An Investigation into the Implementation and Performance of Spectrally Shaped Orthogonal Frequency Division Multiplex

By

David Lahiri Bhatoolaul

B.A. (Hons)

A thesis submitted to the University of Plymouth

for the degree of

DOCTOR OF PHILOSOPHY

School of Electronic, Communication

and Electrical Engineering.

Faculty of Technology

September 1999

An Investigation into the Implementation and Performance of Spectrally Shaped Orthogonal Frequency Division Multiplex

David Lahiri Bhatoolaul

Orthogonal Frequency Division Multiplex (OFDM) is a flexible, robust multi-carrier modulation scheme. The orthogonal spectral shaping and spacing of OFDM sub-carriers ensure that their spectra can be over-lapped without leading to undesirable inter-carrier interference. Conventional OFDM systems have non-band limited Sinc(x) shaped sub-carrier spectra. An alternative form of OFDM, referred to hereafter as Spectrally Shaped OFDM, employs band limited Nyquist shaped sub-carrier spectra. The research described in this thesis investigates the strengths and weaknesses of Spectrally Shaped OFDM as a potential modulation scheme for future mobile radio applications.

From this research a novel Digital Signal Processing architecture for modulating and demodulating Spectrally Shaped OFDM sub-carriers has been derived which exploits the combination of a complex Discrete Fourier Transform (DFT) and PolyPhase Network (PPN) filter. This architecture is shown to significantly reduce the minimum number of computations required per symbol compared to previous designs.

Using a custom coded computer simulation, the effects of varying the key parameters of the novel architecture's PolyPhase Filter (PPN) filter on the overall system complexity, spectral performance and system signal-to-distortion have been extensively studied. From these studies it is shown that compared to similar conventional OFDM systems, Spectrally Shaped OFDM systems possess superior out-of-band spectral qualities but significantly worse Peak-to-Average-Power-Ratio (PAPR) envelope performance. It is also shown that the absolute value of the end PPN filter coefficients (dependent on the roll-off factor of the sub-carrier spectral shaping) dictate the system signal-to-distortion ratio when no time-domain windowing of the PPN filter coefficients is applied. Finally the effects of a both time and frequency selective fast fading channels on the modulation scheme's uncoded Bit Error Rate (BER) versus Signal-to-Noise (SNR) performance are simulated. The results obtained indicate that Spectrally Shaped OFDM is more robust (lower BER) to frequency-selective fading than time-selective fading.

LIST OF CONTENTS

LIST OF FIGURES

Number *Page*

LIST OF EQUATIONS

LIST OF TABLES

LIST OF ABBREVIATIONS

AFD	Average Fade Duration
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BP	Band Pass
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
DAB	Digital Audio Broadcast
DECT	Digitally Encoded Cordless Telephony
DFT	Discrete Fourier Transform
DQPSK	Differential Quadrature Phase Shift Keyed
DVB	Digital Video Broadcast
FD-DQPSK	Frequency Domain - DQPSK
FDMA	Frequency Domain Multiple Access
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
GMSK	Gaussian Minimum Shift Keying
GSM	Groupe Special Mobile
HF	High Frequency
ICI	Inter-Carrier Interference

IDFT	Inverse Discrete Fourier Transform
ISI	Inter-Symbol Interference
LCR	Level Crossing Rate
LPF	Low Pass Filter
MSK	Minimum Shift Keying
OFDM	Orthogonal Frequency Division Multiplex
PA	Power Amplifier
PDF	Probability Density Function
PSD	Power Spectral Density
PSK	Phase Shift Keying
PPN	Poly Phase Network
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RRC	Root Raised Cosine
SNR	Signal-to-Noise Ratio
TDMA	Time Division Multiple Access
UMTS	Universal Mobile Telecommunication System
WCDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network
WLL	Wireless Local Loop

ACKNOWLEDGEMENTS

Firstly I would like to thank my supervisor, Dr. Graham Wade, for his unwavering support throughout the long course of this research.

Secondly, I am grateful to all my colleagues at Lucent Technologies who provided technical assistance and encouragement to continue and complete this research.

Thirdly, I would like to thank my parents for all their help and encouragement.

Last but not least, I thank my beautiful wife, Shakti Bhatoolaul. Without her loving support and patience, this thesis would not have been completed.

DECLARATION

The work described in this thesis was first carried out between October 1993 and October 1996, at the School of Electronic, Communication and Electrical Engineering, University of Plymouth, Plymouth, England. This work was then continued and completed between October 1996 and June 1999, whilst at Lucent Technologies, Optimus House, Windmill Hill Business Park, Swindon, England.

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

Relevant scientific seminars and conferences were regularly attended at which work was often presented; external institutions were visited for consultation purposes, and papers prepared for publication.

The work presented in this thesis is solely that of the author.

Signed.....

Date.....1st SEPTEMBER 1999.....

CHAPTER ONE

1. Introduction

This chapter describes the aims and objectives of the research and gives a brief introduction to each chapter within this thesis.

1.1 Introduction

Over the past decade, the increasingly world-wide and computerised nature of today's businesses has placed pressure on the telecommunications industry to improve and expand the availability, speed and immediacy of its services. Throughout the world this has led to a growing demand for mobile and portable wireless telecommunications. Mobile telephone, radio, data, navigational and paging services have seen spectacular growth in recent years. Cellular mobile radio systems, such as the pan-European GSM service, have enjoyed extremely rapid rates of growth. In fact GSM operators foresee 25 million users of mobile telecommunications services by the year 2000. By this time the predicted European GSM terrestrial radio cellular network infrastructure will only be able to economically provide coverage for 45-50% of the land, corresponding to 80% of the working population. At least half of the remaining 20% of the European mobile market are predicted to turn to the increasing number of Land Mobile Satellite Systems (LMSS) currently being deployed. In addition to these long-range mobile radio products, there are a number of short-range limited mobility/portable radio products and standards bringing the wireless communications into the home and office. For example, there are the DECT, IEEE802.11 and WaveLan standards that make short-range limited mobility wireless communications of 2 Mbits and above possible indoors.

All these products and standards are being evolved to increase their flexibility and power. Specific areas undergoing development include the network architecture, the control protocols and the radio transmission technology. One aspect of radio transmission technology that has recently been the subject of intense world-wide study, debate and controversy is the modulation scheme. From these deliberations, two schemes have emerged that are set to revolutionise mass-market wireless communications well into the twenty-first century. These are Direct Sequence Wideband Code Division Multiplex Access (W-CDMA) and Orthogonal Frequency Division Multiplex (OFDM).

Direct Sequence W-CDMA has been recently adopted by the Asian, European and North American wireless communication governing bodies as the preferred

modulation scheme for their third-generation cellular systems due to appear early next century. In contrast to current second-generation cellular systems, such as IS-136 and GSM, this modulation scheme will have the flexibility to support not only the existing low bit-rate (< 13 kbps) voice services but also new medium (< 144 kbps) to high bit rate (< 2Mbps) services such as web-browsing and real-time video. Key reasons for the selection of W-CDMA ahead of other technologies, include the simplicity of frequency planning, its robustness against narrow-band interference, its support of macro diversity and 'soft' limit on capacity. The book by Viterbi (ref. [81]) provides a good coverage of the spread spectrum concepts that form the foundation of CDMA systems.

The two areas of wireless communications where OFDM has made the greatest impact over the past few years are Wireless Local Area Networks (W-LAN) and Digital Audio and Video Broadcasting (DAB and DVB). Both of the major next generation W-LAN systems, European HiperLAN II and North American IEE802.11, have been designed to share the same OFDM air-interface. These systems have been developed to supply very high bit rate (< 20 Mbps) services in small cellular networks covering places such as hospitals, airports, university campuses and offices. OFDM has also been universally accepted as the preferred modulation scheme for the broadcast of terrestrial and satellite digital audio and video services, because of its high spectral efficiency, scalability, simple implementation and robustness to inter-symbol interference which enables the deployment of so-called single frequency networks (see ref. [88]).

The objectives of the research conducted for this dissertation are focused on identifying and investigating the strengths and weaknesses of a relatively un-researched variant of OFDM, referred to as Spectrally Shaped OFDM, for future generations of wireless mobile radio communications.

1.2 Orthogonal Frequency Division Multiplex

Multi-carrier modulation schemes, such as OFDM, map the symbols received from a single high rate, 'N/T_s', symbol stream to a number, 'N', of parallel low rate, '$1/T_s$', symbol streams. Each of these low rate symbol streams is then modulated onto a separate sub-carrier. When compared to similar single carrier systems, this serial-to-parallel approach to modulation has two advantages,

- The increased symbol period per sub-carrier, improves the systems' resilience to Inter-Symbol Interference (ISI).

- With the application of simple symbol coding and inter-leaving scheme, the effects of narrow band distortion, generated by jammers and frequency selective fading, can be greatly reduced.

The major disadvantage of all multi-carrier modulation schemes, including OFDM, is the high peak-to-average power ratios (PAPR) of their modulated signals. This particular property can make linear amplification of these signals extremely difficult and inefficient compared to single carrier 'constant envelope' schemes such as GMSK.

To avoid introducing undesirable inter-carrier interference (ICI), the adjacent sub-carriers of conventional multi-carrier modulation schemes are spaced sufficiently far apart to ensure that their spectra do not overlap. For OFDM systems, special conditions of 'orthogonality' allow the spectra of adjacent sub-carriers to overlap without introducing ICI at the points in time where the demodulated sub-carrier output signals should ideally be sampled to determine the transmitted symbols. This overlapping of adjacent sub-carrier spectra greatly improves the bandwidth efficiency (see chapter 3) of OFDM systems compared to conventional multi-carrier systems. However this added advantage only comes at the expense of increased sensitivity to time and frequency synchronisation errors.

The 'conditions of orthogonality' mentioned above, dictate not only the spacing and shaping of sub-carriers but also the manner in which symbols are actually modulated onto them. Two sets of these conditions are now briefly described, one

for conventional OFDM, hereafter referred to as Sinc(x) OFDM, and one for Spectrally Shaped OFDM.

1.2.1 Sinc(x) OFDM

The key features of a Sinc(x) OFDM system that ensure orthogonality between overlapping sub-carriers are summarised below and highlighted in figure 1.1. Note that Sinc(x) OFDM is more commonly referred to just as 'OFDM'.

- Assuming a symbol rate of '$1/T_s$' per sub-carrier, each sub-carrier should be spaced '$1/T_s$' apart from its neighbouring sub-carriers. Given this condition the frequency 'f_c' of the 'c^{th}' sub-carrier is given by '$f_c = c/T_s$'.

- The real and imaginary components of complex QPSK (or QAM) symbols are modulated simultaneously by the orthogonal *'$\sin(2\pi f_c t)$'* and *'$\cos(2\pi f_c t)$'* parts of a sub-carrier.

- Each sub-carrier should be Sinc(x) spectrally shaped. In the time-domain this shaping is obtained by truncating the modulating sinusoid using a 'T_s' long rectangular window. This truncation function is performed by *'u(t)'* in figure 1.1.

Together these requirements ensure that the demodulator sub-carrier outputs are free of ICI, as shown by equation 1.1 for the 'cth' sub-carrier.

$$\int_0^{T_s} [a_n \cos(2\pi f_n t) + b_n \sin(2\pi f_n t)] \cos(2\pi f_c t) dt = \begin{cases} 0 & n \neq c \\ \frac{a_n}{2} & n = c \end{cases}$$

$$\int_0^{T_s} [a_n \cos(2\pi f_n t) + b_n \sin(2\pi f_n t)] \sin(2\pi f_c t) dt = \begin{cases} 0 & n \neq c \\ \frac{b_n}{2} & n = c \end{cases}$$

Equation 1.1: Orthogonality of Sinc(x) OFDM

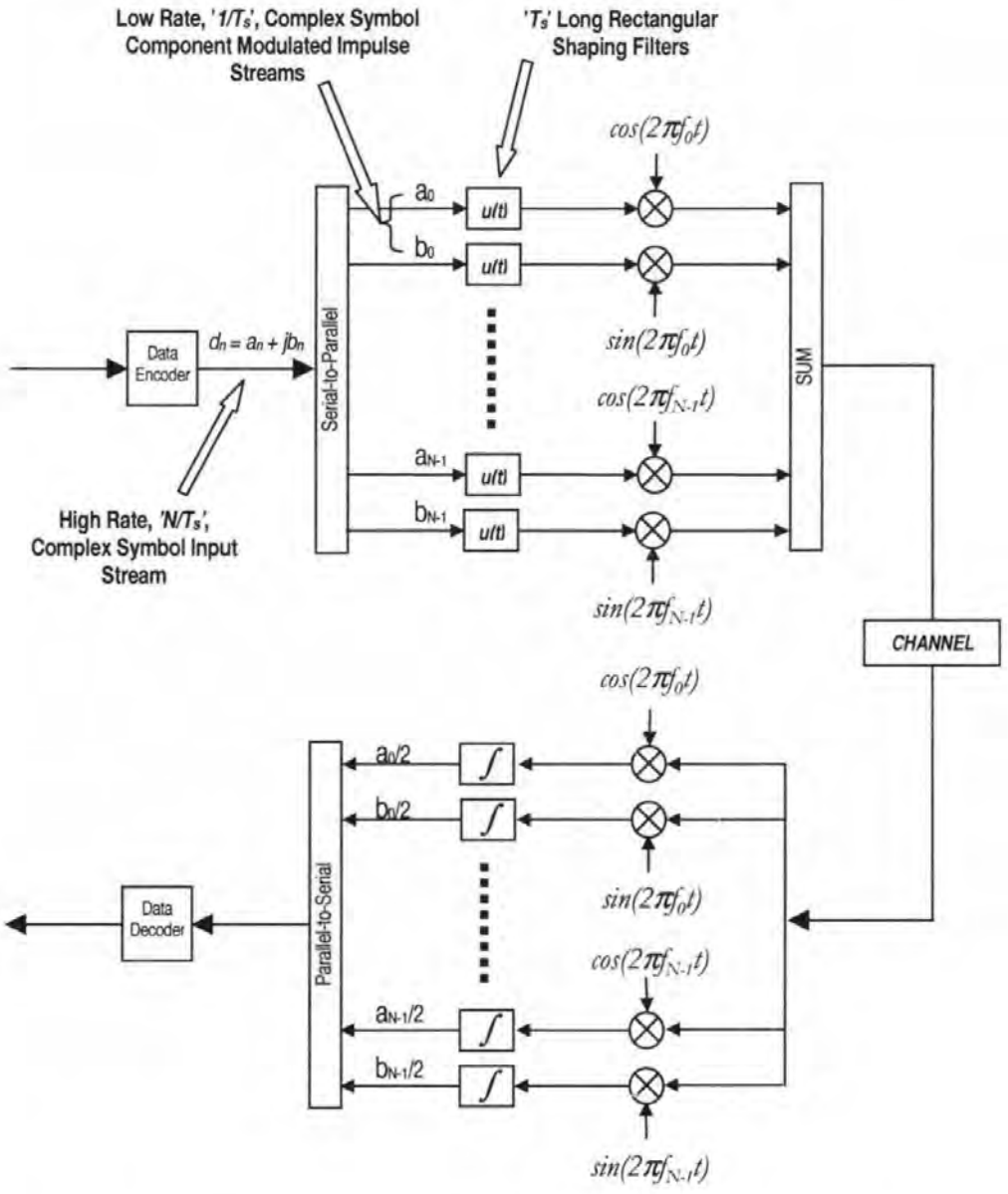


Figure 1.1: Modulation and Demodulation of Sinc(x) OFDM

1.2.2 Spectrally Shaped OFDM

The key features of a Spectrally Shaped OFDM system that ensure orthogonality between overlapping sub-carriers are summarised below and highlighted in figure 1.2.

- Assuming a symbol rate of '$1/T_s$' per sub-carrier, each sub-carrier should be spaced '$1/T_s$' apart from its neighbouring sub-carriers. This is the same as Sinc(x) OFDM.

- There is an offset in time, of '$T_s/2$', between when the real and imaginary components of complex QPSK (or QAM) symbols are modulated onto the orthogonal '$\sin(2\pi f_c t)$' and '$\cos(2\pi f_c t)$' parts of a particular sub-carrier. This delay is a subtle difference between Sinc(x) and Spectrally Shaped OFDM.

- A classic Nyquist filter, such as the Root Raised Cosine, is applied to spectrally shape each sub-carrier. This is a fundamental difference between Sinc(x) and Spectrally Shaped OFDM. In figure 1.2 this shaping is performed by the filter function '$h(t)$'.

R. W. Chang originally derived these system requirements in his 1966 paper (ref. [13]) entitled 'Synthesis of Band-Limited Orthogonal Signals for Multi-channel Data Transmission'. An alternative and novel derivation of these conditions for orthogonality, including the shaping of the sub-carrier shaping filter and the value of the offset required between the real and imaginary parts of a modulated symbol, is provided in Appendix A.

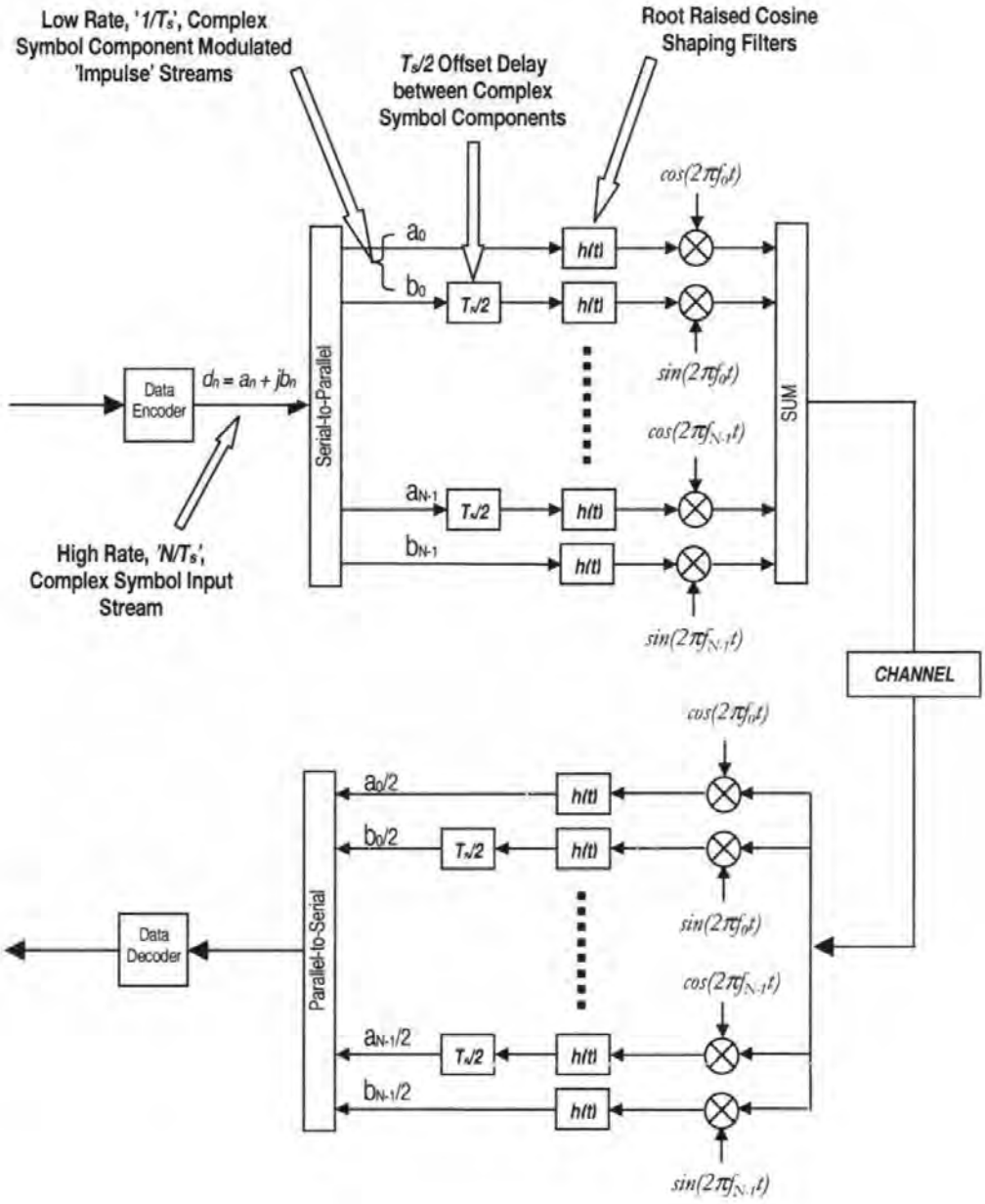


Figure 1.2: Modulation and Demodulation of Spectrally Shaped OFDM

1.2 Aims and Objectives of the Research

The main aims of this research project were to identify and investigate the strengths and weaknesses of Spectrally Shaped OFDM as a potential modulation scheme for future mobile radio applications.

The specific objectives are:

- To develop an efficient DSP architecture for the modulation and the demodulation of Spectrally Shaped OFDM sub-carriers.
- To develop an understanding of how the various parameters within this DSP architecture influence the overall performance of the system, in terms of the demodulator output signal-to-distortion (self-noise) level and the modulated signal spectral qualities.
- To identify what sort of mobile radio channels Spectrally Shaped OFDM could be best optimised for.

1.3 Thesis Structure

This thesis describes the research conducted in the pursuit of the aims and objectives summarised in section 1.2.

In Chapter 2, a novel digital signal processing (DSP) architecture is derived from first principles for the modulation and demodulation of Spectrally Shaped OFDM sub-carriers. This architecture dramatically reduces the DSP hardware power required to implement a practical modem. One factor that has fuelled the recent intensive research effort into Sinc(x) OFDM is the simplicity of its digital implementation using a DFT. In comparison, even though it is widely recognised as having superior spectral properties to Sinc(x) OFDM, Spectrally Shaped OFDM has suffered from a distinct lack of research principally due to the high complexity of its digital implementation.

In Chapter 3, the consequences of varying the characteristics of the sub-carrier shaping filter and number of sub-carriers of a Spectrally Shaped OFDM system, on the spectrum efficiency, implementation complexity and signal-to-distortion (self-noise) level are investigated. From this investigation, the optimum characteristics of the sub-carrier shaping filter required by the novel DSP architecture derived in chapter 2, are deduced. The results presented in this chapter clearly show how the properties of this filter have a crucial role in determining the complexity and performance of the overall system.

Chapter 4 outlines novel research carried out to determine the effects of fast fading radio channels on the uncoded Bit Error Rate (BER) performance of comparable Spectrally Shaped and Sinc(x) OFDM systems. From this analysis, a subset of mobile radio channel types are identified where Spectrally Shaped OFDM could potentially out-perform other modulation schemes such as Sinc(x) OFDM.

Chapter 5 provides a summary of the key findings of this research. Where appropriate these are identified as being either strengths or weaknesses of this modulation scheme. To conclude this chapter, and hence thesis, the limitations of the research conducted are discussed and used to help define new directions for future study.

CHAPTER TWO

2. Design of a Spectrally Shaped OFDM Modem

In this chapter, a novel Digital Signal Processing (DSP) architecture is derived from first principles for the modulation and demodulation of Spectrally Shaped OFDM sub-carriers.

2.1 Introduction

In this chapter a novel DSP architecture is presented that dramatically reduces the processing power required to implement a practical Spectrally Shaped OFDM digital modem. The cost of implementing modems for this modulation scheme has been cited by many researchers, including L. Cimini (ref. [14]), as being the main reason why they have selected to study Sinc(x) OFDM modulation schemes instead. This is even though they recognise that to match the out-of-band spectral properties of Spectrally Shaped OFDM, Sinc(x) OFDM systems require additional band-pass filtering.

In section 2.3 a novel DSP architecture is derived from first principles for the modulation and demodulation of Spectrally Shaped OFDM sub-carriers. This architecture almost halves the number of computations required per symbol, compared to the previous architecture published (ref. [35]). Prior to this section, section 2.2 highlights the two key features of existing OFDM modem designs that are applied to derive the architecture presented in Section 2.3. Section 2.4 closes the chapter with a summary of the key features and advantages of this architecture.

2.2 Design Features of Existing OFDM Modems

Over the past 40 years a number of methods have been developed to implement both Sinc(x) and Spectrally Shaped OFDM modems. In this section, two key principles exploited by many of these methods, the use of the complex digital domain and the DFT-PolyPhase Network filter, are highlighted and discussed, before being applied in section 2.3 for the derivation of a novel DSP architecture.

2.2.1 The Real and Complex Digital Signal Processing Domains

The earliest publications outlining the conceptual theory behind orthogonal multi-carrier systems can be traced back to the late 1950's and early 1960's. At that time, researchers such as Harmuth (ref. [29]), R.W.Chang (ref. [12][13]) and R. Mosier (ref. [51]), were mainly restricted to theorising about their discoveries due to the limitations of that period's technology. Nevertheless R. Mosier was able to produce a simple implementation of his Kineplex (to become known as OFDM) system using analogue resonators and RF oscillators.

Due to the lack of powerful DSP hardware, research into orthogonal multi-carrier systems remained on a theoretical basis for 20 years until the 1980's. During that time, several key discoveries were made that would eventually pave the way for practical systems to be constructed and studied. Probably the most significant of these being the computationally efficient baseband implementation of a 'N' sub-carrier Sinc(x) OFDM modem in the real (as opposed to complex) digital domain using a '2N' point discrete Fourier Transform (DFT) algorithm, by S. Weinstein and P. Ebert in 1971 (ref. [86]). This design was later refined by L. Cimini in 1985 (ref. [14]), who essentially halved the number of computations required for modulation and demodulation of the same 'N' sub-carrier modem, by using a 'N' point DFT operating at the same rate but in the complex (rather than real) digital domain. The factor of 2 difference between the DFT sizes required for the real and complex digital OFDM modem implementations arises from the effect the symmetrical and asymmetrical characteristic of the sampled truncated OFDM signal sinusoid, has on the output signal of a complex DFT.

2.2.2 The Digital DFT-PPN Sub-System

By the 1980's, digital signal processing technology had become sufficiently powerful to allow practical test-beds for simple multi-carrier systems, such as Sinc(x) OFDM, to be developed. This led to the explosion of interest in Sinc(x) OFDM, that has only recently culminated in the adoption of this variety of OFDM for DAB, DVB and WLAN systems. In contrast, research into Spectrally Shaped OFDM has lagged some way behind, simply due to the relative high complexity of its digital implementation. This is even though many key proponents of Sinc(x) OFDM, such as L. Cimini (ref. [14]), recognise that to attain the spectral properties of Spectrally Shaped OFDM, Sinc(x) OFDM systems require additional post-processing in the form of band-pass filtering. In fact during the 1980's only three papers (ref. [34][35][71]) were identified as advancing the understanding of the digital implementation of Spectrally Shaped multi-carrier modulation schemes. The key feature shared by all these modem designs is a sub-system known as the DFT-PolyPhase Network (DFT-PPN).

A PolyPhase Network typically consists of 'N' parallel sub-filters, each with 'M' coefficients. The coefficients of these PPN sub-filters are derived from decimating the 'MN' long Finite Impulse Response (FIR) of a 'common' or 'prototype' filter, 'h(n)'. The point at which the decimation of the original 'h(n)' FIR coefficients begins, is offset by 'i' for the 'i^{th}' sub-filter, so that no one coefficient from the 'prototype' filter is repeated within the PPN. When combined with a DFT, the PPN produces a sub-system that was originally developed for implementing highly efficient digital filter banks (ref. [76][77][79]) and trans-multiplexers (ref. [73][66][47]) which convert FDM to TDM signals.

The modem design (figure 2.1) presented by B. Hirosaki in his 1981 paper (ref. [35]), was the first published that exploited the combination of a DFT-PPN sub-system to modulate and demodulate Spectrally Shaped OFDM sub-carriers. For this design, a 16 channel complex DFT-PPN sub-system with some minor pre- and post-processing stages is used to modulate and demodulate 12 Spectrally Shaped OFDM sub-carriers. To take into the account the '$T_s/2$' offset, required between the real and imaginary components of the transmitted complex symbols, and to allow operation in the real digital domain, the DFT-PPN sub-systems are clocked at '$2/T_s$'.

The performance of the system was later improved (ref. [34]) by adding double sampling automatic equalisers to the demodulator PPN-DFT sub-system sub-carrier outputs.

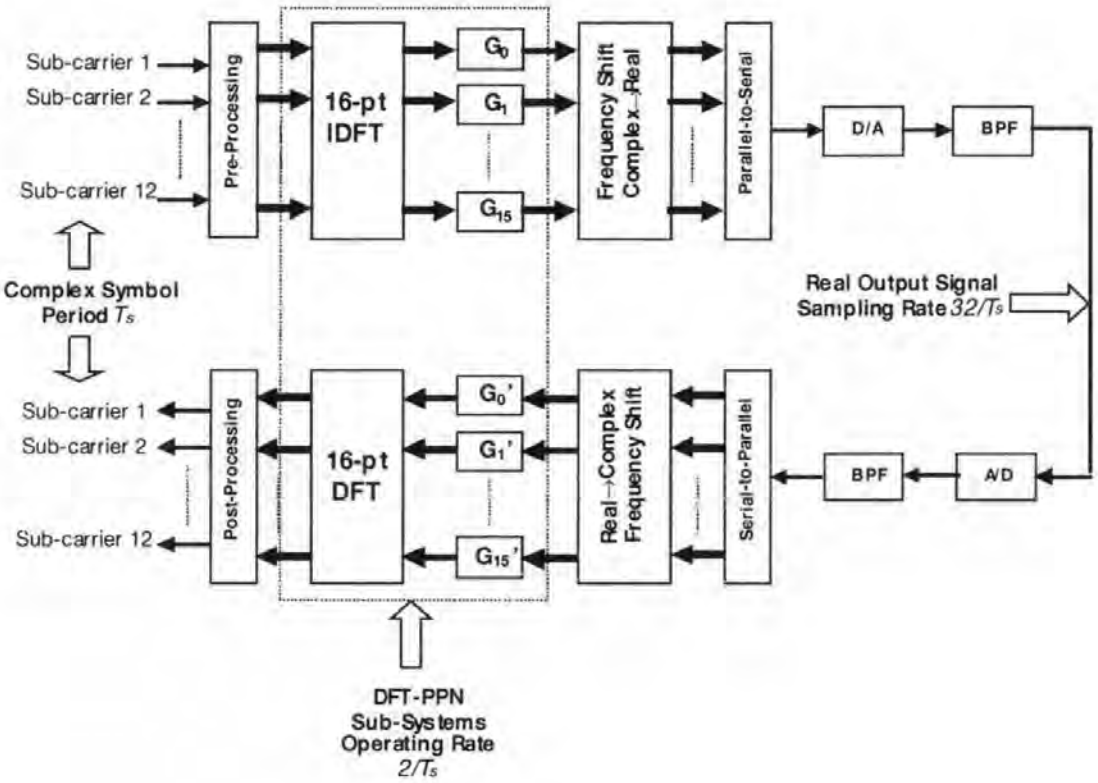
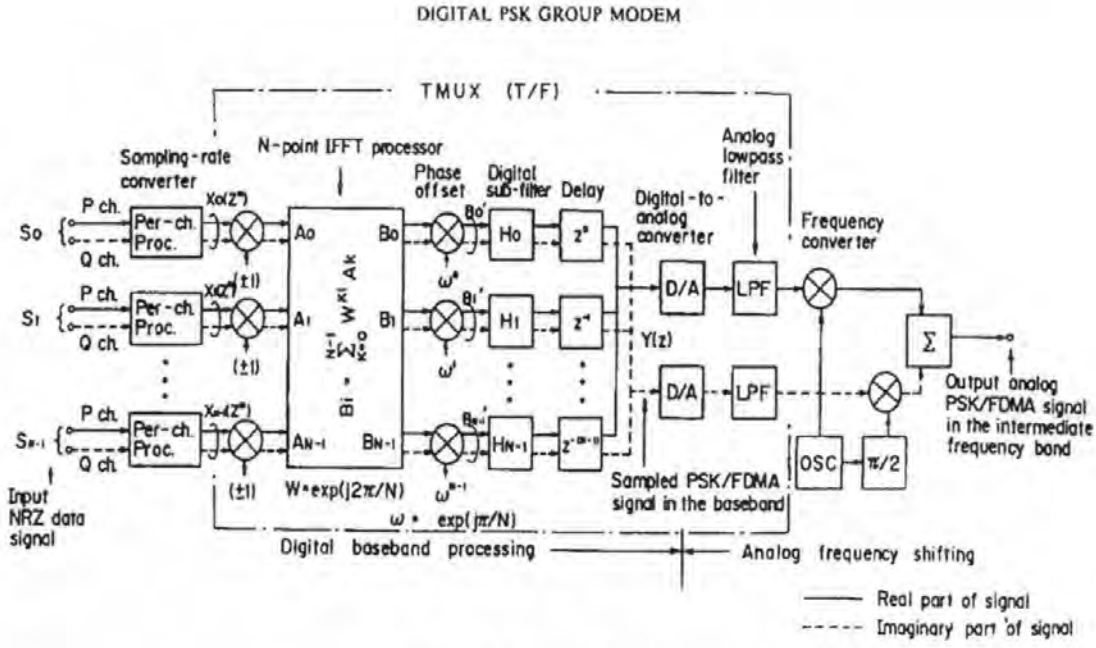


Figure 2.1: Hirosaki's Spectrally Shaped OFDM Modem

In 1988, Takahata et al, published a paper (ref. [71]) outlining their design (see figure 2.2), simulation and testing of a Spectrally Shaped FDM modem intended for cost-effective satellite applications. As a result of the sub-carriers not overlapping (hence FDM and not OFDM), the design shown in figure 2.2 avoids much of the DFT-PPN sub-system pre- and post-processing phases, required by Hirosaki's design, to maintain orthogonality. In addition, by operating the overall design in complex digital domain rather than the real digital domain, the 'N' point DFT-PPN sub-systems of Takahata's system need only be clocked at the symbol rate, '$1/T_s$'.



Overall configuration of the PSK group modulator

Figure 2.2: Takhata's Spectrally Shaped *FDM* Modulator

2.3 Implementation of a Spectrally Shaped OFDM Modem

In this section a novel DSP architecture is derived from first principles for the modulation and demodulation of Spectrally Shaped OFDM sub-carriers that dramatically reduces the number of operations required per symbol. The feature of this architecture that differentiates it from the only other architecture (section 2.1, figure 2.1) published at the time this was originally derived, is that it is designed to operate in the complex digital domain rather than the real digital domain.

2.3.1 Analogue Model of Modulator

Figure 2.3 shows the simplest analogue implementation of a Spectrally Shaped OFDM modulator and demodulator. Once every symbol period, 'T_s', the real and imaginary parts, 'a+jb', of the complex QPSK symbols transmitted along a particular sub-carrier, modulate the amplitude of separate real valued impulse streams. These impulse streams then undergo filtering by filters, 'h(t)', with identical Root Raised Cosine (RRC) finite impulse responses of length 'MT_s' (figure 2.4a). Figure 2.4b shows the spectrum of a Root Raised Cosine (and the comparable Raised Cosine) shaped sub-carrier with a roll-off factor of 0.5. Note that Root Raised Cosine sub-carrier specturm becomes narrower and more rectangular as the roll-off factor (α) decreases. The pair of real valued RRC shaped waveforms produced for each sub-carrier by filtering, are then offset in time by '$T_s/2$' relative to each other, before being modulated up to the appropriate sub-carrier frequency, '$\omega_c=2\pi f_c$', by multiplication with '$\cos(\omega_c t)$' and '$\sin(\omega_c t)$' terms.

If the time offsets of '$T_s/2$' between the real and imaginary components of the QPSK symbols are ignored, then the generation of sub-carrier output signal described above in the real domain, can also be described in the complex domain as follows. Once every symbol period, 'T_s', the QPSK symbols transmitted along a particular sub-carrier, modulate the phase of a complex impulse stream which is filtered by a complex version of RRC filter ('h(t)') used above, with all-zero imaginary coefficients. The complex output of this filter is then modulated up to the appropriate sub-carrier frequency by multiplication with a complex exponential, '$\exp\{-j2\pi f_c t\}$'. Thus a complex signal can be obtained, whose real part is identical to that generated by the real valued implementation described above assuming no '$T_s/2$' offset delays.

This simple and novel step of moving from the real to the complex domain to describe the most basic implementation of the modulator and demodulator has a profound effect on the computational efficiency of the digital implementation derived thereafter. It is comparable to the step performed by L. Cimini (ref. [14]) to derive his complex DFT implementation of a Sinc(x) OFDM modem.

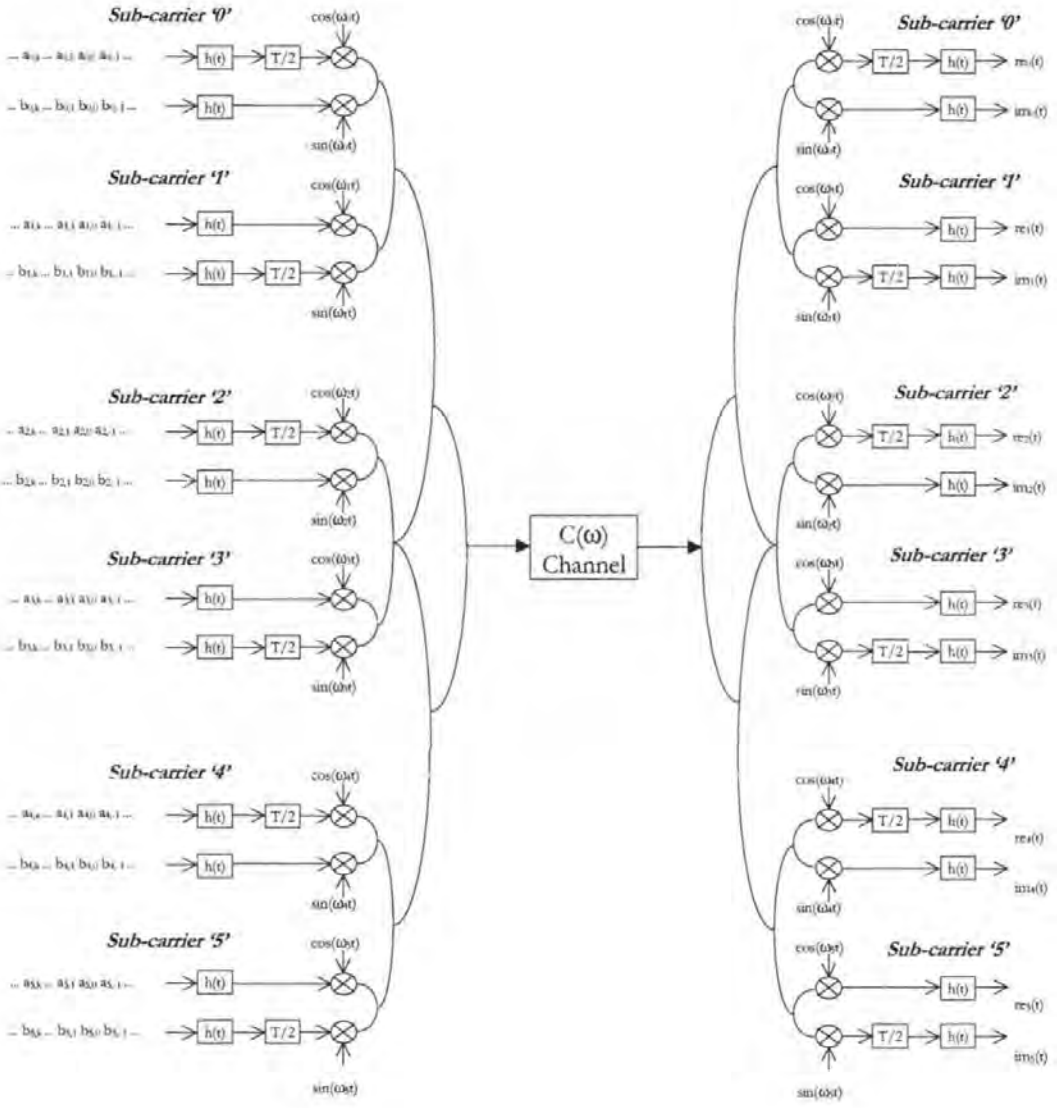


Figure 2.3: Model of a Spectrally Shaped OFDM System

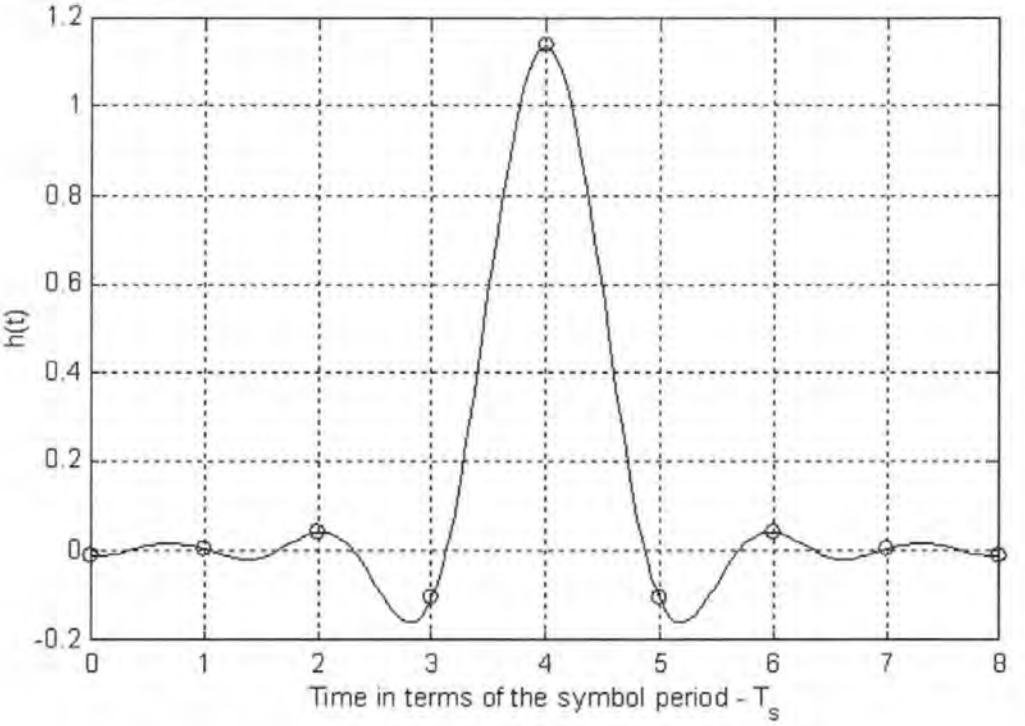


Figure 2.4a: RRC Sub-carrier Filter FIR (M = 8, Roll-off Factor = 0.5)

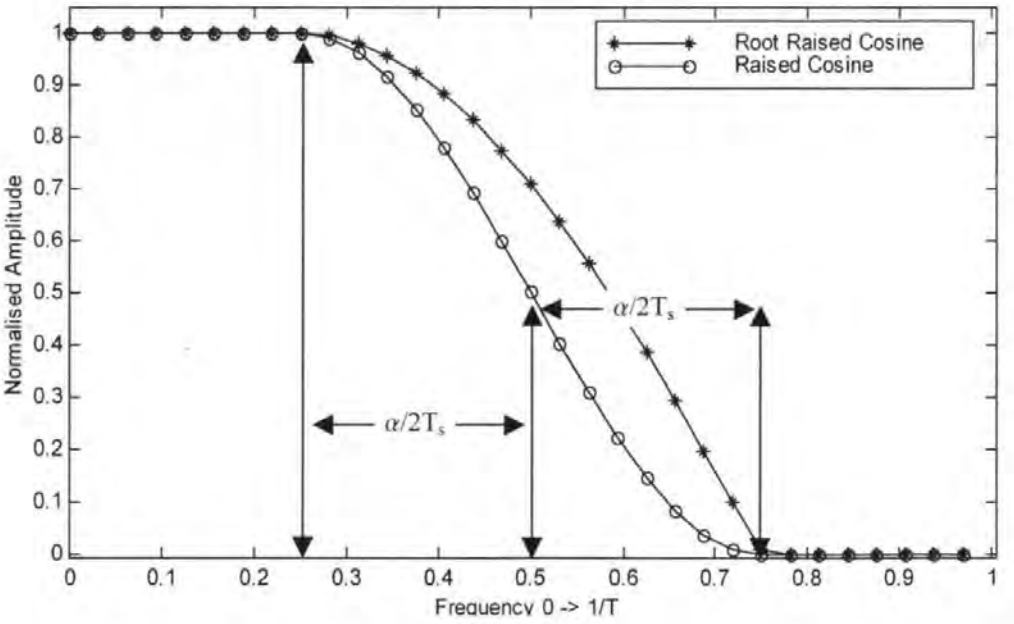


Figure 2.4b: RRC & RC Sub-carrier Spectrums (Roll-off Factor α= 0.5)

Due to the filter having a causal impulse response 'M' times longer than the symbol period (figure 2.4), symbols spaced up to 'MT_s' apart contribute to the filter output at any point in time. It therefore follows that the complex output, '$b_c(t)$', of the 'c^{th}' sub-carrier shaping filter prior to modulation up to the sub-carrier frequency 'f_c' is given by equation 2.1. In this equation '$x_c(r)$' represents the 'r^{th}' complex QPSK symbol to be modulated onto the 'c^{th}' sub-carrier.

$$b_c(t) = \begin{cases} \sum_{k=0}^{M} x_c(r-k) \cdot h((t - r \cdot T_s) + k \cdot T_s) & for \;\; t_s = r.T_s \\ \sum_{k=0}^{M-1} x_c(r-k) \cdot h((t - r \cdot T_s) + k \cdot T_s) & for \;\; r \cdot T_s < t < (r+1) \cdot T_s \end{cases}$$

Equation 2.1: Analogue Sub-carrier Modulated Output Signal

After, '$b_c(t)$', is multiplied by a complex exponential to modulate it up to the appropriate sub-carrier frequency, 'f_c', it is combined with the other 'N-1' sub-carrier '$b_c(t)$' signals to give the 'N' sub-carrier complex modulator output signal 'o(t)'.

$$o(t) = \begin{cases} \sum_{c=0}^{N-1} \left(\sum_{k=0}^{M} x_c(r-k) \cdot h((t - r \cdot T_s) + k \cdot T_s) \right) \cdot \exp\{j2\pi f_c t\} & for \; t = r \cdot T_s \\ \sum_{c=0}^{N-1} \left(\sum_{k=0}^{M-1} x_c(r-k) \cdot h((t - r \cdot T_s) + k \cdot T_s) \right) \cdot \exp\{j2\pi f_c t\} & for \; r \cdot T_s < t < (r+1) \cdot T_s \end{cases}$$

Equation 2.2: Modulated System Complex Output Signal

The value of 'f_c', in equation 2.2 is defined as '$f_c = f_m + c/T_s$' to ensure orthogonality between the sub-carriers, where 'f_m' is the fixed centre frequency of the first sub-carrier. From this point onwards without any loss of generality, 'f_m' is set to zero, so as to simplify further analysis.

2.3.2 Digital Implementation of Modulator

To obtain a digitally sampled form of equation 2.2, the analogue time index 't' is mapped to '$(i/Nv + r)T_s$' (equation 2.3), where 'N' is the total number of sub-carriers and 'v' the over-sampling factor.

$$t \rightarrow \left(\frac{i}{Nv} + r\right) \cdot T_s \qquad where\ i \in \{0,1,...,Nv-1\}\ and\ r\ is\ an\ integer$$

Equation 2.3: Modulator Analogue-to-Digital Mapping Function

After simplification, the substitution of equation 2.3 into 2.2, yields an expression (equation 2.4) for the digitally sampled Spectrally Shaped OFDM modulator output signal, '$o_d(i, r)$'. Note that the '$T_s/2$' offset delays necessary to maintain true orthogonality between the overlapping adjacent sub-carriers, ignored in equations 2.1, 2.2 and 2.4, are to be analysed later without loss of generality now.

$$o_d((\frac{i}{Nv}+r)\cdot T_s) = \begin{cases} \sum_{k=0}^{M}\left(\sum_{c=0}^{N-1} x_c(r-k)\cdot \exp\{j2\pi c\cdot\frac{i}{Nv}\}\right)\cdot h((\frac{i}{Nv}+k)\cdot T_s) & for\ i=0 \\ \sum_{k=0}^{M-1}\left(\sum_{c=0}^{N-1} x_c(r-k)\cdot \exp\{j2\pi c\cdot\frac{i}{Nv}\}\right)\cdot h((\frac{i}{Nv}+k)\cdot T_s) & for\ i\in\{1,2,...,Nv-1\} \end{cases}$$

Equation 2.4: Sampled Modulator Output Signal

From equation 2.4 it can be deduced how a DFT-PPN structure can be applied to synthesise the modulator output signal, '$o_d(i,r)$'. For example, the inner summation term (*'c = 0...N-1'*) corresponds to the output of a 'Nv' point complex IDFT whose first 'N' inputs at an instance in time defined by 'r', are the 'N' sub-carrier complex QPSK symbols ('a + jb'). The remaining 'N(v-1)' inputs of this IDFT are always zero. If these IDFT producible terms are replaced by '$f_i(r-k)$', where function '$f_i(r)$' represents the IDFT operation described above, then equation 2.5 can be obtained.

$$o_d((\frac{i}{Nv}+r)\cdot T_s) = \begin{cases} \sum_{k=0}^{M} f_i(r-k)\cdot h((\frac{i}{Nv}+k)\cdot T_s) & for\ i=0 \\ \sum_{k=0}^{M-1} f_i(r-k)\cdot h((\frac{i}{Nv}+k)\cdot T_s) & for\ i\in\{1,2,...,Nv-1\} \end{cases}$$

Equation 2.5: Digital Modulation incorporating DFT function

For a given value of 'i', equation 2.5, represents the convolution integral of sequence '$f_i(r)$' with a PolyPhase Network (PPN) sub-filter '$h_i(n)$', whose real

coefficients are defined by equation 2.6. Note that all of the parallel PPN sub-filters, '$h_i(n)$', except '$h_0(n)$', have finite impulse responses which are asymmetric, and that 'M' must be even to ensure the overall PPN filter has a linear phase response.

$$h_i(n) = \begin{cases} h(n \cdot T_s) & \text{for } i = 0 \quad \& \quad n = 0,1,2,\ldots,M \\ h(n \cdot T_s + (\frac{i}{Nv} \cdot T_s)) & \text{for } i \in \{1,2,3,\ldots,Nv\text{-}1\} \quad \& \quad n = 0,1,2,\ldots,M\text{-}1 \end{cases}$$

Equation 2.6: Coefficients of Modulator PPN Sub-filters

Since the DFT operation '$f_i(r)$' in equation 2.5, occurs effectively instantly once all 'N' sub-carrier QPSK symbols have been received, the following PPN sub-filter operations occur simultaneously. Therefore to ensure that the outputs of these sub-filters are produced in the correct order, a set of staggered delays are required after the PPN. Figure 2.5 shows how the combination of these staggered delays, PPN sub-filters ($h_i(n)$) and DFTs ($f_i(r)$) need to be combined to generate the digitally sampled Spectrally Shaped OFDM modulator output signal defined by equation 2.5.

