Design of a Fixed-Point Polar Receiver for OFDM-Based Wireless LAN

by

Amro Faisal Mohammed Altamimi
B.Sc., King Fahd University of Petroleum & Minerals, 2005

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Masters of Applied Science

in the Department of Electrical & Computer Engineering

Supervisory Committee

Design of a Fixed-Point Polar Receiver for OFDM-Based Wireless LAN

by

Amro Faisal Mohammed Altamimi
B.Sc, King Fahd University of Petroleum & Minerals, 2005

Supervisory Committee

Dr. Daler Rakhmatov, Department of Electrical & Computer Engineering
Supervisor

Dr. Michael McGuire, Department of Electrical & Computer Engineering
Co-Supervisor

Dr**.** Jianping Pan, Department of Computer Science
Outside Member

Abstract

Supervisory Committee

Dr. Daler Rakhmatov, Department of Electrical & Computer Engineering
Supervisor

Dr. Michael McGuire, Department of Electrical & Computer Engineering
Co-Supervisor

Dr. Jianping Pan, Department of Computer Science
Outside Member

This thesis studies implementation-related issues in OFDM-based digital receivers, using the IEEE 802.11a WLAN standard as a specific wireless technology, where the data rate ranges from 6 Mbps to 54 Mbps. Our goal is to expose and exploit the possibility of scaling of the receiver computational complexity in relation to variable data rate requirements. To facilitate such computational scalability, we propose and evaluate the use of the polar coordinates during data processing in the frequency domain. We also evaluate the impact of various fixed-point precision settings during data processing in both the time domain and the frequency domain. We have found that for the 6-Mbps and 54-Mbps data rates the appropriate fixed-point word length should be 15 bits and 20 bits, respectively. While evaluating different fixed-point precision settings, we found that simulations times were prohibitively long. To address this issue, we also propose an alternative 5-step simulation procedure that significantly reduces the simulation time needed to evaluate any given fixed-point setting option.

Table of Contents

List of Tables

List of Figures

Acknowledgments

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

{ قَالُواْ سُبۡحَٰنَكَ لَا عِلۡمَ لَنَآ إِلَّا مَا عَلَّمۡتَنَآ إِنَّكَ أَنتَ ٱلۡعَلِيمُ ٱلۡحَكِيم }

In the name of God, the most beneficent, the most merciful

{"They said: Be glorified! We have no knowledge saving that which Thou hast taught us. Lo! Thou, only Thou, art the Knower, the Wise"}.

I thank God for blessing me with the knowledge I have attained and I pray that my work be of benefit for those who come after me.

I would like to thank Dr. Daler Rakhmatov for his supervision, time and continued support in my masters program and thesis. I would also like to thank Dr. Michael McGuire for his valuable feedback and recommendations in my thesis preparation.

I would also like to thank the government of Saudi Arabia for their generous scholarship and funding of my Masters program. My special thanks and gratitude to the Custodian of the two holy mosques: King Abdullah bin Abdulaziz for spearheading the scholarship program and making it available for many Saudis to pursue their potential.

Dedication

To my parents, for teaching me the value of knowledge and patience as well as being a constant source of inspiration and motivation.

To my family and friends, for their many years of friendship, support and encouragement.

Glossary of Terms

WLAN	Wireless Local-Area Networks	AGC	Automatic Gain Control
LTE	Long Term Evolution	A/D	Analog to Digital
IEEE	Institute of Electrical and Electronic Engineers	LS	Least Square
OFDM	Orthogonal Frequency Division Multiplexing	MLE	Maximum Likelihood
WiMAX	Worldwide Interoperability for Microwave Access	CTF	Channel Transfer Function
CORDIC	Coordinate Rotation Digital Computer	MMSE	Minimum Mean Square Error
ISI	Inter Symbol Interference	CSI	Channel State Information
ICI	Inter Carrier Interference	RFO	Residual Frequency Offset
PAPR	Peak to Average Power Ratio	RM	Rotation Mode
CFO	Carrier Frequency Offset	VM	Vectoring Mode
SFO	Sampling Frequency Offset	FPGA	Field Programmable Gate Array
PN	Pseudo-Random Noise	LUT	Look Up Table
TDMA	Time Division Multiple Access	FER	Frame Error Rate
CDMA	Code Division Multiple Access	BER	Bit Error Rate
FDM	Frequency Division Multiplexing	AWGN	Additive White Gaussian Noise
ADC	Analog to Digital Converter	Ds	Delay Spread
DAC	Digital to Analog Converter	LOS	Line Of Sight
WiFi	Wireless Fidelity	SNR	signal-to-noise ratios
PHY	physical layer	FFT	Fast Fourier Transform
PLCP	Physical Layer Convergence Procedure	IFFT	Inverse Fast Fourier Transform
PMD	Physical Medium Dependent	CP	Cyclic Prefix
MAC	Medium Access Control	GI	Guard Interval
PSDU	PHY Service Data Units	DFT	Discrete Fourier Transform
PSK	Phase Shift Keying	IDFT	Inverse Discrete Fourier Transform
BPSK	Binary – PSK	STS	Short Training Sequence
QPSK	Quadrature – PSK	LTS	Long Training Sequence
QAM	Quadrature Amplitude Modulation		

