Synchronization*, December 1995.

[12] J. van de Beek, M. Sandell, P.O. Borjesson, "ML Estimation of Timing and Frequency Offset in Multicarrier Systems", *Research Report TULEA 1996:09, Division of Signal Processing, Lulea University*, April 1996.

[13] J. van de Beek, M. Sandell, P.O. Borjesson, "On Synchronization in OFDM Systems Using the Cyclic Prefix", *Proceedings of the RVK'96*, June 1996.

[14] J. van de Beek, M. Sandell, P.O. Borjesson, et al., "A Conceptual Study of OFDM-based Multiple Access Schemes: Part 4 - Time and Frequency tracking", *Technical Report Tdoc 250/96, ETSI STC SMG2 meeting no 20*, December 1996.

[15] J. van de Beek, M. Sandell, P.O. Borjesson, "ML Estimation of Time and Frequency Offset in OFDM Systems", *IEEE Transactions on Signal Processing*, July 1997.

[16] D. Landstrom, S. Wilson, J. van de Beek, P. Odling, P. O. Borjesson, "Symbol time offset estimation in coherent OFDM systems," *COST 259 Technical Document (98)86*, September 1998.

[17] S. Marin, et al. "Paper 3232-05–Medium-range terrestrial propagation study at 28 GHz," *SPIE Proceedings Vol. 3232*, November 1997.

[18] T. M. Schmidl and D. C. Cox, "Robust Frequency and Timing Synchronization for OFDM," *IEEE Transactions on Communications*, December 1997.

[19] H. Meyr, M. Moeneclaey, S. Fechtel. *Digital Communication Receivers* 1998: John Wiley and Sons.

[20] II. Liu and U. Tureli, "A High-Efficiency Carrier Estimator for OFDM Communications," *IEEE Communications Letters*, April 1998.

[21] M. Morelli and U. Mengali. " An Improved Frequency Offset Estimator for OFDM Applications," *IEEE Communications Letters*, March 1999.

[22] T. Pollet and M. Peeters, Alcatel. " Synchronization with DMT Modulation," *IEEE Communications Magazine*, April 1999.

[23] ETSI EN 300 744 v.1.2.1 (1999-07). *Digital Video Broadcasting(DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television.* July 1999: European Broadcasting Union.

[24] K. Sheikh, D. Gesbert, D. Gore, A. Paulraj. "Smart Antennas for Broadband Wireless Access Networks," *IEEE Communications Magazine*, November 1999.

[25] M. Morelli, A. N. D'Andrea, U. Mengali. "Frequency Ambiguity Resolution in OFDM Systems," *IEEE Communications Letters*, April 2000.

[26] N. Lashkarian, S. Kiaei. "Globally Optimum ML Estimation of Timing and Frequency Offset in OFDM Systems," *IEEE Comms. Letters*, 2000.

[27] R. Gallager. Lecture Notes for 6.401/6.450– *Introduction to Digital Communication*, Fall 2000.

[28] "FPGA Polyphase Filter Bank Study and Implementation", *http://www.icsl.ucla.edu/ ipl/pphase.html*