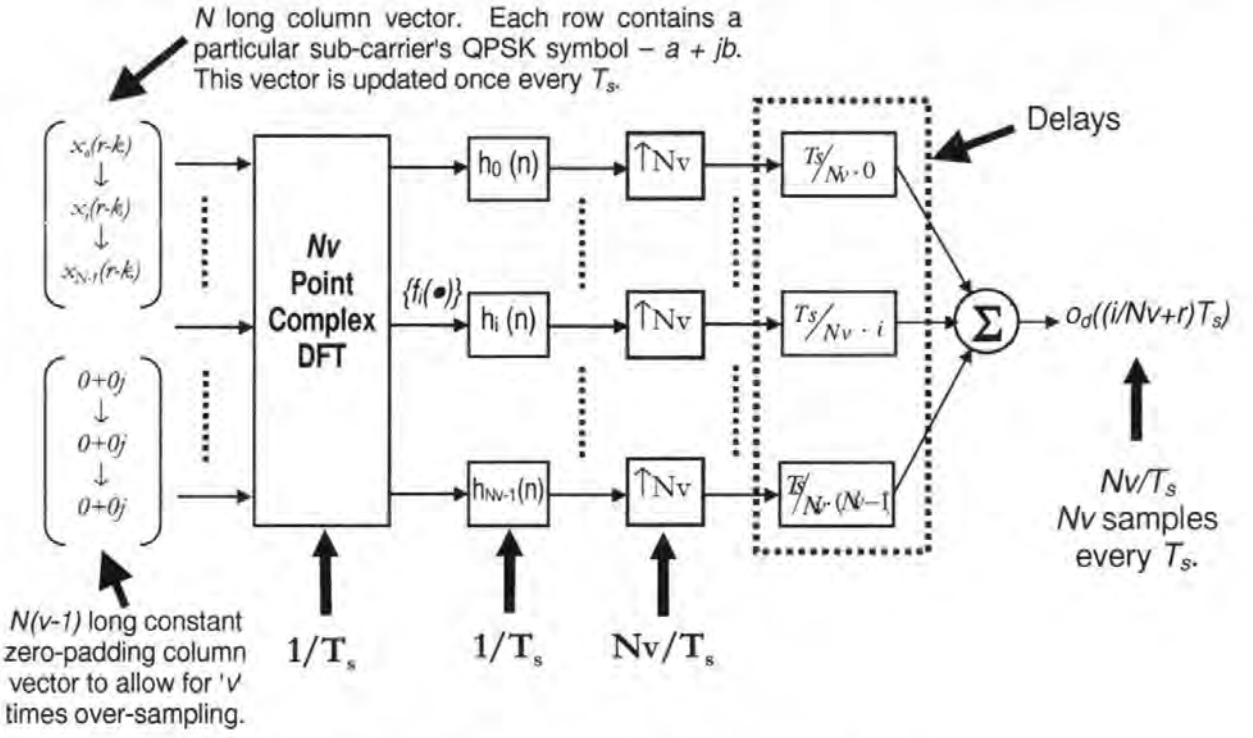


Figure 2.5: Modulation of Non-Offset Symbol Components

The DFT-PPN structure shown in figure 2.5 can only be used to modulate the real or imaginary halves of sub-carrier QPSK symbols not offset by a delay of '$T_s/2$'. To process the remaining halves of symbols, which for the set-up shown in figure 2.3 include the real part of sub-carrier '0' and the imaginary part of sub-carrier '1', a separate modified DFT-PPN sub-system operating in parallel is required. Modifying this parallel DFT-PPN sub-system by adding a '$T_s/2$' delay, does not, by itself, produce the desired effect as shown in equation 2.7.

$$o_d\left(\left(\frac{i}{Nv}+r-\tfrac{1}{2}\right)\cdot T_s\right)=\begin{cases}\left(\sum_{k=0}^{M}\left(\sum_{c=0}^{N-1}x_c(r-k)\cdot\exp\{-j2\pi c\cdot\frac{i}{Nv}\}\cdot\exp\{j\pi c\}\right)\cdot h\left(\left(\frac{i}{Nv}+k-\tfrac{1}{2}\right)\cdot T_s\right)\right) & for\ i=0\\ \left(\sum_{k=0}^{M-1}\left(\sum_{c=0}^{N-1}x_c(r-k)\cdot\exp\{-j2\pi c\cdot\frac{i}{Nv}\}\cdot\exp\{j\pi c\}\right)\cdot h\left(\left(\frac{i}{Nv}+k-\tfrac{1}{2}\right)\cdot T_s\right)\right) & for\ i\in\{1,2,\ldots,Nv-1\}\end{cases}$$

Equation 2.7: Anomaly introduced by *'$T_s/2$'* Offset Delay

From equation 2.7, it can be seen that the inclusion of the '$T_s/2$' offset delay introduces undesirable phase shifts of 'πc' to the output of the DFT terms in the square brackets. These phase offsets can be effectively equalised out by multiplying the input symbols into the offset DFT by equal but opposite phase offsets of '$\exp\{j\pi c\}$' as shown by equation 2.8.

$$x_c(r-k)\rightarrow x_c(r-k)\cdot\exp\{j\pi c\}$$

Equation 2.8: Correction of *'$T_s/2$'* Offset Delay Anomaly

The list below summarises the three key stages of pre-processing that input QPSK symbols must undergo before being processed by the two parallel DFT-PPN sub-systems.

- Zero-padding with *'N(v-1)'* zero values, to allow for *'v'* times over-sampling.
- Separation of real and imaginary QPSK symbol parts for non-offset and *'$T_s/2$'* offset DFT-PPN sub-system processing.

- Multiplication of separated QPSK symbol components for processing by offset DFT-PPN structure, by '*exp{jπc}*' phase shift factors.

Figure 2.6 shows how the various digital sub-system processes described above can be synthesised to create a flexible DSP architecture for the modulation of QPSK symbols onto Spectrally Shaped OFDM sub-carriers.

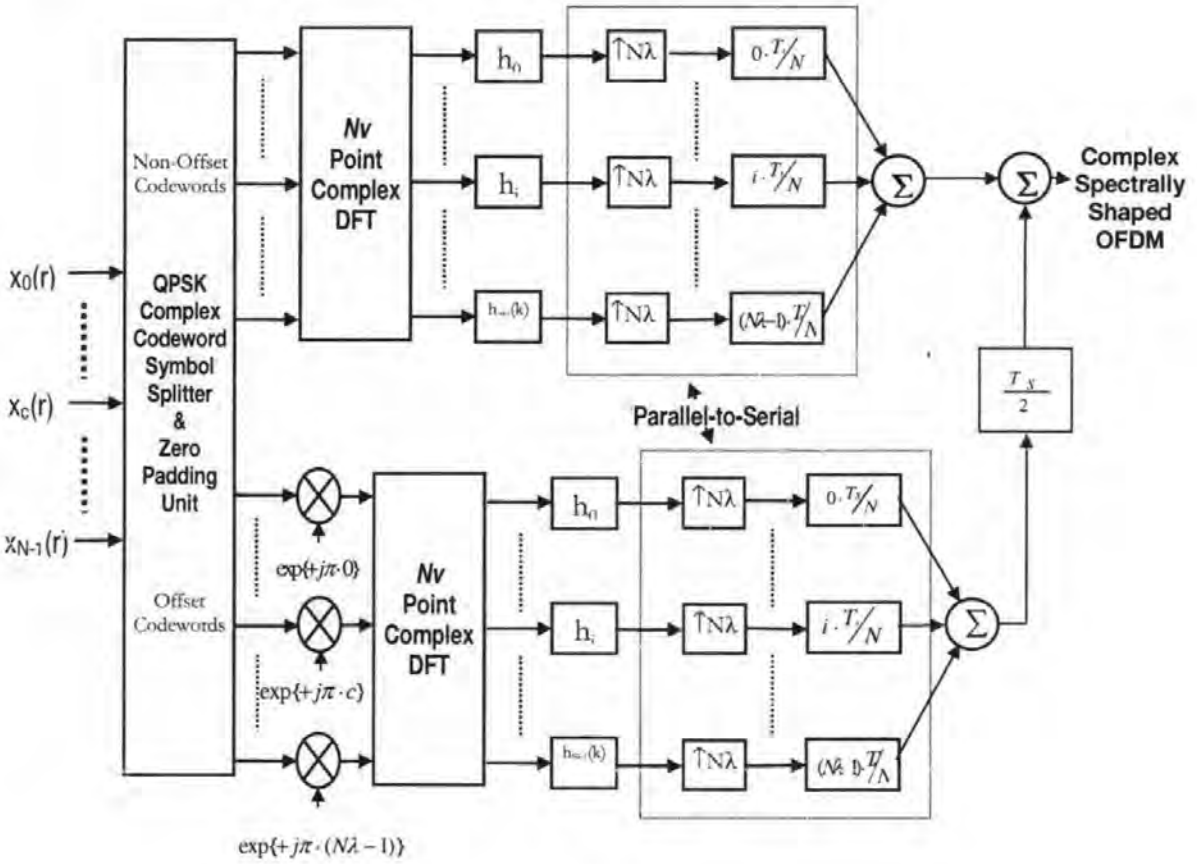


Figure 2.6: Detailed DFT- PPN Schematic of Modulator

2.3.3 Improved Digital Implementation of Modulator

For the original 'N' sub-carrier Spectrally Shaped OFDM modulator presented in figures 2.6 and 2.7, two N-channel DFT-PPN sub-systems operate in parallel to produce the desired complex sampled signal. In this sub-section a novel method is described for exploiting the alternating real and imaginary pattern of the inputs (figure 2.7) into these sub-systems to reduce the overall DSP burden of the original design. Note that for the time being, without loss of generality, the most basic implementation of the system, i.e. that without any over-sampling, is considered.

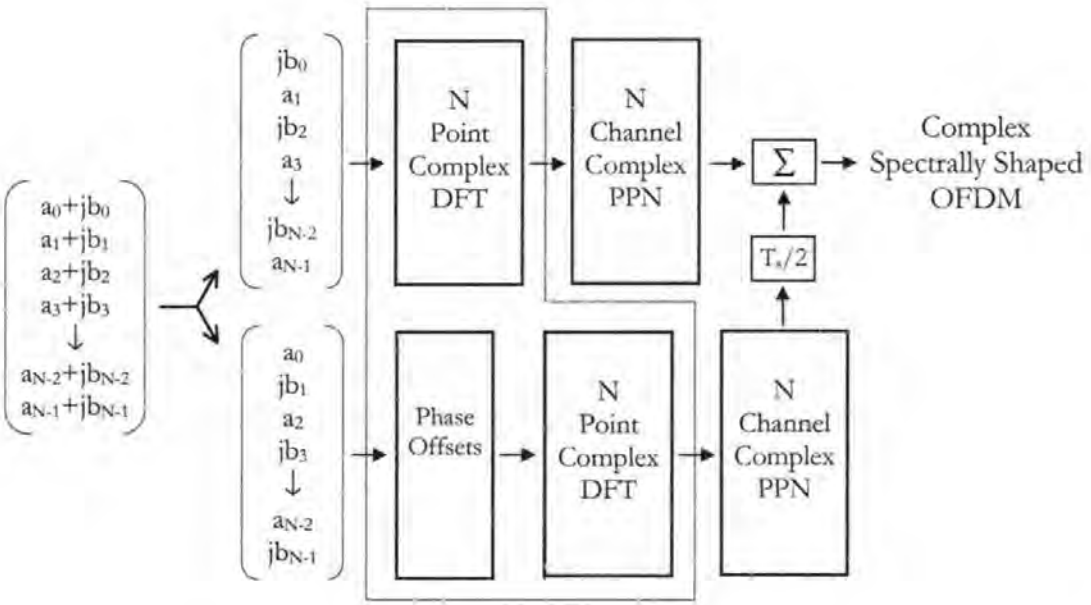


Figure 2.7: Simplified DFT- PPN Schematic of Modulator

From standard decomposition theory (ref. [8][45]) it can be shown that a 'N' point complex DFT is equivalent to two 'N/2' point complex DFTs whose outputs are combined by a set of 'twiddle' factors. Figure 2.8 shows how such an arrangement can be used to replace the 'N' point DFT in either of the sub-systems. Note how for this modified DFT-PPN structure, each of the 'N/2' DFTs has either entirely real or imaginary input sequences.

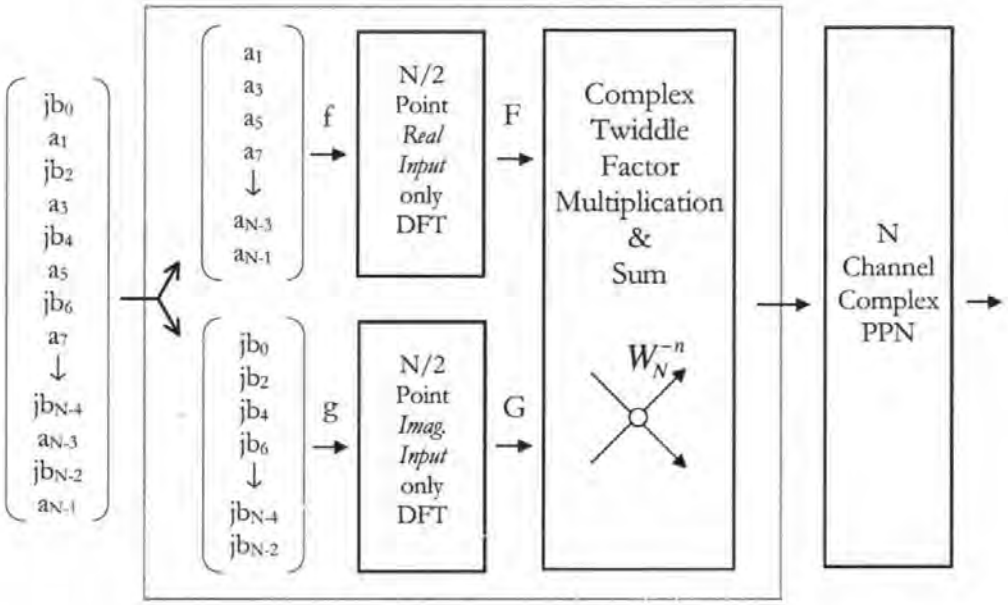


Figure 2.8: Double Complex N/2 DFT Structure

The completely real or imaginary nature of the input sequences of the two complex 'N/2' point DFTs, allows them to be replaced by a single complex 'N/2' point DFT with some minor post-processing as illustrated by figure 2.9. A full derivation showing how this is possible is provided in Appendix B. Figure 2.10 shows how this single 'N/2' point DFT with associated pre- and post-processing phases, can replace the 'N' point DFT in the original DFT-PPN sub-system structure (figure 2.7) without any loss of functionality.

The radix-2 decimation-in-time FFT (ref. [21],[58]), requires '$2N\log_2 N$' real multiplications and '$3N\log_2 N$' real additions to perform a single 'N' point complex DFT. Therefore if the associated pre- and post-processing phases operations are ignored, the implementation of a 'N/2' point FFT in place of the original 'N' point FFT, will more than halve the number of computations required for the DFT phase for each of the sub-systems.

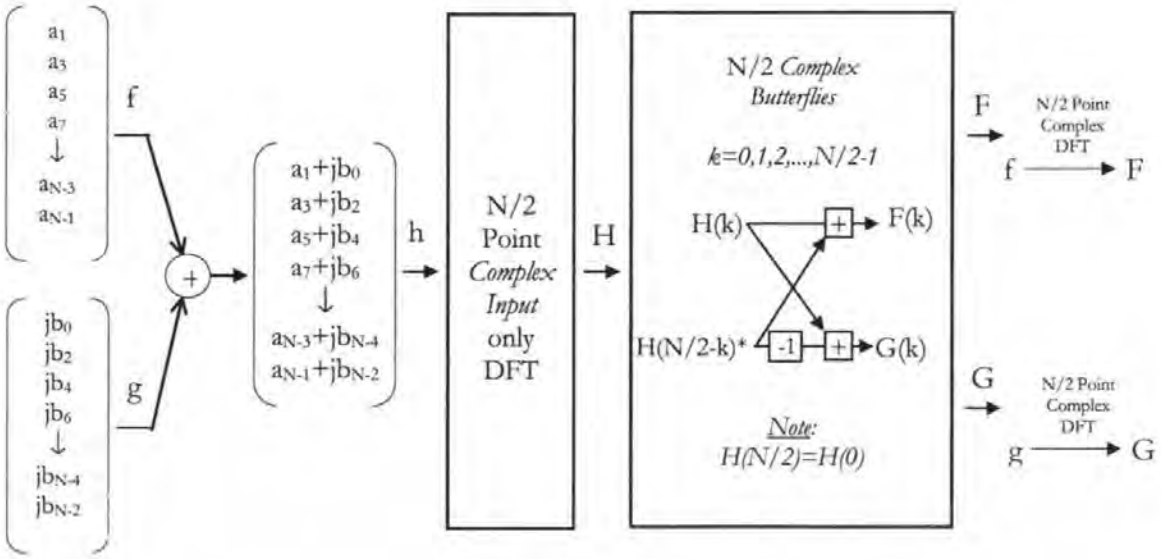


Figure 2.9: Single Complex N/2 DFT Structure

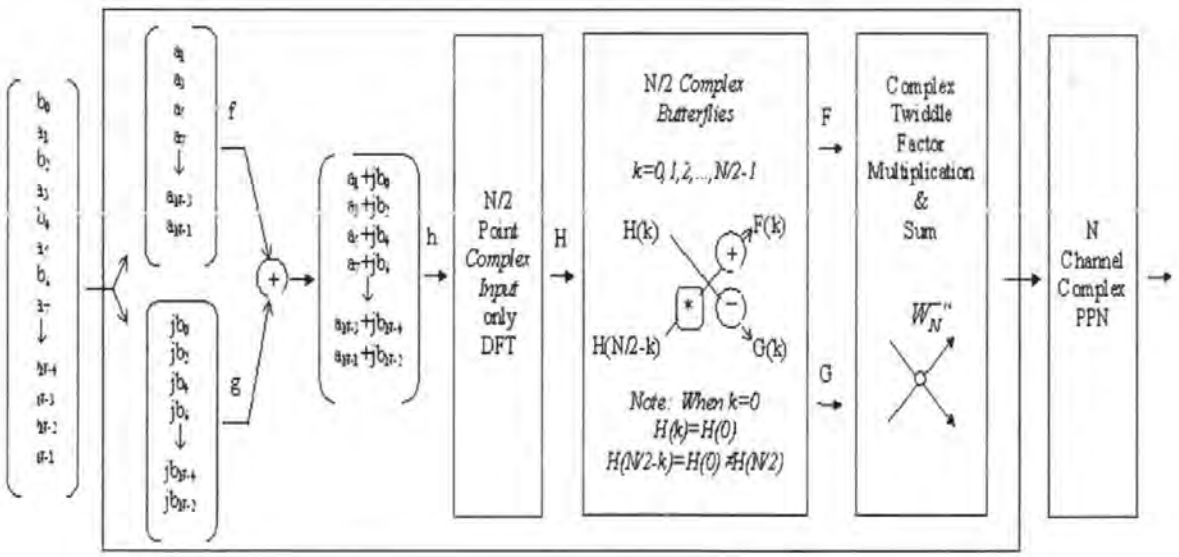


Figure 2.10: Modified N/2 DFT- PPN Sub-System

2.3.4 Analogue Model of Demodulator

Figure 2.11 shows the complex analogue baseband model of a Spectrally Shaped OFDM demodulator (compare with figure 2.3).

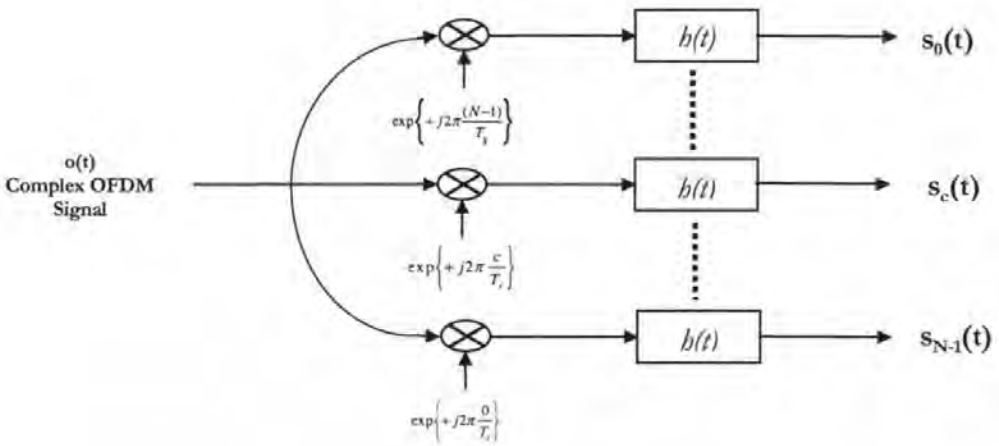


Figure 2.11: Complex Analogue Model of Demodulator

The orthogonality of the system ensures that both the non-offset and '$T_s/2$' offset real and imaginary components of a complex QPSK symbol transmitted along the 'c^{th}' sub-carrier, can be demodulated by sampling the corresponding demodulator sub-carrier filter branch output, '$s_c(t)$' (equation 2.9), at a frequency of '$2/T_s$'.

$$s_c(t) = h(t) * (o(t) \cdot \exp\{j2\pi f_c t\})$$

Equation 2.9: Demodulated Sub-Carrier Complex Output Signal

2.3.5 Digital Implementation of Demodulator

To obtain a digitally sampled form of '$s_c(t)$', from which the PPN-DFT implementation of the demodulator can be deduced, the time index 't' is mapped to '$(\alpha N - \beta)T_s/N$' using equation 2.10.

$$t \rightarrow (\alpha \cdot N - \beta) \cdot \frac{T_s}{N} \qquad \text{where } \beta \in \{0, 1, 2, ..., N-1\} \,\&\, \alpha \text{ is an integer}$$

Equation 2.10 : Demodulator Analogue-to-Digital Mapping Function

$$s_c((\alpha - \frac{\beta}{N}) \cdot T_s) = \sum_{i=0}^{MN} h(i \cdot \frac{T_s}{N}) \cdot o_c((\alpha \cdot N - i) \cdot \frac{T_s}{N} - \beta \cdot \frac{T_s}{N}) \cdot \exp\{-2\pi \cdot \frac{c}{N} \cdot i) \cdot \exp\{-2\pi \cdot \frac{c}{N} \cdot \beta)$$

Equation 2.11 : Sampled Demodulator Sub-carrier Output

For a given value of 'β', equation 2.11 yields samples of '$s_c(t)$' spaced 'T_s' apart, that are offset from the start time ('t = 0') by '$T_s\beta/N$'. Therefore when '$\beta = 0$', assuming correct synchronisation, equation 2.11 effectively defines a sampled sequence consisting of the demodulated components of symbols transmitted along sub-carrier 'c' without a '$T_s/2$' offset delay. Similarly when '$\beta = N/2$' the same equation defines a sampled sequence consisting of the demodulated components of symbols transmitted along sub-carrier 'c' with a '$T_s/2$' offset delay.

By re-expressing the convolution summation term within equation 2.11 using the mapping described by equation 2.12, an expression for '$s_c(\alpha, \beta)$' can be derived from which the PPN-DFT based DSP architecture for a digital demodulator can be deduced.

$$\sum_{i=0}^{MN} x(i) = \sum_{n=0}^{N-1} \sum_{m=0}^{f(n)} x(m \cdot N + n) \qquad f(n) = \begin{cases} M & for\ n = 0 \\ M - 1 & for\ n \in \{1,2,3,...,N-1\} \end{cases}$$

Equation 2.12: Convolution Integral Mapping

After simplification the substitution of equation 2.12 into equation 2.11 gives the following expression.

$$s_c((\alpha-\frac{\beta}{N})\cdot T_s)=\sum_{n=0}^{N-1}\left[\sum_{m=0}^{f(n)}h((m)\cdot T_s+\frac{n}{N}\cdot T_s)\cdot o((\alpha-m)\cdot T_s-(n+\beta)\cdot\frac{T_s}{N})\right]\cdot\exp\{-j2\pi\cdot\frac{c}{N}\cdot n)\cdot\exp\{-j2\pi\cdot\frac{c}{N}\cdot\beta)$$

Equation 2.13: Expanded Sampled Demodulator Sub-carrier Output

For fixed values of 'n' and 'β', the term in square brackets represents the convolution of sequence $\{o_c(\alpha, m)\}$ with a PolyPhase Network sub-filter '$h_i(n)$'. This filtering process is repeated 'N' times, giving rise to a 'N' sub-filter PolyPhase Network (PPN) filter. The sub-filters of this PPN have coefficients (equation 2.14) derived from the original sub-carrier RRC shaping filter (h(t)).

$$h_i(n)=\begin{cases}h(n\cdot T_s) & where\ i=0\ \&\ n=0,1,2,\ldots,M\\ h(n\cdot T_s+i\cdot\frac{T_s}{N}) & where\ i\in\{1,2,3,\ldots,N-1\}\ \&\ n=0,1,2,\ldots,M-1\end{cases}$$

Equation 2.14: Coefficients of Demodulator PPN Sub-filters

To produce the correct input sequences, $\{o_c(\alpha, m)\}$, for these sub-filters, the demodulator input signal has to be delayed and then sub-sampled at a rate of '$1/T_s$'. The value of a delay is chosen to ensure that the outputs of the PPN sub-filters are generated simultaneously. These delay values must also take into account that the first PPN sub-filter has one extra coefficient compared to the other sub-filters.

With 'β' set to zero, a complex DFT of the 'N' simultaneously generated PPN output samples produces '$s_c(\alpha.T_s)$'. Assuming correct synchronisation, the 'c^{th}' complex output sample from this DFT equals the non-offset part (real or imaginary component) of the complex QPSK symbol modulated on to the 'c^{th}' sub-carrier.

With a non-zero 'β', a complex DFT of the PPN output samples introduces undesirable phase shift terms of '$\exp\{-j2\pi.c.\beta/N\}$' to the desired DFT output samples. These phase shift terms can be removed by either multiplying the 'c^{th}' DFT output by the equal but opposite exponential term or by shuffling around the PPN input sequence into the DFT. The equivalence of these techniques is shown by equation 2.15.

$$\exp\{-j^{2\pi}/_{N}\cdot c\cdot\beta\}\cdot\left[\sum_{n=0}^{N-1}x(n)\exp\{-j^{2\pi}/_{N}\cdot c\cdot n\}\right]=$$
$$\sum_{n=0}^{\beta-1}x(n+\beta)\exp\{-j^{2\pi}/_{N}\cdot c\cdot n\}+\sum_{n=\beta}^{N-1}x(n-\beta)\exp\{-j^{2\pi}/_{N}\cdot c\cdot n\}$$

Equation 2.15: Equivalence of Demodulator Phase Correction Techniques

Figure 2.12 shows how the various digital signal processes described above can be synthesised to create a flexible PPN-DFT based DSP architecture for the demodulation of QPSK symbols modulated onto Spectrally Shaped OFDM sub-carriers. Note that in the complete demodulator architecture, two of the PPN-DFT structures shown in figure 2.11 (one with '$\beta = 0$' and the other with '$\beta = N/2$') operate in parallel with the same input signal to produce the non-offset and offset components of the modulated QPSK symbols. When interleaved in time, the sub-carrier outputs of these two parallel sub-systems produce a digital version of the analogue signal described by equation 2.9, sampled at twice the symbol rate, '$2/T_s$'. Over-sampling can be achieved by adding further sub-systems in parallel to the two already defined with '$\beta = \{0, N/2\}$'. For example to achieve four times over-sampling of the signal described by equation 2.9, a total of four sub-systems operating in parallel would be required with '$\beta = \{0, N/4, N/2, 3N/4\}$'.

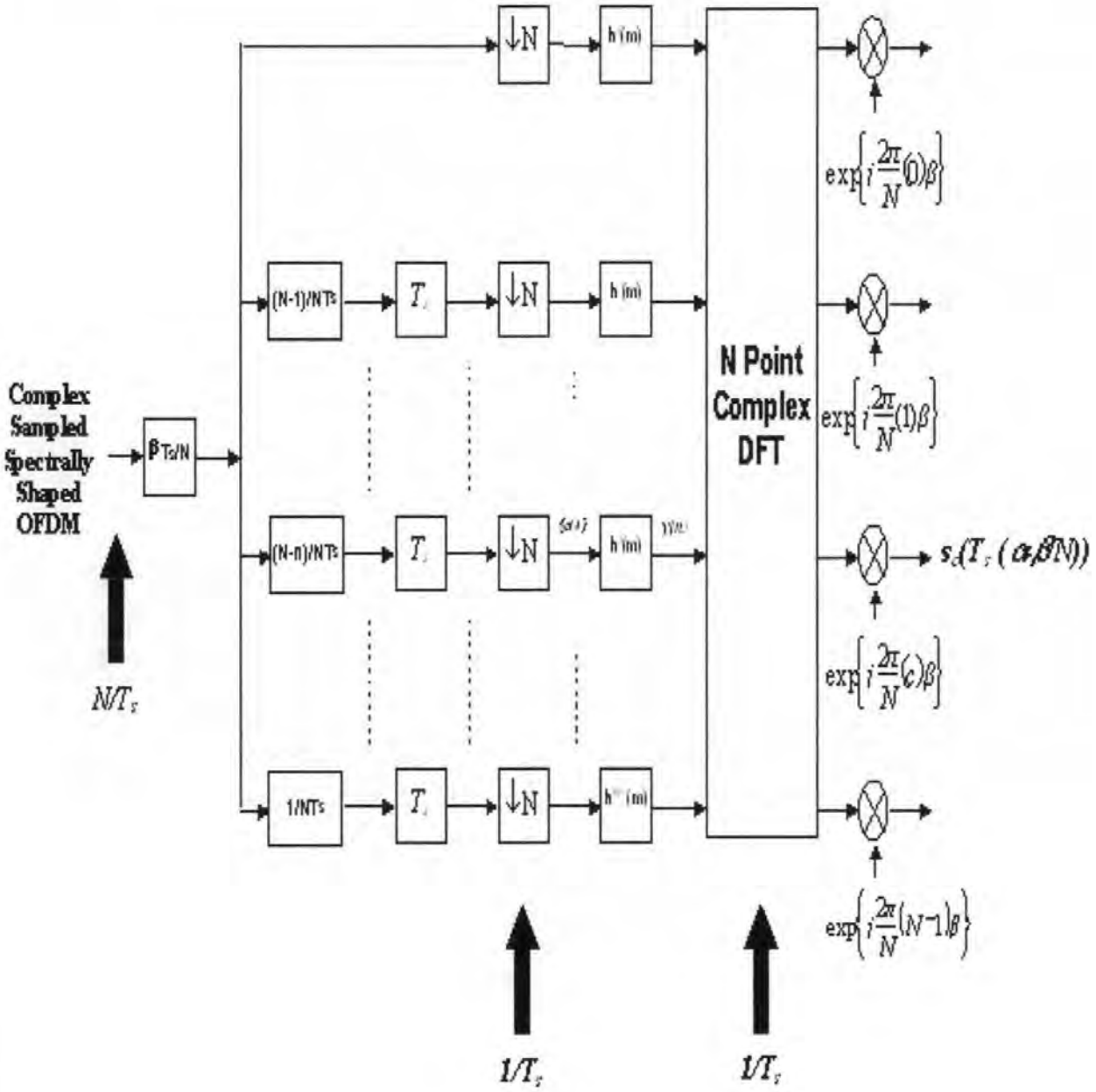


Figure 2.12: Demodulator PPN-DFT Sub-System

2.4 Conclusions

In this chapter, a flexible DFT-PPN based DSP architecture for a digital Shaped OFDM modulator and demodulator has been derived, which is capable of,

- Over-sampling.
- Modulating QPSK symbols. Note that it is possible to extend the analysis described in this chapter to prove that the architecture presented is also capable of modulating multi-phase and multi-level complex QAM symbols.
- Being dimensioned for any number of sub-carriers.
- Supporting orthogonal RRC and other (non-orthgonal) sub-carrier shaping functions.

This architecture has been fully tested and verified (with up to 512 sub-carriers and 16x over-sampling) using both a work-station based communication system simulation package known as SPW and a custom-coded PC based C-code model described in Appendix C. All Spectrally Shaped OFDM results presented in this thesis have been obtained from the C-coded model of the novel DSP architectures presented in this chapter.

The key feature of this design that differentiates it from previous designs, in particular Hirosaki's (ref. [34],[35]), is that it operates fully in the complex digital domain. Due to this difference in the domains in which these two designs operate, the digital hardware processing power required to implement this novel architecture is far less than the existing architecture. Table 2.1 highlights the particular features of this novel architecture that significantly reduce its computational requirements. Another advantage of this novel architecture, highlighted in the final row of table 2.1, is that far fewer PPN coefficients are required to achieve the same Signal-to-Distortion performance as the existing design. Further detail regarding both these design features can be found in chapter 3.

Feature	*Existing Digital Architecture*	*Novel Digital Architecture*
Minimum Sampling Rate of Modulated Signal	$2N/T_s$	N/T_s
Number and Size of DFT required for Modulator DFT-PPN per T_s	*2 N point complex DFTs*	*2 N/2 point complex DFTs*
Number and Size of PPN required for Modulator DFT-PPN per T_s	*2 NM point PPNs* *where* M ≈ 14	*2 NM' point PPNs* *where* M' ≈ 8

Table 2.1: Key Differences between existing and novel Design

Soon after the work for this chapter had been completed, G. Cariolaro and F. Vagliani published (ref. [10]) an alternative derivation of the same DSP architecture for a Spectrally Shaped OFDM modem.

CHAPTER THREE

3. Design and Optimisation of System Parameters

In this chapter the consequences of varying the number of sub-carriers and characteristics of the RRC sub-carrier pulse shaping filter, on the spectrum efficiency, implementation complexity and output signal-to-distortion ratio, of a Spectrally Shaped OFDM system are examined.

3.1 Introduction

A modulation scheme encodes message information onto a carrier wave by varying in a deterministic manner any one or more of the following wave characteristics, amplitude, frequency and phase. The key criteria used to assess the suitability of a modulation scheme for a particular application include:

- Spectrum Efficiency
- Implementation Complexity
- Communication Performance

No single modulation scheme can perform well in all the above criteria, and it is the responsibility of the system engineer to trade-off certain criteria against others according to the application requirements, the limitations of the technology available, costs and the available bandwidth.

In this chapter, the consequences of varying the following three Spectrally Shaped OFDM system parameters,

- The number of sub-carriers.
- The number of RRC sub-carrier pulse shaping filter coefficients.
- The roll-off factor and windowing of RRC sub-carrier pulse shaping filter coefficients.

on the spectrum efficiency, implementation complexity and output signal-to-distortion level, are theoretically and experimentally analysed. In section 3.2 the spectral properties of a range of Spectrally Shaped OFDM systems are determined. The implementation complexity of this modulation scheme in the digital baseband and RF processing domains are then examined in sections 3.3 and 3.4 respectively. Finally in section 3.5 the benefits and drawbacks of the spectral and implementation characteristics of Spectrally Shaped OFDM, are summarised and compared with similar Sinc(x) OFDM systems.

3.2 Spectrum Efficiency

Wireless radio channel bandwidth is becoming an increasingly valuable resource. In fact, many government organisations responsible for assigning bandwidth are currently engaged in auctioning allocations of new spectrum to the possible service providers of 3rd generation public cellular systems, such as UMTS. In this section a theoretical expression along with a set of simulation results are presented that show how the 'in-band' bandwidth and 'out-of-band' spectral efficiencies of a Spectrally Shaped OFDM system vary with respect to the number of sub-carriers and the roll-off factor of the RRC sub-carrier shaping function.

3.2.1 Bandwidth Efficiency

The bandwidth efficiency, 'η', of a modulation scheme is defined as the number of bits that can be transmitted per hertz. This is equivalent to the ratio of the modulation data rate (R bits/s) to required bandwidth (B hertz).

$$\eta = \frac{R}{B}$$

Equation 3.1: Bandwidth Efficiency

The overall bandwidth, 'B', occupied by a 'N' sub-carrier Spectrally Shaped OFDM system with sub-carrier bandwidths of '2δ' and a symbol period of 'T_s' is given by equation 3.2.

$$B = \frac{N-1}{T_s} + 2\delta$$

Equation 3.2: Spectrally Shaped OFDM Bandwidth (Part i)

To maintain orthogonality, the adjacent overlapping sub-carriers of a Spectrally Shaped OFDM system are RRC shaped and are spaced apart by '$1/T_s$'. Given these conditions, an expression (equation 3.3) for 'δ' can be derived in terms of the roll-off factor of the RRC sub-carrier shaping function.

$$\delta = \frac{1+a}{2T_s}$$

Equation 3.3: Spectrally Shaped OFDM Roll-Off Factor Spectrum

From substitution of equation 3.3 into 3.2, equation 3.4 can be formed expressing for bandwidth occupied by the main 'in-band' lobe (see figure 3.2) of a 'N' sub-carrier Spectrally Shaped OFDM system.

$$B = \frac{N}{T_s}\left(1 + \alpha/N\right)$$

Equation 3.4: Spectrally Shaped OFDM Bandwidth (Part ii)

The data rate for such a system assuming the sub-carriers are modulated by M-PSK symbols at a rate of '$1/T_s$', is given by equation 3.5.

$$R = \frac{N \log_2 M}{T_s}$$

Equation 3.5: Spectrally Shaped OFDM Bit-Rate

From combining equations (3.2), (3.3) & (3.5), the bandwidth efficiency for an 'N' sub-carrier Spectrally Shaped OFDM system can be derived.

$$\eta = \frac{\log_2 M}{\left(1 + \frac{\alpha}{N}\right)}$$

Equation 3.6: Spectrally Shaped OFDM Bandwidth Efficiency

For QPSK transmission (M = 4), 'η' approaches the Nyquist asymptotic limit of 2 bits/hertz when 'N' is large and 'α' is small.

3.2.2 Spectral Efficiency

In theory the bandwidth efficiency of a modulation scheme, equation 3.1, defines the minimum amount of bandwidth required to transmit a single 'bit' of information. However due to the non-ideal practical implementation of the idealised filters assumed by the equation 3.3, the bandwidth required in reality is much greater due to the surplus energy radiated in out-of-band side-lobes. These side-lobes generate undesirable inter-carrier interference (ICI) in adjacent carriers, which limit how closely together adjacent carriers can be placed, thereby limiting the overall multi-carrier system bandwidth efficiency. Sidelobe energy can best be analysed using the signal's Power Density Spectrum (PDS). Figure 3.1 shows an example of the PDS for a single carrier QPSK modulation scheme with 0.22 roll-off factor Root Raised Cosine (RRC) carrier shaping function. Due to the non-ideal finite length implementation of the RRC pulse shaping filter (8 symbol periods long), slowly decaying side-lobes are observed which would not be present if ideal filtering were possible.

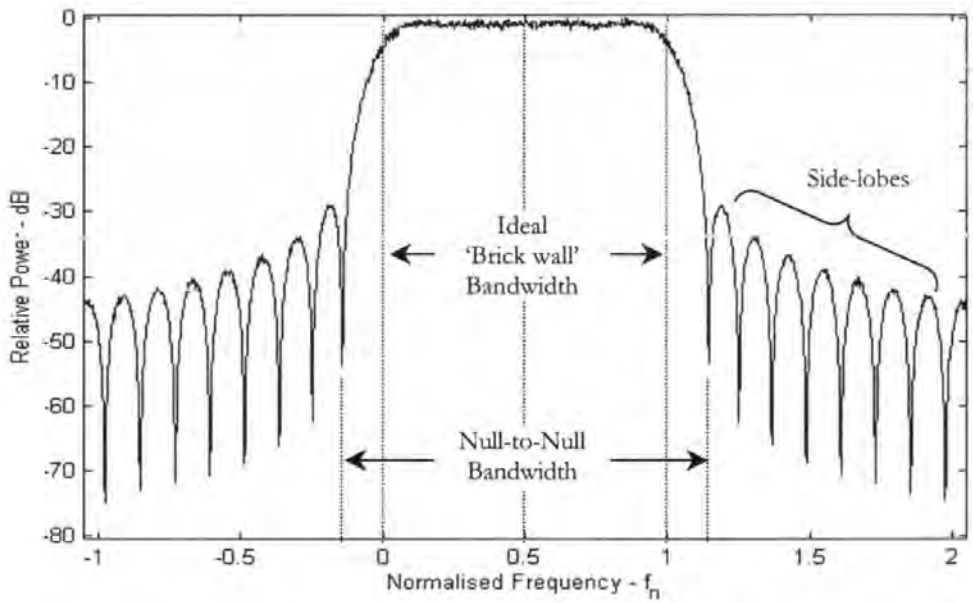


Figure 3.1 : PDS for Single-Carrier QPSK System with roll-off of 0.22

The side-lobe energies for different modulation schemes can decay in very different ways. Several quantitative definitions for measuring how rapidly they fall-off exist, including the half-power bandwidth, null-to-null bandwidth, 99% energy containment bandwidth and the 35 dB bandwidth. A more complex definition commonly used throughout the wireless industry, is Adjacent Channel Protection (ACP). ACP is the ratio of the transmitted power to the power measured at the output of the receiver filter for the adjacent channel. This quantity combines a measure of out-of-band emission with receiver sensitivity. Ultimately none of these quantitative measures can summarise the side-lobe characteristics of a modulation scheme as well as a PDS. Hence to determine the out-of-band spectral performance behaviour with respect to the roll-off factor and the number of sub-carriers a set of Power Density Spectra plots (figures 3.2-3.5) have been generated.

3.2.3 Power Density Spectra Analysis

In order to analyse and compare the bandwidth and spectral efficiency of this modulation scheme with different number of channels and values of the filter roll-off factor, a number of simulations were performed (using the model described in chapter 2) to generate a set of finite length modulator output signals. These signals were then processed by a standard MatLab PDS function to generate comparable power density spectra plots. All the systems simulated in this section are conceptually parameterised, such that they all share the same raw bit-rate across the idealised 'brick-wall' frequency bandwidth of '1'. Therefore a single carrier QPSK system with a symbol rate of '1 s^{-1}', has a spectrum that is comparable to a N sub-carrier QPSK OFDM system with a sub-carrier symbol rate of '1/N s^{-1}'. Given this normalisation of the frequency axis it is possible to deduce the 'in-band' bandwidth efficiency graphically, by dividing the number of bits/symbol by the main lobe null-to-null bandwidth, as shown in figure 3.1 for the Single Carrier QPSK RRC modulation scheme.

To ensure comparable results, eliminate spurious effects and minimise the overall computational load within the limits of the available resources, the following steps were taken.

- All systems were over-sampled 8 times to avoid aliasing in the PDS region of interest.

- The PPN Filters for all the simulated systems were dimensioned to have 8.0 taps per sub-carrier filter regardless of the total number of system sub-carriers. This figure of 8 taps per sub-filter was chosen to minimise the self-noise introduced by the finite length of the filter and arithmetic quantisation effects (see section 3.3.2 for further information).

- The overall RRC FIR coefficients, used to derive coefficients of the PPN sub-filter simulated, were truncated to the desired length using a simple rectangular window.

- The length of output signal generated was fixed to 400000 complex samples (assuming 8 times over-sampling).

- The total length of the MatLab derived PDS generated for all systems was fixed to 8192. This provided a reasonable trade-off between the maximum resolution (frequency bin width) of the PDS and the noise limiting averaging performed across the 400000 length input signal.

To investigate the effects of changing the number of sub-carriers and the roll-off factor of the sub-carrier RRC pulse shaping function the following features of the simulation systems were varied.

- To determine the effects, shown in figures 3.2 to 3.3, of varying the number of sub-carriers between 4 and 32 on the spectral performance, the roll-off factor of the simulated systems RRC PPN FIR was fixed to 1.0.

- To determine the effects, shown in figures 3.4 to 3.5, of varying the roll-off factor between 1.0 and 0.25 on the spectral performance, the number of sub-carriers of the simulated systems was fixed to 4.0.

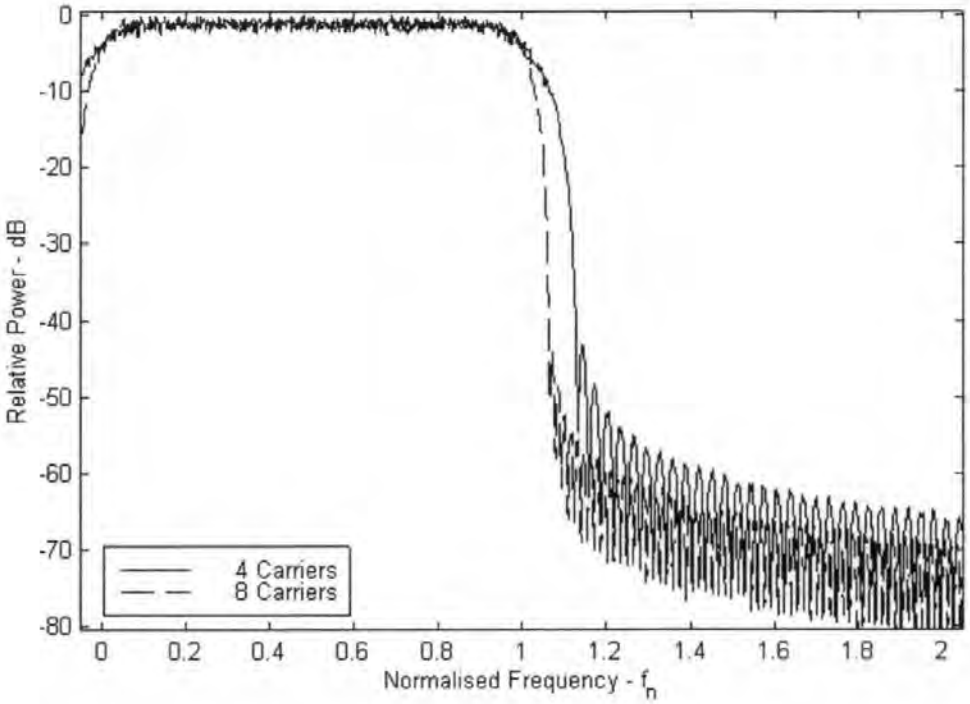


Figure 3.2 : PDS for 4 & 8 Sub-Carrier System

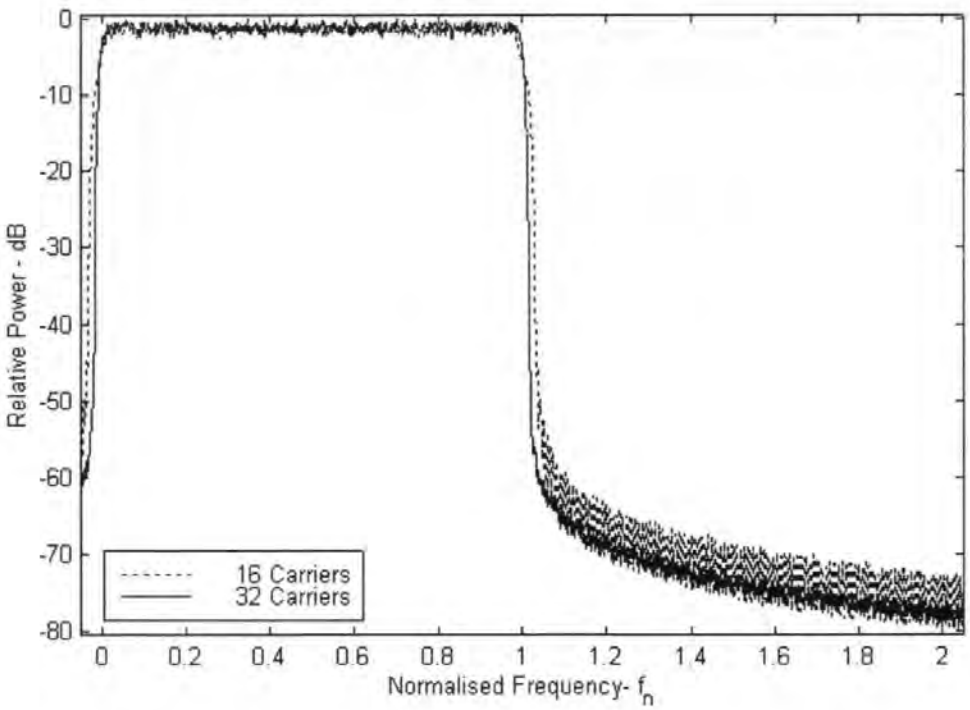


Figure 3.3 : PDS for 16 & 32 Sub-Carrier System

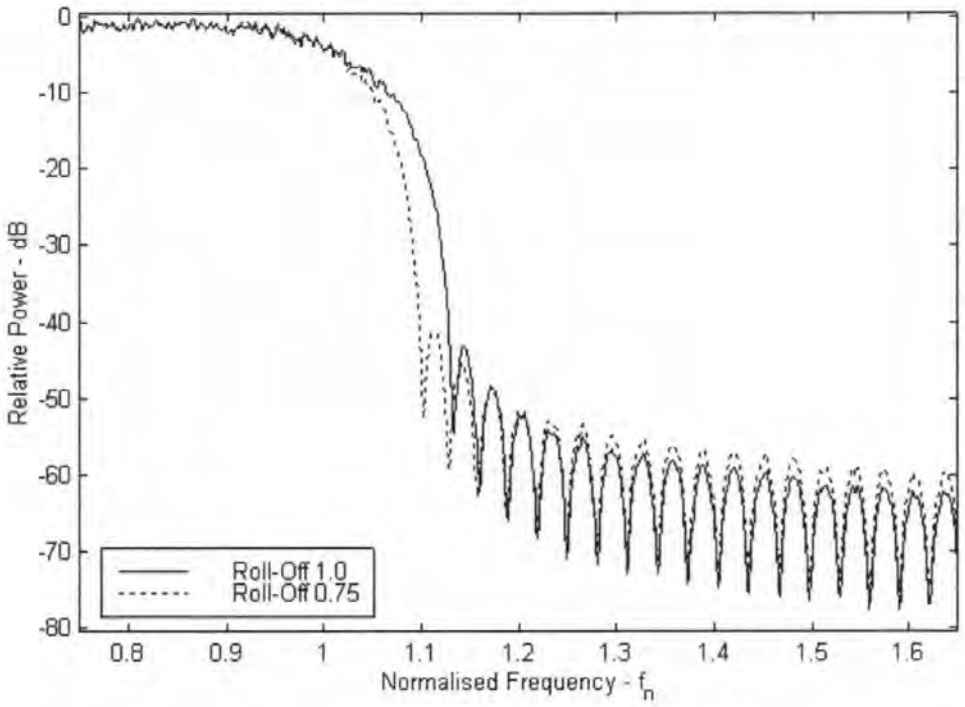


Figure 3.4 : PDS for 1.0 & 0.75 RRC Roll-Off Factor Systems

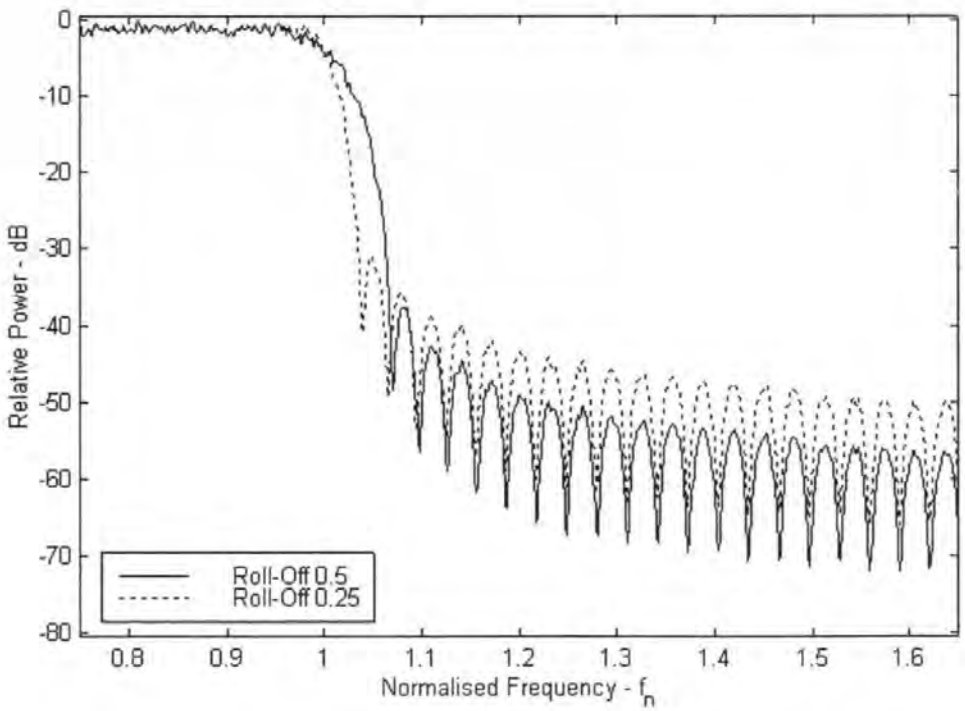


Figure 3.5 : PDS for 0.5 & 0.25 RRC Roll-Off Factor Systems

From figures 3.2 and 3.3 it can been clearly seen that increasing the number of sub-carriers has two beneficial effects.

- Improves the in-band bandwidth efficiency, as predicted by equation 3.6.
- Reduces the power level of the out-of-band side-lobe emissions.

From figure 3.2, where the side-lobes are still visible given the resolution of the Power Density Spectrum relative to the width of the side-lobes themselves, one can determine the main lobe null-to-null bandwidth for the 4 and 8 channel systems as being approximately 1.2 and 1.1 respectively. This corresponds to bandwidth efficiencies of 1.67 and 1.82 bits/Hz for the 4 and 8 channel system respectively. These values verify those obtained from the theoretical expression (equation 3.6) for bandwidth efficiency. Therefore even with only 4 carriers and the 'worst' possible roll-off factor of 1.0, this modulation scheme has spectral qualities comparable to the best 0.22 RRC single carrier QPSK modulation scheme (figure 3.1). Even with only 32 sub-carriers figure 3.3 confirms that 'in-band' bandwidth efficiency of a simple Spectrally Shaped system quickly approaches the asymptotic limit of 2 bits/Hz predicted by equation 3.6.

Figures 3.4 and 3.5, show that decreasing the roll-off factor of the RRC sub-carrier pulse shaping for a Simple Spectrally Shaped OFDM system,

- Improves the in-band bandwidth efficiency.
- Increases the undesirable power level of the out-of-band side-lobe emissions.

The side lobe power level at the ideal adjacent carrier centre frequency ($f_{normalised}$ = 1.5) from the centre frequency of the main carrier ($f_{normalised}$ = 0.5), ranges from -62 dB to -50 dB as the roll-off factor of the pulse shaping falls from 1.0 to 0.25. This improvement of 12 dB in the side-lobe power level obtained by applying a roll-off factor of 1.0 instead of 0.25, will in most practical systems be far useful than the improvement in bandwidth efficiency obtained from adopting the much lower roll-off factor.

Theoretical power density spectra for similarly dimensioned Sinc(x) OFDM systems were generated using the equation 3.7 (ref. [70] pg. 196), to allow comparison with those already obtained for Spectrally Shaped OFDM.

$$S_{vv}(f) = \sum_{n=0}^{N-1} \left| H_a\left(f - \frac{1}{T}\left(n - \frac{N-1}{2} \right) \right) \right|^2$$

Equation 3.7: Sinc(x) OFDM Bandwidth

Assuming that $H_a(f)$ is equal to basic sinc(f) function, equation 3.7 describes the ideal PDS for a N sub-carrier sinc(x) OFDM modulation scheme without windowing or secondary band-pass filtering to reduce the excessive side-lobe growth.

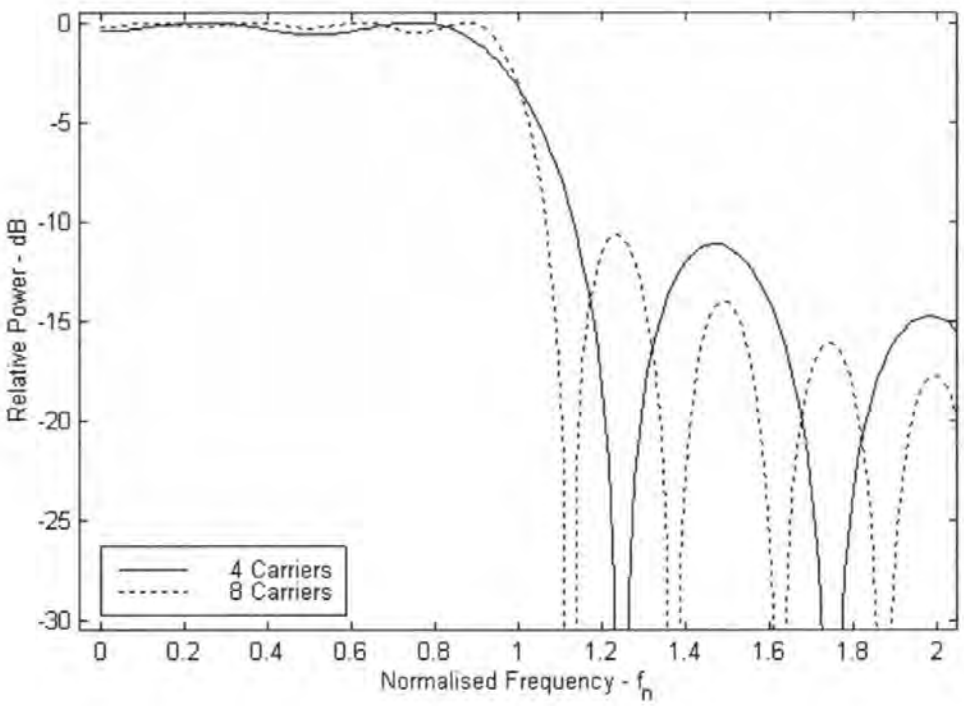


Figure 3.6 : PDS for 4 & 8 carrier Sinc(x) OFDM

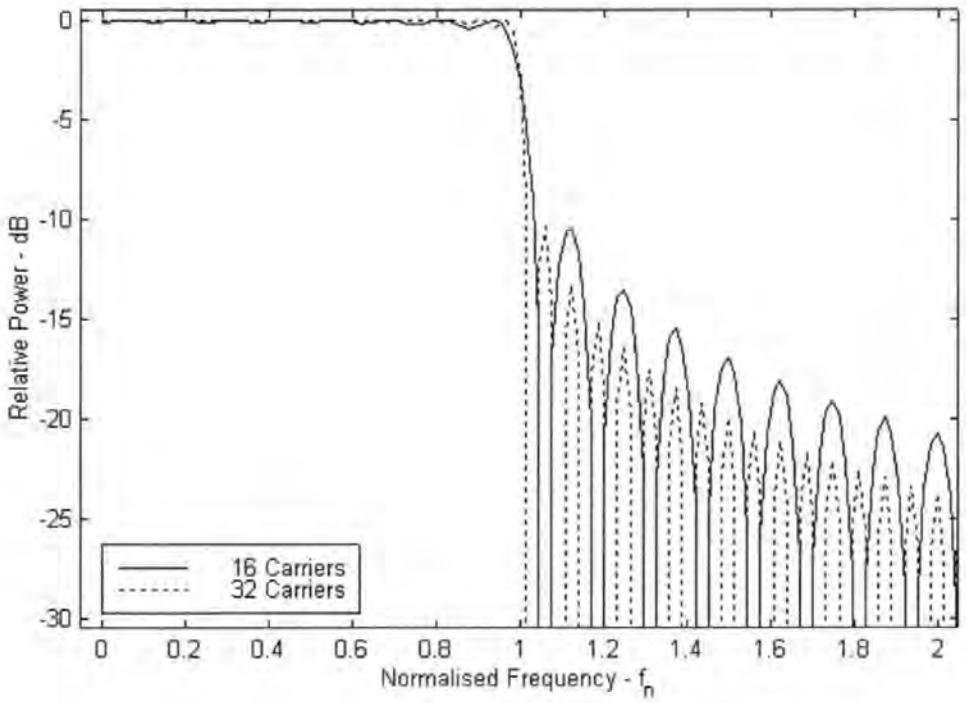


Figure 3.7 : PDS for 16 & 32 carrier Sinc(x) OFDM

Comparing figures 3.6 and 3.7 with those for Spectrally Shaped OFDM (figures 3.2 to 3.5), it can be seen that these OFDM schemes, with identical numbers of sub-carriers, share very similar in-band bandwidth efficiencies, but have dramatically different out-of-band spectral side-lobe levels. For non-ideal practical 4-32 channel systems, the maximum side-lobe energy at the idealised adjacent carrier centre frequency ($f_{normalised}$ = 1.5) for Spectrally Shaped OFDM is already down in the –60 to –70 dB range whereas for the ideal theoretical Sinc(x) OFDM systems it is still in the –10 to -20 dB range.

To reduce these excessively high side-lobes, Sinc(x) OFDM systems often employ a combination of the following techniques.

- Time domain windowing (typically the Tukey function) of the modulated Sinc(x) OFDM symbol samples – this reduces the side-lobe level but at the expense of reducing 'in-band' bandwidth efficiency. This form of windowing should not be

confused with that applied to truncate the RRC FIR used to derive the coefficients of the Spectrally Shaped OFDM PPN sub-filters.

- Secondary filtering in the RF domain – though highly effective this has the drawback of increasing in-band noise and can be costly.

- Dummy carriers – this is where the end sub-carriers are simply not used. This technique trades 'in-band' bandwidth (or throughput) efficiency for 'out-of-band' spectral efficiency.

In summary, Spectrally Shaped OFDM has better spectral and bandwidth efficiencies than comparable Single Carrier QPSK or Sinc(x) OFDM systems without the need for secondary filtering. For even a low number of sub-carriers (4) and high roll-off factor (1.0), this modulation scheme already has spectral properties as good as the best 0.22 roll-off factor single carrier QPSK systems. As the number of sub-carriers is increased, these properties only improve further, with the scheme's main-lobe approaching an ideal 'brick-wall' spectrum with as few as 32 carriers, giving approximately a 20% improvement in bandwidth efficiency over single carrier QPSK (0.22 RRC). An interesting system design trade-off observed, was the effect of varying the roll-off factor of the PolyPhase Network. Decreasing this parameter improves bandwidth efficiency but at the expense of increasing the side-lobe emissions.

3.3 BaseBand Implementation Complexity

When designing the receiver or transmitter hardware for a particular modulation scheme, the engineer has to take into account both the RF and base-band processing component costs and limitations. In this section the key processing requirements for a Spectrally Shaped OFDM modem in the digital baseband domain are investigated and quantified where possible.

3.3.1 Baseband Digital Processing Requirements

The critical quantities that have to be taken into account when designing and dimensioning digital hardware for baseband processing tasks such as the modulation and demodulation, are the maximum rate of real multiplications and additions and the number of storage elements required. From the descriptions of the DSP implementation of modulator and demodulator for Spectrally Shaped OFDM presented in the previous chapter, functions (table 2.1) have been derived to quantify these processing function and element requirements in terms of,

- M : The length of RRC FIR in symbol periods, used to derive the coefficients of the Spectrally Shaped OFDM PPN sub-filters. This also corresponds to the number of taps per each PolyPhase Network Sub-Filter assuming no over-sampling is applied.

- N : The Number of Data Transmitting Channels (sub-carriers) assuming no over-sampling.

- T_s : Modulation Symbol Period, which also equals the inverse of the sub-carrier frequency domain spacing.

Processing Function/Element	***MODULATOR # Processing Function/Element required per symbol period (T_s)***	***DEMODULATOR # Processing Function/Element required per symbol period (T_s)***
Real Multiplications	$2N \log_2 \frac{N}{2} + 4(NM + 1)$	$4N \log_2 N + 4(NM + 1)$
Real Additions	$3N \log_2 \frac{N}{2} + 4NM$	$6N \log_2 N + 4NM$
Memory Elements	$2N(2M + 1) + 4$	$4N(2M + 1) + 4$

Table 3.1: Baseband Processing Costs

The following assumptions were made to derive the functions listed in table 3.1,

- Minimum over-sampling is assumed for both modulator and demodulator.
- Standard Radix-2 FFT implementation. This restricts N to being a power of 2.
- Trivial multiplication's included (thereby making these estimates somewhat pessimistic).
- 4 real multiplication's and 6 additions per radix two complex 'butterfly' multiplication.
- Real valued PolyPhase Filter Network coefficients are assumed.
- A memory element in this context is defined as a real valued read-write memory register. This definition excludes the category of 'static' memory elements required for storing constants (the filter coefficients and FFT twiddle factors).

3.3.2 Design of PolyPhase Network Filter

From table 3.1 and figure 3.8, it can be seen that 'M', the number of taps per each of the 'N' PPN sub-filters, has a significant effect on the complexity of the overall system. For example raising 'M' from 7 to 8 for an 'N = 32' system would increase the complexity (total number of operations per symbol period) of the system by 10%. As well as reducing the overall implementation complexity, minimising 'M' also reduces the overall delay between when a symbol is modulated and demodulated. The performance enhancing advantages obtained from schemes, which rely on rapid feedback between the transmitter and receiver (e.g. fast closed loop power control) increase as the size of this delay falls. Unfortunately these benefits obtained from minimising 'M' have to be traded against the increased internally generated distortion introduced into the system by shortening the RRC FIR of PPN filter relative to the ideal infinitely long RRC FIR.

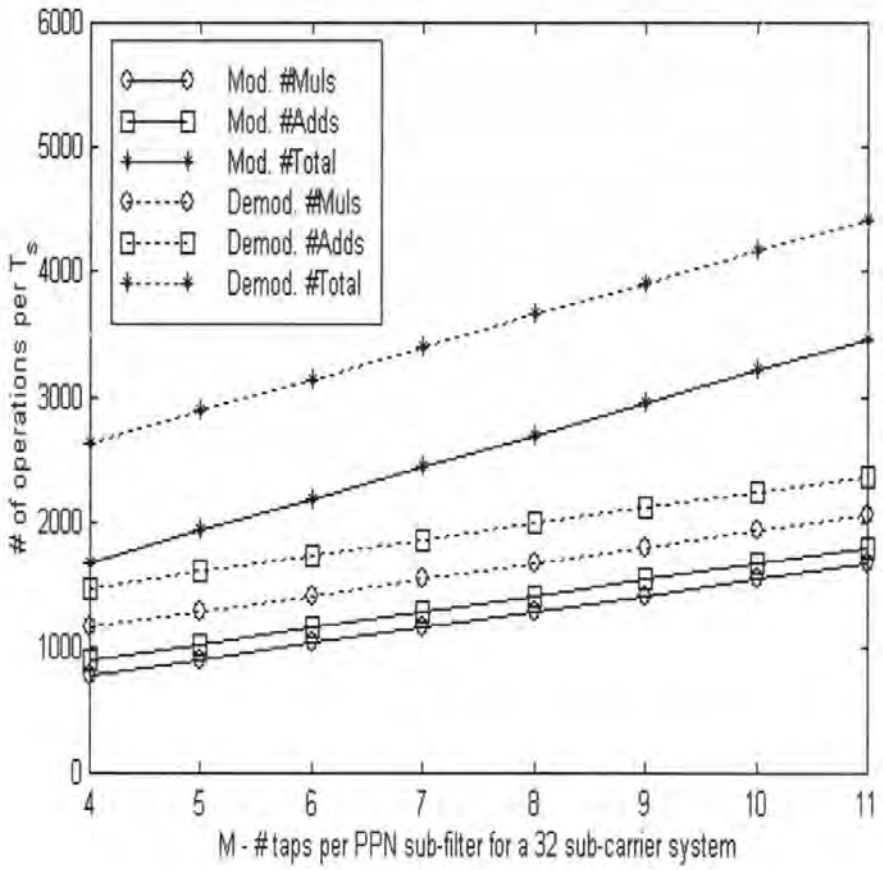


Figure 3.8: Implementation Complexity Modulator

Regarding the optimum number of PPN taps, Hiroskai in his 1981 paper, ref. [31], states,

> *"... the required order M of the FIR is almost proportional to the number of sub-bands L. If as an example, allowable distortion-to-signal (D/S) of –30 dB is assumed, then the required order M can well be approximated by 14L."*

Note that Hirosaki's notation is different from the one presented in this thesis, thus Hirosaki's 'M' and 'L' correspond to 'MN+1' and 'N' respectively. Given this mapping of notation, the above text states that 14 taps per PPN sub-filter is sufficient to obtain a Distortion-to-Signal (D/S) Ratio of –30 dB and that this figure is effectively independent of the number of sub-carriers. In this context the D/S ratio defines the power of mean 'error vector' between the ideal and actual demodulated symbols to the power of the ideal demodulated complex data symbol, which would be $\sqrt{2}$ for sub-carriers modulated with '$\pm 1 \pm i$' QPSK symbols. This definition assumes no external channel or RF phase induced distortion.

Figure 3.9 shows the S/D (the inverse of the D/S ratio) ratio versus roll-off factor characteristic for five systems sharing the same 'Hamming' (as opposed to Rectangular) window truncated RRC FIR (used to derive the PPN coeffcients) and number of taps (M = 8 in this case) per PPN sub-filter but with differing numbers of channels ($N \in \{ 4, 8, 16, 32, 64 \}$). These results verify Hirosaki's conclusion that the maximum achievable S/D ratio is dependent on the number of taps per PPN sub-filter.

Other design parameters of the PolyPhase Network that affect the overall system's signal-to-distortion ratio performance, spectral efficiency and implementation complexity, include the value of the roll-off factor, choice of RRC FIR truncation function and spacing between the filter taps themselves.

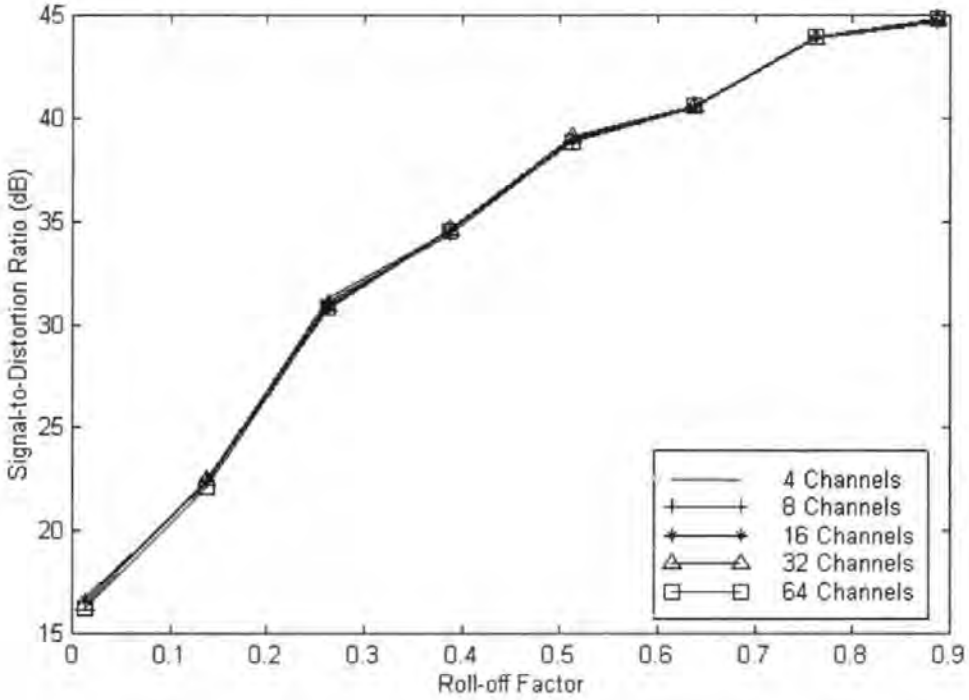


Figure 3.9: S/D Ratio versus Roll-off factor Characteristic.

In 1998, Takahata et al, published a paper, ref. [71], outlining their design of an orthogonal but non-overlapping multi-carrier QPSK modem based on the DFT-PPN structure. In this paper, experimental results are presented and analysed to determine the,

- Minimum number of filter taps per PPN sub-filter to achieve a given Signal-to-Distortion (S/D) level.
- Optimum value of the roll-off factor given a fixed number of filter taps to maximise the Signal-to-Distortion (S/D) level.

From their results, Takahata et al, concluded that for their particular modem,

"...a tap length of 7 is required to achieve a D/U (Desired to Undesired signal power ratio) of 27 dB in the noise budget."

"the relationship between the D/U and the roll-off factor, which suggests that a roll-off factor of 0.3-0.4 is optimum for a tap length of 7."

To verify some of these conclusions, a number of simulations were performed. The first of these experiments was to verify Takahata's conclusion regarding the relationship between the D/U ratio (the same as the S/D ratio) and the roll-off factor of the PPN stated immediately above. Figure 3.10 shows how the S/D ratio varies with respect to the roll-off factor and RRC FIR truncation function for a simple 16 sub-carrier system, with 8 taps per PPN sub-filter. The differences between these results and Takahata's can be attributed to the different system characteristics (such as the number and spacing of sub-carriers and the filter size), the arithmetic accuracy applied and the larger set of values of the roll-off factor simulated here.

The results presented in figure 3.10 confirm Takahata's main finding, that when the RRC FIR (used to derive the PPN coeffcients) is truncated using a simple rectangular window, there exist values of roll-off factor that maximise and minimise the S/D ratio. To investigate why these peaks and troughs appear (something not explained by other researchers) a plot (figure 3.11) of the absolute values of the end coefficients of the RRC FIR was generated. From comparing figures 3.10 and 3.11, it can be seen the troughs occur at roll-off factors (0.36, 0.62 & 0.87) which generate PPN filter end coefficients with absolute values equal to zero.

The results presented in figure 3.10, also show that the simple application of a Hamming window on the PPN filter coefficients, smoothes out the peaks and troughs in the S/D characteristic, but at the cost of lowering the maximum achievable S/D for a given roll-off factor. In addition, this curve shows that increasing the roll-off factor of a fixed sized PPN filter leads to an increase in the maximum achievable S/D ratio.

For a given number of PPN filter taps, increasing the roll-off factor and/or removing any form of windowing improves the S/D ratio. However both these techniques to improve the S/D ratio bring about potentially undesirable side-effects to the spectrum of the modulated signal.

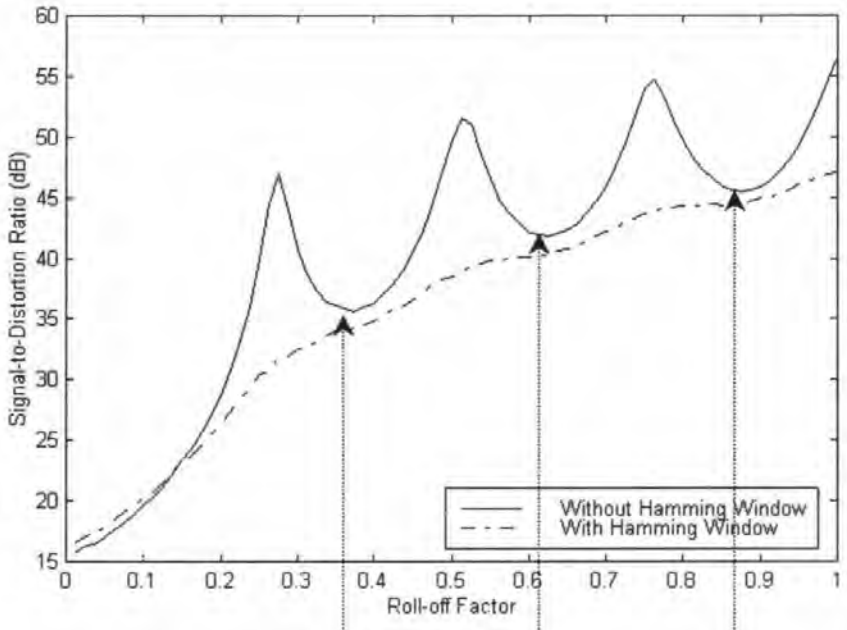


Figure 3.10: PPN Windowing & Roll-off Effects of S/D Ratio

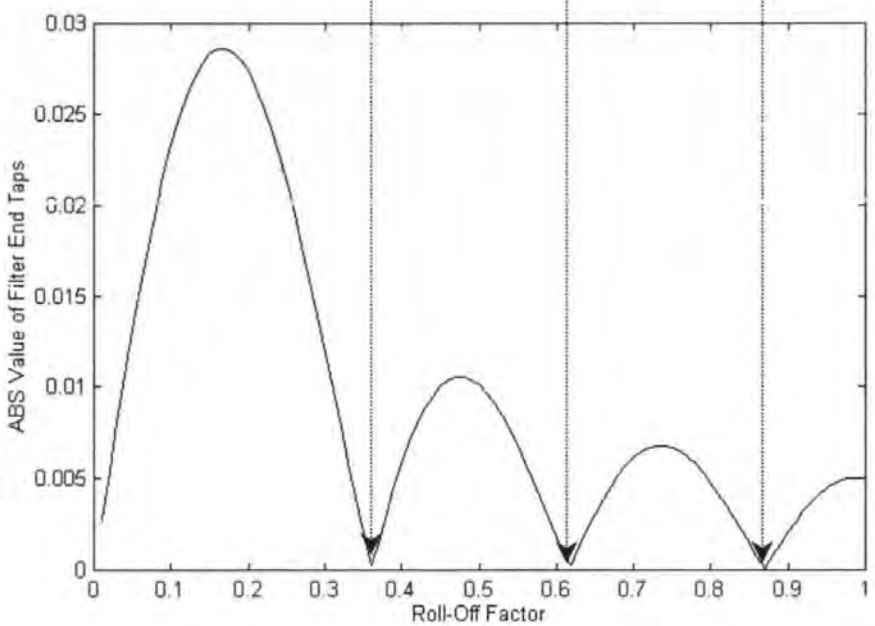


Figure 3.11: Absolute Values of PPN filter network end coefficient

Finally, in figure 3.12, the effects of varying the oversampling of the PPN sub-filter taps for a 16 sub-carrier (N = 16) system with a constant number of taps per sub-filter (and hence constant number of PPN taps in total) are shown. These results indicate that for Spectrally Shaped OFDM systems restricted to a fixed number of PPN taps, the S/D ratio can be maximised by reducing the over-sampling to a minimum.

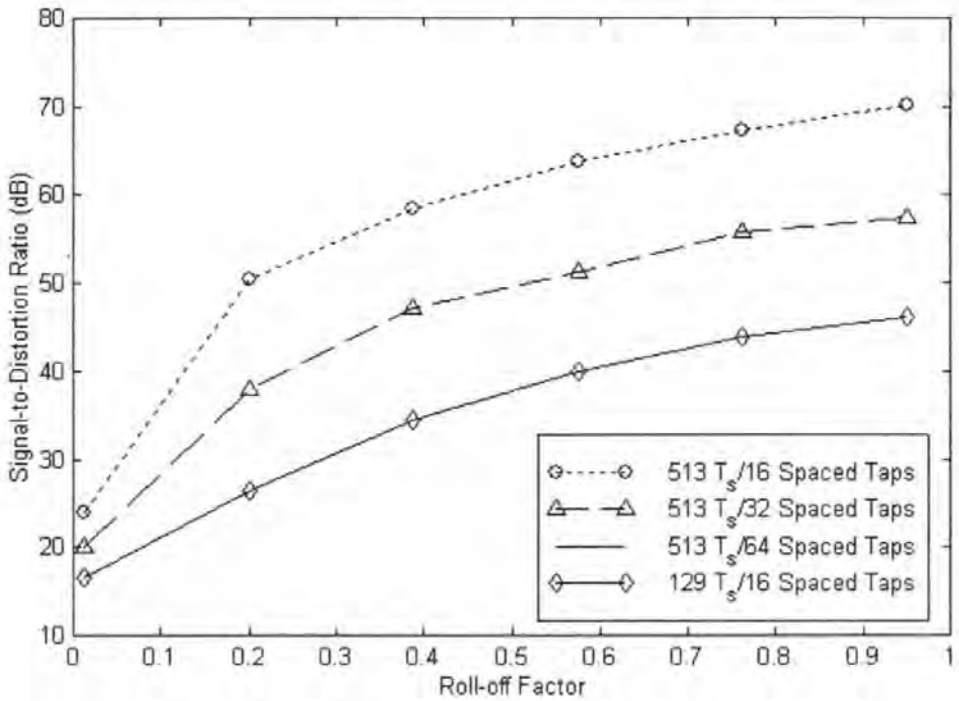


Figure 3.12: Effects of varying PPN Filter Over-sampling Factor

3.4 RF Implementation Complexity

The complexity of the hardware design of the RF front ends of the modulator and demodulator depend largely on the requirements for filtering, synchronisation and power amplification. Some of these requirements, such as filtering, are strongly influenced by external standards bodies, who impose strict regulations on the allowable out-of-band emissions of a system. Others, such as power amplification, are more influenced by the choice of modulation scheme. In this section we examine the linear power amplifier requirements of Spectrally Shaped OFDM, and compare them with those for Sinc(x) OFDM.

3.4.1 Power Amplifier Design

There are 3 classes, A, B and C, of power amplifier (PA) that are commonly used in industry. Class C amplifiers are defined as those for which the power is likely to be on for less than ½ of the time. This class of power amplifiers is predominantly used in the handsets of wireless mobile terminals, where,

- To fit into hand held terminals they must be extremely compact
- To minimise the power drain on the mobile terminal battery unit, they must be extremely power efficient.

Due to these requirements on size and power efficiencies these amplifiers invariably have non-linear power amplification characteristics as shown in figure 3.13.

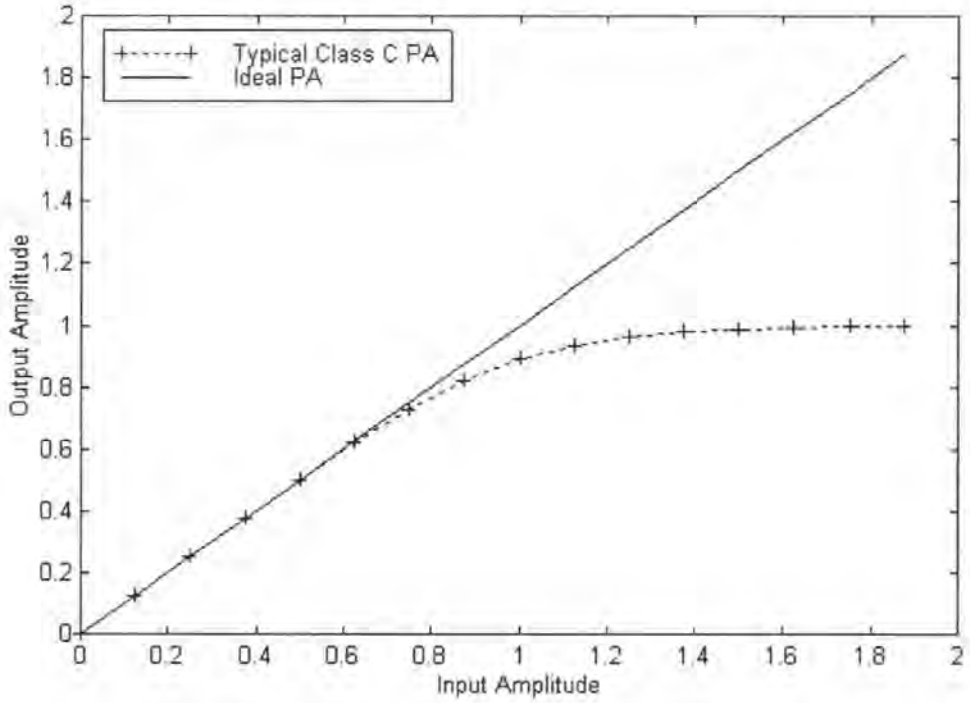


Figure 3.13: Typical Class C PA AM-AM Characteristic

In addition to having non-linear input-output amplitude characteristics, these PAs also having non-linear input amplitude-to-output phase characteristics (not shown here). To avoid generating signal distortion, the operating point of the amplifier has to be carefully chosen to ensure that the maximum amplitude of the input signal never or very rarely enters the non-linear zone. For this particular PA characteristic, it can be seen that the maximum input amplitude for linear amplification must be set to around 0.66. For 'constant envelope' modulation schemes, such as GMSK, whose signal envelope amplitude remain constant, 1.0 would then represent the most efficient PA operating point. This would yield a PA operating efficiency of approximately 100%. For a variable envelope or linear modulation schemes, such as the Spectrally shaped OFDM, Sinc(x) OFDM or single carrier Offset QPSK, the operating point is dictated by the range of possible input signal amplitudes. For single carrier Offset QPSK the ratio of maximum-to-minimum input amplitude is approximately 1.414 (3 dB). For this signal to be linearly amplified by this particular

amplifier, its input signal amplitude would have to be scaled to be between 0.46 and 0.66. This would correspond to an operating point of 0.46, which in turn yields a PA operating efficiency of 46%. In general, amplification of linear modulation schemes is far less efficient than for constant envelope modulation schemes. Constant envelope schemes however suffer from limited bandwidth efficiencies.

3.4.2 Peak to Average Power Ratio (PAPR)

The dynamic range, or peak-to-average power ratio (PAPR), of a modulation scheme's signal envelope defines how efficiently it can be amplified. To determine the PA amplifier requirements for this modulation scheme a number of simulations were carried out to determine approximate values of the modulated signal's PAPR for different number of sub-carriers (N). Common features of all of the 'N' sub-carrier systems simulated are listed below,

- The length of the randomly generated input sequence was fixed to *'2000000*N'* random bits.
- The signal was over-sampled 8 times.
- The number of taps per PolyPhase sub-filter was fixed to 8 taps.
- The roll-off factor of the RRC sub-carrier shaping function was fixed to 0.5.
- None of the coefficients of the RRC sub-carrier shaping PPN filters were windowed.

The results of the PAPR simulations described above are presented in figure 3.14 along with the theoretical PAPR characteristic (equation 3.8 obtained from ref. [38]) of comparable Sinc(x) OFDM systems

$$PAPR\ dB = 10\log_{10} N$$

Equation 3.8: Sinc(x) OFDM Peak-to-Average Power

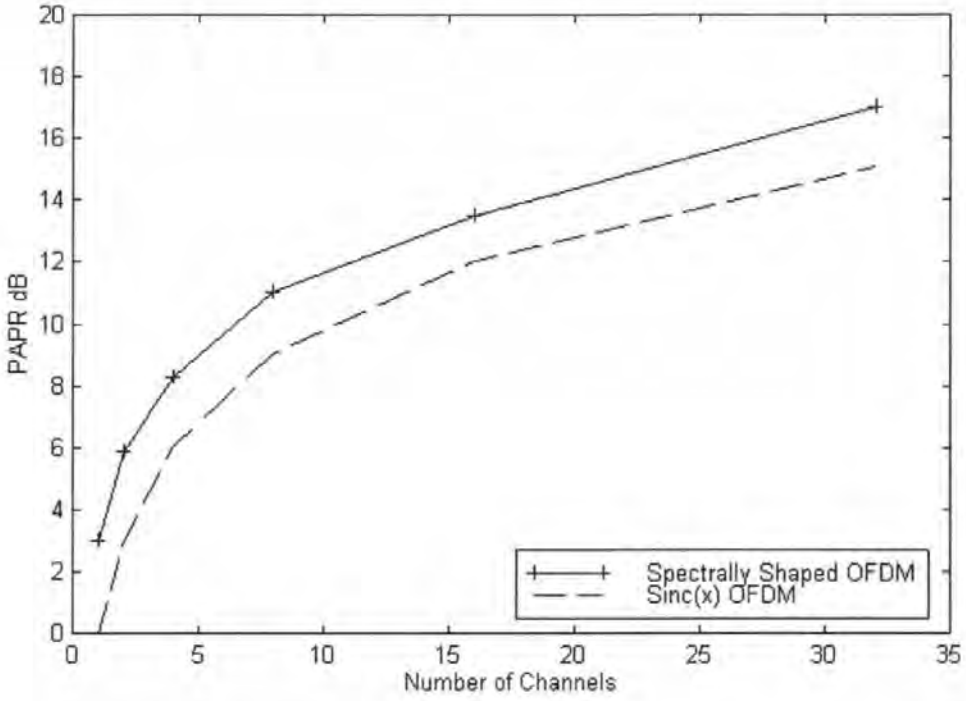


Figure 3.14 : PAPR for Spectrally Shaped and Sinc(x) OFDM

These results show that Spectrally Shaped OFDM has a PAPR characteristic that is between 1-3 dB worse than Sinc(x) OFDM systems with 32 channels or less. The measured PAPR of 3 dB for the single sub-carrier case (corresponding to a conventional simple Offset QPSK modulation scheme) verifies the accuracy of the simulation model for at least the single carrier case. As the number of sub-carriers increases however, the accuracy of the result falls due to the fixed finite length of the random input bit sequence used for all simulations.

3.5 Conclusions

In summary, new results have been provided in this chapter that conclusively prove Spectrally Shaped OFDM has much better spectral and bandwidth efficiencies than comparable single carrier QPSK or Sinc(x) OFDM systems without the need for secondary filtering, dummy end sub-carriers or windowing. However these advantages, which are most apparent with systems employing a low number if sub-carriers (<64), have to be offset against the increased implementation complexity, the 1-3 dB worse PAPR characteristic and the high delay (due to the PPN filter) between when a symbol is modulated and demodulated.

Assuming that the communications performance is similar for systems sharing the same number of sub-carriers, then based on the disadvantages stated above, there would seem little reason for choosing Spectrally Shaped OFDM over Sinc(x) OFDM with windowing and secondary band-pass filtering for commercial wireless applications. If however Spectrally Shaped OFDM can achieve much better communications performance (the subject of chapter 4) in wireless channels with say fewer channels (reducing the problems of PAPR and implementation complexity) than equivalent Sinc(x) OFDM systems, then there could be case for justifying this modulation scheme ahead of others.

Also in this chapter, new results are presented (figures 3.10 and 3.11 of section 3.3.2), which show that windowing of the coefficients of the PPN sub-carrier pulse shaping filter actually degrades the S/D ratio at the demodulator output. Optimum values of the roll-off factor of the RRC sub-carrier pulse-shaping filter function without windowing for maximising the S/D ratio have also been identified, and justified, using these new results.

CHAPTER FOUR

4. Performance in Fast Fading Radio Channels

This chapter outlines the methods and findings of an investigation carried out to determine the effects of different fast fading radio channels on the uncoded Bit Error Rate performance of comparable Sinc(x) and Spectrally Shaped OFDM systems.

4.1 Introduction

This chapter outlines research carried out to determine the effects of fast fading radio channels on the uncoded Bit Error Rate (BER) performance of comparable Spectrally Shaped and Sinc(x) OFDM systems. For the majority of mobile radio channels, the fast fading phenomenon leads to an inescapable degradation in system performance. Section 4.2 provides descriptions of the origins of this phenomenon and the different effects it leads to depending on the radio channel characteristics. Research into Spectrally Shaped OFDM has up to now concentrated on developing efficient methods for modulation and demodulation. Since all these methods, like the method developed in chapter 2, are generally considered too complicated for practical systems, it appears that researchers have been deterred from exploring other aspects of this modulation scheme, such as those investigated in this and the previous chapter. This is in contrast to comparable Sinc(x) OFDM systems, for which a considerable amount of published material exists detailing the research carried out to develop and optimise various system design features and principles, such as the dimensioning of the symbol duration and guard interval, to improve the Bit Error Rate performance in various fading channels. Some of these Sinc(x) OFDM design features and principles are examined in section 4.3 to see if they can be applied to Spectrally Shaped OFDM systems. In Section 4.4 the simulation models developed to explore the strengths and weaknesses of comparable Spectrally Shaped and Sinc(x) OFDM systems in a variety of fast fading channels are described. The results obtained from experiments performed using these models are presented and discussed in section 4.5. Key conclusions regarding the relative strengths and the weakness of the BER performance of both these OFDM systems in different fast fading channels are summarised in section 4.6.

4.2 Mobile Radio Channels

Of all the phenomena that mobile radio channels suffer, multipath propagation presents the greatest design challenge for radio systems engineers. It can lead to severe rapidly varying fluctuations in both the strength and phase of the received signal known as 'fast fading'. Unfortunately in many environments it cannot be avoided, since it is the only practical method of ensuring that radio signal energy arrives at the receiver for demodulation, when there is no clear line-of-sight path between the transmitter and receiver. This section begins with a brief description of the mobile radio channel characteristics that lead to multipath propagation. Highlighted are the quantifiable radio channel characteristics, delay spread and Doppler frequency, that dictate the nature of fast fading experienced by a radio signal. Thereafter, the two categories of fast fading, referred to as time and frequency selective, are described.

4.2.1 Multipath Propagation

Due to ground irregularities and generally low antenna elevations, objects such as trees, buildings, vehicles and even people can obstruct the direct path between the transmitter and receiver. On contact with these physical objects, radio waves will undergo one or several of the following phenomena,

1. Diffraction
2. Scattering
3. Reflection
4. Absorption

Together these phenomena lead to the generation of a continuum of partial copies of the original transmitted wave, referred to as multipath components, which arrive at the receiver diffused across the space, time and frequency domains. This phenomenon, represented in figure 4.1, is known as multipath propagation.

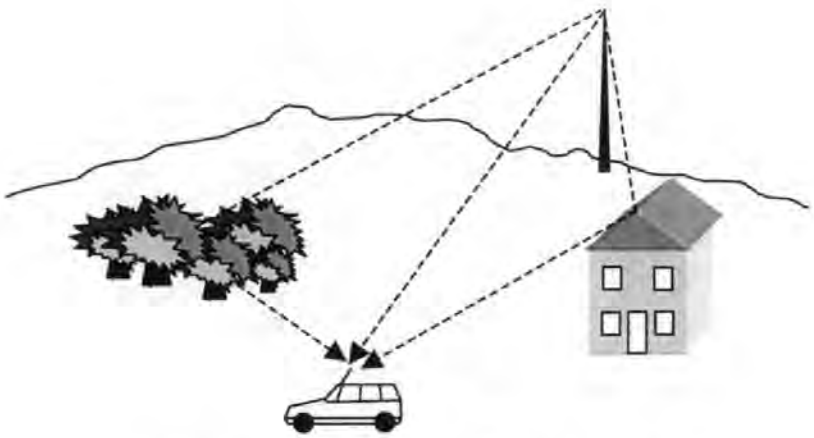


Figure 4.1: Multipath Propagation

The extent of spreading, i.e. the maximum difference in arrival times, of these partial copies in time, is known as the 'delay spread' of the channel. These individual multipath components combine either constructively or destructively at the receiver, to produce a copy of the original transmitted signal whose phase and strength can be severely distorted depending on the nature of the environment and relative motion between the receiver and transmitter.

In environments where the multipath conditions remain static, i.e. where both the mobile station and reflectors are stationary, the receiver can find itself at a location suffering continuous destructive interference known as a fade, which may only be wavelengths away from a location experiencing continuous constructive interference. When the multipath conditions are constantly changing, i.e. when either the mobile station and/or reflectors are moving, then the combination of multipaths leads to the phenomena of fast fading. The severity of this fast fading therefore strongly depends on the velocity, 'v', of the mobile as well as the radio carrier wavelength, 'λ'. These 2 quantities along with the angle 'α' between the incident carrier wave and direction of the mobile's motion define the Doppler frequency for a particular radio link (see equation 4.1).

$$f_d = \frac{v}{\lambda}\cos(\alpha)$$

Equation 4.1: Doppler Frequency

The Doppler frequency and the channel delay spread are shown in the following sections to have a fundamental role defining the nature of the fast fading suffered by a particular radio link. Depending on the relationship between the bandwidth of the transmitted signal and the delay spread of the channel, this fast fading process is said to be either time or frequency selective.

4.2.2 Time Selective Fading

A radio channel is described as being time, or non-frequency, selective if the delay spread of the received multipaths is small compared to the inverse of the transmitted signal bandwidth. In the majority of these scenarios it can usually be assumed that the multipath components arrive at the receiver at approximately the same time having all travelled different but equally long trajectories. At any one instant each of these component waves can be considered as a randomly phase and attenuation distorted copy of the original transmitted wave. The composite sum of these multipath components forms the received signal. The envelope of this signal has an average power 'σ' and amplitude, 'x', which varies with time (rather than frequency) at a rate dictated by the channel Doppler frequency as shown in figure 4.2.

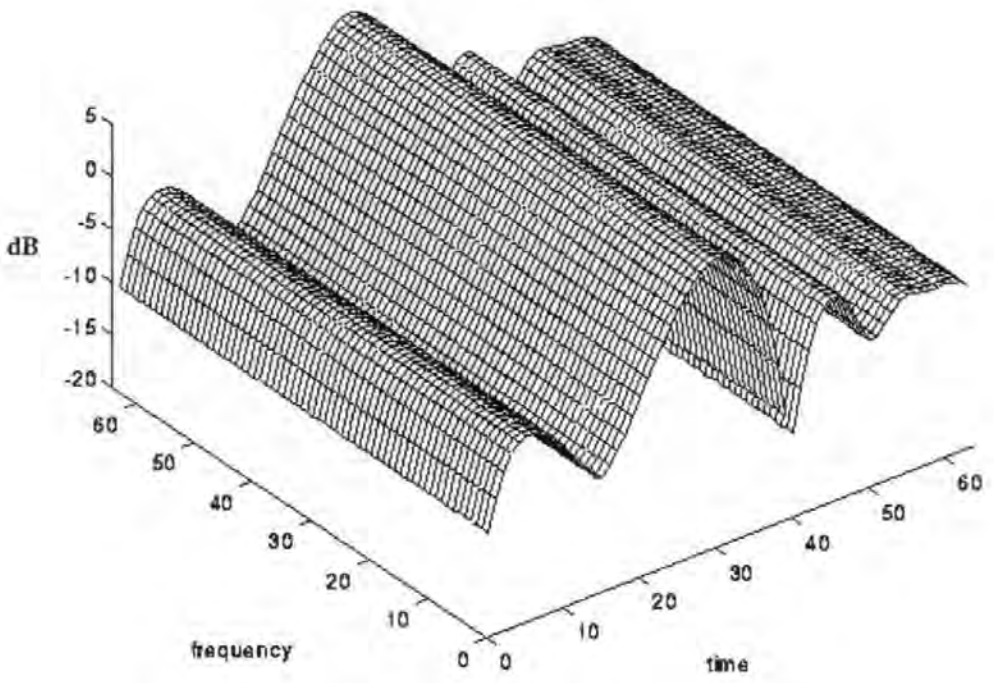


Figure 4.2: Time-Selective Fast Fading Channel

By virtue of the central limit theorem it can be shown (see ref. [2][36][57]), that the distribution of the envelope amplitude, 'x', of the received signal approaches the Rayleigh Density function (equation 4.2) as the number multipaths increases.

$$p_z(x) = \frac{x}{\sigma^2} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} \qquad x \geq 0$$

Equation 4.2: Rayleigh Distribution

The above function assumes that all thes received multipath components experience uniformly distributed fluctuations in phase and attenuation. Analyses of real channels have proven this distribution to be a simple yet accurate model. For channels where additional channel anomalies exist, such as the existence of a single strong dominant path or diffused scatters, a number of alternative distribution functions, such as the Rician and Suzuki, have been devised. In this chapter, research is restricted to the Rayleigh distribution modelled time-selective fading channel, since it is the most common experienced and arguably the most difficult to design for.

Using the Rayleigh distribution model for multipath propagation, in addition to knowledge of the radio link Doppler frequency, 'f_d', it is possible to derive the average fade duration and level crossing rate functions that quantitatively describe the nature of the fading. The average fade duration (equation 4.3 from ref. [69][70]) defines how long the envelope remains below a specified level whilst the level crossing rate (equation 4.4 from ref. [69][70]) defines how often the envelope crosses a specified level. For equations 4.3 and 4.4, the 'specified level' is defined as a ratio of the received signal level to the average signal level, 'ρ'.

$$\bar{t} = \frac{e^{\rho^2} - 1}{\rho f_d \sqrt{2\pi}}$$

Equation 4.3: Average Fade Duration

$$N_r = \sqrt{2\pi} f_d \rho e^{-\rho^2}$$

Equation 4.4: Envelope Crossing Level

From the above equations it can be seen that the maximum Doppler frequency, 'f_d', is the key radio link characteristic that defines both the average fade duration and

the level crossing rate. The larger this frequency, the deeper and narrower the fades become.

With equations 4.3 and 4.4, radio systems designers can begin to judge the severity of fades for a particular channel and begin optimising major modulation scheme parameters and features, such the symbol period, bandwidth and dynamic range of the power control mechanism. For example, for a single carrier modulation scheme to operate without the need of complex equalisation in the time domain, the fading across the symbol waveform in time should be kept as 'flat' as possible. This requires the average fade duration to be much greater than the symbol period.

$$T_s << \frac{1}{f_d}$$

Equation 4.5: Ideal Symbol Period – Doppler Frequency Relationship

The advantage of shortening the symbol period, 'T_s', duration to satisfy equation 4.5, has to be offset against the disadvantage of increasing the signal bandwidth. Increasing this bandwidth can cause the channel to become more frequency selective (see next section), which eventually leads to more severe fading across the bandwidth of the signal and increased inter-symbol interference.

4.2.3 Frequency Selective Fast Fading

A radio channel is described as being frequency selective if the delay spread of the received multipaths is large compared to the inverse of the transmitted signal bandwidth. The fading phenomenon now in addition to varying the signal attenuation and phase characteristics across time, also varies these characteristics across the instantaneous bandwidth of the signal. Figure 4.3 shows an example of a received signal's envelope characteristic in a predominantly frequency selective channel.

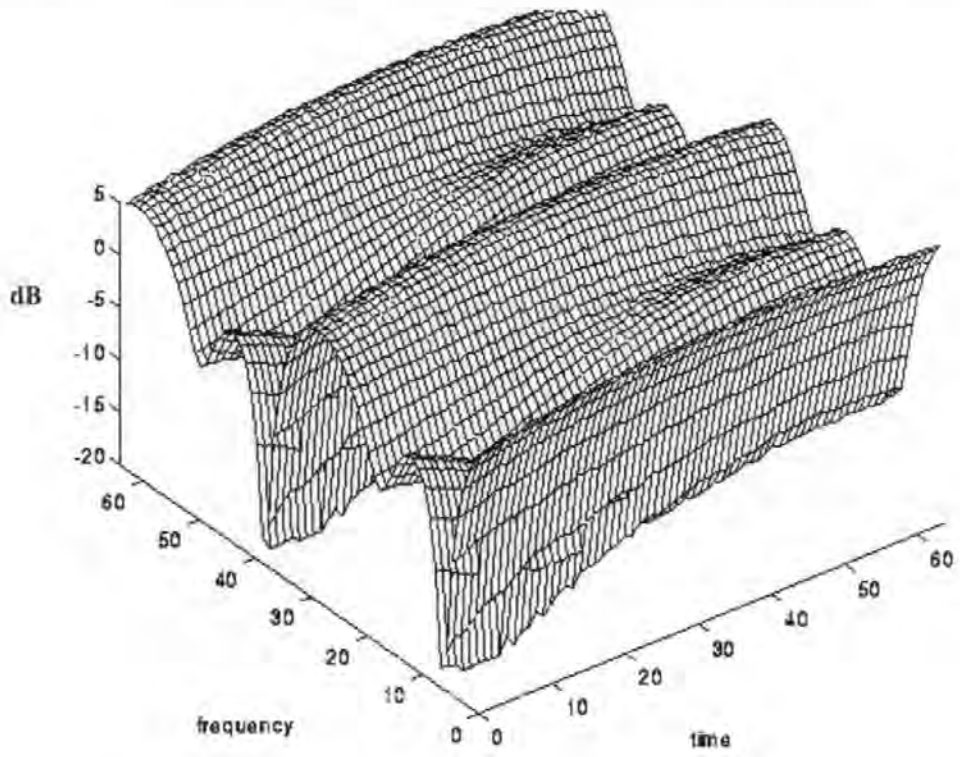


Figure 4.3: Frequency Selective Fast Fading Channel

The coherence bandwidth, 'B_c', of a channel is defined as the bandwidth in which either the amplitudes or the phases of two received signals have a high degree of similarity. In general, it is inversely proportional to the maximum channel delay spread, 'Δ_t'. Equation 4.6 (ref. [2]) gives an example of the coherence bandwidth (correlation factor = 0.5) for a theoretical two-path model (figure 4.10) where the received mean envelope powers of both paths are equal.

$$B_c = \frac{1}{4\Delta_t}$$

Equation 4.6: Coherence Bandwidth for simple 2 Path Channel Model

To avoid complicated 2-dimensional frequency and time channel estimation schemes required to cope with frequency selective fading generated ISI, the carrier bandwidth of a modulation scheme should be dimensioned to be much smaller than the coherence bandwidth. This condition is met when the modulation scheme carrier symbol period is much larger than the delay spread of the channel.

$$\Delta_t << T_s$$

Equation 4.7: Ideal Symbol Period – Delay Spread Relationship

Satisfying equation 4.7 ensures that the fading across the bandwidth of an individual carrier is essentially flat, just as equation 4.5 ensures that the time-selective fading is flat across the duration of a carrier's symbol period. Together these conditions (equation 4.8) give ideal upper and lower bounds on the symbol period duration for an individual carrier in terms of the delay spread and Doppler frequency of the channel.

$$\Delta_t << T_s << \frac{1}{f_d}$$

Equation 4.8: Ideal Symbol Period Upper and Lower Bounds

4.3 Sinc(x) OFDM System Design

Sinc(x) OFDM, is a flexible, bandwidth efficient modulation scheme that in recent years has been chosen for a number of commercial wireless mobile applications, including the Digital Broadcast of Terrestrial and Satellite Broadcast Television and Radio and the next generation European HIPERLAN II and IEEE 802.11 Wireless LAN systems. In common with Spectrally Shaped OFDM, it shares the following fundamental properties,

- It is a linear multi-carrier modulation scheme.
- Its sub-carriers overlap orthogonally in frequency.
- Its sub-carrier symbol period is inversely proportional to its sub-carrier bandwidth spacing.
- FFTs are employed in its most efficient digital modulator and demodulator architectures.

Over the past 15 years several techniques have been developed to enhance the performance of Sinc(x) OFDM against time and frequency selective fading. In this section, the most important of these techniques are studied to discover if they can be applied to Spectrally Shaped OFDM. Figure 4.4 shows a generic Sinc(x) OFDM system incorporating the key techniques to be described in this section.

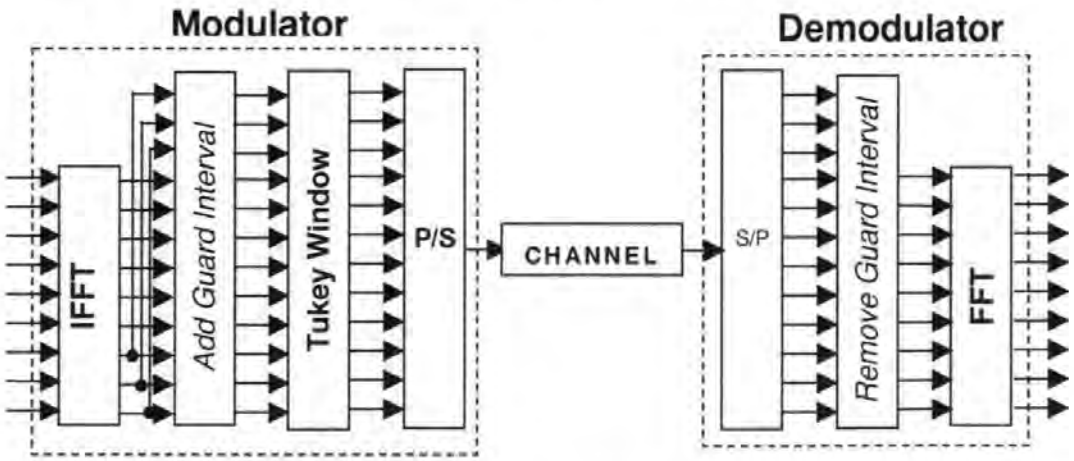


Figure 4.4: Generic Sinc(x) OFDM System Model

4.3.1 Sinc(x) OFDM Symbol Period & Channel Spacing

As with other multi-carrier schemes, Sinc(x) OFDM consists of mapping a single high rate bit stream onto a number, 'N', of parallel symbol streams, each of which is modulated onto an orthogonally overlapping sub-carrier. The sub-carrier symbol period, 'T_s', is therefore 'NB_{sc}' times greater than the original single input stream bit period, where 'B_{sc}' is the number of bits per sub-carrier symbol. To ensure orthogonality of the overlapping sub-carriers for both Spectrally Shaped and Sinc(x) OFDM, the adjacent sub-carrier spacing must be equal to the inverse of the sub-carrier symbol period. Therefore as the sub-carrier symbol period increases, the sub-carrier spacing decreases.

This relationship between the sub-carrier spacing, 'B_{sc}', and symbol period, 'T_s', for an OFDM system restricted to occupying a fixed bandwidth, ensures that as the number of sub-carriers is increased, the bandwidth of individual sub-carriers becomes narrower whilst their symbol period increases. Whilst this lengthening of the OFDM sub-carrier symbol period increases the system's performance against frequency selective fading it only does so by narrowing the sub-carrier bandwidth which conversely makes it less resilient to time-selective fading.

Therefore when dimensioning either the sub-carrier symbol period or bandwidth for a conventional OFDM system, a careful compromise has to be made between the desired system resilience to frequency and time selective fading. In 1989, R. Halbert and B. Floch published research, reference [26], outlining their optimisation of key OFDM parameters for future Digital Terrestrial Broadcasting systems. From the results of their study, they proposed the following empirical rule (equation 4.9) for defining the upper and lower bounds on the sub-carrier symbol period, 'T_s', given a channel with a maximum Doppler frequency of 'f_d' and delay spread of 'Δ_t'.

$$10\ \Delta_t < T_s < \frac{0.02}{f_d}$$

Equation 4.9: Sinc(x) OFDM Symbol Period Boundaries

Independent studies carried out by other researchers, including E. Casas [11], L. Thibault [72] and D. Harvatin [36], generally support R. Halbert's and B. Floch's conclusions.

Given the range of 'acceptable' values for 'T_s' stated in equation 4.9, a designer will generally opt for as short as possible symbol (i.e. as few as possible sub-carriers) since this helps reduce,

- Both the modulator and demodulator FFT implementation complexity.
- The sensitivity of the systems to offsets between modulator and demodulator carrier frequencies.
- The Peak-to-Average Power Problem (see chapter 3).

In summary, the 'useful' symbol period for conventional Sinc(x) OFDM systems is inversely proportional to sub-carrier spacing. Therefore defining either one of these quantities automatically defines the other. To minimise the required equalisation to a simple single complex multiplier per sub-carrier symbol period, these quantities are dimensioned so that the fading processes that occur across the area of time and frequency space occupied by a symbol are as flat as possible.

This principle behind the dimensioning of the Sinc(x) OFDM symbol period and hence sub-carrier bandwidth separation, should be equally applicable to Spectrally Shaped OFDM since they share the same inverse relationship between their sub-carrier spacing and symbol period. However the limits on the optimum symbol period, stated by equation 4.9, will vary to reflect the difference in ability of the modulation schemes to handle equivalent frequency and time selective fast fading channel conditions.

4.3.2 Sinc(x) OFDM Guard Interval

In frequency selective channels with high delay spread, multipath can lead to severe Inter-Symbol Interference (ISI). Increasing indefinitely the sub-carrier symbol period helps reduce the proportion of a symbol corrupted by ISI, but only at the expense of reducing the system's robustness against time selective fading. To combat ISI without increasing the 'useful' symbol period, a 'Guard Interval' is often inserted between successive OFDM symbols at the transmitter as illustrated by figures 4.4 and 4.5. This symbol appendage simply consists of a copy of the last fraction (typically the last 20%) of the original FFT generated symbol's samples. Provided the resultant guard interval period is longer than the delay spread of the channel, all the ISI generated 'tails' from the previous symbol will be absorbed by this section and therefore not affect the 'useful' symbol energy once it is discarded prior to demodulation.

The guard interval increases an OFDM system's tolerance to ISI without impairing the orthogonality of its sub-carriers. This is possible because the waveforms of successive symbols on a particular sub-carrier do not overlap in time (unlike those for Spectrally Shaped OFDM). In addition, this design feature also provides a useful source of coarse timing synchronisation when there are enough sub-carriers (see ref. [84][85]). The only drawbacks associated with the guard interval are that (dpending on the method employed to insert the interval) it reduces the overall data throughput and increases the power that the transmitter has to transmit per unit (symbol) of user information.

Conceptually there exist two methods in which the guard interval can be added to a simple OFDM system. The first maintains the overall data throughput by effectively compressing the desired 'useful' symbol period. This method (applied for the Sinc(x) OFDM systems simulated here) increases the bandwidth of sub-carriers and the real-rime processing power (higher sampling rate) required for modulation and demodulation. The second method simply appends the redundant guard interval onto the existing symbol thereby increasing the overall symbol period and reducing the data throughput.

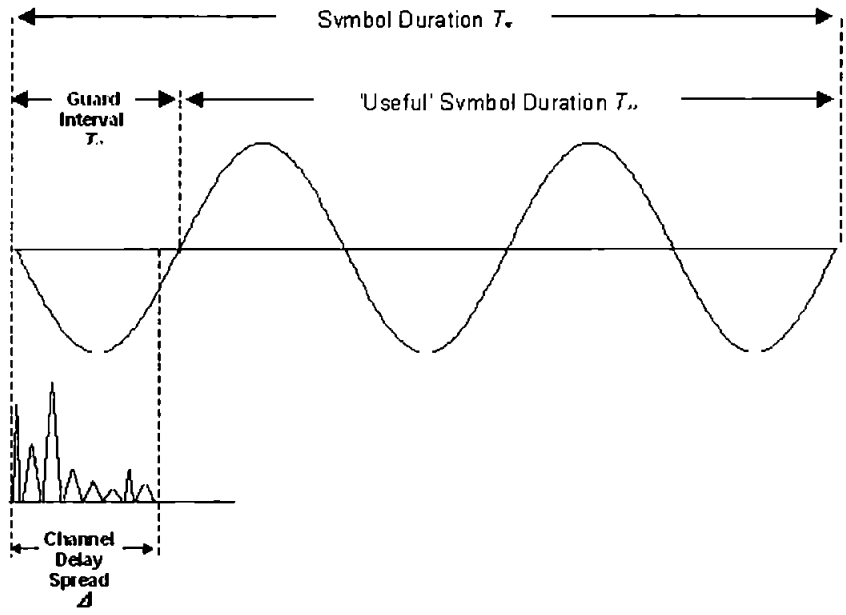


Figure 4.5: Sinc(x) OFDM Symbol Waveform with Guard Interval

Unlike the symbol waveforms for Sinc(x) OFDM, those for Spectrally Shaped OFDM overlap in time. Although this ensures narrower sub-carrier bandwidths than Sinc(x) OFDM systems, it makes the implementation of an orthogonal guard interval impossible. Alternative more complicated techniques that could be applied to Spectrally Shaped OFDM systems to combat frequency selective fading, include enhanced channel estimation and equalisation schemes.

4.3.3 Sinc(x) OFDM Channel Estimation

Fast fading can lead to phase rotations and attenuation of the transmitted symbols that are large enough for them to be incorrectly identified by the receiver. To combat this phenomenon radio systems include channel estimation schemes. These schemes fall into one of two categories, referred to as coherent or non-coherent. All channel estimation schemes perform better the flatter the fading becomes across the area of time and frequency occupied by a symbol.

Assuming identical symbol period to sub-carrier spacing relationships, Sinc(x) and Spectrally Shaped OFDM systems divide up the available time-frequency grid in the same way. Therefore channel estimation schemes designed for one system should be equally applicable to the other. However since the symbol waveforms for the two types of OFDM differ greatly in both the frequency and time domains, the effectiveness of the same channel estimation scheme will vary between the two systems.

4.3.3.1 Coherent Channel Estimation

Coherent channel estimation schemes require special 'pilot' symbols to be transmitted amongst the existing data symbols, whose positions on the time-frequency grid are already known by the receiver. The algorithm uses these symbols, to explicitly estimate the phase (and attenuation if necessary) distortion caused by the channel at particular symbol locations in the time-frequency grid. When optimising the performance of a Sinc(x) OFDM coherent channel estimation scheme the following factors have been identified by several researchers (ref. [64],[68][75]) as being critical.

- The type of channel fading.
- The number and power of pilot symbols relative to data symbols.
- The distribution of pilot symbols across the time-frequency grid.

The interpolation algorithm applied to derive channel estimates for the data symbols located between the pilot symbols.

The time-frequency grid drawn in figure 4.6, shows the pilot symbol locations for a simple frequency domain biased coherent channel estimation scheme simulated in section 4.5. For this particular scheme, the 'N' adjacent sub-carriers of an arbitrary OFDM system are split into 'L' sub-groups or 'pilot blocks' each containing 'N/L' symbols in the frequency domain. Out of these 'N/L' symbols, the 2 outer most symbols are used as dedicated QPSK pilots. The remaining 'N/L – 2' symbols per pilot block are used to carry user data. The demodulator performs linear interpolation between the pilot symbols to obtain channel estimates for the data symbols within that particular block as shown in figure 4.7.

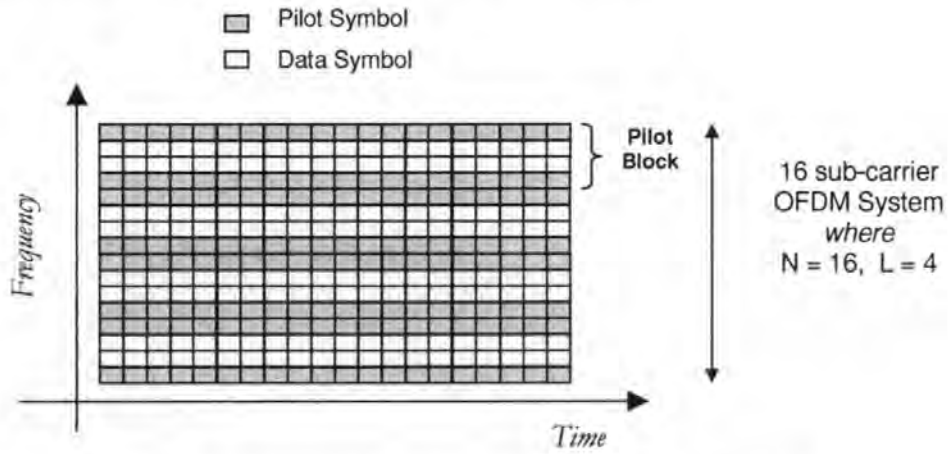


Figure 4.6: Coherent Channel Estimation Scheme Time-Frequency Grid

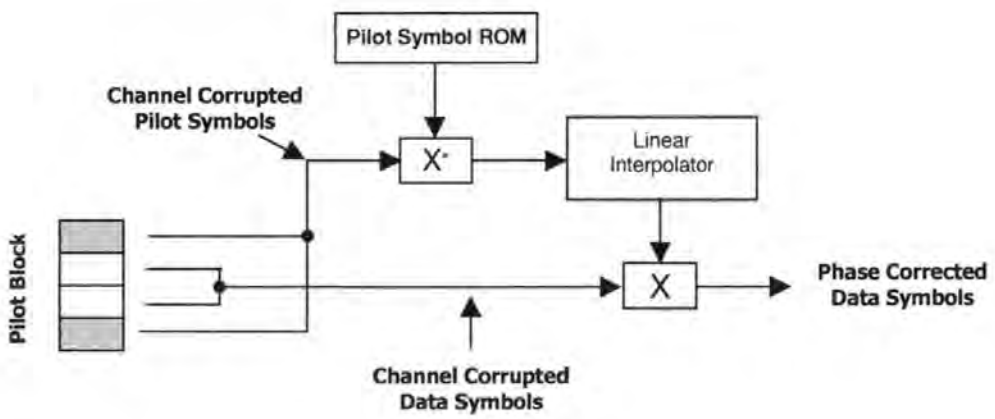


Figure 4.7: OFDM Demodulator for Coherent Channel Estimation

The narrowest fade bandwidth that this particular scheme can handle is approximately equal to the bandwidth occupied by 2 adjacent pilot blocks or '2N/L' adjacent sub-carriers. When applied to an OFDM system with a fixed number, 'N', of sub-carriers, the performance of this channel estimation scheme against frequency selective fading increases as the ratio of the pilot to data symbols increases. The disadvantage associated with increasing the pilot-to-data symbol ratio is that it reduces the overall data throughput.

4.3.3.2 Non-Coherent Channel Estimation

Non-coherent schemes are similar to coherent schemes in the sense that they both require pilot or reference symbols to be transmitted. However unlike coherent schemes, no interpolation operation is required and only one 'reference' symbol needs to be transmitted per pilot block. Figure 4.8 shows the time-frequency grid for a simple frequency domain biased non-coherent channel estimation scheme known as Frequency Domain Differential Phase Shift Key (FD-DPSK).

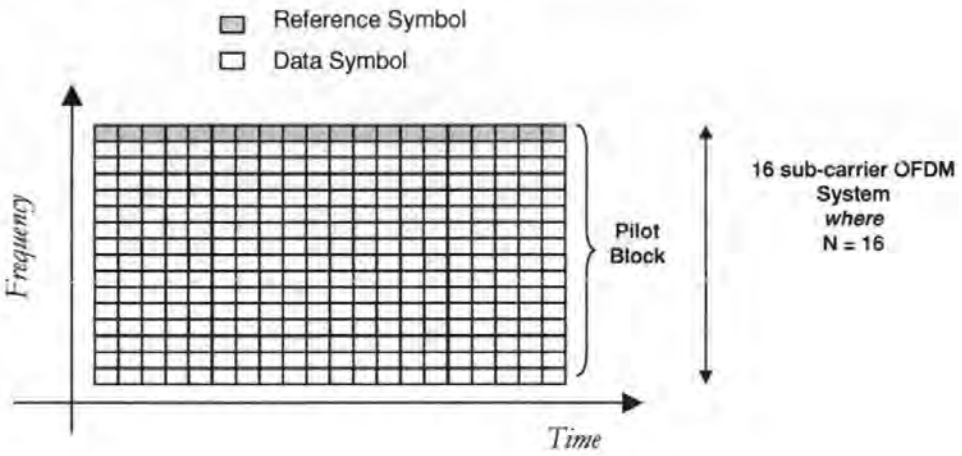


Figure 4.8: Non-Coherent Channel Estimation Scheme Time-Frequency Grid

The data symbols for non-coherent channel estimation schemes are encoded using a rule, that allows the receiver to demodulate and decode symbols whilst simultaneously compensating for fading, without any explicit calculation of values for the extent of the channel fading having to be made. For the FD-DPSK non-

coherent channel estimation scheme represented in figure 4.8, a typical encoding rule would be that given by equation 4.10. Here 'n' refers to the time index (effectively one pilot block or column of the grid in figure 4.8) and 'k' refers to the sub-carrier (effectively one row of the grid in figure 4.8).

$$z_k(n) = y_k(n)\, y^*_{k-1}(n)$$

Equation 4.10: Non-Coherent Channel Estimation Encoding Rule

Non-coherent channel estimation schemes are more efficient in terms of data throughput and are invariably simpler to implement than coherent schemes, particularly when the symbol constellation applied is not multi-level (e.g. QPSK, 8-PSK etc). Although these schemes generally out-perform coherent schemes in rapidly changing channels, in stationary channels they can suffer up to a 3 dB degradation (section 4.5, figure 4.11) in performance compared to the ideal coherent schemes.

4.3.4 Sinc(x) OFDM Channel Coding & Interleaving

If the overall bandwidth occupied by any multi-carrier modulation scheme (Sinc(x) or Spectrally Shaped OFDM) is significantly greater than the coherence bandwidth of the channel, then simple interleaving and coding of data bits across all of the available sub-carriers can exploit the natural channel frequency and time selectivity (or diversity), to improve BER performance.

4.3.5 Digital Audio Broadcasting Sinc(x) OFDM Systems

The Sinc(x) OFDM modulation scheme has in recent years been adopted for a wide variety of commercial digital wireless communication systems. One such system, is Digital Audio Broadcast (DAB). The OFDM based technology for this application was developed and standardised by a consortium of European companies for the broadcast of future digital radio and data services. In this sub-section, features of the DAB OFDM air-interface (ref. [23]) are used to illustrate how the design principles outlined in section 4.3, can be applied to create a highly flexible and robust air interface.

Due to the variety of deployment scenarios it has been designed to handle, ranging from terrestrial to satellite channels situated almost anywhere between 50 MHz and 3 GHz, there are four 'modes' of Sinc(x) OFDM air-interface defined. The key parameters for each of these modes, listed in table 4.1, have been chosen to maximise the system performance in a particular deployment scenario whilst minimising overall system complexity. All four modes can be implemented using essentially the same hardware but clocked at different rates.

DAB Sinc(x) OFDM Transmission Mode	*Number of Carriers*	*Carrier Spacing kHz*	*Symbol Duration μs*	*Guard Interval μs*
I	*1536*	*1*	*1246*	*246*
II	*384*	*4*	*312*	*62*
III	*192*	*8*	*156*	*32*
IV	*768*	*2*	*623*	*123*

Table 4.1: Description of DAB Mode Parameters

The parameters for mode II are optimised for terrestrial broadcast channel scenarios where the expected delay spread can range from 500 ns (indoor) to 15 μs (hilly countryside) and the Doppler frequency can range from 0 Hz (stationary) up to 18.5 Hz (200 km/h @ 100 MHz). For example consider the 250 μs value of the 'useful' OFDM symbol period (figure 4.5), this satisfies the empirical rule, summarised by equation 4.9, that defines upper and lower limits of the Sinc(x) OFDM symbol period for a given set of simplified channel conditions,

$$150\mu s < 250\mu s < 1080\mu s$$

Equation 4.11: DAB Sinc(x) OFDM Symbol Duration

The length of the guard interval for all these modes is equal to exactly 25% of original symbol period. For mode II this gives a guard interval length of 62 μs. This figure is large enough to eliminate not only all the 'natural' ISI created by the worst rural channels (up to 30 μs) but also the artificial ISI created from the deployment of so called Single Frequency Networks. In common with the other three modes, mode II employs frequency domain differential QPSK (FD-DPSK) and is designed to occupy a minimum bandwidth of 1.536 MHz (this excludes guard bands). The choice of FD-DPSK ensures good channel estimation performance in the majority of rapidly varying channels that can be expected. The bandwidth of 1.536 MHz is sufficiently large compared to the typical channel coherence bandwidth of ≈100 kHz, for the natural diversity, or selectivity, of the channel fading to be exploited using straightforward coding and interleaving of symbols across time and frequency.

4.4 Simulation Models & Strategy

In this section the simulation models developed to investigate the robustness of comparable Spectrally Shaped and Sinc(x) OFDM systems against channel phenomena, such as time and frequency selective fading, are described. These models are also used, to compare the effectiveness of simple coherent and non-coherent channel estimation schemes (see section 4.3.3) for the two types of OFDM.

4.4.1 Introduction

The Bit-Error-Rate (BER) versus Signal-to-Noise Ratio (SNR) characteristic of a modulation scheme provides a quantifiable measure of its robustness to a particular channel scenario given a certain set of system features (e.g. channel estimation, coding and interleaving schemes). When compared with similar characteristics obtained for radio channels with different attributes, such as Doppler frequencies, the trends and limits (including error floors) of system performance with respect to that channel attribute can be determined. Similarly, the BER-SNR characteristic can be used to assess the effectiveness of different system features such as a guard interval or channel estimation scheme. Modelling the effects of frequency selective fading on system performance analytically is widely considered too complex.

> *"The use of simulation as a design tool, when doubly spread channel performance cannot be modelled mathematically, is demonstrated."*
>
> D. T. Harvatin (ref. [30])

Hence for this investigation, a C-coded simulation model (Appendix C) of both a Sinc(x) and Spectrally Shaped OFDM system, is used to derive the BER-SNR performance characteristic for different channels and system features.

4.4.2 Description of Baseband Equivalent Simulation System (BESS)

Figure 4.9 shows a block diagram of the basic C-coded model used for the analysis and simulation of both the Spectrally Shaped and Sinc(x) OFDM systems. The c-code for this simulation engine is listed in Appendix C. Expanded block diagrams of the modulator and demodulator blocks for the Spectrally Shaped OFDM system can be found in chapter 2 and for the Sinc(x) OFDM system in figure 4 of section 3 of this chapter.

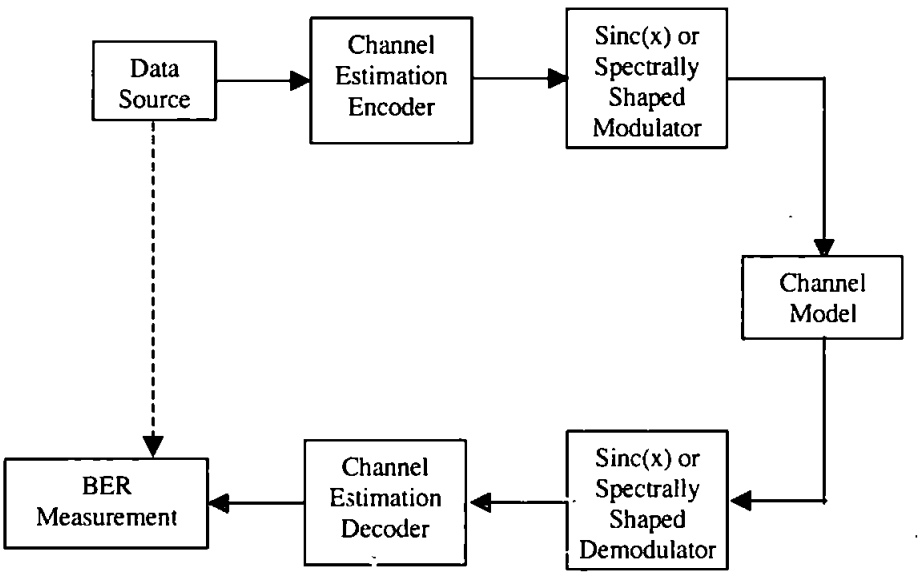


Figure 4.9: Block Diagram of Generic Simulation System

The data source generates a pseudo-random bipolar sequence. At any one time, the values of the data bits generated, are independent of any previous bits and are equally likely to be either '+1' or '-1'. Successive pairs of bits are then used to form complex QPSK symbols, which are collected together to create 'N' long complex vectors, where 'N' is equal to the number of OFDM sub-carriers. These complex vectors are then passed through the Channel Estimation Encoder Block that either inserts pilot symbols or differentially encodes the existing data symbols using the

techniques described earlier. Next the data symbols are modulated onto either Sinc(x) or Spectrally Shaped OFDM sub-carriers.

The resultant signal is then fed into a Mobile Channel block that simulates either a simple AWGN channel, a time-selective fading channel using a single Jakes Rayleigh fading generator or a frequency selective fading channel using a simple 2-path model as shown in figure 4.10.

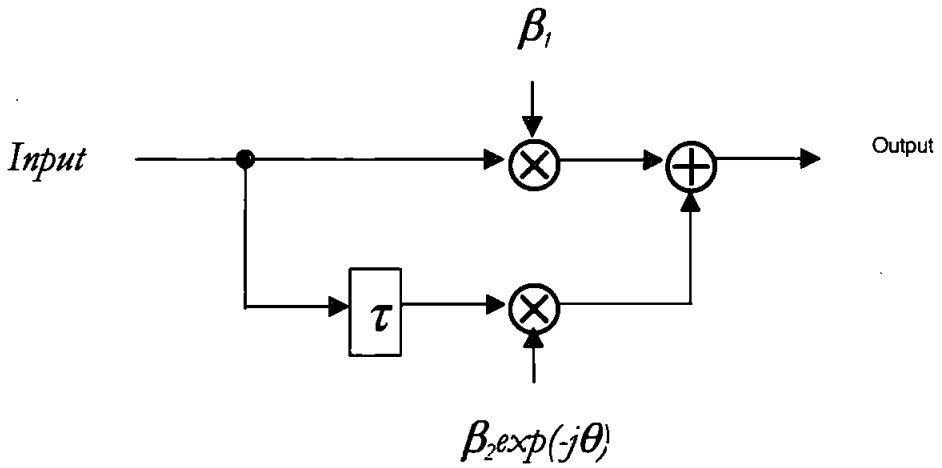


Figure 4.10: Simple 2-Path Frequency Selective Channel Model

For the frequency selective fading channel model, the input signal is separated into two components, one weighted by 'β_1' and the other delayed by 'τ' (the effective delay spread of the channel) and weighted by '$\beta_2 \exp(-j\theta)$'. Both 'β_1' and 'β_2' are Rayleigh distributed random variables. The relative phase 'θ' between the two rays is uniformly distributed between 0 and 2π. The outputs from the two rays are summed at the receiver input.

Equations 4.12 and 4.13 give the impulse and frequency response respectively for the 2-path channel model assuming a fixed set of values for the 'β_1', 'β_2', 'θ' and 'τ'

$$h(t) = \beta_1 \delta(t) + \beta_2 \exp(-j\theta)\delta(t-\tau)$$

Equation 4.12: 2-Path Channel Model Impulse Response

$$H(f) = \beta_1 + \beta_2 \exp(-j\theta)\exp(-j2\pi f\tau)$$

Equation 4.13: 2-Path Channel Model Frequency Response

The parameter, 'τ', determines the spacing between frequency nulls (coherence bandwidth) and the breadth of any one null; 'θ' determines the frequency of the first null; and 'β_1' and 'β_2' determine the depth of the null. To investigate how the two OFDM modulation schemes performed in frequency selective channels with differing coherence bandwidths, the simplest form of this 2-path channel model is applied, where '$\beta_1 = \beta_2 = 0.5$', '$\theta = 0$' and 'τ' is adjusted between '0' and 'T_s' (the OFDM symbol period).

For both the time and frequency selective fading channels, additive white noise is added to the signal after it has been subjected to fading and before it enters the receiver. The variance (power) of this noise signal adjusted to provide the specified Signal-to-Noise Ratio (SNR) given that the signal power, in the absence of multipath and amplification, is calibrated to be 1.0 at the receiver entry point.

After passing through the channel model and being subjected to noise the OFDM signal passes straight into the demodulation block. This block simulates the digital baseband processing necessary to demodulate either the Spectrally Shaped or Sinc(x) OFDM modulated input signal, as described in chapter 2 for a Spectrally Shaped OFDM system and as illustrated in figure 4.4 for a Sinc(x) OFDM system. In addition to perfect synchronisation in frequency and time (fixed to the first multipath) being assumed no attempt is made to simulate the non-linear RF filtering, down-conversion and amplification effects that would occur in a practical implementation. The demodulated signal is then passed through the channel estimation block, which attempts to correct the 'random' symbol phase rotations caused by the fading channel, using the methods described in section 4.3.3. Finally the phase corrected demodulated QPSK data symbols are sampled and compared with those actually transmitted to calculate the BER for a given SNR.

4.5 Simulation Results & Discussions

This section outlines the experiments performed using the simulation model described in section 4.4, to analyse the behaviour of the comparable Sinc(x) and Spectrally Shaped OFDM modulation schemes in Guassian, time-selective and frequency-selective channels. For all these experiments the E_b/N_0 values quoted do not take into account pilot symbol redunancy. Also the roll-off factor for all the Spectrally Shaped OFDM systems simulated is fixed to 0.25. The results obtained from these experiments are formally presented and discussed. From these discussions, theories and conclusions are drawn regarding the relative strengths and weaknesses of the two modulation schemes.

4.5.1 AWGN Channels

4.5.1.1 Experiments

To investigate and compare the effects of the AWGN channel on the BER performance of a variety of Conventional and Spectrally Shaped OFDM systems, a number of experiments were carried out. Common features of all the simulation models constructed for these experiments are listed below.

- 128 QPSK Carriers.
- 1x Over-sampling.
- Perfect frequency and timing synchronisation assumed.
- E_b/N_0 measured in 1 dB steps from 0 dB up to 8 dB.
- For a given value of E_b/N_0 the simulation cycle is repeated a sufficient number of times to obtain a constant number (1000) of bits in error.

Different features (such as the type of channel estimation) of the Sinc(x) and Spectrally Shaped OFDM simulation models constructed for these experiments are listed in tables 4.2 and 4.3. The second column for both of these tables states the type of channel estimation scheme applied to a particular experiment set-up. Where the coherent channel estimation scheme (originally described in section 4.3) has been applied, the pilot-to-data symbol ratio is also quoted. For the Sinc(x) OFDM

system experiments additional columns are included that state whether or not a guard interval and Tukey Window pulse shaping are also applied.

Experiment Name	Type of Channel Estimation	Guard Interval	Tukey Window Pulse Shaping
Simple	None	None	None
Simple Non-Coherent	Non-coherent FD-DPSK	None	None
Shaped Non-Coherent	Non-coherent FD-DPSK	25%	5%
Shaped Low Coherent	Coherent 1:32	25%	5%
Shaped Medium Coherent	Coherent 1:16	25%	5%
Shaped High Coherent	Coherent 1:8	25%	5%

Table 4.2: Sinc(x) OFDM AWGN Experiments

Experiment Name	Type of Channel Estimation
Simple	None
Non-Coherent	Non-coherent FD-DPSK
Low Coherent	Coherent 1:32
Medium Coherent	Coherent 1:16
High Coherent	Coherent 1:8

Table 4.3: Spectrally Shaped OFDM AWGN Experiments

4.5.1.2 Results

The results from the experiments outlined in tables 4.2 and 4.3 are presented in figures 4.11 to 4.12 respectively.

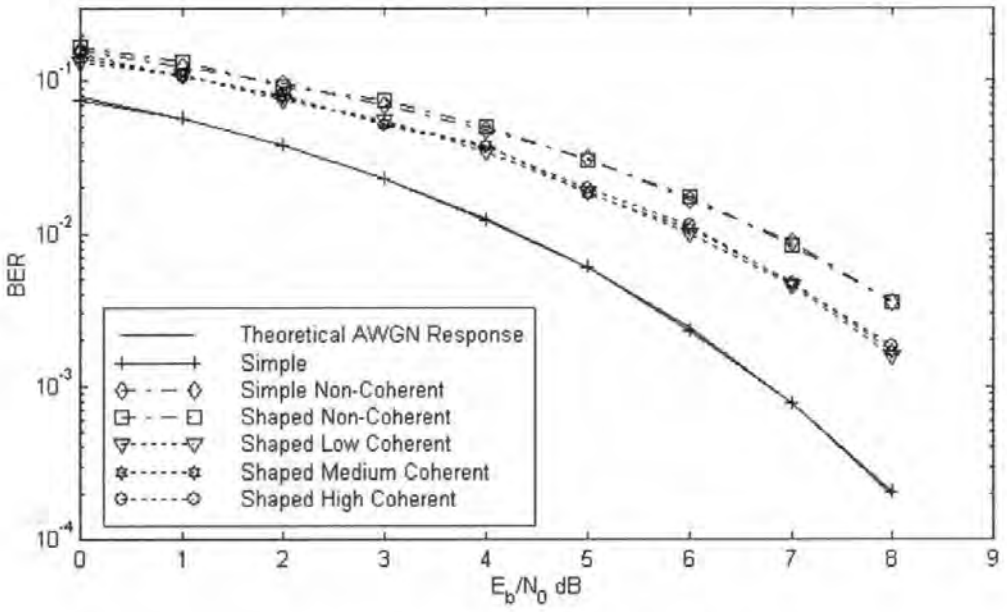


Figure 4.11: Sinc(x) OFDM in AWGN

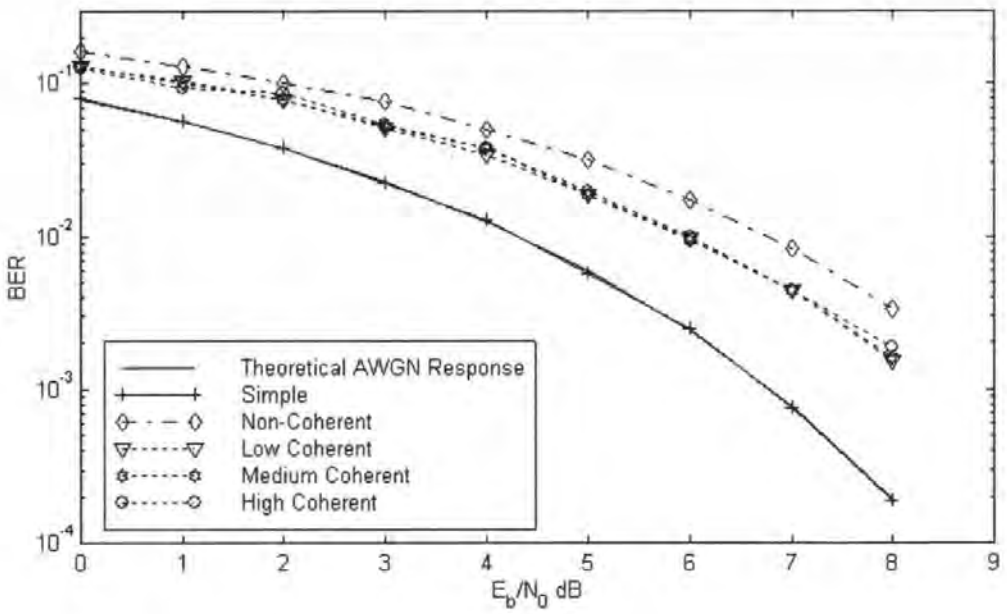


Figure 4.12: Spectrally Shaped OFDM in AWGN

4.5.1.3 Observations

From the experimentally determined BER versus E_b/N_0 characteristics, presented in figures 4.11 and 4.12, across the range E_b/N_0 equals 0 dB to 8 dB a number of observations can be made.

Without any form of channel estimation both Sinc(x) and Spectrally Shaped OFDM produce a BER characteristic that closely matches that predicted by theory.

Coherent detection performs better than non-coherent differential detection. The difference between the particular coherent and non-coherent schemes adopted here is in the range of 0.75 - 1.0 dB.

The performance of the coherent scheme applied here is independent of pilot to data symbol ratio.

The performance of the non-coherent scheme applied here for Sinc(x) OFDM is independent of inclusion of the Guard Interval and Tukey Pulse Shaping.

There is no discernible BER performance advantage between Sinc(x) and Spectrally Shaped OFDM systems with equivalent channel estimation schemes in AWGN channels.

4.5.1.4 Discussions

The close agreement between the results obtained for both variants of OFDM without channel estimation, with those predicted by theory, verify the reliability of the basic simulation model. With equivalent channel estimation schemes these modulation schemes exhibit almost identical BER behaviour in the presence of AWGN assuming perfect time and frequency synchronisation. Therefore on the basis of simply their BER performance in AWGN there is little to choose between the two schemes. However when you consider their relative spectral qualities and implementation complexities, arguments (already stated in previous chapters) can be formulated to support one scheme ahead of the other.

4.5.2 Time Selective Channels

4.5.2.1 Experiments

To investigate and compare the effects of the time-selective or flat fading channel on the BER performance of a variety of Sinc(x) and Spectrally Shaped OFDM systems, a number of experiments were carried out. Common features of all the simulation models constructed for these experiments are listed below.

- 128 QPSK Carriers
- 1x Over-Sampling
- The Symbol Period, t_s, for all experiments is kept constant at 1.0.
- Perfect frequency and timing synchronisation assumed.
- For a given experimental set-up the BER performance is measured for values of E_b/N_0 ranging from 0 dB up to 40 dB, in steps of 4 or 2 dB.
- For a given experimental set-up, the BER performance is repeatedly measured across the range of E_b/N_0 defined above, for 4 different values of the Doppler frequency ($f_d \in \{0.25,\ 0.125,\ 0.0625,\ 0.03125\}$).
- To obtain BER measurements with approximately the same margin of error for each value of E_b/N_0, the simulation cycle is repeated a sufficient number of times to obtain a minimum number (100000) of bits in error for each value of E_b/N_0.
- Time-selective fading generated using a conventional Jakes Fading Simulator with 16 oscillators.

Features of the Sinc(x) and Spectrally Shaped OFDM system models that were varied between experiments are described in tables 4 and 5.

Experiment Name	Type of Channel Estimation	Guard Interval	Tukey Window Pulse Shaping
Simple	Ideal	None	None
Simple Non-Coherent	Non-coherent FD-DPSK	None	None
Shaped Non-Coherent	Non-coherent FD-DPSK	25%	5%
Shaped Low Coherent	Coherent 1:32	25%	5%
Shaped Medium Coherent	Coherent 1:16	25%	5%
Shaped High Coherent	Coherent 1:8	25%	5%

Table 4.4: Sinc(x) OFDM Flat Fading Experiments

Experiment Name	Type of Channel Estimation
Simple	None
Non-Coherent	Non-coherent FD-DPSK
Low Coherent	Coherent 1:32
High Coherent	Coherent 1:8

Table 4.5: Spectrally Shaped OFDM Flat Fading Experiments

4.5.2.2 Results

The results to the experiments outlined in tables 4.4 and 4.5 are presented in figures 4.13 to 4.18. Derived from this set of results are figures 4.19 and 4.20, which capture how the BER characteristic error floor (visible in figures 4.13-4.18) varies with respect to Doppler frequency for the experiments listed above.

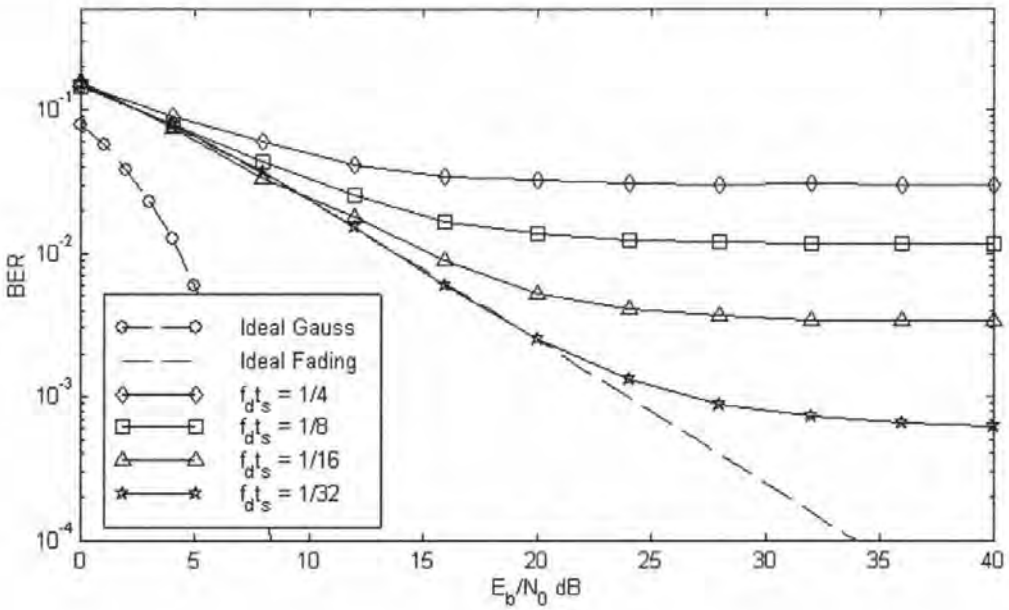


Figure 4.13: Simple Ideal Sinc(x) OFDM

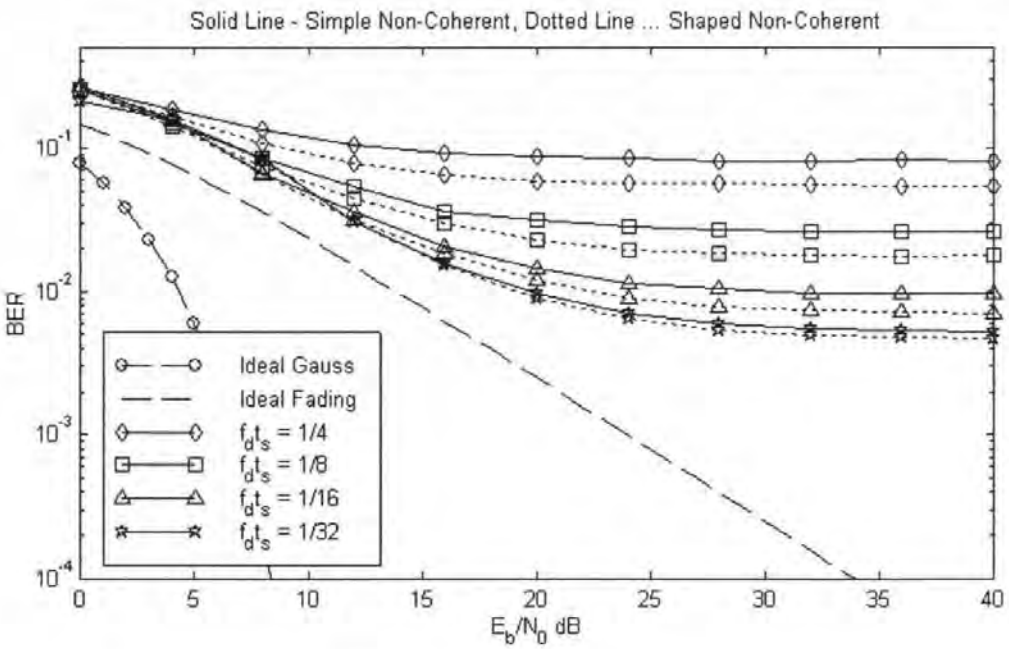


Figure 4.14: Simple & Shaped Non-Coherent Sinc(x) OFDM

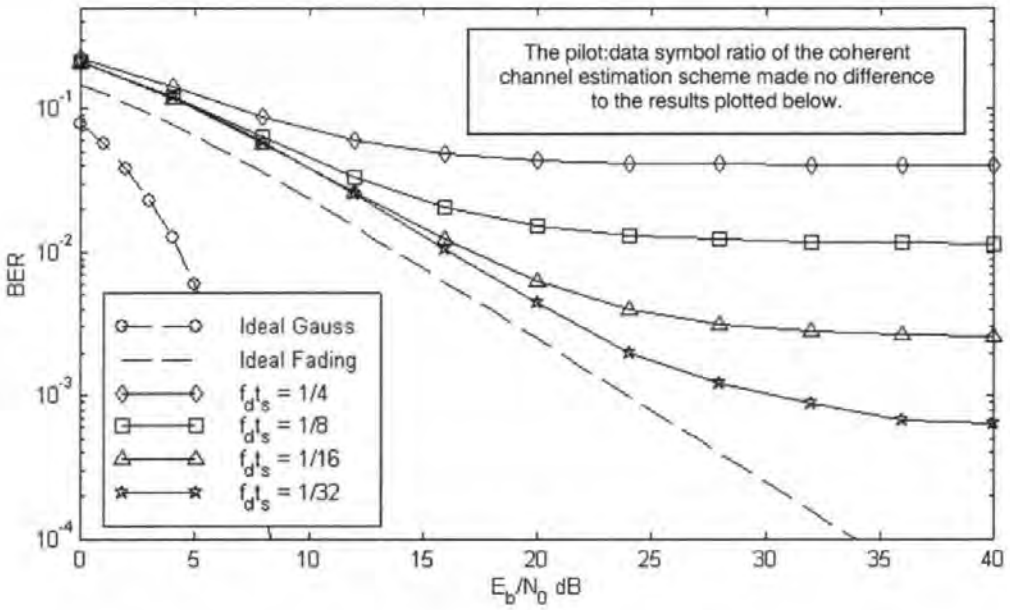


Figure 4.15: Coherent Sinc(x) OFDM

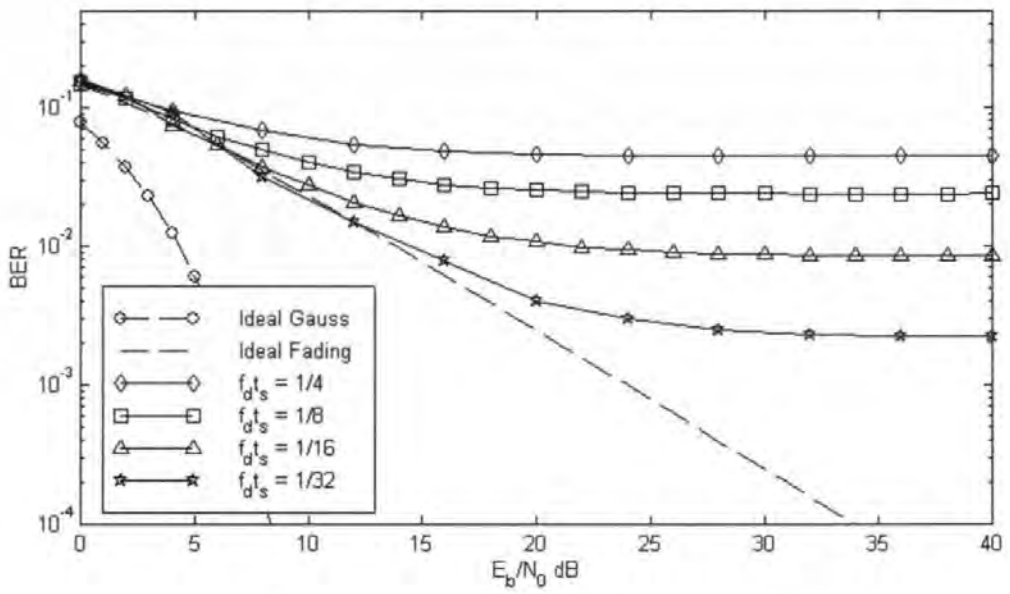


Figure 4.16: Ideal Spectrally Shaped OFDM

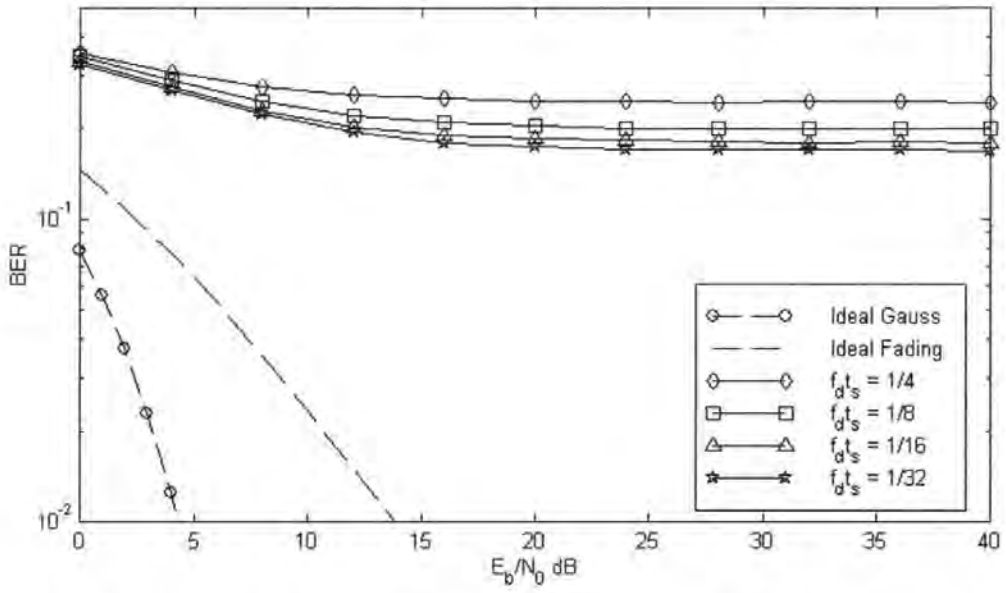


Figure 4.17: Non-Coherent Spectrally Shaped OFDM

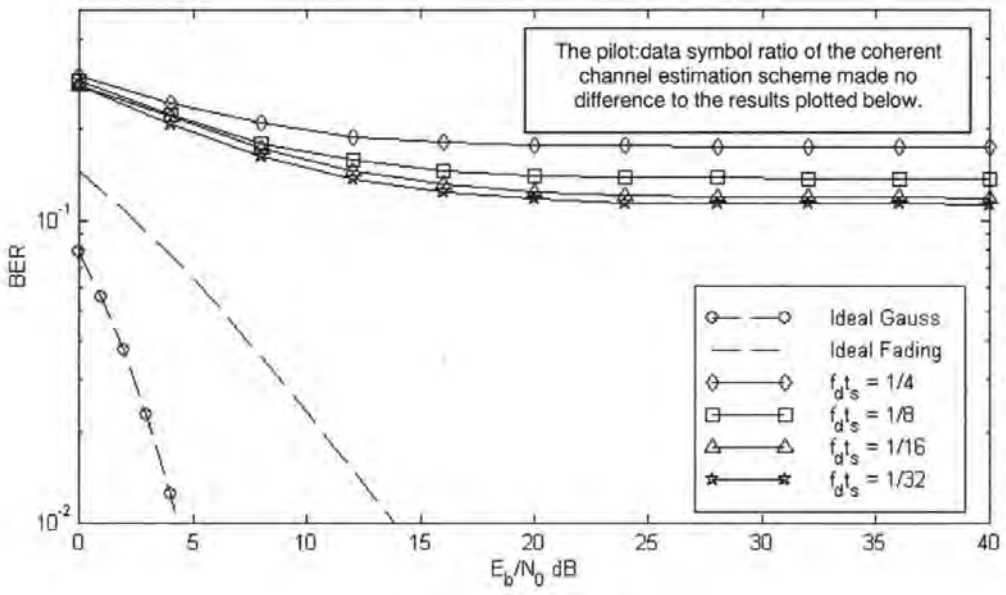


Figure 4.18: Coherent Spectrally Shaped OFDM

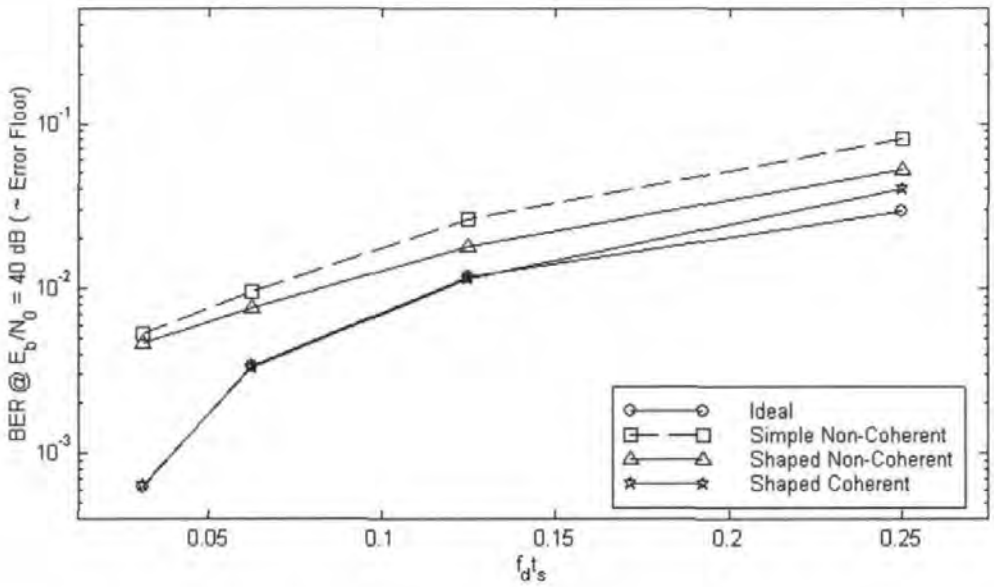


Figure 4.19: Sinc(x) OFDM Error Floor in Flat Fading Channels

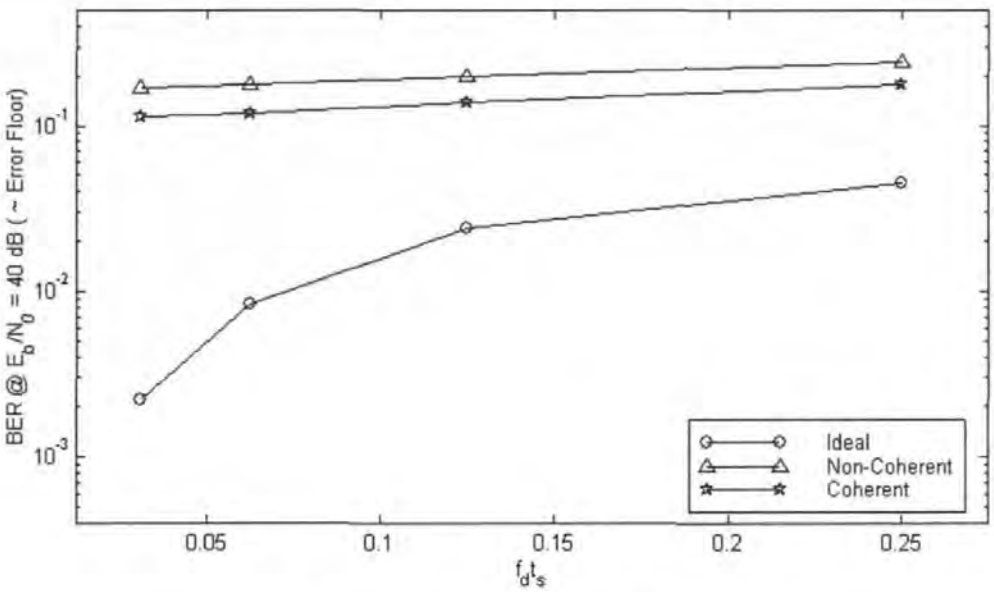


Figure 4.20: Spectrally Shaped OFDM Error Floor in Flat Fading Channels

4.5.2.3 Observations

From the experimentally determined BER versus E_b/N_0 characteristics, presented in figures 4.13 to 4.18, across the range E_b/N_0 equals 0 dB to 40 dB a number of observations can be made.

As the values of E_b/N_0 and f_d decrease, the BER characteristic for both Sinc(x) and Spectrally Shaped OFDM approaches and eventually follows the theoretical flat fading bound assuming ideal channel estimation.

Regardless of the value of f_d or the type of channel estimation applied, as the value of E_b/N_0 increases, the BER characteristics for both types of OFDM, fall until a constant level or 'error floor' is reached. The point at which this 'error floor' first occurs varies depending on the system features and the channel.

The value of the error floor increases, as the channel time-selective fading becomes more severe (i.e. as the Doppler frequency increases).

For both types of OFDM, the coherent channel estimation scheme performs better than the comparable non-coherent channel estimation scheme.

For both types of OFDM, the performance of the coherent channel estimation scheme is independent of the actual pilot to data symbol ratio.

Figure 4.14 shows that the inclusion of a guard interval improves the BER characteristic for the Sinc(x) OFDM system.

With comparable non-ideal frequency domain biased coherent and non-coherent channel estimation schemes, the BER characteristic of the Sinc(x) OFDM system significantly out-performs that for the Spectrally Shaped OFDM system.

4.5.2.4 Discussions

The results presented in figures 4.13 to 4.18 indicate that as the Doppler frequency rises, i.e. as the channel begins to vary in time more rapidly, the ability of the demodulator for the Sinc(x) OFDM system to track and compensate for the fading process, becomes increasingly poorer. Eventually a point is reached where, even with infinite signal to noise ratio and ideal channel estimation, an error floor is reached. Up to now, this time-selective channel phenomena had only been predicted and observed only for Sinc(x) OFDM systems, by researchers such as Harvatin [30] and Tufvesson [75]. Figures 4.16 to 4.18 confirm for the first time Spectrally Shaped OFDM also exhibits similar error floors but at significantly higher levels, even when ideal channel estimation is assumed. This difference in behaviour is probably due to the fact that the symbol time-domain waveform for Spectrally Shaped OFDM extends (and overlaps) over many (8 in this simulation) symbol periods whereas for Sinc(x) OFDM the equivalent symbol waveform is contained within 1 symbol period. Therefore whereas a single symbol period long time-domain fade for Sinc(x) OFDM may only corrupt the waveform of 1 or 2 symbols, for Spectrally Shaped OFDM the same duration of fade would corrupt (to varying degrees) the overlapping waveforms of up to 8 successive symbols. With improved channel estimation schemes this difference may become less significant. However since even when ideal channel estimation is assumed (figures 4.13 and 4.16) there is still a difference, it is the authors' opinion that the BER performance of Spectrally Shaped OFDM in time-selective channels will never exceed that of comparable Sinc(x) OFDM systems.

Figures 4.19 and 4.20 indicate that the coherent channel estimation schemes perform better than the comparable non-coherent scheme applied across channels that vary rapidly and slowly. Researchers agree that in general this should only be true for channels that vary slowly with time or frequency, but figures 4.19 and 4.20 indicate that this is also true for channels that vary rapidly (high Doppler frequency). The most likely reason for this anomaly, is the fact that the channel estimation schemes applied were not optimised to function in time selective channels, hence why the pilot density of the particular coherent scheme simulated made no difference to the measured BER performance.

It was observed in figures 4.15 and 4.18 that the coherent channel estimation pilot-to-data symbol ratio made no difference to the BER characteristic for both Sinc(x) and Spectrally Shaped OFDM. This is due to the time-selective fading being flat across the bandwidth of the modulated signal and the channel estimation scheme performing interpolation between pilot symbols across the frequency domain only. With a coherent channel estimation scheme that interpolated across time, one would expect the quality of the channel estimate to improve as pilot-to-data symbol ratio increased.

In figure 4.14, the BER characteristic can be seen to improve with the inclusion of a guard interval. This is due to the method employed to insert a guard interval (see section 4.3.2) which maintains the overall data throughput by effectively compressing the desired symbol period (at the expense of increasing the sub-carrier bandwidth). Since the desired symbol period (see figure 4.5) is now shorter, it is less severly affected by time-selective fast fading. Note that had the alternative method of inserting the guard interval stated in section 4.3.2 been simulated, there would be no difference in performance.

4.5.3 Frequency Selective Channels

4.5.3.1 Experiments

To investigate and compare the affects of a frequency *and* time selective fading channel on the BER performance of a variety of Sinc(x) and Spectrally Shaped OFDM systems, a number of experiments were carried out. Common features of the simulation models constructed for these experiments are listed below.

- 128 QPSK Carriers
- 1x Over-Sampling
- The Symbol Period, T_s, is kept constant, at 1.0, for all experiments.
- A simple 2-path fading channel model (figure 4.10) is applied to generate the frequency selective fading.
- The receiver is time and frequency synchronised to demodulate the earliest path from the 2-path channel model throughout the duration of an experiment even though the second path will at times be significantly stronger than the first path.
- For a given experimental set-up the BER performance is measured for values of E_b/N_0 ranging from 0 dB up to 40 dB, in steps of 4 dB.
- To obtain BER measurements with approximately the same margin of error for each value of E_b/N_0, the simulation cycle is repeated a sufficient number of times to obtain a minimum number (100000) of bits in error for each value of E_b/N_0.
- For a given experimental set-up, the BER performance is repeatedly measured across the range of E_b/N_0 defined above, for a number of different values of the separation 'τ' between the 2 paths (effectively the channel spread) of the frequency selective channel model (figure 4.10).
- Two independent Jakes fading simulators, each with 16 oscillators, are used to generate Rayleigh fading with a Doppler frequency, 'f_d', of 0.125 on each path of the 2-path channel model.

- The relative phase rotation, 'θ" between the first and second path of the 2-path channel model is kept at zero for all experiments.

- The mean powers, 'β_2' and 'β_2', of the 2 separate paths of the frequency selective fading channel model are both fixed to 0.5 for all experiments.

Features of the Sinc(x) and Spectrally Shaped OFDM system models that were varied between experiments are described in tables 4.6 and 4.7.

Experiment Name	Type of Channel Estimation	Guard Interval	Tukey Window Pulse Shaping
Shaped Non-Coherent	Non-coherent FD-DPSK	25%	5%
Shaped Low Coherent	Coherent 1:32	25%	5%
Shaped High Coherent	Coherent 1:8	25%	5%

Table 4.6: Sinc(x) OFDM System Model Attributes

Experiment Name	Type of Channel Estimation
Non-Coherent	Non-coherent FD-DPSK
Low Coherent	Coherent 1:32
High Coherent	Coherent 1:8

Table 4.7: Spectrally Shaped OFDM System Model Attributes

4.5.3.2 Results

The results of the experiments outlined in tables 4.6 and 4.7 are presented in figures 4.21 to 4.26. These show how the BER performance, for a particular OFDM system set-up, varies with respect to E_b/N_0 in one dimension and the channel delay spread

of the channel (normalised to the 'useful' OFDM symbol duration) in the other dimension.

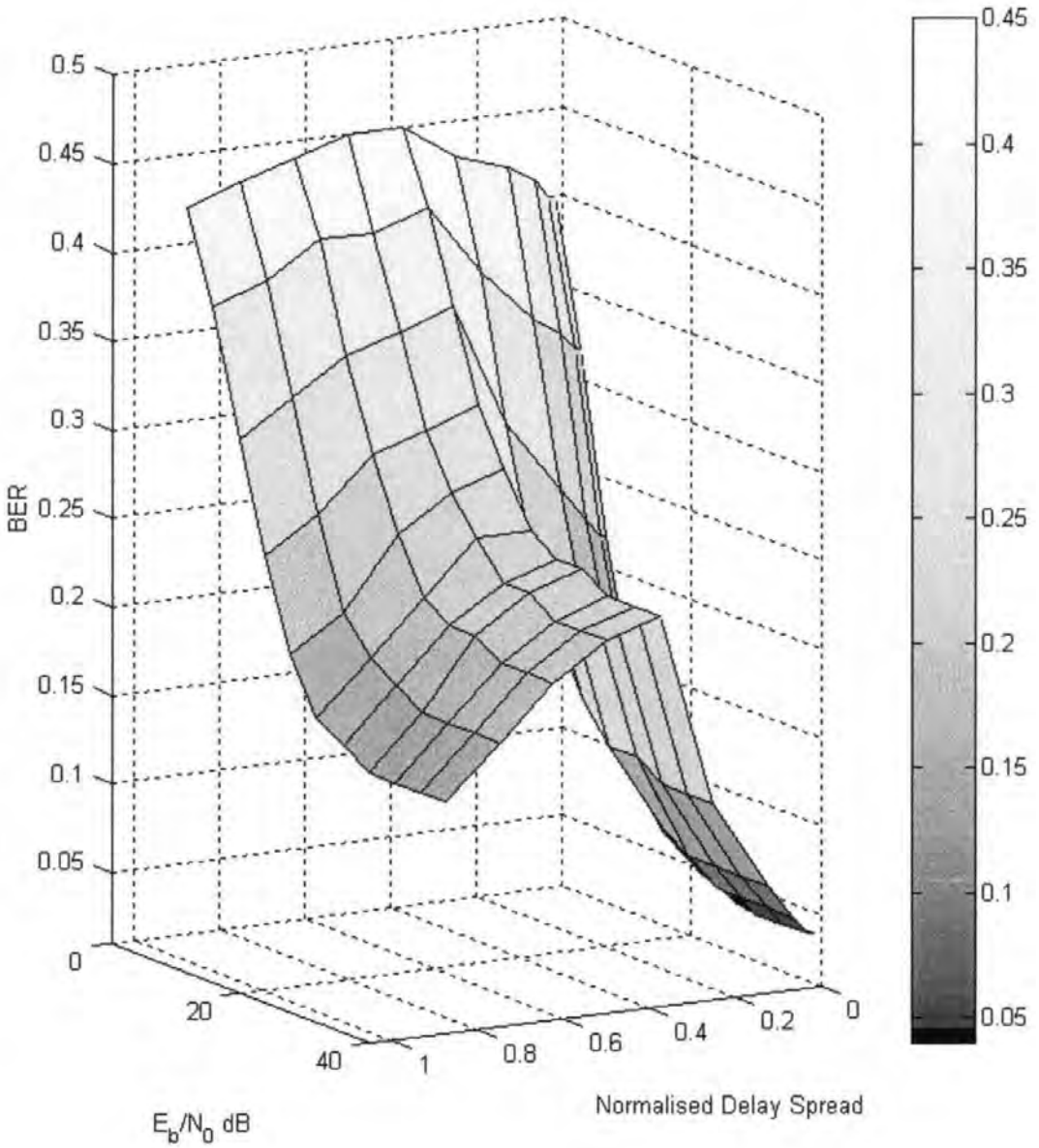


Figure 4.21: Non-Coherent Sinc(x) OFDM

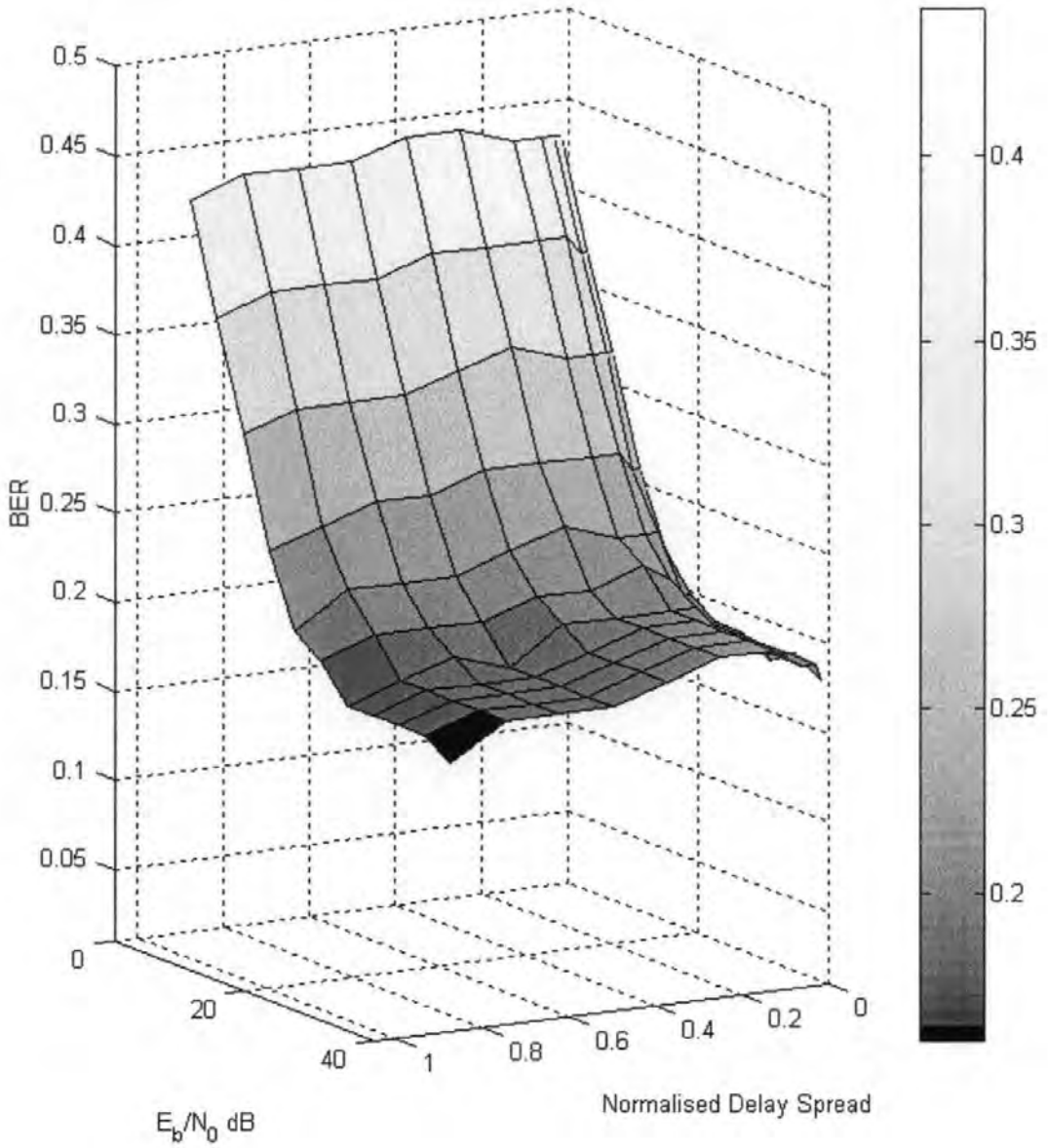


Figure 4.22: Low Pilot Density Coherent Sinc(x) OFDM

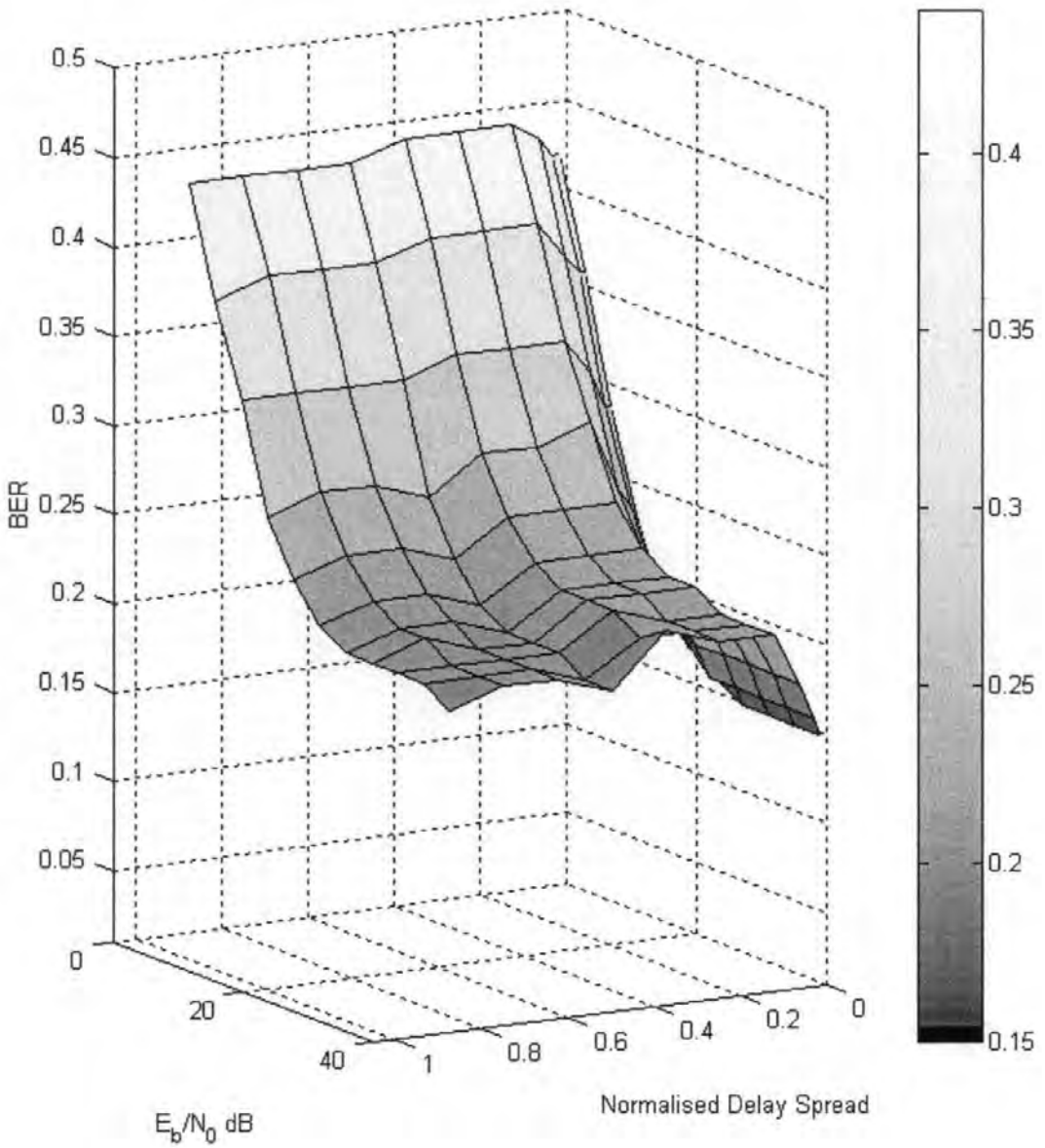


Figure 4.23: High Pilot Density Coherent Sinc(x) OFDM

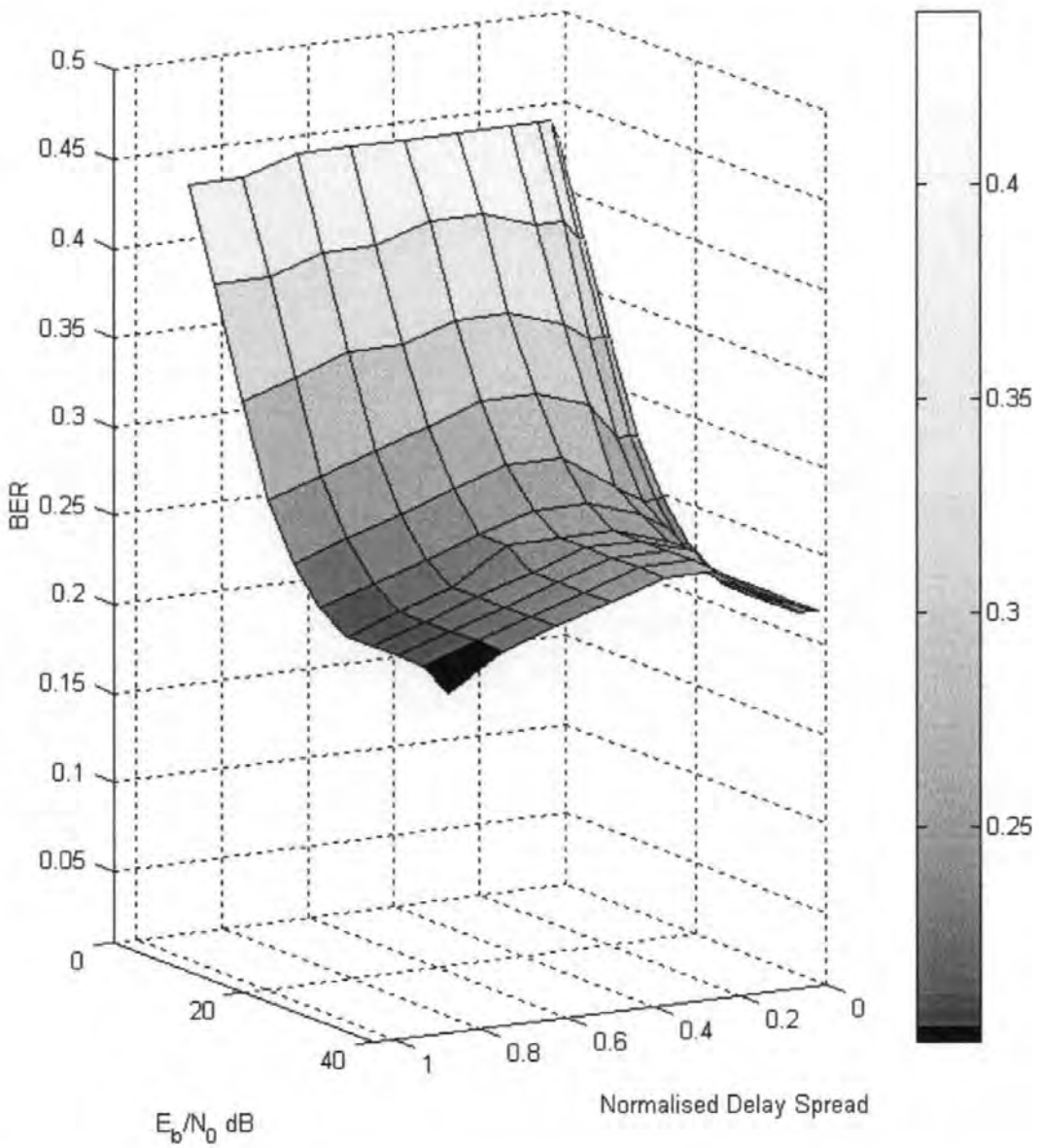


Figure 4.24: Non-Coherent Spectrally Shaped OFDM

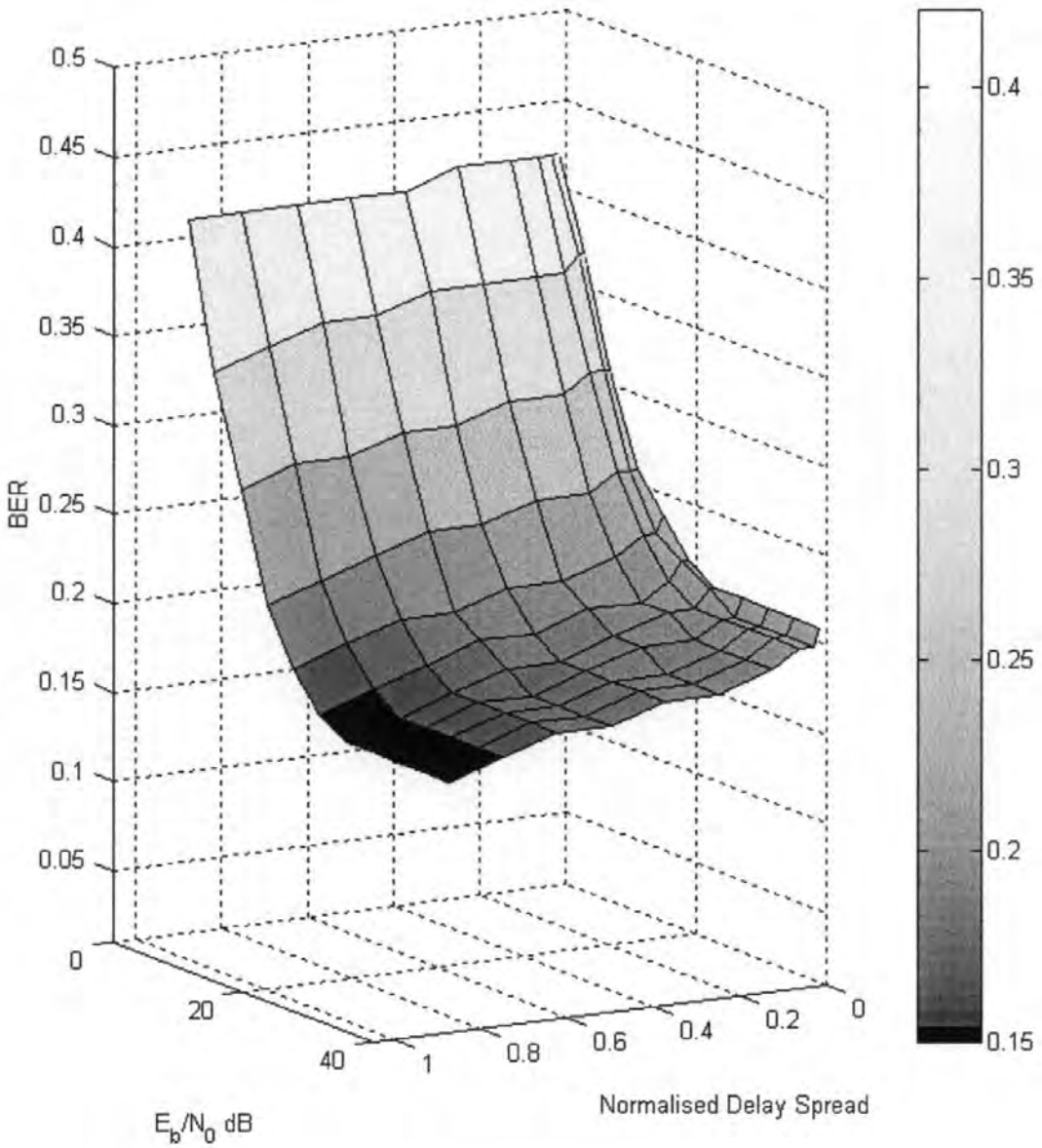


Figure 4.25: Low Pilot Density Coherent Spectrally Shaped OFDM

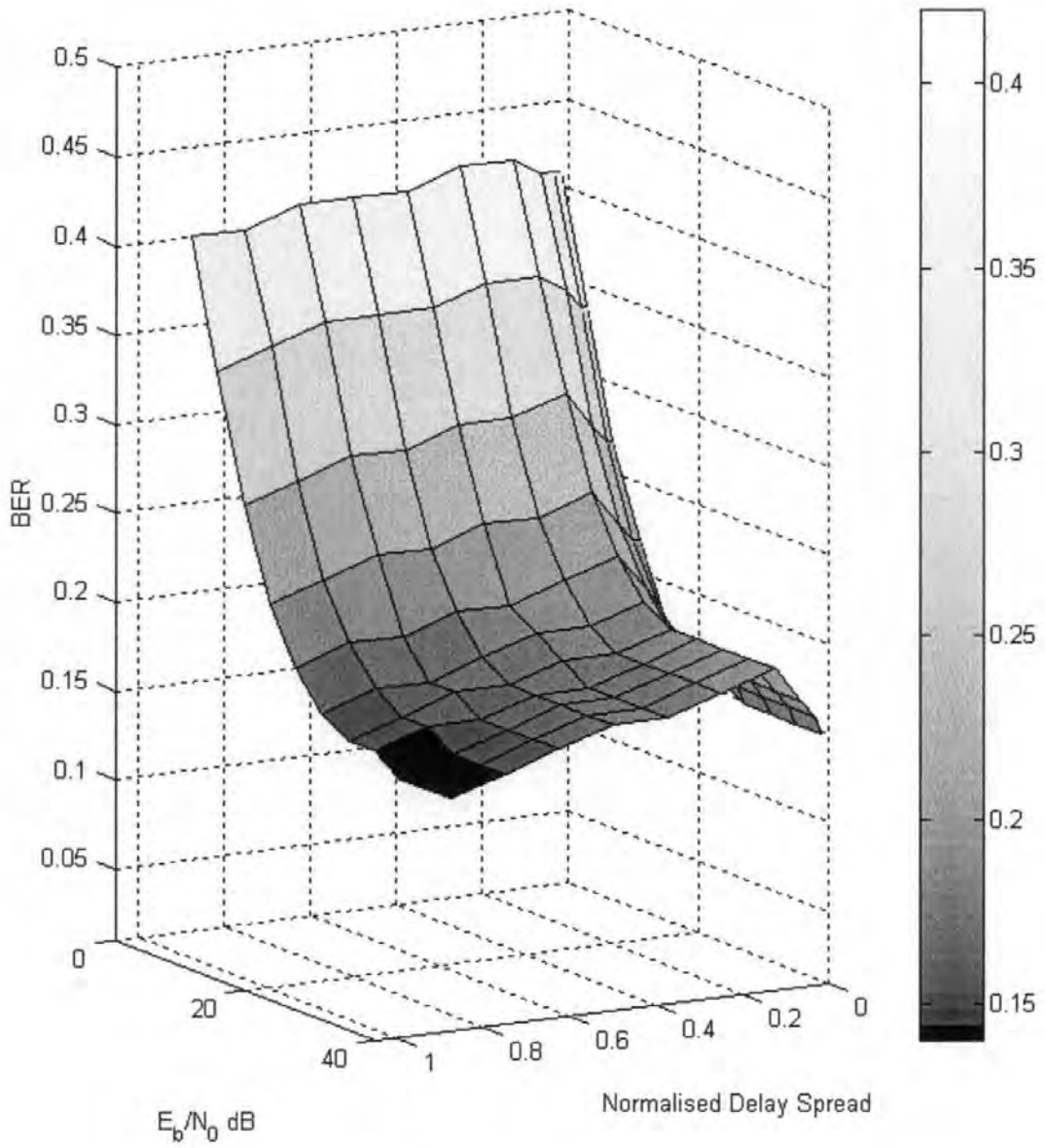


Figure 4.26: High Pilot Density Coherent Spectrally Shaped OFDM

4.5.3.3 Observations

From the experimentally determined BER versus E_b/N_0 and channel delay spread results presented in figures 4.21 to 4.26, a number of observations have been made which are summarised below.

Regardless of the value of channel delay spread or the type of channel estimation applied, as the value of E_b/N_0 increases, the BER for both Sinc(x) and Spectrally OFDM systems falls until a constant level or 'error floor' is reached.

With equivalent non-coherent channel estimation schemes, both Sinc(x) and Spectrally Shaped OFDM systems share a similar error floor versus channel delay spread characteristic, which rises to a peak BER value (just below 0.25) when the normalised channel delay spread (for the 2 path model simulated) is approximately of 0.5. The only significant difference between this characteristic for the two OFDM systems, is the value of lowest error floor values measured. For the Sinc(x) OFDM system the lowest error floor measured is below 0.05 whereas for Spectrally Shaped OFDM system it is much higher (and hence worse) at around 0.2.

With equivalent low pilot-to-data symbol ratio coherent channel estimation schemes, both Sinc(x) and Spectrally Shaped OFDM systems share a similar error floor versus channel delay spread characteristic which generally falls as the normalised channel delay spread rises.

With equivalent high pilot-to-data symbol ratio coherent channel estimation schemes, both Sinc(x) and Spectrally Shaped OFDM systems share a similar error floor versus channel delay spread characteristic. This characteristic exhibits a peak error floor value for a normalised delay spread somewhere between 0.1 and 0.2. For both OFDM systems, this error floor versus channel delay spread characteristic is generally slightly better (lower error floor values) than the comparable characteristic for the same OFDM system but with a low pilot-to-data symbol ratio coherent channel estimation.

The results presented in figures 4.24 to 4.26 for the Spectrally Shaped OFDM system indicate that the non-coherent channel estimation scheme applied is

significantly less effective than either of the coherent schemes given the range of channels measured.

The results presented in figures 4.21 to 4.23 for the Sinc(x) OFDM system indicate that error floor versus channel delay spread characteristic for the non-coherent channel estimation is significantly better than either of the coherent schemes for the lower and higher ranges of the channel delay spread. However for values of the channel delay spread between these ranges, i.e. 0.4 to 0.6, the same non-coherent channel estimation scheme performs slightly worse than the coherent schemes.

4.5.3.4 Discussions

Several of the figures show that the value of the error floor exhibits a peak for a channel normalised delay spread greater than zero but significantly less than 1 symbol period. This suggests that there are two phenomena dictating the behaviour of this characteristic.

The first of these phenomena can be attributed to the effectiveness of the channel estimation scheme to combat the frequency selective fades. As the channel frequency selective fading becomes more severe (i.e. with increasing channel delay spread), the ability of the channel estimation scheme to correctly compensate for the larger symbol phase rotations generated deteriorates. Eventually a 'transition' point is reached where the average fade bandwidth is narrower than the minimum fade bandwidth that the channel estimation can effectively compensate for. Beyond this point the channel estimation scheme ceases to be effective.

If this were the only phenomenon dictating the error floor performance, one might expect that beyond this point the measured error floor would rise (or worsen) even further. However the results clearly indicate that this is not the case. This suggests that another phenomena, which effectively helps reduce the error floor value as the channel delay spread increases, is acting in parallel to the one described above.

The varying degree of multi-path (effectively intra-symbol) interference generated by the second path of the frequency selective channel model may be the cause of this second phenomenon. As the separation in time, or channel delay spread, between this 'interfering' path and the first path increases, the worst case intra-symbol interference that this second path can generate falls. Note that for these simulations, the receiver was permanently synchronised to demodulate the first path regardless of its instantaneous quality compared to the second path. In real systems there would be some form of searcher/tracking algorithm, whose function would be to ensure that only the strongest path or paths were ever being demodulated.

Researchers, such as Hikmet Sari (ref. [65]), have observed that non-coherent channel estimation schemes are generally better than comparable coherent

schemes in channels more rapidly varying channels. Whereas the results for the Sinc(x) OFDM system appear to support this observation, the corresponding results for the Spectrally Shaped OFDM system do not. This difference in behaviour is most likely due to lower bounds of BER performance being dictated by the relatively strong time-selective nature of the time and frequency selective 2-path channel model. As can be seen from the results obtained for the purely time selective channel, the BER performance of Spectrally Shaped OFDM is much worse than that for comparable Sinc(x) OFDM systems sharing the same frequency domain biased channel estimation algorithms.

The spectrums of Sinc(x) OFDM sub-carriers in theory extend infinitely in frequency domain (with a Sinc(x) spectrum) if no deliberate filtering or windowing is applied. Therefore a fade, with for example, a bandwidth equal to the orthogonal sub-carrier separation, '$1/T_s$', corrupts the spectrum of all the adjacent sub-carriers with spectrums overlapping the fade region. The extent to which the overlapping spectrum of an adjacent sub-carrier is corrupted decreases the further away that sub-carrier is from the centre of the fade. In contrast, the spectrums of Spectrally Shaped OFDM sub-carriers have a finite spectrum that only overlaps with the two adjacent sub-carriers. This means that a fade with the same '$1/T_s$' bandwidth corrupts at most the spectrums of three sub-carriers. Therefore one would expect a Spectrally Shaped OFDM system to be more resilient to frequency selective fading than a Sinc(x) OFDM system, especially as the bandwidth of fades become narrower. However the results obtained from the experiments conducted here indicate otherwise.

Though the results confirm that Spectrally Shaped OFDM is better at handling frequency selective channels than time selective channels (with the channel estimation schemes applied), they do not show that it out-performs comparable Sinc(x) OFDM systems for the particular time and frequency selective channel simulated as predicted above. This is probably due to the time selective nature of the modelled channel being so strong that the advantage of Spectrally Shaped OFDM has in handling the frequency selective fades is completely masked out.

4.6 Conclusions

In this chapter a number of experiments have been performed to analyse and compare the uncoded BER performance of comparable Sinc(x) and Spectrally Shaped OFDM systems in a number of different radio channels. Though experiments like these have been performed before for several Sinc(x) OFDM systems, they have up to now not been performed for comparable Spectrally Shaped OFDM systems. Thus the novel results obtained, provide the first evidence to prove that, even with ideal channel estimation, Sinc(x) OFDM out-performs Spectrally Shaped OFDM in time-selective channels.

When comparable systems sharing the same frequency domain biased channel estimation schemes, are tested in a channel with frequency as well as time selective fading, Sinc(x) OFDM only marginally out-performs Spectrally Shaped OFDM. The degradation in performance suffered by a Sinc(x) OFDM system, when the frequency selective fading is added to the originally time selective radio channel, is much greater than that suffered by a comparable Spectrally Shaped OFDM system. This suggests that Spectrally Shaped OFDM may be more robust to frequency selective fading than Sinc(x) OFDM. Further experiments with more predominantly frequency selective channels are required to confirm this possible performance advantage of Spectrally Shaped OFDM.

Given these findings the only sorts of channel where a Spectrally Shaped OFDM system may be able to out-perform a Sinc(x) OFDM system, are ones which exhibit a high degree of frequency selective fading and a low degree of time selective fading. These channels are typically experienced by Wireless Local Loop systems, which provide high bandwidth cable-like services to stationary home and office located transceivers. However even for these applications, it may prove to be more cost-effective to invest extra processing power into enhancing the channel estimation, equalisation, interleaving and coding schemes of an existing relatively simple Sinc(x) OFDM system.

CHAPTER FIVE

5. Conclusions

In this chapter the achievements of the research conducted in the pursuit of the objectives outlined in Chapter 1, are summarised and critically examined.

5.1 Achievements of the Research Programme

The achievements of the research conducted in the pursuit of the objectives outlined in Chapter 1, are numbered and summarised below.

1. A novel proof for the 'conditions of orthogonality' that ensure zero Inter-Carrier Interference between over-lapping sub-carriers of a Spectrally Shaped OFDM system, has been derived from first principles (Appendix A).

2. A novel DSP architecture for the modulation and demodulation of Spectrally Shaped OFDM sub-carriers has been developed (chapter 2). This scalable design requires significantly fewer (up to half) operations per symbol period compared to the previous design (ref. [35]) published. It also has the flexibility to support both over-sampling and the transmission of QAM symbols. Nevertheless the complexity of even this new more efficient implementation, is arguably still too high for it *not* to be considered as a weakness when compared to other modulation schemes, such as Sinc(x) OFDM.

The effects of varying the number of sub-carriers and the characteristics of the filter section of the novel implementation architecture presented in chapter 2, have been extensively studied through simulation in chapter 3. From the results obtained,

3. The excellent spectral properties of Spectrally Shaped OFDM have been verified.

4. It has been discovered that windowing of the filter coefficients has a detrimental effect on the demodulated signal-to-distortion level.

5. Values of the Root Raised Cosine filter roll-off factor have been determined that minimise the demodulated signal-to-distortion level, when windowing of the filter coefficients is not applied.

6. The delay between when a symbol is modulated and demodulated, incurred as a result of the PolyPhaseNetwork filter component of the digital implementation, has been identified as a new fundamental weakness of this modulation scheme. This delay will seriously limit any performance enhancing advantages obtained

from schemes, which rely on rapid feedback between the transmitter and receiver (e.g. fast closed loop power control).

7. The peak-to-average power ratio of the Spectrally Shaped OFDM signal has been found to be between 1-3 dB worse than comparable Sinc(x) OFDM systems with the same number of sub-carriers.

From the investigations conducted to determine the BER performance of Spectrally Shaped OFDM in a variety of fast fading channels (chapter 4), a number of observations have been made.

8. It has been verified that Sinc(x) OFDM systems out-perform Spectrally Shaped OFDM systems in time-selective fast fading channels.

9. It has been shown that Spectrally Shaped OFDM is more robust to frequency-selective fading than time-selective fading.

In summary the key strengths of Spectrally Shaped OFDM identified and, to a limited extent, verified from the research conducted, include,

- It's excellent in-band and out-of-band spectral properties.

- Its robustness to predominantly frequency selective fading channels.

The major disadvantages of Spectrally Shaped OFDM are,

- The high complexity of its practical digital implementation.

- The high peak-to-average power ratio of its modulated signal. This is between 1-3 dB worse than comparable Sinc(x) OFDM systems.

- The high delay between when a symbol is modulated and demodulated.

5.2 Limitations of the Research

Key limitations of the research conducted are listed below.

The study of the demodulator sub-carrier output Signal-to-Distortion levels was limited to the ideal floating-point arithmetic domain, and therefore excluded the effects of fixed-point quantisation and scaling. From an attempt made to implement this modulation scheme on a fixed-point DSP development board, the overall system performance was found to be highly sensitive to overflow errors as a result of incorrect scaling.

The high peak-to-average power ratio (PAPR) of a Spectrally Shaped OFDM modulated signal, has been identified from this research as potentially a major weakness of this modulation scheme. Further analysis (perhaps analytical) is required to determine how seriously this could affect the overall system performance when real amplifiers and coding with interleaving is taken into account.

From the investigation into the performance of this modulation scheme in fast fading channels, it was identified that channel estimation schemes that are known to be very effective for Sinc(x) OFDM are not necessarily as effective for Spectrally Shaped OFDM. With alternative channel estimation schemes it is likely that difference in BER performance noted between Sinc(x) and Spectrally Shaped OFDM systems would have been far smaller, particularly in strongly frequency selective channels.

Further research is required to determine whether of not the roll-off factor (fixed at 0.25 for the simulations outlined in chapter 4) of the Root Raised Cosine Spectrally Shaped OFDM sub-carriers has an influence on the overall BER performance of the system in fast fading channels.

5.3 Suggestions and Scope for Future Work

From the research conducted a number of directions for future studies have been identified.

One major weakness of existing Sinc(x) OFDM systems, is their high sensitivity to timing and frequency synchronisation errors. This weakness of Sinc(x) OFDM has led to it not being adopted for applications where a large number of mobile users need to simultaneously transmit to same base station using adjacent sub-carriers. Note for DAB and DVB OFDM applications there are no 'uplink' radio transmissions, and for WLAN OFDM systems, only one user transmits to the base station at any one time (except during random access procedures). Further research is necessary to determine if Spectrally Shaped OFDM is as sensitive to these sorts of synchronisation errors as say Sinc(x) OFDM.

In chapter 4 it was shown that the effectiveness of the same channel estimation scheme can vary dramatically between comparable Sinc(x) and Spectrally Shaped OFDM systems. It is hypothesised that this phenomenon is due to the different domains in which the two OFDM variants exhibit a 'finite characteristic'. These characteristics for Sinc(x) and Spectrally Shaped OFDM respectively are, the finite duration of the truncated sinusoid used to modulate symbols in the time-domain, and the finite bandwidth of sub-carriers in the frequency-domain. If this 'time-frequency duality' hypothesis is correct, it would mean that near-optimal channel estimation, tracking, equalisation and inter-leaving schemes for Spectrally Shaped OFDM could be simply generated by taking existing well-developed schemes for Sinc(x) OFDM and reversing the domain of operation. Further research could be conducted to test and confirm this hypothesis.

From this research, Spectrally Shaped OFDM has been identified as being most suited to high bandwidth, low mobility Wireless Local Loop (WLL) applications. WLL systems, such as the commercially available AirLoop system, provide high bandwidth cable-like services to stationary home and office located transceivers. An investigation could be carried out to explore in detail the advantages and disadvantages associated with adopting this modulation scheme for this particular application.

APPENDIX A - PROOF OF SYSTEM ORTHOGONALITY

In this appendix the system design features that ensure orthogonality between the overlapping sub-carriers of a Spectrally Shaped OFDM system are mathematically proven.

DESCRIPTION OF SYSTEM

Figure A.1 below, shows a simple model of a complex modulator for a single Spectrally Shaped sub-carrier. From this model, equation A.1 can be derived describing the *'ith'* sub-carrier modulated signal, *'$o_i(t)$'*, in terms of;

$a_i + jb_i$: Complex QPSK (Or QAM) modulated onto the 'ith' sub-carrier.

$h_i'(t)$: Pulse shaping filter for real part of 'ith' sub-carrier symbol.

$h_i''(t)$: Pulse shaping filter for imaginary part of 'ith' sub-carrier symbol.

$\omega_i (=2\pi f_i)$: Modulating frequency of 'ith' sub-carrier.

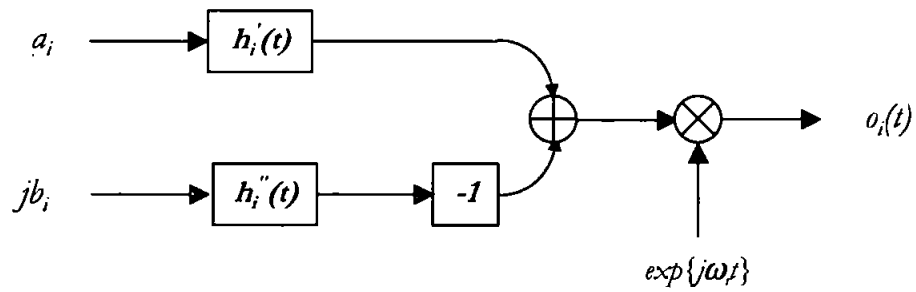


Figure A.1: *'ith'* Sub-Carrier Modulator

$$o_i(t) = [a_i h_i'(t) - jb_i h_i''(t)]\exp\{j\omega_i t\}$$

Equation A.1: *'ith'* Sub-Carrier Modulated Signal

Note that for the time being, equation A.1 makes no assumptions regarding the exact values of the offset delay (now effectively incorporated into the filter function), channel spacing and filter shaping function required to maintain orthogonality.

Figure A.2 below, shows the corresponding model of a complex demodulator for the 'k^{th}' Spectrally Shaped sub-carrier.

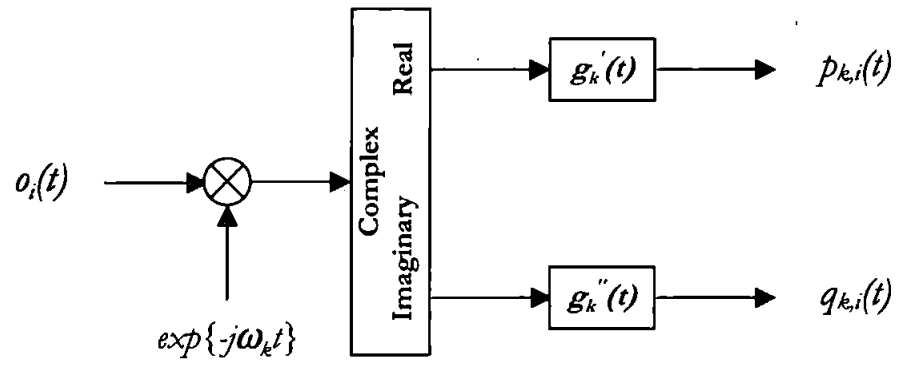


Figure A.2: 'k^{th}' Sub-Carrier Demodulator

From the models shown in figures A.1 and A.2, equations A.2 and A.3 can be derived to describe the real and imaginary parts of the 'k^{th}' sub-carrier demodulated signal, '$p_{k,i}(t)$' and '$q_{k,i}(t)$', assuming a modulated input signal, '$o_i(t)$', from the 'i^{th}' sub-carrier. For these equations, 'ω_c' is equal to the bandwidth separation ('ω_i-ω_k') between the 'i^{th}' and 'k^{th}' sub-carriers, and '$*$' represents a convolution integral operation.

$$p_{k,i}(t) = [a_i h_i'(t)\cos(\omega_c t) + b_i h_i''(t)\sin(\omega_c t)] * g_k'(t)$$

Equation A.2: Real Part of 'k^{th}' Sub-Carrier Demodulated Signal

$$q_{k,i}(t) = [a_i h_i'(t)\sin(\omega_c t) - b_i h_i''(t)\cos(\omega_c t)] * g_k''(t)$$

Equation A.3: Imaginary Part 'k^{th}' Sub-Carrier Demodulated Signal

When '$k \neq i$', equations A.2 and A.3 define the Inter-Carrier Interference (ICI) suffered by the 'k^{th}' demodulated sub-carrier, due to the modulated signal from the 'i^{th}' sub-carrier. In complete contrast, when '$k=i$', these same equations define the desired output signals for 'k^{th}' demodulated sub-carrier.

To obtain simplified expressions for the spectra of '$p_{k,i}(t)$' and '$q_{k,i}(t)$', the filter functions, $h_i'(t)$, $h_i''(t)$, $g_i'(t)$ *and* $g_i''(t)$, will be assumed to be derived from a single common filter '$h_0(t)$' in the manner summarised by table A.1. Note that the functions defined this table, include an assumption of where 'offset' delays (α, β) are required between the real and imaginary parts of modulated symbols.

$h_i'(t)$	$h_0(t-\beta) \leftrightarrow H_0(\omega)\exp(-j\omega\beta)$ ***i = Even Integer***	$h_0(t) \leftrightarrow H_0(\omega)$ ***i = Odd Integer***
$h_i''(t)$	$h_0(t-\beta) \leftrightarrow H_0(\omega)$ ***i = Even Integer***	$h_0(t) \leftrightarrow H_0(\omega)\exp(-j\omega\beta)$ ***i = Odd Integer***
$g_k'(t)$	$h_0(t-\alpha) \leftrightarrow H_0(\omega)\exp(-j\omega\alpha)$ ***k = Even Integer***	$h_0(t) \leftrightarrow H_0(\omega)$ ***k = Odd Integer***
$g_k''(t)$	$h_0(t-\alpha) \leftrightarrow H_0(\omega)$ ***k = Even Integer***	$h_0(t) \leftrightarrow H_0(\omega)\exp(-j\omega\alpha)$ ***k = Odd Integer***

Table A.1: Assumptions regarding System Filter Definitions

Given the filter definitions presented in table A.1, the spectra, '$Q_{k,i}(\omega)$' *and* '$P_{k,i}(\omega)$', can be derived in terms of 'β', 'α', and '$H_0(\omega)$' for all permutations of 'i' and 'k', as shown in table A.2.

***i* odd, *k* odd**	$P_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\left\{\frac{a_i}{2}+\frac{b_i}{2j}\exp\{-j(\omega-\omega_c)\beta\}\right\}\ldots$ $+ H_0(\omega)H_0(\omega+\omega_c)\left\{\frac{a_i}{2}-\frac{b_i}{2j}\exp\{-j(\omega+\omega_c)\beta\}\right\}$
	$Q_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2j}-\frac{b_i}{2}\exp\{-j(\omega-\omega_c)\beta\}\right\}\ldots$ $- H_0(\omega)H_0(\omega+\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2j}+\frac{b_i}{2}\exp\{-j(\omega+\omega_c)\beta\}\right\}$
***i* odd, *k* even**	$P_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2}+\frac{b_i}{2j}\exp\{-j(\omega-\omega_c)\beta\}\right\}\ldots$ $+ H_0(\omega)H_0(\omega+\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2}-\frac{b_i}{2j}\exp\{-j(\omega+\omega_c)\beta\}\right\}$
	$Q_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\left\{\frac{a_i}{2j}\exp\{-j(\omega-\omega_c)\beta\}-\frac{b_i}{2}\right\}\ldots$ $- H_0(\omega)H_0(\omega+\omega_c)\left\{\frac{a_i}{2j}\exp\{-j(\omega+\omega_c)\beta\}+\frac{b_i}{2}\right\}$
***i* even, *k* odd**	$P_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\left\{\frac{a_i}{2}\exp\{-j(\omega-\omega_c)\beta\}+\frac{b_i}{2j}\right\}\ldots$ $+ H_0(\omega)H_0(\omega+\omega_c)\left\{\frac{a_i}{2}\exp\{-j(\omega+\omega_c)\beta\}-\frac{b_i}{2j}\right\}$
	$Q_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2j}-\frac{b_i}{2}\exp\{-j(\omega-\omega_c)\beta\}\right\}\ldots$ $- H_0(\omega)H_0(\omega+\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2j}+\frac{b_i}{2}\exp\{-j(\omega+\omega_c)\beta\}\right\}$
***i* even, *k* even**	$P_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2}\exp\{-j(\omega-\omega_c)\beta\}+\frac{b_i}{2j}\right\}\ldots$ $+ H_0(\omega)H_0(\omega+\omega_c)\exp\{-j\omega\alpha\}\left\{\frac{a_i}{2}\exp\{-j(\omega+\omega_c)\beta\}-\frac{b_i}{2j}\right\}$
	$Q_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)\left\{\frac{a_i}{2j}\exp\{-j(\omega-\omega_c)\beta\}-\frac{b_i}{2}\right\}\ldots$ $- H_0(\omega)H_0(\omega+\omega_c)\left\{\frac{a_i}{2j}\exp\{-j(\omega+\omega_c)\beta\}+\frac{b_i}{2}\right\}$

Table A.2: $P_{k,i}(\omega)$ and $Q_{k,i}(\omega)$ Spectra

Using table A.2, it will now be shown how the system requirements for zero Inter-Symbol Interference (ISI) and zero Inter-Carrier Interference (ICI), restrict to sub-carrier filter shaping to be Nyquist and the magnitude of the 'offset' delays, 'β' and 'α', to being '$T_s/2$'.

REQUIREMENTS FOR ZERO INTER-SYMBOL INTERFERENCE

The optimum design of the pulse/sub-carrier shaping filter, '$h_0(t)$', can be derived from examining the spectra of the *desired* demodulated signals, '$p_{k,i}(t)$' and '$q_{k,i}(t)$' when '$k=i$'. Generalised expressions for these spectra can be found in table A.2. After these expressions are simplified to take into account that '$\omega_c = \omega_i - \omega_k = 0$', equations A.4 and A.5 are obtained.

$$\left.\begin{aligned} P_{k,i}(\omega) &= a_i H_0(\omega)^2 \\ Q_{k,i}(\omega) &= -b_i H_0(\omega)^2 \exp\{-j\omega(\beta+\alpha)\} \end{aligned}\right\} \quad k = i = odd\ integer$$

Equation A.4: Desired Signal Spectra (Odd Sub-carriers)

$$\left.\begin{aligned} P_{k,i}(\omega) &= a_i H_0(\omega)^2 \exp\{-j\omega(\beta+\alpha)\} \\ Q_{k,i}(\omega) &= -b_i H_0(\omega)^2 \end{aligned}\right\} \quad k = i = even\ integer$$

Equation A.5: Desired Signal Spectra (Even Sub-carriers)

From the above equations, it can be deduced that to ensure zero Inter-Symbol Interference, the squared match filter response, $H_0(\omega)^2$, must obey Nyqiust conditions. The Root Raised Cosine filter is the most commonly used filter that obeys this condition.

Note that the '$exp\{-j\omega(\alpha+\beta)\}$' factor present in both of the above equations, translates to a time shift of '$\alpha+\beta$' in the time-domain.

REQUIREMENTS FOR ZERO INTER-CARRIER INTERFERENCE

For the real valued Inter-Carrier Interference (ICI) terms, defined by *'$p_{k,i}(t)$'* and *'$q_{k,i}(t)$'* when *'$k \neq i$'*, to be zero (i.e. orthogonal) at the instants (once every symbol period T_s) when the desired signals, *'$p_{k,i}(t)$'* and *'$q_{k,i}(t)$'* for *'$k=i$'*, are sampled to determine the transmitted symbols, *'$a+jb$'*, their spectra ($Q_{k\neq i}(\omega)$ & $P_{k\neq i}(\omega)$) must satisfy the condition defined by equation A.6.

$$\left.\begin{aligned} \sum_{k=-\infty}^{\infty} Q_{k\neq i}(\omega - \omega_s k) = 0 \\ \sum_{k=-\infty}^{\infty} P_{k\neq i}(\omega - \omega_s k) = 0 \end{aligned}\right\} \quad where\ \omega_s = {}^{2\pi}\!/_{T_s}\ and\ k = integer$$

Equation A.6: Zero ICI Condition

Without loss of generality, one expression for $P_{k\neq i}(\omega)$ from table A.2 will now be tested the 'Zero ICI Condition' expressed by equation A.6, to determine what values of 'α' and 'β' ensure system orthogonality.

The spectra, $P_{k\neq i}(\omega)$, for the case when '*i*' is odd and '*k*' is even, can be expressed as follows (see Table A.2).

$$P_{k,i}(\omega) = H_0(\omega)H_0(\omega-\omega_c)g(\omega) + H_0(\omega)H_0(\omega+\omega_c)f(\omega)$$

Equation A.7: $P_{k\neq i}(\omega)$ *when 'i'* is odd and *'k'* is even

Where,

$$g(\omega) = \exp\{-j\omega\alpha\}\left\{\frac{a_i}{2} + \frac{b_i}{2j}\exp\{-j(\omega-\omega_c)\beta\}\right\}$$

$$f(\omega) = \exp\{-j\omega\alpha\}\left\{\frac{a_i}{2} - \frac{b_i}{2j}\exp\{-j(\omega+\omega_c)\beta\}\right\}$$

Equation A.8: *g(ω) and f(ω) Terms*

To ensure zero Inter-Symbol Interference, it has already been shown that '$H_0(\omega)$' must be a Nyquist filter. The Root Raised Cosine Nyquist filter applied for sub-carrier shaping in a Spectrally Shaped OFDM system, has a band-limited spectrum, '$H_0(\omega)$', that is evenly symmetric about '$\omega = 0$'. This characteristic means that the $H_0(\omega)H_0(\omega-\omega_c)$ and $H_0(\omega)H_0(\omega+\omega_c)$ terms in equation A.7, each have band-limited symmetrically even spectra which are centred about '$+\pi(i-k)/T_s$' and '$-\pi(i-k)/T_s$' respectively. This allows equation A.6 to be simplified to,

$$P_{k,i}(\omega) + P_{k,i}(\omega - \omega_s) = 0 \qquad where \begin{cases} \omega_s = \dfrac{2\pi}{T_s} \\ \omega = 0 \rightarrow \dfrac{2\pi}{T_s} \end{cases}$$

Equation A.9: Simplified Zero ICI Condition

Given our knowledge of ω_s, ω_c and the spectral characteristics of $H_0(\omega)$, the substitution of equation A.7 into A.9 can be simplified dramatically to derive the condition expressed by equation A.10.

$$g(\omega) + f(\omega - \omega_c) = 0$$

Equation A.10: Zero ICI Condition Requirement

From equations A.10 and A.8 it can be shown that 'α' and 'β' must satisfy,

$$1 + \exp\{j\omega_c\alpha\} = \exp\{j\omega_c\beta\} - \exp\{j\omega_c\alpha\} = 0$$

$$given \quad \omega_c = \omega_s = \frac{2\pi}{T_s}$$

$$\Rightarrow \alpha = \beta = \frac{T_s}{2}$$

Equation A.11: $T_s/2$ Offset Delay Value

Hence it can be shown that to ensure orthogonality in a Spectrally Shaped OFDM system employing band-limited Root Raised Cosine Nyqiust filters, a delay of Ts/2 is required between the real and imaginary components of symbols modulated and demodulated on a particular sub-carrier. The location and values of these delays are highlighted in figures 1.1 and 2.3.

APPENDIX B - DFT SIZE REDUCTION

In this appendix, it is shown how the processing performed by the two complex 'N/2' point DFTs in the sub-system originally illustrated by figure 2.8, can be achieved using a single 'N/2' point DFT with some minor pre- and post-processing.

DERIVATION OF SIMPLIFIED N/2 POINT DFT SUB-SYSTEM

Define the two sub-sequences entering the two 'N/2' complex DFTs, shown in figure B.1 (a copy of figure 2.8), as *{f}* and *{g}*.

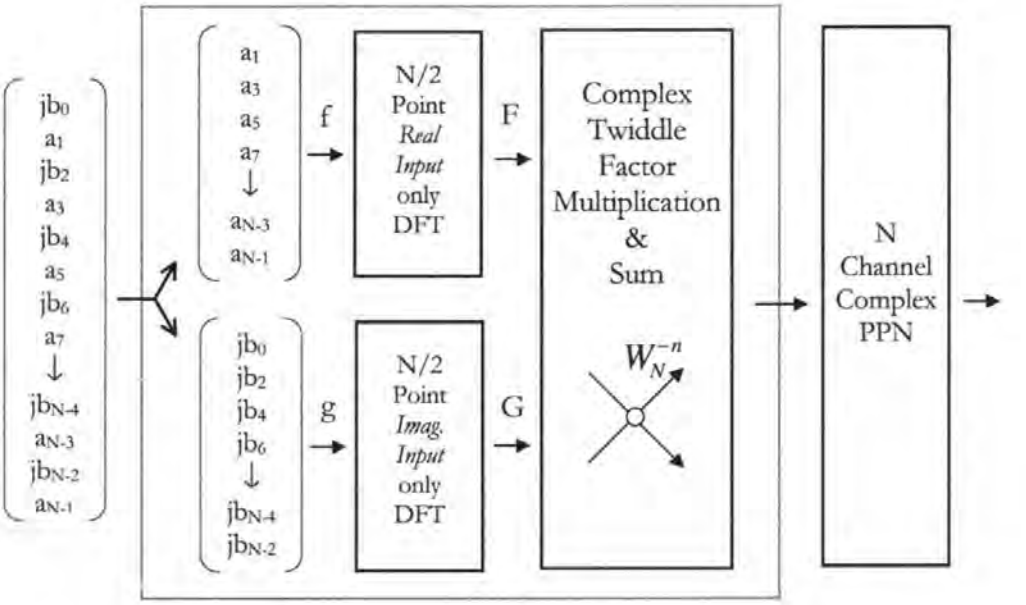


Figure B.1: Double Complex *'N/2'* DFT Sub-System

These sequences, which are either entirely real or imaginary, have complex DFTs *F* and *G* respectively.

$$f \xleftrightarrow{DFT} F \qquad g \xleftrightarrow{DFT} G$$

Equation B.1: DFT of *f* & g

Define a new complex input sequence, *{h}*, derived from *{f}* and *{g}*, as,

$$h = f + g$$

Equation B.2: Definition of Sequence *h*

Due to linearity, the DFT, *H*, of *{h}* is given by,

$$\left.\begin{array}{l} H = F + G \\ \Rightarrow H(k) = F(k) + G(k) \end{array}\right\} \; for \;\; k = 0,1,2,...,M-1 \;\; and \;\; M = \frac{N}{2}$$

Equation B.3: DFT Linearity Property

The DFT of a conjugated complex sequence, such as *h*(n)*, is,

$$h^*(n) \xleftrightarrow{DFT} H^*(M-k)$$

Equation B.4: DFT of a Conjugated Sequence

Given equation B.4, *H*(k)*, can be defined in terms of *F(k)* and *G(k)*.

$$H^*(M-k) = F^*(M-k) + G^*(M-k)$$

Equation B.5: Expression for *H*(k)*

Since the sequences *{f}* and *{g}* are made up of only real and imaginary components respectively, their DFTs have the following properties.

$$F(k) = F^*(M-k)$$
$$G(k) = -G^*(M-k)$$

Equation B.6: Symmetrical Properties of DFTs with real or imaginary inputs only

From equations B.5 and B.6, the following expression can be derived,

$$H^*(M-k) = F(k) - G(k)$$

Equation B.7: Simplified Expression for *H*(k)*

Using equations B.3 and B.7, it can now be shown (equation B.8) that the processing performed by 2 'N/2' point complex DFTs to obtain *F* and *G*, can be achieved using a single 'N/2' DFT, *H*, with input sequence *{h} = {f} + {g}*.

$$\left.\begin{aligned} F(k) &= \frac{H(k) + H^*(M-k)}{2} \\ G(k) &= \frac{H(k) - H^*(M-k)}{2} \end{aligned}\right\} \quad for\ k = 0,1,2,\ldots,M-1$$

Equation B.7: Simplified Expression for *F(k) & G(k)*

Note that when k = 0, H*(M-k) reduces to H*(0) and not H*(M).

APPENDIX C - SOFTWARE

This appendix contains the C-code used to build a Spectrally Shaped OFDM simulation engine. This engine was used to test the DSP implementation described in chapter 2 and to provide a full range the system performance results. The C-code listed includes the complete set of 9 Header files required and 1 example of a Main file, used to derive the BER performance in an AWGN channel. This ANSI-C compatible code was compiled and run on a 166 MHz Pentium PC using the Borland C++ (Version 5.01) development tool. Not included in this appendix are the almost identical code listings created for Sinc(x) OFDM simulation engine.

F_RANDOM.H

```
/* f_random.h
 *
 * 3 Random Number Generator Functions Defined.
 * Each is built around a pseudo-random number generator function - RanInt.
 *
 * 'RanDouble()'       returns a random double number between 0 and 1.
 * 'RanGauss(sigma)'  returns a Gaussian double number with variance sigma.
 * 'RanBipolar()'     returns a uniformly distributed int equal to +1/-1.
 *
 * Version : 1.0
 * Date : 5th June 1997
 * Orignal Author : Krishnamurthy
 * Editor : David Bhatoolaul
 *ANSI C compatible
 * Edited and Tested with Borland C/C++ PC version 5.0
 *
 * Changes to original Code :
 *
 *  void SetSeed() ---> void SetSeed(long, long);
 *     To allow compilation by Borland C/C++ Version 5.0 in ANSI C mode.
 *  double RanGauss() ----> double RanGauss( double sigma)
 *     Original RanGauss function generated only normalised gaussian samples.
 *
 */

#define Gen1 06651126160    /* degree 30, No. 19, Peterson and Weldon p. 492 */
#define Largest1 07777777777
#define Test1 00000000001   /* LSB */
#define Gen2 00004064275    /* degree 29, No. 21, Peterson and Weldon p. 491 */
#define Largest2 03777777777
#define Test2 02000000000   /* MSB */
#define Normal 1            /* log2((Largest1+1)/(Largest2+1)) */

long State1, State2;        /* To set seed, set these (non-zero) */

/* SetSeed sets the seeds to i & j */
void SetSeed(long i, long j)
{
 if (i != 0) State1 = i; else State1 = 1;
 if (j != 0) State2 = j; else State2 = 1;
}

/* RanInt returns a random non-negative integer number between 1 and Largest1 */
long RanInt()
{
 long i;
 void SetSeed(long, long);
 if ((State1 == 0) || (State2 == 0)) SetSeed(0, 0);
 i = State1 & Test1;                    /* Get LSB of State 1 */
 State1 >>= 1;                          /* Shift State1 right */
 if (i != 0) State1 ^= Gen1;            /* XOR Generator1     */

 i = State2 & Test2;                    /* Get MSB of State 2 */
 State2 = (State2 << 1) & Largest2;
                                        /* Shift State2 left and mask */
 if (i != 0) State2 ^= Gen2;            /* XOR Generator2 */
 return (State1 ^ (State2 << Normal));
}


/* RanDouble returns a random double number between 0 and 1 */
double RanDouble( void )
{
 long RanInt( void );
 return ((double) RanInt() / Largest1);
}

/*
 * RanGauss():
 * A Gaussian random number generator based on uniform [0, 1] numbers.
```

```
 * (See Knuth Vol. 2, 2cd Ed., page 117, Algorithm P.)
 */

double RanGauss( double sigma)
{
  double x, y, s, RanDouble();
  s = 1.;
  while (s >= 1.) {
    x = 2. * RanDouble() - 1.;
    y = 2. * RanDouble() - 1.;
    s = x * x + y * y;
  }
  x *= sqrt(-2. * log(s) / s);
  return ((x*sigma));
}

/* RanBipolar returns a uniformly distributed int equal to +1/-1. */

int RanBipolar(void)
{
  double RanDouble();

  if(RanDouble() < 0.5) return(-1);
  else return(1);
}
```

CMPLX.H

```
typedef struct complex{ double re, im; } complex;

/*****************************************************************************/
complex add_cpx( complex ajb, complex cjd ){

  complex ans;
  ans.re = ajb.re + cjd.re;
  ans.im = ajb.im + cjd.im;
  return(ans);
}

/*****************************************************************************/
complex minus_cpx( complex ajb, complex cjd ){

  complex ans;
  ans.re = ajb.re - cjd.re;
  ans.im = ajb.im - cjd.im;
  return(ans);
}

/*****************************************************************************/
complex multiply_cpx( complex ajb, complex cjd ){

  complex ans;
  ans.re = ajb.re*cjd.re - ajb.im*cjd.im;
  ans.im = ajb.re*cjd.im + ajb.im*cjd.re;
  return(ans);
}

/*****************************************************************************/
complex scalar_multiply_cpx( complex ajb, double factor ){

  complex ans;
  ans.re = ajb.re*factor;
  ans.im = ajb.im*factor;
  return(ans);
}

/*****************************************************************************/
double power_cpx( complex ajb ){

  double ans;
  ans = (ajb.re*ajb.re) + (ajb.im*ajb.im);
  return(ans);
}

/*****************************************************************************/
complex inverse_cpx (complex ajb){

  double scale_factor;
  complex ans;
  scale_factor = 1.0/(sqrt(power_cpx(ajb)));
  ans.re = scale_factor*ajb.re;
  ans.im = -1.0*scale_factor*ajb.im;
  return(ans);
}
```

```
/*****************************************************************************/
double arg_cpx (complex ajb){

  double ans, offset, re_part, im_part, theta;
  re_part = fabs(ajb.re);
  im_part = fabs(ajb.im);
  theta = atan(im_part/re_part);
  if ( (ajb.re >= 0.0) && (ajb.im >= 0.0) ) {
    ans = theta;
  } else if ( (ajb.re < 0.0) && (ajb.im >= 0.0) ) {
    ans = M_PI - theta ;
  } else if ( (ajb.re < 0.0) && (ajb.im < 0.0) ) {
    ans = M_PI + theta;
  } else if ( (ajb.re >= 0.0) && (ajb.im < 0.0) ) {
    ans = 2.0*M_PI - theta;
  }

  return(ans*180.0/M_PI);

}

/*****************************************************************************/
complex powarg2cpx ( double cpx_pow, double cpx_arg){

  complex ans;
  double theta;

  ans.re = 0.0;
  ans.im = 0.0;
  theta = 0.0;

  if (cpx_arg >= 2.0*M_PI) {
    while (cpx_arg >= 2.0*M_PI) cpx_arg -= 2.0*M_PI;
  }

  if ((cpx_arg >= 0.0) && (cpx_arg < M_PI_2)){
    theta = cpx_arg;
    ans.re = cpx_pow*cos(theta);
    ans.im = cpx_pow*sin(theta);
  }

  if ((cpx_arg >= M_PI_2) && (cpx_arg < 2.0*M_PI_2)) {
    theta = cpx_arg - 1.0*M_PI_2;
    ans.re = -1.0*cpx_pow*sin(theta);
    ans.im = cpx_pow*cos(theta);
  }

  if ((cpx_arg >= 2.0*M_PI_2) && (cpx_arg < 3.0*M_PI_2)) {
    theta = cpx_arg - 2.0*M_PI_2;
    ans.re = -1.0*cpx_pow*cos(theta);
    ans.im = -1.0*cpx_pow*sin(theta);
  }

  if ((cpx_arg >= 3.0*M_PI_2) && (cpx_arg < 4.0*M_PI_2)) {
    theta = cpx_arg - 3.0*M_PI_2;
    ans.re = cpx_pow*sin(theta);
    ans.im = -1.0*cpx_pow*cos(theta);
  }

  return(ans);
}

/*****************************************************************************/
```

DEFINE.H

```
#define NUM_DATA_CHANNELS       4
#define OVER_SAMPLING_FACTOR    8
#define RRC_SYMBOL_DURATION     8.0
#define RRC_ROLL_OFF            0.25
#define SYMBOL_PERIOD           1.0

#define NUM_PILOTS              32

#define JAKES_OSCILLATORS       16
#define P1_2_DIFF               112
#define P1_POWER                0.5
#define P2_POWER                0.5
#define P1_DOP_SAM              0.125
#define P2_DOP_SAM              0.125
#define P1_PHASE                0.0
#define P2_PHASE                0.0
#define P1_INDEPENDENT          1.0
#define P2_INDEPENDENT          4.0
#define P1_ALPHA                0.0
#define P2_ALPHA                0.0

#define BER_COUNT               100
#define EbN0_COUNT              1
#define SAM_COUNT               400000
```

STRUCT.H

```
/*************************************************************************/
typedef struct simulation_parameters{

  int num_data_channels;
  int over_sampling_factor;
  int num_channels;
  int num_filt_taps;
  int num_ppn_taps;
  int filt_symbol_duration;
  int num_oscillators;
  int path_diff;
  int pilots;
  double true_symbol_period;
  double true_sample_period;
  double filt_roll_off;
  double dop_sym_ratio1;
  double dop_sym_ratio2;
  double jakes_phase1;          /** Path 1 Static Phase Distortion **/
  double jakes_phase2;          /** Path 2 Static Phase Distortion **/
  double power1;                /** Path 1 Static Power Distortion **/
  double power2;                /** Path 2 Static Power Distortion **/
  double alpha1;                /** Path 1 Jakes Generator Alpha **/
  double alpha2;                /** Path 2 Jakes Generator Alpha **/
  double independent1;          /** Path 1 Jakes Generator Independence Factor **/
  double independent2;          /** Path 2 Jakes Generator Independence Factor **/

} simulation_parameters;

/*************************************************************************/
typedef struct simulation_memory{

  long int *symbol_number;      /** Simulation Counting Information **/
  long int *fft_clock;          /** Number of times FFT has been clocked **/
  complex  *init_qpsk;          /** Initial QPSK Symbol Input **/
  complex  *encode_qpsk;        /** Differentially Encoded QPSK **/
  complex*upper_branch;         /** Upper Branch DFT-PPN storage branch **/
  complex*lower_branch;         /** Lower Branch DFT-PPN storage branch**/
  double*filter_coefficients;   /** 1-D RRC Filter Coefficients Storage **/
  double*filter_array;          /** 2-D RRC Filter Coefficients Storage **/
  complex*upper_filt_mem;       /** 2-D RRC PPN Tap Value Storage **/
  complex*lower_filt_mem;       /** 2-D RRC PPN Tap value Storage **/
  complex  *mod_out;            /** Modulator Output **/
  complex  *old_mod_out;        /** Modulator Output **/
  complex  *prev_lower_branch;  /** Previous Modulator Output **/
  long int *ber;                /** BER Error count **/
  long int *demod_sym_num;      /** Number of symbols demodulated **/
  double   *demod_noise_pow;    /** Accumulated Distortion Power **/
  complex  *fading_taps1;       /** Fading Tap Memory **/
  complex  *fading_taps2;       /** Second Multipath Component **/

} simulation_memory;

/*************************************************************************/
typedef struct demodulator_memory{

  complex*quad_buffer;          /** Simulation Counting Information **/
  complex  *reference_qpsk;     /** Initial QPSK Symbol Input **/
  complex*beta0_branch;         /** Upper Branch DFT-PPN storage branch **/
  complex*beta1_branch;         /** Lower Branch DFT-PPN storage branch **/
  complex*beta2_branch;         /** Upper Branch DFT-PPN storage branch **/
  complex*beta3_branch;         /** Lower Branch DFT-PPN storage branch **/
  complex*beta0_filt_mem;       /** 2-D RRC PPN Tap Value Storage **/
  complex*beta1_filt_mem;       /** 2-D RRC PPN Tap value Storage **/
  complex*beta2_filt_mem;       /** 2-D RRC PPN Tap Value Storage **/
  complex*beta3_filt_mem;       /** 2-D RRC PPN Tap value Storage **/
  complex  *demod_out;          /** Modulator Output **/
  complex  *demod_chan_est;
  int jitter;

} demodulator_memory;
```

INIT.H

```
/**********SINC FUNCTION*****************/
double sinc(double x)
{
if(x == 0.0) return(1);
else return ((sin(x)/x));
}

/********FILTER FUNCTION*****************/
void RRCfilter(simulation_parameters sys_para, simulation_memory sys_mem){
  FILE *out;
  int t, dt, time_span, samples;
  double PI, alpha, *RRCfilt, window, RRCfilt_energy, RRCfilt_power, theory_power, top_part, low_part1, low_part3, os_factor;
  RRCfilt_energy = 0.0;
  RRCfilt_power = 0.0;
  theory_power = 0.0;
  top_part = 0.0;
  low_part1 = 0.0;
 /* low_part2 = 0.0; */
  low_part3 = 0.0;
  os_factor = 0.0;
  PI = M_PI;
  RRCfilt = sys_mem.filter_coefficients;
  alpha = sys_para.filt_roll_off;
  time_span = sys_para.filt_symbol_duration;
  samples = sys_para.num_channels;
  top_part = sys_para.num_filt_taps;
  os_factor = (log10(2.0*sys_para.over_sampling_factor)/log10(2.0));
  low_part1 = pow(sys_para.over_sampling_factor,os_factor);
/*  low_part2 = sys_para.num_data_channels*pow(2.0, (sys_para.num_data_channels-2));  */
  low_part3 = pow(2, ((os_factor-2)*(os_factor-2)))/(sys_para.num_data_channels*sys_para.num_data_channels);

  theory_power = top_part/(1.0*low_part1);


    out = fopen("c:\\Thesis\\Datafile\\filter.asc", "w+");
    if(out==NULL){
      printf("Can't open file.\n");
    }

    for(t = 0; t <= time_span*samples; t++){
        dt = (t-((time_span*samples)/2));
        RRCfilt[t] = alpha*cos(((PI*dt)/samples)+(PI/4))*sinc(((PI*alpha*dt)/samples) : (PI/4))
              + alpha*cos(((PI*dt)/samples)-(PI/4))*sinc(((PI*alpha*dt)/samples)-(PI/4))
              + (1-alpha)*sinc((PI*(1-alpha)*dt)/samples);
        window = (0.54 + 0.46*cos((PI*dt)/(time_span*samples)));      /*WINDOW FUNCTION*/
     /*   RRCfilt[t] = RRCfilt[t]*window;          */
        /*fprintf(out, "%f\n", RRCfilt[t]);*/
    }

    for(t = 0; t <= time_span*samples; t++){
        RRCfilt_energy += RRCfilt[t]*RRCfilt[t];
    }

    RRCfilt_power = RRCfilt_energy/(time_span*samples+1);

    for(t = 0; t <= time_span*samples; t++){
        RRCfilt[t] = RRCfilt[t]/sqrt(RRCfilt_power*theory_power*low_part3);
        fprintf(out, "%f\n", RRCfilt[t]);
    }


  fclose(out);
return;
}

/*************************************************************************/
void PPNfilter(simulation_parameters sys_para, simulation_memory sim_mem){
   int hori_index, vert_index, filt_index, array_index;
```

```
    for( vert_index = 0; vert_index < sys_para.num_channels; vert_index++){
     for( hori_index = 0; hori_index < (sys_para.filt_symbol_duration + 1); hori_index++){
       array_index = vert_index*(sys_para.filt_symbol_duration + 1) + hori_index;
        filt_index = vert_index + hori_index*sys_para.num_channels;
        if(filt_index < sys_para.num_ppn_taps){
          sim_mem.filter_array[array_index] = sim_mem.filter_coefficients[filt_index];
        } else  {
          sim_mem.filter_array[array_index] = 0.0;
        }
     }
    }
}

/*****************************************************************************/
```

INPUT.H

```
/*************************************************************************/
void INPUT_DATA(simulation_parameters sys_para, simulation_memory sim_mem, int index ){

  int c, d;
  d = sys_para.num_data_channels;

  for(c = 0; c < d; c++){
    if (c < d){
      switch (random(4)) {
      case 0 : sim_mem.init_qpsk[c].re = 1.0;
           sim_mem.init_qpsk[c].im = 1.0;
           break;
      case 1 : sim_mem.init_qpsk[c].re = -1.0;
           sim_mem.init_qpsk[c].im = 1.0;
           break;
      case 2 : sim_mem.init_qpsk[c].re = 1.0;
           sim_mem.init_qpsk[c].im = -1.0;
           break;
      case 3 : sim_mem.init_qpsk[c].re = -1.0;
           sim_mem.init_qpsk[c].im = -1.0;
           break;
      }
    }
  }
}

/*************************************************************************/
void input_branch_generator(simulation_parameters sys_para, simulation_memory sim_mem){

  int c, d;
  d = sys_para.num_channels;

  for(c = 0; c < d; c++){
      sim_mem.lower_branch[c].re = c;
      sim_mem.lower_branch[c].im = 1.0;
  }
}


/*************************************************************************/
void input_zero_generator(simulation_parameters sys_para, simulation_memory sim_mem ){

  int c, d;
  d = sys_para.num_channels;

  for(c = 0; c < d; c++){
      sim_mem.init_qpsk[c].re = 0.0;
      sim_mem.init_qpsk[c].im = 0.0;
  }
}

/*************************************************************************/
void input_impulse_generator(simulation_parameters sys_para, simulation_memory sim_mem, int index ){

  int c, d;
  d = sys_para.num_channels;

  for(c = 0; c < d; c++){
      sim_mem.init_qpsk[c].re = 0.0;
      sim_mem.init_qpsk[c].im = 0.0;
  }

  if (index == 0){
  sim_mem.init_qpsk[0].re = -1.0;
  sim_mem.init_qpsk[0].im = 1.0;
  }

  if (index == 3){
  sim_mem.init_qpsk[1].re = -1.0;
  sim_mem.init_qpsk[1].im = -1.0;
```

```
  }

  if (index == 5){
  sim_mem.init_qpsk[2].re = 1.0;
  sim_mem.init_qpsk[2].im = 1.0;
  }

}

/*****************************************************************************/
```

MODULATE.H

```
/*****************************************************************************/
void CLEAR_filt_mem(simulation_parameters sys_para, simulation_memory sim_mem ){
  int dash;
  complex cpx_zero;

  cpx_zero.re = 0.0;
  cpx_zero.im = 0.0;

    for(dash = 0; dash < (sys_para.num_ppn_taps + sys_para.num_channels - 1); dash++){
      sim_mem.upper_filt_mem[dash] = cpx_zero;
      sim_mem.lower_filt_mem[dash] = cpx_zero;
    }
}
/*****************************************************************************/
void CLEARmem(simulation_parameters sys_para, simulation_memory sim_mem ){

  int symbol_num;

  for (symbol_num = 0; symbol_num < sys_para.num_channels; symbol_num++){
      sim_mem.upper_branch[symbol_num].re = 0.0;
      sim_mem.upper_branch[symbol_num].im = 0.0;
      sim_mem.lower_branch[symbol_num].re = 0.0;
      sim_mem.lower_branch[symbol_num].im = 0.0;
      sim_mem.mod_out[symbol_num].re = 0.0;
      sim_mem.mod_out[symbol_num].im = 0.0;
  }
}

/*****************************************************************************/
void DIFFencode(simulation_parameters sys_para, simulation_memory sim_mem ){

  int dash;
  double scale_factor;
  complex data_word, prev_word;

  scale_factor = sqrt(0.5);

  sim_mem.encode_qpsk[0] = sim_mem.init_qpsk[0];
  for (dash = 1; dash < sys_para.num_data_channels; dash++){
      data_word = sim_mem.init_qpsk[dash];
      prev_word.re = scale_factor*sim_mem.encode_qpsk[dash-1].re;
      prev_word.im = scale_factor*sim_mem.encode_qpsk[dash-1].im;
      sim_mem.encode_qpsk[dash] = multiply_cpx(prev_word, data_word);
  }
}
/*****************************************************************************/
void DUMMYencode(simulation_parameters sys_para, simulation_memory sim_mem ){

  int dash;

  for (dash = 0; dash < sys_para.num_data_channels; dash++){
      sim_mem.encode_qpsk[dash] = sim_mem.init_qpsk[dash];
  }
}


/*****************************************************************************/
void COHencode(simulation_parameters sys_para, simulation_memory sim_mem ){

  int num_blocks, block_size, dash;
  complex pilot_symbol;

  num_blocks = sys_para.pilots/2;
  block_size = sys_para.num_data_channels/num_blocks;

  pilot_symbol.re = 1.0;
  pilot_symbol.im = 1.0;


  for (dash = 0; dash < sys_para.num_data_channels; dash++){
```

```
    sim_mem.encode_qpsk[dash] = sim_mem.init_qpsk[dash];
  }

  for (dash = 0; dash < sys_para.num_data_channels; dash++){
    if ( dash%block_size == 0 ) {
       sim_mem.encode_qpsk[dash] = pilot_symbol;
      /*  sim_mem.encode_qpsk[dash + block_size - 1] = pilot_symbol;    */
       sim_mem.encode_qpsk[dash + block_size - 1].re = 1.0;
       sim_mem.encode_qpsk[dash + block_size - 1].im = 1.0;
    }
  }
}

/*************************************************************************/
void PREprocessing(simulation_parameters sys_para, simulation_memory sim_mem ){

  int symbol_num;

  for (symbol_num = 0; symbol_num < sys_para.num_data_channels; symbol_num++){
    /*sim_mem.upper_branch[symbol_num].re = sim_mem.init_qpsk[symbol_num].re;
      sim_mem.upper_branch[symbol_num].im = sim_mem.init_qpsk[symbol_num].im;
      sim_mem.lower_branch[symbol_num].re = sim_mem.init_qpsk[symbol_num].re;
      sim_mem.lower_branch[symbol_num].im = sim_mem.init_qpsk[symbol_num].im;
      */
    if (symbol_num%2 == 0){
      sim_mem.upper_branch[symbol_num].re = 0.0;
      sim_mem.upper_branch[symbol_num].im = sim_mem.encode_qpsk[symbol_num].im;
      sim_mem.lower_branch[symbol_num].re = 1.0*sim_mem.encode_qpsk[symbol_num].re;
      sim_mem.lower_branch[symbol_num].im = 0.0;
    } else if (symbol_num%2 != 0){
      sim_mem.upper_branch[symbol_num].re = sim_mem.encode_qpsk[symbol_num].re;
      sim_mem.upper_branch[symbol_num].im = 0.0;
      sim_mem.lower_branch[symbol_num].re = 0.0;
      sim_mem.lower_branch[symbol_num].im = -1.0*sim_mem.encode_qpsk[symbol_num].im;
    }
  }
}


/*************************************************************************/
/*****
/*****                  Full DFT Program
/*****
/*****  This program/function accepts a complex array of known length
/*****    N, and returns the un-normalised DFT of the same array.
/*****    The algorthim applied to perform the DFT is a basic
/*****              radix 2 DIF Cooley-Tukey.
/*****
/*************************************************************************/


void DFT( simulation_parameters sys_para, complex *array, double IDFT) {

  int M, N2, s, j, i, index, bit_num, p, N;
  double PI;
  N = sys_para.num_channels;
  PI = M_PI;
  M = ceil(log(N)/log(2.0));              /** Number of flow graph stages **/
  N2 = N;

    for (s = 1; s <= M; s++ ){
      int N1, S;
      double E;
      N1 = N2;
      N2 = N2/2;
      E = (2*PI)/N1;
      S = N/N1;
        for(j = 0; j < N2; j++){
          double A, tfc, tfs;
          A = j*E;
          tfc = cos(A);
          tfs = IDFT*sin(A);
            for(i = 0; i < S; i++){
```

```
            int k, l;
            double XT, YT;
            k = i*N1 + j;
            l = k + N2;
            XT = array[k].re - array[l].re;
            array[k].re = array[k].re + array[l].re;
            YT = array[k].im - array[l].im;
            array[k].im = array[k].im + array[l].im;
            array[l].re = tfc*XT + tfs*YT;
            array[l].im = tfc*YT - tfs*XT;
          }
        }
    }
/*****************************************************************************/
/*****   This section sorts the DFT signal flow graph complex output     *****/
/*****    array produced above, into the correct order for output.       *****/
/*****      It is based on a straightforward bit-reversal                *****/
/*****    i.e. array location 10100011 goes to 11000101.(check??)        *****/
/*****************************************************************************/

        for(index = 0; index < N; index++){
          complex temp;
          int index1, M;
          index1 = index;
          M = ceil(log(N)/log(2.0));
            for(bit_num = 0; bit_num < (int) M/2; bit_num++){
                int bit_value1, bit_value2;
                bit_value1 = (int) pow(2, bit_num)&index;
                bit_value2 = (int) pow(2, (M-1-bit_num))&index;
                index1 = index1 - (bit_value1 + bit_value2);
                bit_value1 = bit_value1 << (M-(bit_num*2+1));
                bit_value2 = bit_value2 >> (M-(bit_num*2+1));
                index1 = index1 + (bit_value1 + bit_value2);
            }
            if ( index1 > index ){
                temp.re = array[index].re;
                temp.im = array[index].im;
                array[index].re = (array[index1].re);
                array[index].im = (array[index1].im);
                array[index1].re = (temp.re);
                array[index1].im = (temp.im);
            } else {
                array[index1].re = (array[index1].re);
                array[index1].im = (array[index1].im);
            }

        }


    if(IDFT == 1.0){
        for(p = 0; p < N; p++){
          array[p].re = (array[p].re)/(1.0*N);
          array[p].im = (array[p].im)/(1.0*N);
        }
    } else {
        for(p = 0; p < N; p++){
          array[p].re = (array[p].re);
          array[p].im = (array[p].im);
        }
    }

}


/*****************************************************************************/
void DFTprocessing(simulation_parameters sys_para, simulation_memory sim_mem ){

  DFT( sys_para, sim_mem.upper_branch, 1.0);
  DFT( sys_para, sim_mem.lower_branch, 1.0);

}
```

```
/*************************************************************************/
void DATAshuffle(simulation_parameters sys_para, simulation_memory sim_mem){

  int filt_coef, filt_size, chan_num, num_chan;

  filt_size = sys_para.filt_symbol_duration + 1;
  num_chan = sys_para.num_channels;

  for( chan_num = 0; chan_num < num_chan; chan_num++){
    for( filt_coef = (filt_size-1); filt_coef > 0; filt_coef--){
      sim_mem.upper_filt_mem[filt_coef + chan_num*filt_size].re = sim_mem.upper_filt_mem[filt_coef + chan_num*filt_size -
1].re;
      sim_mem.upper_filt_mem[filt_coef + chan_num*filt_size].im = sim_mem.upper_filt_mem[filt_coef + chan_num*filt_size -
1].im;
      sim_mem.lower_filt_mem[filt_coef + chan_num*filt_size].re = sim_mem.lower_filt_mem[filt_coef + chan_num*filt_size -
1].re;
      sim_mem.lower_filt_mem[filt_coef + chan_num*filt_size].im = sim_mem.lower_filt_mem[filt_coef + chan_num*filt_size -
1].im;
    }
    sim_mem.upper_filt_mem[chan_num*filt_size].re = sim_mem.upper_branch[chan_num].re;
    sim_mem.upper_filt_mem[chan_num*filt_size].im = sim_mem.upper_branch[chan_num].im;
    sim_mem.lower_filt_mem[chan_num*filt_size].re = sim_mem.lower_branch[chan_num].re;
    sim_mem.lower_filt_mem[chan_num*filt_size].im = sim_mem.lower_branch[chan_num].im;
  }
}

/*************************************************************************/
void CONVppn(simulation_parameters sys_para, simulation_memory sim_mem){

  int dash, number, array_loc;
  complex complex_zero, upper_out, lower_out;

  complex_zero.re = 0.0;
  complex_zero.im = 0.0;

  for(dash = 0; dash < sys_para.num_channels; dash++){
    upper_out = complex_zero;
    lower_out = complex_zero;
    for(number = 0; number < (sys_para.filt_symbol_duration+1); number++){
      array_loc = number + dash*(sys_para.filt_symbol_duration + 1);
      upper_out.re += sim_mem.upper_filt_mem[array_loc].re*sim_mem.filter_array[array_loc];
      upper_out.im += sim_mem.upper_filt_mem[array_loc].im*sim_mem.filter_array[array_loc];
      lower_out.re += sim_mem.lower_filt_mem[array_loc].re*sim_mem.filter_array[array_loc];
      lower_out.im += sim_mem.lower_filt_mem[array_loc].im*sim_mem.filter_array[array_loc];
    }
    sim_mem.upper_branch[dash].re = upper_out.re;
    sim_mem.upper_branch[dash].im = upper_out.im;
    sim_mem.lower_branch[dash].re = lower_out.re;
    sim_mem.lower_branch[dash].im = lower_out.im;
  }
}


/*************************************************************************/
void POSTprocessing(simulation_parameters sys_para, simulation_memory sim_mem ){

  int index, half_way;

  half_way = (int) (0.5*sys_para.num_channels);


  for( index = 0; index < sys_para.num_channels; index++){
    sim_mem.old_mod_out[index].re = sim_mem.mod_out[index].re;
    sim_mem.old_mod_out[index].im = sim_mem.mod_out[index].im;
  }


  for( index = 0; index < half_way; index++){
    sim_mem.mod_out[index].re = sim_mem.upper_branch[index].re + sim_mem.prev_lower_branch[index+half_way].re;
    sim_mem.mod_out[index].im = sim_mem.upper_branch[index].im + sim_mem.prev_lower_branch[index+half_way].im;
  }
```

```
  for( index = half_way; index < sys_para.num_channels; index++){
    sim_mem.mod_out[index].re = sim_mem.upper_branch[index].re + sim_mem.lower_branch[index-half_way].re;
    sim_mem.mod_out[index].im = sim_mem.upper_branch[index].im + sim_mem.lower_branch[index-half_way].im;;
  }

  for( index = 0; index < sys_para.num_channels; index++){
    sim_mem.prev_lower_branch[index].re = sim_mem.lower_branch[index].re;
    sim_mem.prev_lower_branch[index].im = sim_mem.lower_branch[index].im;
  }
}

/*****************************************************************************/

void MODULATOR(simulation_parameters sys_para, simulation_memory sim_mem ){

  int dash;

  COHencode( sys_para, sim_mem );

  /*** RUBBISH PRINTING ***
  printf("\n\n Init QPSK  *** ");
  for(dash = 0; dash < sys_para.num_data_channels; dash++){
    printf("%2.0lf/%2.0lf ", (sim_mem.init_qpsk[dash].re), (sim_mem.init_qpsk[dash].im));
  }
  printf("\n Encode QPSK *** ");
  for(dash = 0; dash < sys_para.num_data_channels; dash++){
    printf("%2.0lf/%2.0lf ", (sim_mem.encode_qpsk[dash].re), (sim_mem.encode_qpsk[dash].im));
  }
  getch();

  /*** END OF PRINT ***/
  PREprocessing( sys_para, sim_mem );
  DFTprocessing(sys_para, sim_mem);
  DATAshuffle( sys_para, sim_mem);
  CONVppn(sys_para, sim_mem);
  POSTprocessing(sys_para, sim_mem);
}


/*****************************************************************************/

void COH_MOD(simulation_parameters sys_para, simulation_memory sim_mem ){

  COHencode( sys_para, sim_mem );
  PREprocessing( sys_para, sim_mem );
  DFTprocessing(sys_para, sim_mem);
  DATAshuffle( sys_para, sim_mem);
  CONVppn(sys_para, sim_mem);
  POSTprocessing(sys_para, sim_mem);
}

/*****************************************************************************/

void DIFF_MOD(simulation_parameters sys_para, simulation_memory sim_mem ){

  DIFFencode( sys_para, sim_mem );
  PREprocessing( sys_para, sim_mem );
  DFTprocessing(sys_para, sim_mem);
  DATAshuffle( sys_para, sim_mem);
  CONVppn(sys_para, sim_mem);
  POSTprocessing(sys_para, sim_mem);
}
```

```
/*****************************************************************************/

void SIM_MOD(simulation_parameters sys_para, simulation_memory sim_mem ){

  DUMMYencode( sys_para, sim_mem );
  PREprocessing( sys_para, sim_mem );
  DFTprocessing(sys_para, sim_mem);
  DATAshuffle( sys_para, sim_mem);
  CONVppn(sys_para, sim_mem);
  POSTprocessing(sys_para, sim_mem);
}

/*****************************************************************************/
void PrintSystemMemory(simulation_parameters sys_para, simulation_memory sim_mem){

  int dash, number;

 /* CLEARmem(sys_para, sim_mem);              */
 /* input_impulse_generator(sys_para, sim_mem);  */
  PREprocessing( sys_para, sim_mem );

  printf("\n\n******* Init QPSK ***************\n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.init_qpsk[dash].re,sim_mem.init_qpsk[dash].im);
  }
  printf("\n Upper Branch \n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.upper_branch[dash].re, sim_mem.upper_branch[dash].im);
  }
  printf("\n Lower Branch \n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.lower_branch[dash].re, sim_mem.lower_branch[dash].im);
  }

  DFTprocessing(sys_para, sim_mem);
  printf("\n Upper Branch \n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.upper_branch[dash].re, sim_mem.upper_branch[dash].im);
  }
  printf("\n Lower Branch \n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.lower_branch[dash].re, sim_mem.lower_branch[dash].im);
  }

  DATAshuffle( sys_para, sim_mem);

  printf("\n Upper Filter Memory");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("\n");
    for(number = 0; number < (sys_para.filt_symbol_duration+1); number++){
      printf("(%2.1lf,    %2.1lf)*(%2.2lf)    ",    sim_mem.upper_filt_mem[number+dash*(sys_para.filt_symbol_duration+1)].re,
sim_mem.upper_filt_mem[number+dash*(sys_para.filt_symbol_duration+1)].im,
sim_mem.filter_array[number+dash*(sys_para.filt_symbol_duration+1)]);
    }
  }
  printf("\n Lower Filter Memory");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("\n");
    for(number = 0; number < (sys_para.filt_symbol_duration+1); number++){
      printf("(%2.1lf,    %2.1lf)*(%2.2lf)    ",    sim_mem.lower_filt_mem[number+dash*(sys_para.filt_symbol_duration+1)].re,
sim_mem.lower_filt_mem[number+dash*(sys_para.filt_symbol_duration+1)].im,
sim_mem.filter_array[number+dash*(sys_para.filt_symbol_duration+1)]);
    }
  }

  CONVppn(sys_para, sim_mem);

  printf("\n Upper Branch \n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.upper_branch[dash].re, sim_mem.upper_branch[dash].im);
  }
  printf("\n Lower Branch \n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
```

```
    printf("(%2.2lf, %2.2lf) ", sim_mem.lower_branch[dash].re, sim_mem.lower_branch[dash].im);
  }

  POSTprocessing(sys_para, sim_mem);
  printf("\n MOd Out\n");
  for(dash = 0; dash < sys_para.num_channels; dash++){
    printf("(%2.2lf, %2.2lf) ", sim_mem.mod_out[dash].re, sim_mem.mod_out[dash].im);
  }

}
/*****************************************************************************/
```

CHANNEL.H

```
/*****************************************************************/

void CHANNEL_AWGN( simulation_parameters sys_para, simulation_memory sim_mem, double sigma){

  int dash;

  for(dash = 0; dash < sys_para.num_channels; dash++){
    sim_mem.mod_out[dash].re += RanGauss( sigma );
    sim_mem.mod_out[dash].im += RanGauss( sigma );
  }

}


/*****************************************************************/
/**                                                             **/
/**   jakes_function_array.c                                    **/
/**                                                             **/
/**   Date  :   31-10-98                                        **/
/**   Author :   D. Bhatoolaul                                  **/
/**                                                             **/
/**   int j_nzero        :  Num. of Jakes Oscillators           **/
/**   double j_alpha     :  Jakes Phase Offset                  **/
/**   double doppler     :  Doppler Frequency                   **/
/**   long double start_time:  Start Time                       **/
/**   complex* fading_taps  :  Output Array(pointer)            **/
/**   int array_length   :  Array Length                        **/
/**   double sample_period  :  Output Sample Period             **/
/**                                                             **/
/**                                                             **/
/*****************************************************************/

void jakes_function_array(int j_nzero, double j_alpha, double j_independent, double doppler_freq, long double start_time,
complex *jakes_fading_taps, int array_length, double sample_period){

  int j_n, j1, j2;
  double beta, gamma, j_time, j_doppler;
  complex jakes_out;

  j_doppler = 2.0*M_PI*(doppler_freq);
  j_n = 2*(2*j_nzero+1);
  beta = 0.0;
  gamma = 0.0;
  j2 = 0;
  j_time = start_time;

  while (j2 < array_length){
    jakes_out.re = 0.0;
    jakes_out.im = 0.0;
      for(j1 = 1; j1 < (j_nzero+1); j1++){
      beta =  (M_PI*j1/(1.0*j_nzero)) + ((2.0*M_PI*(j_independent-1.0))/(1.0*j_nzero));
      gamma = cos((2.0*M_PI*j1)/(1.0*j_n));
      jakes_out.re += cos(beta)*cos(gamma*j_doppler*j_time);
      jakes_out.im += sin(beta)*cos(gamma*j_doppler*j_time);
      }
    jakes_fading_taps[j2].re                    =                    ((2.0*jakes_out.re)                    +
(sqrt(2.0)*cosl(j_doppler*j_time)*cos(j_alpha)))/(sqrt(2.0*(j_nzero+(1.0/sqrt(2.0)))));
    jakes_fading_taps[j2].im                    =                    ((2.0*jakes_out.im)                    +
(sqrt(2.0)*sinl(j_doppler*j_time)*sin(j_alpha)))/(sqrt(2.0*(j_nzero+(1.0/sqrt(2.0)))));
    j2 += 1;
    j_time += sample_period;
  }
}
```

```
/*****************************************************************/
/**                                                             **/
/**   int j_nzero        :  Num. of Jakes Oscillators          **/
/**   double j_alpha     :  Jakes Phase Offset                 **/
/**   double doppler     :  Doppler Frequency                  **/
/**   long double start_time:  Start Time                      **/
/**   complex* fading_taps :  Output Array(pointer)            **/
/**   int array_length   :  Array Length                       **/
/**   double sample_period :  Output Sample Period             **/
/**                                                             **/
/*****************************************************************/
/*****************************************************************/
/*****************************************************************/

void CHANNEL_FADING( simulation_parameters sys_para, simulation_memory sim_mem, double sigma){

  int dash, num_osc, array_length;
  double doppler_freq, jakes_alpha_offset, jakes_independent_factor, sample_period;
  long double start_time;
  complex gauss_noise, conj_fading_tap, fading_tap;

  num_osc = sys_para.num_oscillators;
  jakes_alpha_offset = sys_para.jakes_phase1;
  jakes_independent_factor = sys_para.independent1;
  start_time = sim_mem.fft_clock[0]*sys_para.true_symbol_period;
  array_length = sys_para.num_channels;
  sample_period = sys_para.true_sample_period;
  doppler_freq = sys_para.dop_sym_ratio1*sys_para.true_symbol_period;

  jakes_function_array(num_osc,  jakes_alpha_offset,  jakes_independent_factor,  doppler_freq,  start_time,
sim_mem.fading_taps1, array_length, sample_period);

  for(dash = 0; dash < sys_para.num_channels; dash++){
    fading_tap = sim_mem.fading_taps1[dash];
    gauss_noise.re =  RanGauss( sigma );
    gauss_noise.im =  RanGauss( sigma );
    conj_fading_tap.re = fading_tap.re;
    conj_fading_tap.im = -1.0*fading_tap.im;
    /******** Fading Channel **********************/
    sim_mem.mod_out[dash] = multiply_cpx(sim_mem.mod_out[dash], fading_tap);
    sim_mem.mod_out[dash] = add_cpx(sim_mem.mod_out[dash], gauss_noise);
    /******** Perfect Channel Estimation ***********/
 /*   sim_mem.mod_out[dash] = multiply_cpx(sim_mem.mod_out[dash], conj_fading_tap);    */

  }

}

/*****************************************************************/
void CHANNEL_2_PATH( simulation_parameters sys_para, simulation_memory sim_mem, double sigma){

  int dash, num_osc, array_length;
  double  doppler_freq1,  jakes_alpha_offset1,  jakes_independent_factor1,  doppler_freq2,  jakes_alpha_offset2,
jakes_independent_factor2, path_factor1, path_factor2;
  long double start_time;
  complex gauss_noise, conj_fading_tap1, fading_tap1, conj_fading_tap2, fading_tap2;
  complex temp_path1, temp_path2;

  num_osc = sys_para.num_oscillators;
  array_length = sys_para.num_channels;
  start_time = sim_mem.fft_clock[0]*sys_para.true_symbol_period;
  jakes_alpha_offset1 = sys_para.jakes_phase1;
  jakes_alpha_offset2 = sys_para.jakes_phase2;
  jakes_independent_factor1 = sys_para.independent1;
  jakes_independent_factor2 = sys_para.independent2;
  doppler_freq1 = sys_para.dop_sym_ratio1*sys_para.true_symbol_period;
  doppler_freq2 = sys_para.dop_sym_ratio2*sys_para.true_symbol_period;
  path_factor1 = (sys_para.power1);
  path_factor2 = (sys_para.power2);

  jakes_function_array(num_osc,  jakes_alpha_offset1,  jakes_independent_factor1,  doppler_freq1,  start_time,
sim_mem.fading_taps1, array_length, sys_para.true_sample_period);
```

```
  jakes_function_array(num_osc,  jakes_alpha_offset2,  jakes_independent_factor2,  doppler_freq2,  start_time,
sim_mem.fading_taps2, array_length, sys_para.true_sample_period);

  for(dash = 0; dash < sys_para.num_channels; dash++){
    fading_tap1 = sim_mem.fading_taps1[dash];
    fading_tap2 = sim_mem.fading_taps2[dash];
    gauss_noise.re =  RanGauss( sigma );
    gauss_noise.im =  RanGauss( sigma );
    conj_fading_tap1.re = fading_tap1.re;
    conj_fading_tap1.im = -1.0*fading_tap1.im;
    conj_fading_tap2.re = fading_tap2.re;
    conj_fading_tap2.im = -1.0*fading_tap2.im;
    /******** Fading Channel **********************/
    temp_path1 = scalar_multiply_cpx( sim_mem.mod_out[dash], path_factor1 );
    temp_path1 = multiply_cpx( temp_path1, fading_tap1);

    if ((dash - sys_para.path_diff) < 0) {
     temp_path2  =  scalar_multiply_cpx(  sim_mem.old_mod_out[(sys_para.num_channels-sys_para.path_diff+dash)],
path_factor2 );
     temp_path2 = multiply_cpx( temp_path2, fading_tap2);
    } else if ((dash - sys_para.path_diff) >= 0) {
     temp_path2 = scalar_multiply_cpx( sim_mem.mod_out[dash - sys_para.path_diff], path_factor2 );
     temp_path2 = multiply_cpx( temp_path2, fading_tap2);
    }


    temp_path1 = add_cpx(temp_path1, temp_path2);
    sim_mem.mod_out[dash] = add_cpx(temp_path1, gauss_noise);

    /******** Perfect Channel Estimation ***********/
 /*   sim_mem.mod_out[dash] = multiply_cpx(sim_mem.mod_out[dash], conj_fading_tap1);    */


  }

}

/*********************************************************************/
```

DEMODULATE.H

```
/*************************************************************************/
void CLEAR_demod(simulation_parameters sys_para, demodulator_memory demod_mem){
  int dash;

  for(dash = 0; dash < sys_para.num_data_channels; dash++){
    demod_mem.beta0_branch[dash].re = 0.0;
    demod_mem.beta0_branch[dash].im = 0.0;
    demod_mem.beta1_branch[dash].re = 0.0;
    demod_mem.beta1_branch[dash].im = 0.0;
    demod_mem.beta2_branch[dash].re = 0.0;
    demod_mem.beta2_branch[dash].im = 0.0;
    demod_mem.beta3_branch[dash].re = 0.0;
    demod_mem.beta3_branch[dash].im = 0.0;
  }
}

/*************************************************************************/
void CLEAR_quad_buffer(simulation_parameters sys_para, demodulator_memory demod_mem){

 int dash, number;

 for(dash = 0; dash < sys_para.num_data_channels; dash++){
    demod_mem.quad_buffer[dash].re = 0.0;
    demod_mem.quad_buffer[dash].im = 0.0;
 }

 for(number = 0; number < (sys_para.filt_symbol_duration+1);number++){
  for(dash = 0; dash < sys_para.num_data_channels; dash++){
    demod_mem.reference_qpsk[dash + number*sys_para.num_data_channels].re = 0.0;
    demod_mem.reference_qpsk[dash + number*sys_para.num_data_channels].im = 0.0;
  }
 }
}

/*************************************************************************/
void CHAN_info( simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem ){

  int dash, number, tag;

  /*********************************************/
  /*** Shuffle mod_out memory ready for use ***/
  /*********************************************/
    for(dash = 0; dash < sys_para.num_channels; dash++){
    demod_mem.quad_buffer[4*sys_para.num_channels - 1 - dash].re = demod_mem.quad_buffer[3*sys_para.num_channels -
1 - dash].re;
    demod_mem.quad_buffer[4*sys_para.num_channels - 1 - dash].im = demod_mem.quad_buffer[3*sys_para.num_channels
- 1 - dash].im;
    }

    for(dash = 0; dash < sys_para.num_channels; dash++){
    demod_mem.quad_buffer[3*sys_para.num_channels - 1 - dash].re = demod_mem.quad_buffer[2*sys_para.num_channels -
1 - dash].re;
    demod_mem.quad_buffer[3*sys_para.num_channels - 1 - dash].im = demod_mem.quad_buffer[2*sys_para.num_channels
- 1 - dash].im;
    }

    for(dash = 0; dash < sys_para.num_channels; dash++){
    demod_mem.quad_buffer[2*sys_para.num_channels - 1 - dash].re = demod_mem.quad_buffer[1*sys_para.num_channels -
1 - dash].re;
    demod_mem.quad_buffer[2*sys_para.num_channels - 1 - dash].im = demod_mem.quad_buffer[1*sys_para.num_channels
- 1 - dash].im;
    }

    for(dash = 0; dash < sys_para.num_channels; dash++){
    demod_mem.quad_buffer[dash].re = sim_mem.mod_out[sys_para.num_channels - 1 - dash].re;
    demod_mem.quad_buffer[dash].im = sim_mem.mod_out[sys_para.num_channels - 1 - dash].im;
    }
```

```
  /****** Shuffle init_qpsk ready for use *****/


  for(number = sys_para.filt_symbol_duration; number > 0; number--){
  tag = sys_para.num_data_channels*number;
    for(dash = 0; dash < sys_para.num_data_channels; dash++){
    demod_mem.reference_qpsk[tag + dash] = demod_mem.reference_qpsk[tag - sys_para.num_data_channels + dash];
    }
  }

  for(dash = 0; dash < sys_para.num_data_channels; dash++){
    demod_mem.reference_qpsk[dash].re = sim_mem.init_qpsk[dash].re;
    demod_mem.reference_qpsk[dash].im = sim_mem.init_qpsk[dash].im;
  }

}


/*************************************************************************/
void DEMOD_pre( simulation_parameters sys_para, demodulator_memory demod_mem ){

  int dash,  tag0, tag1, tag2, tag3;

/* jit0 = 2;
  jit1 = demod_mem.jitter + (int) (0.25*sys_para.num_channels) ;
  jit2 = demod_mem.jitter + (int) (0.5*sys_para.num_channels);
  jit3 = demod_mem.jitter + (int) (0.75*sys_para.num_channels);     */

  tag0 =  (2*sys_para.num_channels - 1) - 0.5*sys_para.num_channels;
  tag1 =  (2*sys_para.num_channels - 1) - 0.75*sys_para.num_channels;
  tag2 =  (2*sys_para.num_channels - 1) - 0.0*sys_para.num_channels;
  tag3 =  (2*sys_para.num_channels - 1) - 0.25*sys_para.num_channels;

  for(dash = 1; dash < sys_para.num_channels; dash++){
  demod_mem.beta0_branch[sys_para.num_channels - dash].re = demod_mem.quad_buffer[tag0 - dash].re;
  demod_mem.beta0_branch[sys_para.num_channels - dash].im = demod_mem.quad_buffer[tag0 - dash].im;
  demod_mem.beta1_branch[sys_para.num_channels - dash].re = demod_mem.quad_buffer[tag1 - dash].re;
  demod_mem.beta1_branch[sys_para.num_channels - dash].im = demod_mem.quad_buffer[tag1 - dash].im;
  demod_mem.beta2_branch[sys_para.num_channels - dash].re = demod_mem.quad_buffer[tag2 - dash].re;
  demod_mem.beta2_branch[sys_para.num_channels - dash].im = demod_mem.quad_buffer[tag2 - dash].im;
  demod_mem.beta3_branch[sys_para.num_channels - dash].re = demod_mem.quad_buffer[tag3 - dash].re;
  demod_mem.beta3_branch[sys_para.num_channels - dash].im = demod_mem.quad_buffer[tag3 - dash].im;
  }
  demod_mem.beta0_branch[0].re      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.5*sys_para.num_channels].re;
  demod_mem.beta0_branch[0].im      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.5*sys_para.num_channels].im;
  demod_mem.beta1_branch[0].re      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.75*sys_para.num_channels].re;
  demod_mem.beta1_branch[0].im      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.75*sys_para.num_channels].im;
  demod_mem.beta2_branch[0].re      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.0*sys_para.num_channels].re;
  demod_mem.beta2_branch[0].im      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.0*sys_para.num_channels].im;
  demod_mem.beta3_branch[0].re      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.25*sys_para.num_channels].re;
  demod_mem.beta3_branch[0].im      =      demod_mem.quad_buffer[(1*sys_para.num_channels      -      1)      -
0.25*sys_para.num_channels].im;
}

/*************************************************************************/
void DEMOD_datashuffle(simulation_parameters sys_para, demodulator_memory demod_mem){

  int filt_coef, filt_size, chan_num, num_chan;

  filt_size = sys_para.filt_symbol_duration + 1;
  num_chan = sys_para.num_channels;

  for( chan_num = 0; chan_num < num_chan; chan_num++){
    for( filt_coef = (filt_size-1); filt_coef > 0; filt_coef--){
      demod_mem.beta0_filt_mem[filt_coef      +      chan_num*filt_size].re      =      demod_mem.beta0_filt_mem[filt_coef      +
chan_num*filt_size - 1].re;
```

```
      demod_mem.beta0_filt_mem[filt_coef + chan_num*filt_size].im = demod_mem.beta0_filt_mem[filt_coef +
chan_num*filt_size - 1].im;
      demod_mem.beta1_filt_mem[filt_coef + chan_num*filt_size].re = demod_mem.beta1_filt_mem[filt_coef +
chan_num*filt_size - 1].re;
      demod_mem.beta1_filt_mem[filt_coef + chan_num*filt_size].im = demod_mem.beta1_filt_mem[filt_coef +
chan_num*filt_size - 1].im;
      demod_mem.beta2_filt_mem[filt_coef + chan_num*filt_size].re = demod_mem.beta2_filt_mem[filt_coef +
chan_num*filt_size - 1].re;
      demod_mem.beta2_filt_mem[filt_coef + chan_num*filt_size].im = demod_mem.beta2_filt_mem[filt_coef +
chan_num*filt_size - 1].im;
      demod_mem.beta3_filt_mem[filt_coef + chan_num*filt_size].re = demod_mem.beta3_filt_mem[filt_coef +
chan_num*filt_size - 1].re;
      demod_mem.beta3_filt_mem[filt_coef + chan_num*filt_size].im = demod_mem.beta3_filt_mem[filt_coef +
chan_num*filt_size - 1].im;
    }
    demod_mem.beta0_filt_mem[chan_num*filt_size].re = demod_mem.beta0_branch[chan_num].re;
    demod_mem.beta0_filt_mem[chan_num*filt_size].im = demod_mem.beta0_branch[chan_num].im;
    demod_mem.beta1_filt_mem[chan_num*filt_size].re = demod_mem.beta1_branch[chan_num].re;
    demod_mem.beta1_filt_mem[chan_num*filt_size].im = demod_mem.beta1_branch[chan_num].im;
    demod_mem.beta2_filt_mem[chan_num*filt_size].re = demod_mem.beta2_branch[chan_num].re;
    demod_mem.beta2_filt_mem[chan_num*filt_size].im = demod_mem.beta2_branch[chan_num].im;
    demod_mem.beta3_filt_mem[chan_num*filt_size].re = demod_mem.beta3_branch[chan_num].re;
    demod_mem.beta3_filt_mem[chan_num*filt_size].im = demod_mem.beta3_branch[chan_num].im;
  }
}

/*****************************************************************************/
void DEMOD_ppnconv(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem){

  int dash, number, array_loc;
  complex complex_zero, beta0_zero, beta1_zero, beta2_zero, beta3_zero;

  complex_zero.re = 0.0;
  complex_zero.im = 0.0;

  for(dash = 0; dash < sys_para.num_channels; dash++){
    beta0_zero = complex_zero;
    beta1_zero = complex_zero;
    beta2_zero = complex_zero;
    beta3_zero = complex_zero;
    for(number = 0; number < (sys_para.filt_symbol_duration+1); number++){
      array_loc = number + dash*(sys_para.filt_symbol_duration + 1);
      beta0_zero.re += demod_mem.beta0_filt_mem[array_loc].re*sim_mem.filter_array[array_loc];
      beta0_zero.im += demod_mem.beta0_filt_mem[array_loc].im*sim_mem.filter_array[array_loc];
      beta1_zero.re += demod_mem.beta1_filt_mem[array_loc].re*sim_mem.filter_array[array_loc];
      beta1_zero.im += demod_mem.beta1_filt_mem[array_loc].im*sim_mem.filter_array[array_loc];
      beta2_zero.re += demod_mem.beta2_filt_mem[array_loc].re*sim_mem.filter_array[array_loc];
      beta2_zero.im += demod_mem.beta2_filt_mem[array_loc].im*sim_mem.filter_array[array_loc];
      beta3_zero.re += demod_mem.beta3_filt_mem[array_loc].re*sim_mem.filter_array[array_loc];
      beta3_zero.im += demod_mem.beta3_filt_mem[array_loc].im*sim_mem.filter_array[array_loc];
    }
    demod_mem.beta0_branch[dash].re = beta0_zero.re;
    demod_mem.beta0_branch[dash].im = beta0_zero.im;
    demod_mem.beta1_branch[dash].re = beta1_zero.re;
    demod_mem.beta1_branch[dash].im = beta1_zero.im;
    demod_mem.beta2_branch[dash].re = beta2_zero.re;
    demod_mem.beta2_branch[dash].im = beta2_zero.im;
    demod_mem.beta3_branch[dash].re = beta3_zero.re;
    demod_mem.beta3_branch[dash].im = beta3_zero.im;
  }
}


/*****************************************************************************/
void DEMOD_dft(simulation_parameters sys_para, demodulator_memory demod_mem){

  DFT( sys_para, demod_mem.beta0_branch, -1.0);
  DFT( sys_para, demod_mem.beta1_branch, -1.0);
  DFT( sys_para, demod_mem.beta2_branch, -1.0);
  DFT( sys_para, demod_mem.beta3_branch, -1.0);
}
```

```
/*****************************************************************************/
void DEMOD_post(simulation_parameters sys_para, demodulator_memory demod_mem){

  int dash;
  complex factor0, factor1, factor2, factor3;

  for (dash = 0; dash < sys_para.num_channels ;dash++){
  factor0.re = cos(2.0*M_PI*dash*0.5);
  factor0.im = sin(2.0*M_PI*dash*0.5);
  factor1.re = cos(2.0*M_PI*dash*0.25);
  factor1.im = sin(2.0*M_PI*dash*0.25);
  factor2.re = cos(2.0*M_PI*dash*0.0);
  factor2.im = sin(2.0*M_PI*dash*0.0);
  factor3.re = cos(2.0*M_PI*dash*0.75);
  factor3.im = sin(2.0*M_PI*dash*0.75);
  demod_mem.beta0_branch[dash] = multiply_cpx( factor0, demod_mem.beta0_branch[dash]);
  demod_mem.beta1_branch[dash] = multiply_cpx( factor1, demod_mem.beta1_branch[dash]);
  demod_mem.beta2_branch[dash] = multiply_cpx( factor2, demod_mem.beta2_branch[dash]);
  demod_mem.beta3_branch[dash] = multiply_cpx( factor3, demod_mem.beta3_branch[dash]);
  }
}

/*****************************************************************************/
void DEMOD_out(simulation_parameters sys_para, demodulator_memory demod_mem){

  int dash;

  for(dash = 0; dash < sys_para.num_channels; dash++){
    if (dash == 0){
        demod_mem.demod_out[dash].re = demod_mem.beta0_branch[dash].re;
        demod_mem.demod_out[dash].im = demod_mem.beta2_branch[dash].im;
    } else {
         if( (dash%2) == 0 ){
        demod_mem.demod_out[sys_para.num_channels - dash].re = demod_mem.beta0_branch[dash].re;
        demod_mem.demod_out[sys_para.num_channels - dash].im = demod_mem.beta2_branch[dash].im;
         }
         else if( (dash%2) == 1 ){
        demod_mem.demod_out[sys_para.num_channels - dash].im = demod_mem.beta0_branch[dash].im;
        demod_mem.demod_out[sys_para.num_channels - dash].re = demod_mem.beta2_branch[dash].re;
         }
    }

  }

}

/*****************************************************************************/
void DEMOD_scale(simulation_parameters sys_para, demodulator_memory demod_mem){

   int dash;
   double scale_factor;

   scale_factor = 2.0/sys_para.num_data_channels;

   for (dash = 0; dash < sys_para.num_channels; dash++){
      demod_mem.demod_out[dash].re = scale_factor*demod_mem.demod_out[dash].re;
      demod_mem.demod_out[dash].im = scale_factor*demod_mem.demod_out[dash].im;
   }

}
```

```
/*****************************************************************************/
void DEMOD_diff_decode(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory
demod_mem){

  int tag, dash;
  double scale_factor;
  complex curr_word, prev_word;
  tag = sys_para.filt_symbol_duration + 1;

  scale_factor = (0.5);

  if (sim_mem.fft_clock[0] > (tag-1)) {
   demod_mem.demod_out[0].re = scale_factor*demod_mem.demod_out[0].re;
   demod_mem.demod_out[0].im = scale_factor*demod_mem.demod_out[0].im;
   for (dash = (sys_para.num_data_channels - 1); dash > 0; dash--){
     curr_word.re = scale_factor*demod_mem.demod_out[dash].re;
     curr_word.im = scale_factor*demod_mem.demod_out[dash].im;
     prev_word = inverse_cpx(demod_mem.demod_out[dash-1]);
     demod_mem.demod_out[dash] = multiply_cpx(curr_word, prev_word);
   }
  }

}

/*****************************************************************************/
void DEMOD_coh_chan_est(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory
demod_mem){

  int dash, num_blocks, block_num, block_size, start_index, stop_index, symbol_num, index, tag;
  double real_grad, imag_grad, scale_factor;
  complex cpx_start, cpx_end, pilot_symbol, conj_pilot_symbol;

  tag = sys_para.filt_symbol_duration + 1;
  scale_factor = sqrt(0.5);
  num_blocks = sys_para.pilots/2;
  block_size = sys_para.num_data_channels/num_blocks;
  pilot_symbol.re = sqrt(0.5);
  pilot_symbol.im = sqrt(0.5);
  conj_pilot_symbol.re =  1.0*pilot_symbol.re;
  conj_pilot_symbol.im = -1.0*pilot_symbol.im;

  if (sim_mem.fft_clock[0] > (tag-1)) {

   for (block_num = 0; block_num < num_blocks; block_num++){

     start_index = block_num*block_size;
     stop_index  = (block_num+1)*block_size - 1;

     cpx_start = multiply_cpx(conj_pilot_symbol, demod_mem.demod_out[start_index]);
     cpx_end   = multiply_cpx(conj_pilot_symbol, demod_mem.demod_out[stop_index]);

     cpx_start.re = scale_factor*cpx_start.re;
     cpx_start.im = scale_factor*cpx_start.im;
     cpx_end.re = scale_factor*cpx_end.re;
     cpx_end.im = scale_factor*cpx_end.im;

     real_grad = (cpx_end.re - cpx_start.re)/((1.0*block_size)-1.0);
     imag_grad = (cpx_end.im - cpx_start.im)/((1.0*block_size)-1.0);

     demod_mem.demod_chan_est[start_index].re = cpx_start.re;
     demod_mem.demod_chan_est[start_index].im = cpx_start.im;
     demod_mem.demod_chan_est[stop_index].re  = cpx_end.re;
     demod_mem.demod_chan_est[stop_index].im  = cpx_end.im;

     for (symbol_num = 0; symbol_num < (block_size-2); symbol_num++){
       index = block_num*block_size + symbol_num + 1;
       demod_mem.demod_chan_est[index].re = (real_grad*(symbol_num+1) + cpx_start.re);
       demod_mem.demod_chan_est[index].im = (imag_grad*(symbol_num+1) + cpx_start.im);
     }
   }
}
}
```

```
/*****************************************************************************/
void DEMOD_coh_detect(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory
demod_mem){

  int dash, num_blocks, block_num, block_size, start_index, stop_index, symbol_num, index, tag;
  double real_grad, imag_grad, scale_factor;
  complex conj_chan_est;

  tag = sys_para.filt_symbol_duration + 1;
  num_blocks = sys_para.pilots/2;
  block_size = sys_para.num_data_channels/num_blocks;

  if (sim_mem.fft_clock[0] > (tag-1)) {
   for (block_num = 0; block_num < num_blocks; block_num++){
     for (symbol_num = 0; symbol_num < (block_size-2); symbol_num++){
       index = block_num*block_size + symbol_num + 1;
       conj_chan_est.re = demod_mem.demod_chan_est[index].re;
       conj_chan_est.im = -1.0*demod_mem.demod_chan_est[index].im;
       demod_mem.demod_out[index] = multiply_cpx(conj_chan_est, demod_mem.demod_out[index]);
     }
   }
  }

}

/*****************************************************************************/
void DEMOD_coh_compare(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory
demod_mem){

  int tag, block_num, number, symbol_num, num_blocks, block_size, index;
  tag = sys_para.filt_symbol_duration + 1;
  number = sys_para.num_data_channels*(sys_para.filt_symbol_duration);
  num_blocks = sys_para.pilots/2;
  block_size = sys_para.num_data_channels/num_blocks;

  if (sim_mem.fft_clock[0] > (tag-1)) {
   for (block_num = 0; block_num < num_blocks; block_num++){
     for (symbol_num = 0; symbol_num < (block_size-2); symbol_num++){
       index = block_num*block_size + symbol_num + 1;
       if((demod_mem.reference_qpsk[index+number].re*demod_mem.demod_out[index].re) < 0) sim_mem.ber[0] += 1;
       if((demod_mem.reference_qpsk[index+number].im*demod_mem.demod_out[index].im) < 0) sim_mem.ber[0] += 1;
     }
   }
   sim_mem.demod_sym_num[0] += 2*(sys_para.num_data_channels - sys_para.pilots);
  }


}

/*****************************************************************************/
void DEMOD_compare(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem){

  int dash, tag, number;
  complex difference;
  tag = sys_para.filt_symbol_duration + 1;
  number = sys_para.num_data_channels*(sys_para.filt_symbol_duration);
  if ( sim_mem.fft_clock[0] > (tag-1)) {
    for(dash = 0; dash < sys_para.num_data_channels; dash++){
      if((demod_mem.reference_qpsk[dash+number].re*demod_mem.demod_out[dash].re) < 0) sim_mem.ber[0] += 1;
      if((demod_mem.reference_qpsk[dash+number].im*demod_mem.demod_out[dash].im) < 0) sim_mem.ber[0] += 1;
      difference.re = 0.0;
      difference.im = 0.0;
      difference = minus_cpx(demod_mem.reference_qpsk[dash+number], demod_mem.demod_out[dash]);
      sim_mem.demod_noise_pow[0] += power_cpx(difference);
      }
    sim_mem.demod_sym_num[0] += 2*(sys_para.num_data_channels);
  }
}
```

```
/*************************************************************************/
void DEMOD_diff_compare(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory
demod_mem){

  int dash, tag, number;
  complex difference;
  tag = sys_para.filt_symbol_duration + 1;
  number = sys_para.num_data_channels*(sys_para.filt_symbol_duration);
  if ( sim_mem.fft_clock[0] > (tag-1)) {
    for(dash = 1; dash < sys_para.num_data_channels; dash++){
      if((demod_mem.reference_qpsk[dash+number].re*demod_mem.demod_out[dash].re) < 0) sim_mem.ber[0] += 1;
      if((demod_mem.reference_qpsk[dash+number].im*demod_mem.demod_out[dash].im) < 0) sim_mem.ber[0] += 1;
      difference.re = 0.0;
      difference.im = 0.0;
      difference = minus_cpx(demod_mem.reference_qpsk[dash+number], demod_mem.demod_out[dash]);
      sim_mem.demod_noise_pow[0] += power_cpx(difference);
      }
    sim_mem.demod_sym_num[0] += 2*(sys_para.num_data_channels-1);
  }
}

/*************************************************************************/

void DEMODULATOR(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem ){

  int dash, number;
  number = sys_para.num_data_channels*(sys_para.filt_symbol_duration);
  /**************/ CHAN_info( sys_para, sim_mem, demod_mem );

  /**************/ DEMOD_pre(sys_para, demod_mem);

  /**************/ DEMOD_datashuffle( sys_para, demod_mem);

  /**************/ DEMOD_ppnconv(sys_para, sim_mem, demod_mem);

  /**************/ DEMOD_dft(sys_para, demod_mem);

  /**************/ DEMOD_post(sys_para, demod_mem);

  /**************/ DEMOD_out(sys_para, demod_mem);

  /**************/ DEMOD_scale(sys_para, demod_mem);
   /*
    printf("\n\n Refer QPSK  *** ");
    for(dash = 0; dash < sys_para.num_data_channels; dash++){
    printf("%2.0lf/%2.0lf ", demod_mem.reference_qpsk[dash+number].re,demod_mem.reference_qpsk[dash+number].im);
    }
    printf("\n Demod QPSK  *** ");
    for(dash = 0; dash < sys_para.num_data_channels; dash++){
    printf("%2.0lf/%2.0lf ", demod_mem.demod_out[dash].re,demod_mem.demod_out[dash].im);
    }
    getch();


  /**************/ DEMOD_coh_chan_est(sys_para, sim_mem, demod_mem);

  /*  printf("\n Est. QPSK  *** ");
    for(dash = 0; dash < sys_para.num_data_channels; dash++){
    printf("%2.0lf/%2.0lf ", demod_mem.demod_chan_est[dash].re, demod_mem.demod_chan_est[dash].im);
    }

  /**************/ DEMOD_coh_detect(sys_para, sim_mem, demod_mem);

  /*  printf("\n CohDet QPSK *** ");
    for(dash = 0; dash < sys_para.num_data_channels; dash++){
    printf("%2.0lf/%2.0lf ", demod_mem.demod_out[dash].re,demod_mem.demod_out[dash].im);
    }

  /**************/ DEMOD_coh_compare(sys_para, sim_mem, demod_mem);

}
```

```
/*****************************************************************************/
void COH_DEMOD(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem ){

  /**************/ CHAN_info( sys_para, sim_mem, demod_mem );
  /**************/ DEMOD_pre(sys_para, demod_mem);
  /**************/ DEMOD_datashuffle( sys_para, demod_mem);
  /**************/ DEMOD_ppnconv(sys_para, sim_mem, demod_mem);
  /**************/ DEMOD_dft(sys_para, demod_mem);
  /**************/ DEMOD_post(sys_para, demod_mem);
  /**************/ DEMOD_out(sys_para, demod_mem);
  /**************/ DEMOD_scale(sys_para, demod_mem);
  /**************/ DEMOD_coh_chan_est(sys_para, sim_mem, demod_mem);
  /**************/ DEMOD_coh_detect(sys_para, sim_mem, demod_mem);
  /**************/ DEMOD_coh_compare(sys_para, sim_mem, demod_mem);

}

/*****************************************************************************/
void DIFF_DEMOD(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem ){

  /**************/ CHAN_info( sys_para, sim_mem, demod_mem );
  /**************/ DEMOD_pre(sys_para, demod_mem);
  /**************/ DEMOD_datashuffle( sys_para, demod_mem);
  /**************/ DEMOD_ppnconv(sys_para, sim_mem, demod_mem);
  /**************/ DEMOD_dft(sys_para, demod_mem);
  /**************/ DEMOD_post(sys_para, demod_mem);
  /**************/ DEMOD_out(sys_para, demod_mem);
  /**************/ DEMOD_scale(sys_para, demod_mem);
  /**************/ DEMOD_diff_decode(sys_para, sim_mem, demod_mem);
  /**************/ DEMOD_diff_compare(sys_para, sim_mem, demod_mem);

}

/*****************************************************************************/
void SIM_DEMOD(simulation_parameters sys_para, simulation_memory sim_mem, demodulator_memory demod_mem ){

  /**************/ CHAN_info( sys_para, sim_mem, demod_mem );
  /**************/ DEMOD_pre(sys_para, demod_mem);
  /**************/ DEMOD_datashuffle( sys_para, demod_mem);
  /**************/ DEMOD_ppnconv(sys_para, sim_mem, demod_mem);
  /**************/ DEMOD_dft(sys_para, demod_mem);
  /**************/ DEMOD_post(sys_para, demod_mem);
  /**************/ DEMOD_out(sys_para, demod_mem);
  /**************/ DEMOD_scale(sys_para, demod_mem);
  /**************/ DEMOD_compare(sys_para, sim_mem, demod_mem);

}

/*****************************************************************************/
```

MAIN_AWGNBER.C

```
void CLEAR_demod(simulation_parameters sys_para, demodulator_memory demod_mem){


#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <dos.h>


#include "cmplx.h"
#include "struct.h"
#include "define.h"
#include "init.h"
#include "input.h"
#include "modulate.h"
#include "demodulate.h"
#include "f_random.h"
#include "channel.h"

int main(void){

/*************************************************************************/
/********     INITIALISATION PHASE                                ******/
/*************************************************************************/

int index, Eb_N0, dash;
/*long num_samples;    */
double sigma, sigma2, samp_sym, bits_sym, signal_power, ber, power, count, noise_power;

FILE *stream_Eb_N0;
FILE *stream_sym_num;


simulation_parameters sys_para;
simulation_memory sim_mem;
demodulator_memory demod_mem;


static long int *sym_num;
static long int *fft_clock_num;
static complex *init_ip;
static complex *encode_ip;
static complex *up_ip;
static complex *lo_ip;
static double *rrc_filt;
static double *rrc_ppn;
static complex *up_ppn;
static complex *lo_ppn;
static complex *mod_output;
static complex *old_mod_output;
static complex *mod_output_prev;
static long int *demod_ber;

static complex *quad_buffer;
static complex *reference_qpsk;
static complex *beta0_branch;
static complex *beta1_branch;
static complex *beta2_branch;
static complex *beta3_branch;
static complex *beta0_filt_mem;
static complex *beta1_filt_mem;
static complex *beta2_filt_mem;
static complex *beta3_filt_mem;
static complex *demod_out;
static complex *demod_chan_est;
static complex *fading_tap_mem1;
static complex *fading_tap_mem2;
static long int *demod_symbol_number;
static double *demod_noise_power;
```

```
/* DEFINE SIMULATION PARAMETERS */
sys_para.num_data_channels = NUM_DATA_CHANNELS;
sys_para.over_sampling_factor = OVER_SAMPLING_FACTOR;
sys_para.num_channels = sys_para.num_data_channels*sys_para.over_sampling_factor;
sys_para.filt_symbol_duration = RRC_SYMBOL_DURATION;
sys_para.num_filt_taps = sys_para.num_channels*sys_para.filt_symbol_duration + 1;
sys_para.num_ppn_taps = (sys_para.filt_symbol_duration+1)*sys_para.num_channels;
sys_para.pilots = NUM_PILOTS;
sys_para.true_symbol_period = SYMBOL_PERIOD;
sys_para.true_sample_period = sys_para.true_symbol_period/sys_para.num_channels;
sys_para.filt_roll_off = RRC_ROLL_OFF;
sys_para.num_oscillators = JAKES_OSCILLATORS;
sys_para.dop_sym_ratio1 = P1_DOP_SAM;
sys_para.dop_sym_ratio2 = P2_DOP_SAM;
sys_para.jakes_phase1 = P1_PHASE;
sys_para.jakes_phase2 = P2_PHASE;
sys_para.power1 = P1_POWER;
sys_para.power2 = P2_POWER;
sys_para.independent1 = P1_INDEPENDENT;
sys_para.independent2 = P2_INDEPENDENT;
sys_para.alpha1 = P1_ALPHA;
sys_para.alpha2 = P2_ALPHA;
sys_para.path_diff = P1_2_DIFF;


sym_num = (long int*) calloc(sizeof(long int),1);
fft_clock_num = (long int*) calloc(sizeof(long int),1);
init_ip = (complex *) calloc(sizeof(complex), sys_para.num_channels);
encode_ip = (complex *) calloc(sizeof(complex), sys_para.num_channels);
up_ip = (complex *) calloc(sizeof(complex), sys_para.num_channels);
lo_ip = (complex *) calloc(sizeof(complex), sys_para.num_channels);
rrc_filt = (double *) calloc(sizeof(double), sys_para.num_ppn_taps);
rrc_ppn = (double *) calloc(sizeof(double), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
up_ppn = (complex *) calloc(sizeof(complex), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
lo_ppn = (complex *) calloc(sizeof(complex), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
mod_output = (complex *) calloc(sizeof(complex), sys_para.num_channels);
old_mod_output = (complex *) calloc(sizeof(complex), sys_para.num_channels);
mod_output_prev = (complex *) calloc(sizeof(complex), sys_para.num_channels);
demod_ber = (long int *) calloc(sizeof(long int), 1);
demod_symbol_number = (long int *) calloc(sizeof(long int), 1);
demod_noise_power = (double *) calloc(sizeof(double), 1);
fading_tap_mem1 = (complex *) calloc(sizeof(complex), sys_para.num_channels);
fading_tap_mem2 = (complex *) calloc(sizeof(complex), sys_para.num_channels);

sim_mem.symbol_number = sym_num;
sim_mem.fft_clock = fft_clock_num;
sim_mem.init_qpsk = init_ip;
sim_mem.encode_qpsk = encode_ip;
sim_mem.upper_branch = up_ip;
sim_mem.lower_branch = lo_ip;
sim_mem.filter_coefficients = rrc_filt;
sim_mem.filter_array = rrc_ppn;
sim_mem.upper_filt_mem = up_ppn;
sim_mem.lower_filt_mem = lo_ppn;
sim_mem.mod_out = mod_output;
sim_mem.old_mod_out = old_mod_output;
sim_mem.prev_lower_branch = mod_output_prev;
sim_mem.ber = demod_ber;
sim_mem.demod_sym_num = demod_symbol_number;
sim_mem.demod_noise_pow = demod_noise_power;
sim_mem.fading_taps1 = fading_tap_mem1;
sim_mem.fading_taps2 = fading_tap_mem2;


quad_buffer = (complex *) calloc(sizeof(complex), 4*sys_para.num_channels);
reference_qpsk = (complex *) calloc(sizeof(complex), (sys_para.num_data_channels*(sys_para.filt_symbol_duration + 1)));
beta0_branch = (complex *) calloc(sizeof(complex), sys_para.num_channels);
beta1_branch = (complex *) calloc(sizeof(complex), sys_para.num_channels);
beta2_branch = (complex *) calloc(sizeof(complex), sys_para.num_channels);
beta3_branch = (complex *) calloc(sizeof(complex), sys_para.num_channels);
beta0_filt_mem = (complex *) calloc(sizeof(complex), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
```

```
beta1_filt_mem = (complex *) calloc(sizeof(complex), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
beta2_filt_mem = (complex *) calloc(sizeof(complex), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
beta3_filt_mem = (complex *) calloc(sizeof(complex), (sys_para.num_ppn_taps + sys_para.num_channels - 1));
demod_out = (complex *) calloc(sizeof(complex), 1*sys_para.num_channels);
demod_chan_est = (complex *) calloc(sizeof(complex), 1*sys_para.num_data_channels);

demod_mem.quad_buffer = quad_buffer;
demod_mem.reference_qpsk = reference_qpsk;
demod_mem.beta0_branch = beta0_branch;
demod_mem.beta1_branch = beta1_branch;
demod_mem.beta2_branch = beta2_branch;
demod_mem.beta3_branch = beta3_branch;
demod_mem.beta0_filt_mem = beta0_filt_mem;
demod_mem.beta1_filt_mem = beta1_filt_mem;
demod_mem.beta2_filt_mem = beta2_filt_mem;
demod_mem.beta3_filt_mem = beta3_filt_mem;
demod_mem.demod_out = demod_out;
demod_mem.demod_chan_est = demod_chan_est;
demod_mem.jitter = 0;

stream_sym_num = fopen("c:\\Thesis\\Datafile\\chap3a\\scht112s.asc", "w+");
stream_Eb_N0 = fopen("c:\\Thesis\\Datafile\\chap3a\\scht112b.asc", "w+");
if((stream_sym_num==NULL) || (stream_Eb_N0==NULL)){
  printf("Can't open file.\n");
}

RRCfilter(sys_para, sim_mem);
PPNfilter(sys_para, sim_mem);
signal_power = (0.031)*(4.0/NUM_DATA_CHANNELS);

CLEAR_quad_buffer(sys_para, demod_mem);
CLEAR_filt_mem(sys_para, sim_mem);
sim_mem.symbol_number[0] = 0;
sim_mem.fft_clock[0] = 0;
sim_mem.ber[0] = 0;
sim_mem.demod_sym_num[0] = 0;
sim_mem.demod_noise_pow[0] = 0.0;
power = 0.0;
count = 0.0;

signal_power = 1.0;
```

```
/*************************************************************************/
/********     RUN PHASE                                          ******/
/*************************************************************************/

printf("\n SYSTEM CHARACTERISTICS\n **********\n");
printf(" Number of Data Channels  : %d \n", sys_para.num_data_channels);
printf(" Over Sampling Factor     : %d \n", sys_para.over_sampling_factor);
printf(" FFT Size                 : %d \n", sys_para.num_channels);
printf(" Symbol Period            : %1.5f \n", sys_para.true_symbol_period);
printf(" Sample Period(frequency) : %1.5f (%3.5lf)\n", sys_para.true_sample_period, (1.0/sys_para.true_sample_period));
printf(" Number of Pilot Symbols  : %d \n", sys_para.pilots);
printf(" Pilot Density            : %2.2f% \n", 100.0*sys_para.pilots/sys_para.num_data_channels);
printf("\n CHANNEL CHARACTERISTICS\n **********\n");
printf(" Number of Oscillators          : %d  \n", sys_para.num_oscillators);
printf(" Fd*Ts                          : %3.5f \n",  sys_para.dop_sym_ratio1);
printf(" Fd                             : %3.5f \n",  sys_para.dop_sym_ratio1/sys_para.true_symbol_period);
printf(" Samples sep. paths 1 & 2       : %d  \n", sys_para.path_diff);
printf(" Relative Power          (Path1 - Path2) : %2.4f - %2.4f \n", sys_para.power1, sys_para.power2);
printf(" Relative Phase Offset   (Path1 - Path2) : %2.4f - %2.4f \n", sys_para.jakes_phase1, sys_para.jakes_phase1);
printf(" Jakes Independence Factor (Path1 - Path2) : %2.4f - %2.4f\n", sys_para.independent1, sys_para.independent2);
printf(" Jakes Alpha Phase       (Path1 - Path2) : %2.4f - %2.4f\n", sys_para.alpha1, sys_para.alpha2);
gotoxy(1, 24);
printf("Eb_N0  Sigma   Power(%1.5lf)  Count  Errors   BER     Noise Power\n", signal_power);

for (Eb_N0 = 0; Eb_N0 < EbN0_COUNT; Eb_N0 = Eb_N0+1){

  CLEAR_quad_buffer(sys_para, demod_mem);
  CLEAR_filt_mem(sys_para, sim_mem);
  sim_mem.symbol_number[0] = 0;
  sim_mem.fft_clock[0] = 0;
  sim_mem.ber[0] = 0;
  sim_mem.demod_sym_num[0] = 0;
  sim_mem.demod_noise_pow[0] = 0.0;
  power = 0.0;
  count = 0.0;
  samp_sym = sys_para.num_channels/sys_para.num_data_channels;
  bits_sym = 2.0;
  sigma2 = (samp_sym/bits_sym)*signal_power*pow(10,(-0.1*(double)Eb_N0));
  sigma = sqrt(sigma2/2);

    while (sim_mem.symbol_number[0] < SYM_COUNT){
      CLEARmem(sys_para, sim_mem);
      CLEAR_demod(sys_para, demod_mem);
      sim_mem.symbol_number[0] += sys_para.num_data_channels*2;
       sim_mem.fft_clock[0] += 1;

      iNPUT_DATA(sys_para, sim_mem, index);
      SIM_MOD(sys_para, sim_mem);

        for(dash = 0; dash < sys_para.num_channels; dash++){
        gotoxy(1, 26+Eb_N0/1);
        count += 1.0;
        power                 +=                 (sim_mem.mod_out[dash].re*sim_mem.mod_out[dash].re                +
sim_mem.mod_out[dash].im*sim_mem.mod_out[dash].im);
        }

      CHANNEL_AWGN(sys_para, sim_mem, sigma);

      SIM_DEMOD(sys_para, sim_mem, demod_mem);

        if (sim_mem.demod_sym_num[0] > 0){
        noise_power = 0.0;/*(sim_mem.demod_noise_pow[0]/sim_mem.demod_sym_num[0]);   */
        noise_power = 0.0;/*10.0*log10(2.0/noise_power);                          */
        ber = (double) sim_mem.ber[0]/((double) sim_mem.demod_sym_num[0]);
        printf(" %d   %5.3lf   %5.6lf", Eb_N0, sigma, (power/count));
        printf("    %ld    %ld  %1.6lf  %1.4f\n", sim_mem.demod_sym_num[0], sim_mem.ber[0], ber, noise_power);
        }
    }

  fprintf(stream_Eb_N0, "%ld\n", sim_mem.ber[0]);
  fprintf(stream_sym_num, "%ld\n", sim_mem.demod_sym_num[0]);
}
getch();
```

```
/*****************************************************************/
/********    TERMINATION PHASE                            ******/
/*****************************************************************/

free(sym_num);
free(fft_clock_num);
free(init_ip);
free(encode_ip);
free(up_ip);
free(lo_ip);
free(rrc_filt);
free(rrc_ppn);
free(up_ppn);
free(lo_ppn);
free(mod_output);
free(old_mod_output);
free(mod_output_prev);
free(demod_ber);
free(demod_noise_power);
free(demod_symbol_number);
free(fading_tap_mem1);
free(fading_tap_mem2);

free(quad_buffer);
free(reference_qpsk);
free(beta0_branch);
free(beta1_branch);
free(beta2_branch);
free(beta3_branch);
free(beta0_filt_mem);
free(beta1_filt_mem);
free(beta2_filt_mem);
free(beta3_filt_mem);
free(demod_out);
free(demod_chan_est);


fclose(stream_Eb_N0);
fclose(stream_sym_num);

return(0);
}
```

LIST OF REFERENCES

1. Achatz, R. "*Modeling and Simulation of a OFDM Radio Link*", Proceedings for the IEEE Wireless Communications Conference, 1997, pp. 234-239

2. Akaiwa, Y, "*Introduction to Digital Mobile Communication*", Wiley, 1997

3. Bellanger, M.G. "*Digital Filtering by PolyPhase Network: Application Rate Alteration and Filter Banks*", IEEE Transactions on Acoustics, Speech, and Signal Processing, April 1976, Vol. ASSP-24, No. 24

4. Bellanger, M.G. "*TDM-FDM Transmultiplexer: Digital Polyphase and FFT*", IEEE Trans. on Comms., Sept. 1974, Vol. COM-22, No. 9

5. Bingham, J.A.C, "*Multicarrier Modulation for Data Transmission: An Idea whose time has come*", IEEE Communications Magazine, 1990

6. Bladel, M.V. "*Comparison of Single-Carrier and Multi-Carrier QAM for Digital TV Transmission via CATV*", Proceedings of the 6th Tirrenia International Workshop on Digital Communications, Sept. 1993

7. Braun, W.R. "*A Physical Mobile Radio Channel Model*", IEEE Transactions on Vehicular Technology, May 1991, Vol. 40, No. 2

8. Brigham, E.O, "*The Fast Fourier Transform*", Prentice-Hall, 1974

9. Bullock, S.R. "*Transceiver System Design for Digital Communications*", Noble, 1995

10. Cariolaro, G. "*An OFDM System with a Half Complexity*", IEEE Trans. on Communications, Dec. 1994

11. Casas, E.F. "*OFDM for Data Communication Over Mobile Radio FM Channels – Part I: Analysis and Experimental Results*", IEEE Trans. on Comms., May 1991, Vol. 39, No. 5

12. Chang, R.W. "*A Theoretical Study of Performance of an Orthogonal Multiplexing Data Transmission Scheme*", IEEE Trans. on Comm. Technology, Aug. 1968, Vol. COM-16

13. Chang, R.W. "*Synthesis of Band-Limited Orthogonal signals for Multichannel Data Transmission*", The Bell System Technical Journal, Dec. 1966

14. Cimini, L.J, "*Analysis and Simulation of a Digital Mobile Channel Using Orthogonal Frequency Division Multiplexing*", IEEE Trans. on Comms., July 1985, Vol. COM-13, No.7

15. Corden, I.R. "*Efficient Quadrature Multichannel Processor Algorithms for MCD Applications*", International Journal of Satellite Communications, July 1992, Vol. 10, pp. 105-138

16. Corden, I.R. "*Fast transform based complex transmultiplexer algorithm for multiband quadrature digiital modulation schemes*", IEE Proceedings, December 1990, Vol. 137, Pt. 1, No. 6

17. Couasnon, T. "*OFDM for digital TV broadcasting*", Signal Processing, 1994, Vol. 39

18. Darlington, S. "*On Digital Single-Sideband Modulators*", IEEE Transactions on Circuit Theory, August 1970, Vol. CT-17, No. 3

19. Dersch, U. "*Simulations of the Time and Frequency Selective Outdoor Mobile Radio Channel*", IEEE Transactions on Vehicular Technology, August 1993, Vol. 42, No. 3

20. Duhamel, P. "*Existence of a 2^n FFT algorithm with a number of multiplications lower than 2^{n+1}*", Electronics Letters, August 1984, Vol. 20, No. 17

21. Duhamel, P. "*Split-Radix FFT Algorithm*", Electronics Letters, Jan. 1984, Vol. 20, No. 1

22. Enden, A.W.N, "*Discrete-time signal Processing*", Prentice-Hall, 1989

23. ETS 300 401, "*Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*", ETSI, June 1996

24. ETS 300 749, "Digital Video Broadcasting (DVB); Framing Structure, channel coding and modulation for MMDS systems below 10 GHz", ETSI, Dec. 1996

25. Fang, Y. "*New Methods to Generate Vestigal-Sideband Signal for Data Transmission*", IEEE Transactions on Communications, April 1972, Vol. COM-20, No. 2

26. Floch, B. "*Digital Sound Broadcasting to Mobile Receivers*", IEEE Transactions on Consumer Electronics, August 1989, Vol. 35, No. 3

27. Freeny, S.L. "*Design of Digital Filters for an All Digital Frequency Division Multiplex-Time Division Multiplex Translator*", IEEE Transactions on Circuit Theory, Nov. 1971, Vol. CT-18, No. 6

28. Gibby, R.A. "Some Extensions of Nyquist's Telegraph Transmission Theory", The Bell System Technical Journal, Sept. 1965

29. Harmuth, H.F. "*On the Transmission of Information by Orthogonal Time Functions*", July 1960, pp. 248-255

30. Harvatin, D.T. "*Orthogonal Frequency Division Multiplexing Performance in Delay and Doppler Spread Channels*", Proceeding of Vehicular Technology Conference, 1997, Vol. 3, pp. 1644-1647

31. Heideman, M.T. "*On the Number of Multiplications Necessary to Compute a Length-2^n DFT*", IEEE Transactions on Acoustics, Speech, and Signal Processing, Feb. 1986, Vol. ASSP-34, No. 1

32. Heute, U. "*A Digital Filter Bank with PolyPhase Network and FFT Hardware: Measurements and Applications*", Signal Processing, 1981, Vol. 3, pp. 307-319

33. Hirosaki, B. "*A Maximum Likelihood Receiver for an OrthogonallY Multiplexed QAM System*", IEEE Journal on selected areas in communications, Spet. 1984, Vol. SAC-2, No. 5

34. Hirosaki, B. "*Advanced Groupband Data Modem using Orthogonally Multiplexed QAM Technique*", IEEE Transactions on Communications, June 1986, Vol. COM-34, No. 6

35. Hirosaki, B. "An Orthogonally Multiplexed QAM System using the Discrete Fourier Transform", IEEE Transactions on Communications, July 1981

36. Jakes, W.C, "*Mircowave Mobile Communications*", IEEE Press, 1974

37. Jeruchim, M.C. "*Simulation of Communication Systems*", Plenum, 1992

38. Jones, A.E. "*Combined Coding for Error Control and Increased Robustness to System Non-linearities in OFDM*", IEEE, 1995

39. Journal, "*Special Issue on Digital Video Broadcasting*", Electronics & Communication Engineering Journal, Feb. 1997, Volume 9 Number 1

40. Koilpillai, D.R. "*Some Results in the Theory of Crosstalk-Free Transmultiplexers*", IEEE Transactions on Signal Processing, Oct. 1991, Vol. 39, No. 10

41. Kyees, P.J, "*ADSL: A New Twisted-Pair Access to the Information Highway*", April 1995

42. Lassalle, R. "*Principles of Modulation and Channel Coding for Digital Broadcasting for Mobile Receivers*", EBU Technical Review, August 1987

43. Lee, J.H. "*Designing filters for polyphase filter banks*", IEE Proceedings-G, June 1992, Vol. 139, No. 3

44. Lee, W.C.Y, "*Mobile Cellular Telecommunications*", IEE, 1995

45. Lynn, P.A, "*Introductory Digital Signal Processing with Computer Applications*", Wiley, 1994

46. Macario, R. C. V, "*Personal & Mobile Radio Systems*", IEE, 1993

47. Marshall, T.G., "*Special Issue on Transmultiplexers*", IEEE Transactions on Communications, July 1982, Vol. COM-30, No. 7

48. Miller, R.J. "*OFDM Design Considerations using HF Fading Channel*", IEE Colloquium on HF Antennas and Propagation, 1995, pp. 6/1-6/4

49. Molisch, A.F. "*Error Floor of MSK Modulation in a Mobile Radio Channel with Two Independently fading Paths*", IEEE Transactions on Vehicular Technology, May 1996, Vol. 45, No. 2

50. Moose. P.H. "*A Technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction*", IEEE Trans. on Comms., Oct. 1994, Vol. 42, No. 10

51. Mosier, R. R. "*A Data Transmission System using Pulse Phase Modulation*", June 1957, pp. 233-238

52. Muschallik, Claüs. "*Influence of RF Oscillators on an OFDM Signal*", IEEE Trans. on Consumer Electronics, Aug. 1995, Vol. 41, No. 3

53. Nayebi, K. "*Low Delay FIR Filter Banks: Design and Evaluation*", IEEE Transactions on Signal Processing, Jan. 1994, Vol. 42, No.1

54. Nee, R. D. "*Multipath and Multi-transmitter Interference in Spread-Spectrum Communication and Navigation Systems*", Delft University Press, 1995

55. Nikookar, H. "*OFDM Performance Evaluation over measured indoor radio propagation channels*", 4th IEEE International Conference on Universal Personal Communications, 1995, pp. 968-972

56. Ormondroyd, R.F. "*Comparison of time guard-band and coding strategies for OFDM digital cellular radio in multipath fading*", IEEE 47th Vehicular Technology Conference, 1997, Vol. 2, pp. 850-854

57. Parsons, D, "*The Mobile Radio Propagation Channel*", Prentech Press, 1994

58. Proakis, J.G, "*Digital signal Processing Principles Algorithms, and Applications*", Prentice-Hall International, 1996

59. Radiocommunication study Groups, "*The use of OFDM for Terrestrial Broadcasting of Digital Television*", ITU, Nov. 1994

60. Ramachandran, R.P. "Bandwidth Efficient Transmultiplexers, Part I: Synthesis", IEEE Transactions on Signal Processing, Jan. 1992, Vol. 40, No. 1

61. Ramachandran, R.P. "*Synthesis of Bandwidth Efficient OQAM and VSB Transmultiplexers*", IEEE, 1990

62. Reader, D. "*Blind Maximum Likelihood Sequence Detection Over Fast Fading Communication Channels*", University of South Australia, August 1996

63. Rikkert De Koe, O.B.P. "*On Some Extensions's of Nyquist's Telegraph Transmission Theory*", Proceedings Letters, Dec. 1968

64. Said, F. "Linear two dimensional pilot assisted channel estimation for OFDM systems", 6th IEE Conference on Telecommunications, 1998, pp. 32-36

65. Sari, H. "*Transmission Techniques for Digital Terrestrial TV Broadcasting*", IEEE Communications Magazine, Feb. 1995

66. Scheuermann, H. "*A Comprehensive Survey of Digital Transmultiplexing Methods*", Proceedings of the IEEE, Nov. 1981, Vol. 69, No.11

67. Shelswell, P. "*The COFDM modulation system: the heart of digital audio broadcasting*", Electronics & Communication Engineering Journal, June 1995

68. Speth, M. "*Broadband Transmission using OFDM: system performance and receiver complexity*", International Zurich Seminar on Accessin, Transmission, Networking., 1998, pp. 99-104

69. Steele, R, "*Mobile Radio Communications*", IEEE Press, 1992

70. Stuber, G.L, "*Principles of Mobile Communication*", KAP, 1996

71. Takahata, F. "*A PSK Group Modem Based on Digital Signal Processing: Algorithm, Hardware Design, Implementation and Performance*", International Journal of Satellite Communications, March 1998, Vol. 6, pp. 253-266

72. Thibault, L. "*Performance Evaluation of COFDM for Digital Audio Broadcasting Part I: Parametric Study*", IEEE Trans. on Broadcasting, March 1997, Vol. 43, No. 1

73. Tomlinson, M. "*Techniques for digital interfacing t.d.m.-f.d.m. systems*", IEE Proceedings, Dec. 1976, Vol. 123, No. 12

74. Tourtier, Ph.J, "*Multicarrier modem for digital HDTV terrestrial broadcasting*", Signal Processing: Image Communication 5, 1993, pp. 379-403

75. Tufvesson, F. "*Pilot Assisted Channel Estimation for OFDM in mobile cellular systems*", IEEE 47th Vehicular Technology Conference, 1997, Vol. 3, pp. 1639-1643

76. Vaidyanathan, P.P. "*Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial*", Proceedings of IEEE, Jan. 1990, Vol. 78, No. 1

77. Vary, P. "On the Design of Digital Filter Banks Based on a Modified Principle of Polyphase", AEU, 1979, Vol. 33, pp.293-300

78. Vary. P. "*A Short-time Spectrum analyzer with Polyphase-Network and DFT*", Signal Processing, 1980, Vol. 2, pp. 55-65

79. Vetterli, M. "*A Theory of Multirate Filter banks*", IEEE Transactions on Acoustics, Speech, and Signal Processing, March 1987, Vol. ASSP-35

80. Vetterli, M. "*Filter Banks allowing Perfect Reconstruction*", Signal Processing, 1986, Vol. 10, pp. 219-244

81. Viterbi, A.J, "*CDMA Principles of Spread Spectrum Communication*", Addison-Wesley, 1995

82. Vitetta, G.M. "*Maximum Likelihood Detection of Differentially Encoded PSK Signals Transmitted over Rayleigh Frequency-Flat fading Channels*", IEEE, 1994

83. Wahlqvist, M. "*Design and Evaluation of an OFDM-based Proposal for Third Generation Mobile Communication*", Lulea University of Technology, 1998

84. Wahlqvist, M. "*Time Synchronisation in the uplink of an OFDM sytem*", IEEE, 1996

85. Warner, W.D. "*OFDM/FM Frame Synchronisation for Mobile Radio Data Communication*", IEEE Trans. on Vehicular Technology, Aug. 1993, Vol.42, No. 3

86. Weinstein, S.B. "Data Transmission by Frequency-Division Multiplexing using the Discrete Fourier Transform", IEEE Trans. On Comm. Tech., Oct. 1971, Vol. COM-19

87. Wiley, R.E. "*Digital Television: The End of the Beginning*", IEEE Communications Magazine, Dec. 1995

88. Wu, Y. "Orthognal Frequency Division Multiplexing: A multi-carrier modulation scheme", IEEE Trans. on Consumer Electronics, Aug. 1995, Vol. 41, No. 3

LIST OF PUBLICATIONS

Spectrum shaping in *N*-channel QPSK-OFDM systems

D. Bhatoolaul
G. Wade

Indexing terms: Digital filtering, Modern algorithms, Polyphase shaping filters, QPSK-OFDM systems

Abstract: A compact time-domain treatment of a complete spectrum shaped *N*-channel orthogonal FDM system is presented, enabling practical DSP algorithms for modulation and demodulation to be clearly identified. The resulting DSP architecture is an alternative to previously described OQPSK-OFDM systems and directly provides the two complex samples per symbol required for symbol timing recovery. The paper also discusses parameter selection for the polyphase shaping filters and compares simulation results of power spectral density for shaped and unshaped systems. Simulation shows that shaped systems can closely approach the ideal transmission spectrum even for modest values of *N*.

1 Introduction

Orthogonal frequency-division multiplexing (OFDM) using N channels or carriers is a widely recognised method for combating intersymbol interference (ISI) arising from multipath reception since it extends the symbol period by a factor N. It is then usual to employ a guard interval within the symbol period to absorb multipath echoes. On the other hand, a guard interval (and the consequent loss of efficiency) can be avoided if phase tracking is used for each channel, and this is assumed here. Orthogonality between channels permits spectral overlap and this leads to high spectral efficiency. In particular, using Mary PSK modulators and raised-cosine filters (with a roll-off factor γ) for spectral shaping of each channel, the spectral efficiency [1] is

$$\eta = \frac{\log_2 M}{1 + \frac{\gamma}{N}} \tag{1}$$

This gives an asymptotic efficiency of 2 bit/s/Hz for QPSK and the FDM spectrum tends to a rectangular function for large N. For this reason it can be argued that deliberate pulse/spectral shaping is unnecessary and OFDM systems have been developed on this basis [1, 2]. On the other hand, the out-of-band spectrum for modest values of N can be significant for unshaped channels. Assuming a symbol period T_s and channel spacing $1/T_s$ (for orthogonality), the power spectral density (PSD) for N-channel QPSK-OFDM without spectral shaping is

$$P(\omega) = A \sum_{c=0}^{N-1} Sa^2[(\omega - \omega_c)T_s/2] \tag{2}$$

where A is a constant, $Sa(x) = \sin(x)/x$, and $\omega_c = 2\pi c/T_s$ (corresponding to channel c of a baseband OFDM system). The PSD is illustrated in Fig. 1 for $N = 32$ and 512. Clearly, for one FDM bandwidth from the band edge the power spectrum is some 28 dB down for $N = 32$ and still only about 40 dB down for $N = 512$. Also, in practice N (and hence the symbol duration) cannot be increased indefinitely owing to limitations imposed by the coherence time of the channel [2]. This is particularly true for low bit-rate systems, say <1 Mbit/s, where N may well be limited to around 512. In contrast, it is shown that the out of band response can be reduced to negligible proportions using spectral shaping, even for very low values of N.

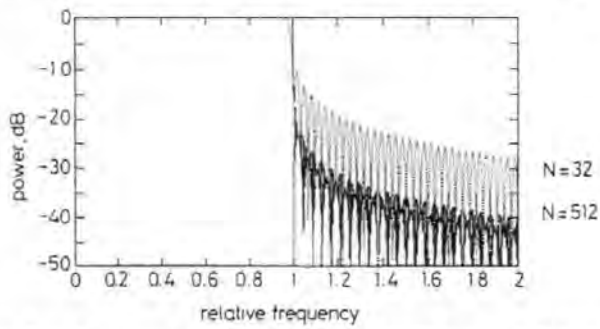


Fig. 1 *Power spectral density for QPSK-OFDM (no spectral shaping)*

The importance of spectral shaping has been recognised by a number of groups [3, 4] and the main objective of this paper is to present a straightforward time-domain analysis of a complete spectrum-shaped system from which detailed DSP algorithms can readily be identified. The paper also discusses parameter selection for practical spectrum-shaped systems, and simulation is used to demonstrate typical improvements in out-of-band spectra compared with Fig. 1.

Unshaped bandpass systems transmitting a vector of N complex symbols at rate $1/T_s$ essentially use an N-point complex FFT(IFFT) operating at the same rate. When channel shaping is introduced one approach is to operate the N-point FFT and filter networks at rate $2/T_s$ [3]. Alternatively, one could use two N-point FFT/filter networks operating in parallel and each clocked at rate $1/T_s$, as discussed in this paper. In turn, due to the data structure, each N-point FFT can be reduced to two $N/4$-point FFTs working at the same rate, representing a considerable saving. This arrangement also directly provides

Paper 2240K (E5, E8), first received 8th March and in final revised form 7th August 1995

The authors are with the Communications Group, SECEE, University of Plymouth, Plymouth PL4 8AA, United Kingdom

the two complex samples per symbol required by symbol timing recovery loops.

The introduction of band-shaping filters $h(t)$ into the conventional QPSK-OFDM system necessitates the use of $T_s/2$ compensating delays, as shown in Fig. 2. These delays ensure orthogonality between channels and the resulting system corresponds to offset QPSK (OQPSK), where $a_c, b_c \in \{\pm 1\}$ for QPSK channel c. We select the usual raised-cosine shaping, so that $h(t)$ in Fig. 2 corresponds to a root raised-cosine filter (RRC). We now derive DSP algorithms for implementing Fig. 2.

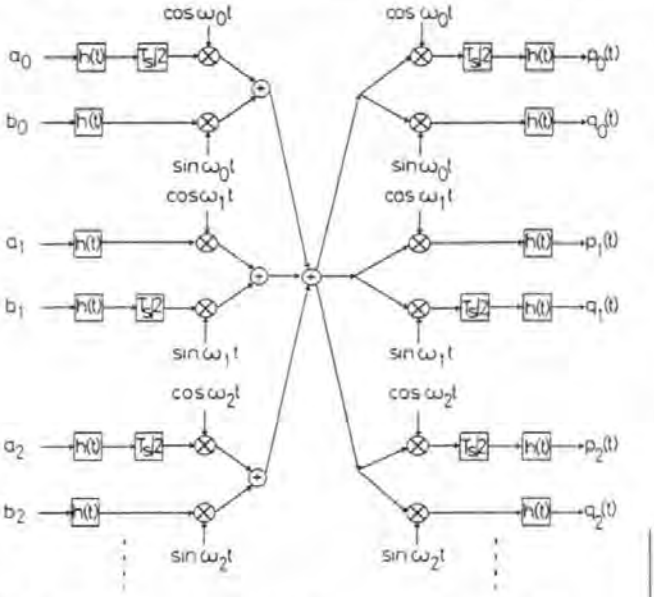


Fig. 2 *Analogue OQPSK-OFDM system* ($\omega_i - \omega_{i-1} = 2\pi/T_s$)

2 Modulator

2.1 DSP implementation

For the present, neglect the $T_s/2$ delays in Fig. 2 to perform a compact analysis using complex signals. Assume that each causal RRC filter has linear phase and spans 0–MT_s s, where M is an even integer. The rth complex codeword $x_c(r) = a_r + jb_r$ entering channel c at $t = rT_s$ generates a complex impulse response centred at $t = (r + M/2)T_s$. Since this response spans M sample periods, impulses spaced up to MT_s seconds apart can contribute to the filter output. It follows that channel c filter output can be expressed as (taking $r = 0$)

$$b_c(t) = \begin{Bmatrix} \sum_{k=0}^{M} x_c(-k)h(t + kT_s) & t = 0 \\ \sum_{k=0}^{M-1} x_c(-k)h(t + kT_s) & 0 < t < T_s \end{Bmatrix} \tag{3}$$

The filter outputs now modulate complex carriers and the modulator outputs are summed to give the complex QPSK-OFDM signal

$$o(t) = \begin{Bmatrix} \sum_{c=0}^{N-1}\left[\sum_{k=0}^{M} x_c(-k)h(t + kT_s)\right]\exp(-j2\pi f_c t) & t = 0 \\ \sum_{c=0}^{N-1}\left[\sum_{k=0}^{M-1} x_c(-k)h(t + kT_s)\right]\exp(-j2\pi f_c t) & 0 < t < T_s \end{Bmatrix} \tag{4}$$

where $f_c = c/T_s$ as before. A sampled form of eqn. 4 can now be obtained by using the mapping

$$t \Rightarrow \frac{i}{N\lambda} T_s \quad 0 \leqslant t < T_s \tag{5}$$

where λ is an oversampling factor and $i \in \{0, 1, 2, \ldots, N\lambda - 1\}$. Applying this to eqn. 4 gives the sampled OFDM signal

$$o\left(\frac{i}{N\lambda} T_s\right) = \begin{Bmatrix} \sum_{k=0}^{M}\left[\sum_{c=0}^{N-1} x_c(-k)\exp\left(-j2\pi c\frac{i}{N\lambda}\right)\right]h\left(\left(\frac{i}{N\lambda} + k\right)T_s\right) & i = 0 \\ \sum_{k=0}^{M-1}\left[\sum_{c=0}^{N-1} x_c(-k)\exp\left(-j2\pi c\frac{i}{N\lambda}\right)\right]h\left(\left(\frac{i}{N\lambda} + k\right)T_s\right) & i \in \{1, 2, \ldots, N\lambda - 1\} \end{Bmatrix} \tag{6}$$

For a fixed value of k the term in square brackets can be considered to be an $N\lambda$-point DFT whose first N inputs are the codewords $x_c(\cdot)$ from the N QPSK channels, while the other inputs are set to zero, see Fig. 3.

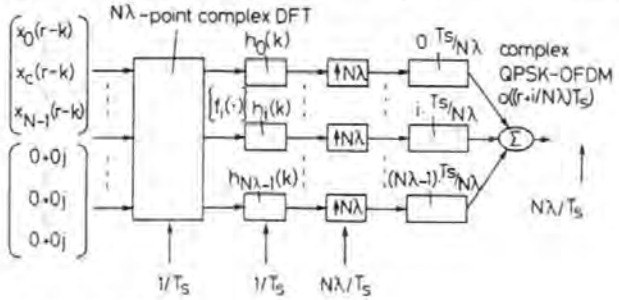


Fig. 3 *Basic OFDM modulator (neglecting $T_s/2$ offset)*

The DFT is clocked every T_s s, corresponding to new sets of N codewords. Denoting the DFT as $f_i(-k)$ reduces eqn. 6 to

$$o\left(\frac{i}{N\lambda} T_s\right) = \begin{Bmatrix} \sum_{k=0}^{M} f_i(-k)h\left(\left(\frac{i}{N\lambda} + k\right)T_s\right) & i = 0 \\ \sum_{k=0}^{M-1} f_i(-k)h\left(\left(\frac{i}{N\lambda} + k\right)T_s\right) & i \in \{1, 2, \ldots, N\lambda - 1\} \end{Bmatrix} \tag{7}$$

For a fixed value of i eqn. 7 is the convolution sum of sequence $\{f_i(\cdot)\}$ with a polyphase filter [5, 6] $h_i(k)$ whose coefficients are defined as follows

$$h_i(k) = \begin{cases} h(kT_s) & i = 0 \text{ and } k = 0, 1, \ldots, M \\ h\left(kT_s + \left(\frac{i}{N\lambda} T_s\right)\right) & i \in \{1, 2, \ldots, N\lambda - 1\} \text{ and } k = 0, 1, \ldots, M-1 \end{cases} \quad (8)$$

where $h(t)$ corresponds to a linear-phase RRC 'prototype' filter. Note that $h_i(k)$ is asymmetric for $i \neq 0$ and that two such filters are required to work in parallel for the real and imaginary DFT outputs. As stated, one could regard eqn. 7 as sequence $\{f_i(\cdot)\}$ feeding filter $h_i(k)$, there being $N\lambda$ such convolutions. On the other hand, over the range 0–T_s s, just one sample is required from each of the $N\lambda$ convolutions to generate the $N\lambda$ point digital OFDM signal in eqn. 7. Therefore since the DFT outputs update simultaneously every T_s s, it is necessary to introduce staggered delays at the filter outputs, as shown in Fig. 3. In turn, this requires zero padding to increase the sample rate. The DFT input vector in Fig. 3 is also zero-padded by $N(\lambda - 1)$ zero-valued samples to provide oversampling by a factor λ. This is useful for minimising aliasing effects when measuring power density during system simulation.

2.2 Modification for offset channels

The foregoing analysis ignored the necessary $T_s/2$ delays in Fig. 2. With reference to Fig. 3 a simple (but apparently redundant) way of incorporating the required offset is to set symbols 'a' in even channels and symbols 'b' in odd channels to zero, and to process them by a separate and modified DFT-PPN system. Considering channel zero for example, Fig. 3 would receive $0 + jb_0$ and the modified subsystem would process $a_0 + j0$.

Suppose a $T_s/2$ delay is placed at the end of the DFT-PPN system in Fig. 3. Using a new mapping similar to eqn. 5 one can now express eqn. 4 in the sampled form

$$o\left(\left(\frac{i}{N\lambda} - \frac{1}{2}\right)T_s\right) = \begin{cases} \sum_{k=0}^{M} \left[\sum_{c=0}^{N-1} x_c(-k) \exp\left(-j2\pi c \frac{i}{N\lambda}\right) \exp\{j\pi c\}\right] h\left(\left(\frac{i}{N\lambda} + k - \frac{1}{2}\right)T_s\right) & i = 0 \\ \sum_{k=0}^{M-1} \left[\sum_{c=0}^{N-1} x_c(-k) \exp\left(-j2\pi c \frac{i}{N\lambda}\right) \exp\{j\pi c\}\right] h\left(\left(\frac{i}{N\lambda} + k - \frac{1}{2}\right)T_s\right) & i \in \{1, 2, \ldots, N\lambda - 1\} \end{cases} \quad (9)$$

According to eqn. 9 the $T_s/2$ delay generates a phase shift of πc on the cth carrier, which can be removed by multiplying the DFT input $x_c(\cdot)$ by a phase offset $\exp\{-j\pi c\}$ i.e. by ± 1. This simple approach to handling the $T_s/2$ offset requires a complex codeword splitter to first extract those parts of a complex symbol requiring the offset and then deliver them to the DFT-PPN processor represented by eqn. 9. The overall modulator is shown in Fig. 4.

The efficiency of the system in Fig. 4 can be improved as follows. Over a symbol period the modulator input is $c_k = a_k + jb_k$, $k = 0, 1, \ldots, N-1$ and the nonzero values applied to the upper DFT correspond to the sequence $\{jb_0, a_1, jb_2, a_3, \ldots, jb_{N-2}, a_{N-1}\}$. Clearly, an N-point DFT of this sequence can be decimated to the form ($\lambda = 1$)

$$X(n) = G(n) + W_N^n H(n) \quad W_N = e^{-j2\pi/N} \quad (10)$$

where $G(n)$ and $H(n)$ are $N/2$-point DFTs of sequences $\{jb_0, jb_2, jb_4, \ldots, jb_{N-2}\}$ and $\{a_1, a_3, a_5, \ldots, a_{N-1}\}$, respectively. Since these are purely real or purely imaginary sequences it follows that each N-point DFT in Fig. 4 can be replaced by two $N/4$-point FFTs, using 'real-input' type algorithms, representing considerable computational saving. Alternatively, it can be shown that each N-point DFT could be replaced by an $N/2$-point complex input FFT followed by $N/2$ complex butterflies. Similar simplification could also be applied to the demodulator.

3 Demodulator

3.1 DSP Implementation

The analogue demodulators in Fig. 2 can be represented by the complex baseband model in Fig. 5. Using this Figure, the output of the channel c filter is

$$s_c(t) = h(t) * (o(t) \exp\{j2\pi f_c t\}) \quad (11)$$

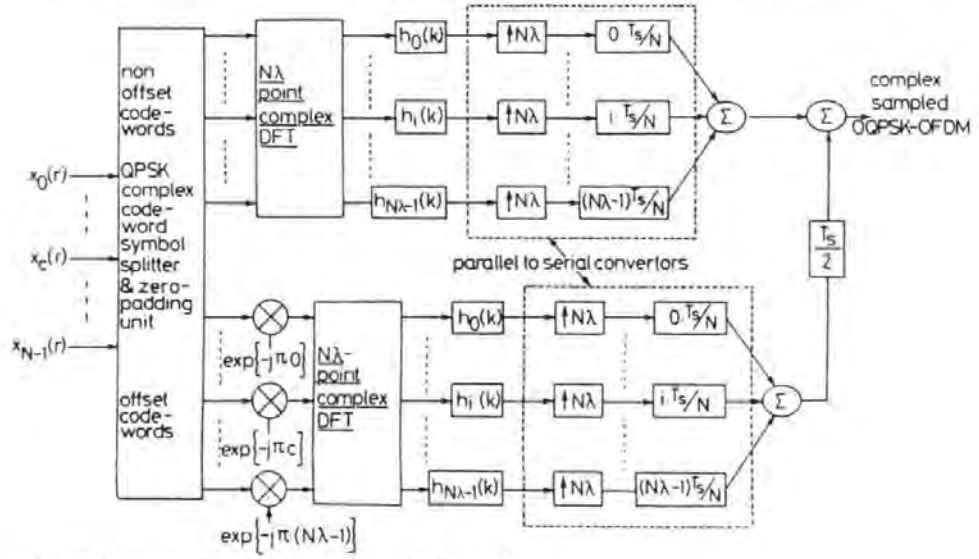


Fig. 4 *Complete modulator for spectrum-shaped OFDM (in practice $\lambda = 1$)*

where $f_c = c/T_s$. Now assume that the input to Fig. 5 is sampled at rate N/T_s, corresponding to the output of Fig. 3 with $\lambda = 1$. Noting that $h(t)$ spans $MN + 1$ samples,

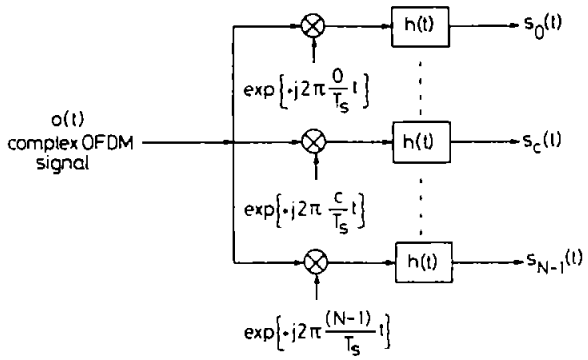


Fig. 5 *Analogue OFDM demodulator*

eqn. 11 now becomes

$$s_c\left(k\frac{T_s}{N}\right) = \sum_{i=0}^{MN} h\left(i\frac{T_s}{N}\right) o\left((k-i)\frac{T_s}{N}\right) \times \exp\left\{j2\pi\frac{c}{N}(k-i)\right\} \quad (12)$$

where k is an integer. Channel c now generates N complex samples every T_s s, although only two of these (spaced by $T_s/2$ s) are required to decode each complex codeword $x_c(r)$. For example, assuming the imaginary components of sequence $\{x_c(r)\}$ are delayed by $T_s/2$ s relative to the real part, then components a_r can be obtained by sampling $s_c(t)$ once every T_s s, while components b_r would be obtained by delaying sampling by $T_s/2$ s. The required subsampling of $s_c(kT_s/N)$ can be achieved through the mapping

$$k \Rightarrow \alpha N - \beta \quad \beta \in \{0, 1, 2, \ldots, N-1\} \quad \text{and } \alpha \text{ is an integer} \quad (13)$$

Applying this mapping to eqn. 12 gives samples of $s_c(t)$ spaced T_s s apart and offset from some time datum by $T_s\beta/N$ s

$$s_c\left(\left(\alpha - \frac{\beta}{N}\right)T_s\right) = \exp\left\{-j2\pi\frac{c}{N}\beta\right\} \sum_{i=0}^{MN} h\left(i\frac{T_s}{N}\right) \times o\left((\alpha N - i)\frac{T_s}{N} - \beta\frac{T_s}{N}\right) \times \exp\left\{-j2\pi\frac{c}{N}i\right\} \quad (14)$$

Given correct synchronisation, eqn. 14 gives the non-delayed components of $x_c(r)$ when $\beta = 0$, and the delayed components when $\beta = N/2$. Using this analysis, the demodulator therefore comprises two subsystems working in parallel, one with $\beta = 0$ and one with $\beta = N/2$.

An efficient DSP realisation of eqn. 14 can be obtained by expressing it in terms of a double summation. First note that for an arbitrary function $x(i)$ one can express a single summation over $MN + 1$ indices as

$$\sum_{i=0}^{MN} x(i) = \sum_{n=0}^{N-1} \sum_{m=0}^{f(n)} x(mN + n) \quad (15)$$

where

$$f(n) = \begin{cases} M & \text{for } n = 0 \\ M - 1 & \text{for } n \in \{1, 2, 3, \ldots, N-1\} \end{cases} \quad (16)$$

Using eqns. 15 and 16 we can re-express eqn. 14 as

$$s_c\left(\left(\alpha - \frac{\beta}{N}\right)T_s\right) = \exp\left\{-j2\pi\frac{c}{N}\beta\right\} \times \sum_{n=0}^{N-1}\left[\sum_{m=0}^{f(n)} h\left(mT_s + \frac{n}{N}T_s\right) \times o\left((\alpha - m)T_s - (n+\beta)\frac{T_s}{N}\right)\right] \times \exp\left\{-j2\pi\frac{c}{N}n\right\} \quad (17)$$

Letting $y(n) = [\cdot]$ gives

$$s_c\left(\left(\alpha - \frac{\beta}{N}\right)T_s\right) = \exp\left\{-j2\pi\frac{c}{N}\beta\right\} \sum_{n=0}^{N-1} y(n) \times \exp\left\{-j2\pi\frac{c}{N}n\right\} \quad (18)$$

$$= \sum_{n=0}^{\beta-1} y(n+\beta) \exp\left\{-j2\pi\frac{c}{N}n\right\} + \sum_{n=\beta}^{N-1} y(n-\beta) \exp\left\{-j2\pi\frac{c}{N}n\right\} \quad (19)$$

Clearly, eqn. 18 represents an N-point DFT of $\{y(n)\}$, and the cth DFT output is phase shifted (multiplied by ± 1) when $\beta = N/2$, see Fig. 6. Alternatively, according to eqn. 19, the phase shifts could be replaced by shuffling the DFT inputs. The complete demodulator requires two subsystems of the form shown in Fig. 6 to be used in parallel, one with $\beta = 0$ and the other with $\beta = N/2$. The appropriate outputs from channel c of each subsystem are then combined to reconstruct the complex sample $x_c(r)$. Also, together, these outputs give two complex samples per symbol as required by phase-tracking loops.

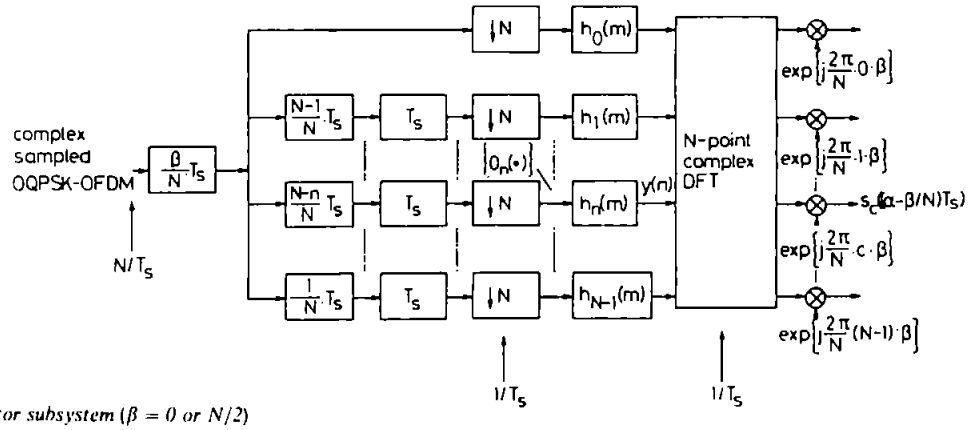


Fig. 6 *Demodulator subsystem ($\beta = 0$ or $N/2$)*

The derivation of $y(n)$, $n = 0, 1, \ldots, N-1$ requires some explanation. According to eqn. 17, these samples result from N filtering operations on sequences $\{o_n(\cdot)\}$, $n = 0, 1, \ldots, N-1$ and, as for the modulator, the required polyphase filters $h_n(m)$ have coefficients derived from a prototype RRC filter $h(t)$ according to

$$h_n(m) = \begin{cases} h(mT_s) & n = 0 \text{ and } m = 0, 1, 2, \ldots, M \\ h\left(mT_s + n\frac{T_s}{N}\right) & n \in \{1, 2, 3, \ldots, N-1\} \\ & \text{and } m = 0, 1, 2, \ldots, M-1 \end{cases} \quad (20)$$

Note from eqn. 17 that filter $h_n(m)$ processes sequence $\{o_n(\cdot)\}$ which is delayed in time by nT_s/N s relative to sequence $\{o_0(\cdot)\}$. Also, note that filter $h_0(m)$ has one extra coefficient (an extra delay of T_s s). It therefore follows that for the output of filter $h_n(m)$ to be synchronous with the output from filter $h_0(m)$, a delay of $(N-n)T_s/N$ must be inserted, as shown in Fig. 6. Unfortunately, inserting this delay aligns filter samples with time index m ($n > 0$) with the sample in channel zero having index $m + 1$. Therefore to maintain a set of N values of $y(n)$ with the same time index m it is necessary to add a further T_s delay to all channels except channel zero.

3.2 PPN coefficients

The prototype RRC impulse response $h(t)$ spans $MN + 1$ samples or M symbol periods and it is necesary to determine a suitable value for M. This can be done by frequency sampling the overall raised-cosine response at $MN + 1$ points; for a normalised sampling frequency the baseband response is then skew-symmetric about frequancy $1/2N$ and the kth sample corresponds to frequency $k/(MN + 1)$. Some γM samples will lie on the actual raised-cosine curve.

Taking the IFFT gives the impulse response which is then windowed (and zero-extended to provide frequency-domain interpolation). The FFT of the zero-extended response is shown in Fig. 7 for $\gamma = 0.5$ and several values

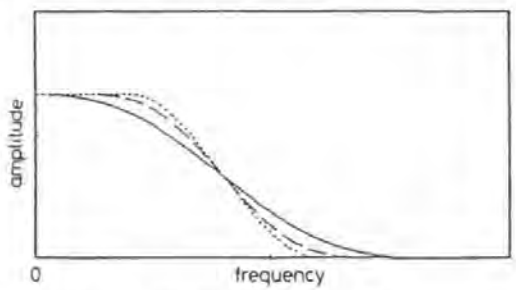


Fig. 7 *Raised-cosine spectrum as a function of M*

——— $M = 4$
– – – – $M = 8$
····· Ideal

of M and it is apparent that $M = 8$ is satisfactory for this value of γ (the response for $M = 16$ is virtually indistinguishable from the ideal response). Frequency sampling and windowing was also used to design the prototype RRC filter, which was then subsampled according to eqns. 8 or 20.

4 Simulated spectra

The complete OQPSK-OFDM system (Figs. 4 and 6) was simulated to examine individual subchannels and the overall power spectral density. Simulation used a high-level modelling package (SPW, Cadence UK) and extensive use was made of custom-coded blocks, written in C. Long records of random data (typically 40,000 samples) were used for power spectrum estimation.

Fig. 8 shows the channel spectra for a single complex impulse $1 + j1$ applied to channel 24; curve (*a*) shows the

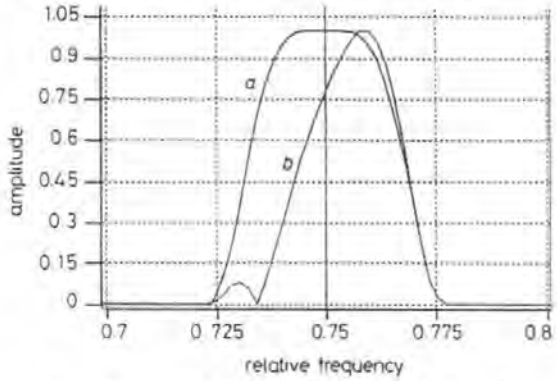


Fig. 8 *Response to complex impulse on channel 24 ($N = 32$)*

a Subsystem response
b Transmitted spectrum

true RRC response of either subsystem in Fig. 4 and curve (*b*) shows the actual transmitted spectrum (all other channel inputs being zero). Fig. 9 illustrates PSD estim-

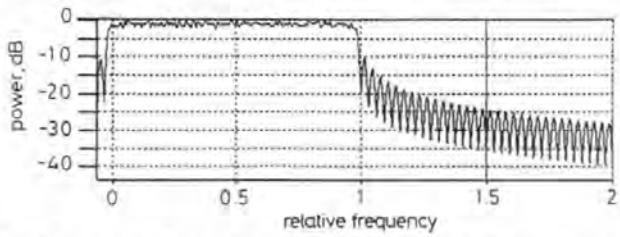


Fig. 9 *Simulated PSD for QPSK-OFDM ($N = 32$): unshaped system*

ates for $N = 32$ obtained from the model using the autocorrelation-FFT method with a Hanning window. Fig. 9 shows the PSD for no channel shaping ($\sin x/x$-shaped spectra), corresponding to Fig. 1. Fig. 10 shows the PSD for RRC transmitted spectra and shows a clear reduction in out-of-band spectra compared to unshaped transmission.

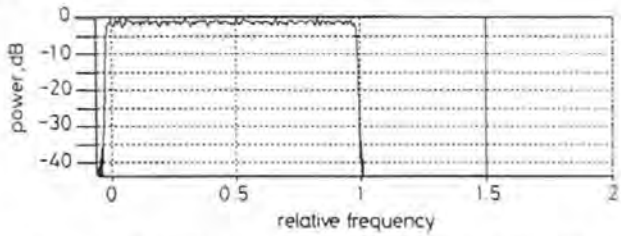


Fig. 10 *Simulated PSD for QPSK-OFDM ($N = 32$): RRC system*

5 Conclusions

To summarise, a time-domain analysis of a complete spectrum-shaped OQPSK-OFDM system has been presented, establishing clear links between theory and DSP implementation. This leads to an alternative architecture working at rate $1/T_s$ compared to rate $2/T_s$ for previously described systems. High-level modelling of the proposed system has demonstrated that the PSD of shaped systems can closely approach the ideal rectangular spectrum even for modest values of N.

6 References

1 Cimini, L.J.: 'Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing', *IEEE Trans.*, 1985, **COM-33**, (7), pp. 665–675

2 FLOCH, B., HALBERT-LASSALLE, R., and CASTELAIN, D.: 'Digital sound broadcasting to mobile receivers', *IEEE Trans.*, 1989, **CE-35**, (3), pp. 493–503
3 HIROSAKI, B., HASEGAWA, S., and SABATO, A.: 'Advanced groupband data modem using orthogonally multiplexed QAM technique', *IEEE Trans.* 1986, **COM-34**, (6), pp. 587–592
4 TAKAHATA, F., YASUNAGA, M., HIRATA, Y., OHSAWA, T., and NAMIKI, J.: 'A PSK group modem based on digital signal processing: algorithm, hardware design, implementation and performance', *Int. J. Satellite Comm.*, 1988, **6**, pp. 253–266
5 BELLANGER, M.G., and DAGUET, J.L.: 'TDM-FDM transmultiplexer: digital polyphase and FFT', *IEEE Trans.*, 1974, **COM-22**, (9), pp. 1199–1205
6 BELLANGER, M.G., BONNEROT, G., and COUDREUSE, M.: 'Digital filtering by polyphase network: application, rate alteration and filter banks', *IEEE Trans.*, 1976, **ASSP-24**, (2), pp. 109–114

COPYRIGHT STATEMENT ©