List of Symbols

X(f)	Signal in frequency-domain	w_T	Window function
x(t)	Signal in time-domain	d	Correlation index
T_{SYM}	OFDM Symbol duration	P(d)	Auto-correlation power
β	Roll off factor	R(d)	Auto-correlation Energy
T_{cp}	Cyclic prefix length	M(d)	Auto-correlation Metric
N_{CBPS}	Number of Coded Bits Per OFDM symbol	Th_r	Correlation rising threshold
N_s	Number of Subcarriers	L	Correlation window size
δf	Subcarrier Spacing	$\Lambda(d)$	Cross-correlation power
N	FFT/IFFT Length	φ_o	Phase offset caused by CFO in time
T_s	Sampling Time	φ_{RFO}	Phase offset caused by RFO in frequency
n_ϵ	Number of offset Samples in time-domain	N_p	Number of pilot subcarriers
ϵ_t	Timing Offset in seconds	p, p_r	Receiver pilot
τ_{MAX}	Multipath Maximum Delay Spread	c, p_k	Known pilot
N_g	Number of Guard band Subcarriers	γ	PN sequence sign identifier
φ_i	Phase rotation at Subcarrier i	D	RFO compensated data
δ_i	Frequency at subcarrier i	C	Channel Transfer Function Inverse
φ_k	Local Frequency offset at each subcarrier k	Th_f	Correlation falling threshold
ϵ_f	Normalized Frequency Offset	E	Equalized data before RFO compensation
r	Received signal in time-domain	t	Time in seconds
s	Transmitted signal in time-domain	f	Frequency in Hertz
h	Channel Impulse Response	n	Time sample number
ω	Additive White Gaussian Noise (AWGN)	k	Frequency subcarrier number
R	Received signal in Frequency-domain	l	OFDM Symbol number
S	Transmitted signal in Frequency -domain	Δf	Frequency Offset
H	Channel Transfer Function	f_c	Center Frequency
I	Inter Carrier Interference (ICI)	c_q	Oscillator Quality factor
W	AWGN in Frequency-domain	σ^2	Variance
D_{freq}	SNR Degradation due to Frequency Offset	N_{DBPS}	number of data bits per OFDM symbol
Es/No	Symbol energy to noise ratio	N_{BPSC}	number of coded bits per subcarrier

T_s'	Perceived Sampling Time at Receiver	N_{SYM}	Number of OFDM Symbols
ζ	Sampling time offset	N_{DATA}	Number of data bits
θ_o	Phase offset due to sampling time shift	N_{PAD}	Number of Padding bits

Chapter 1: Introduction

1.1. Motivation and Objectives

With the ever-increasing need for higher data rates and stable performance, the boundaries of modern telecommunication systems are constantly tested and pushed to further limits. Mobile communication has recently taken the step into its fourth generation of standards, such as WiMAX and LTE (Long Term Evolution), while WLANs (Wireless Local-Area Networks) have recently adopted the IEEE 802.11n standard. These new standards offer data rates around 300 Mb/s, while providing better resistance to environmental effects.

At the heart of the new abovementioned wireless technologies is OFDM (Orthogonal Frequency Division Multiplexing). This thesis investigates hardware-oriented implementation issues arising in OFDM receiver designs and exposes engineering tradeoffs between system performance and computational cost. This thesis then provides technical recommendations for an OFDM receiver design, targeting computational effort reduction with negligible system performance degradation. As typical OFDM receivers support multiple data rates, the first goal was to enable scaling of the receiver computational complexity in relation to variable data rate requirements. In our simulation-based experimental studies, we have chosen the OFDM-based IEEE 802.11a WLAN standard as a demonstration platform. This standard specifies eight data rates: 6, 9, 12, 18, 24, 36, 48, and 54 Mbps. System simulations, used for performance evaluation of various design alternatives, pose a challenge of a different kind: they are very time-consuming, which slows down the design decision-making process itself. Our second goal was to enable fast evaluations of design decisions, with empirical indicators of how optimistic or pessimistic those decisions are in terms of hardware requirements.

1.2. Thesis Contributions

There are three main contributions presented in this thesis:

- First, we propose the use of polar (amplitude-phase) coordinates during processing of complex-valued signals in the frequency domain, as opposed to the common practice of using Cartesian (real-imaginary) coordinates. The advantage of polar coordinates is in their computational scalability in the context of data-rate-dependent OFDM demodulation. For example, for the data rates of 6, 9, 12, and 18 Mbps the amplitude information can be ignored during demodulation (i.e., only the phase information is used), so the receiver may effectively shut down the circuitry that processes the amplitude stream of signal values. This cannot be done when using Cartesian coordinates, as both real and imaginary streams of received signal values are needed during demodulation.

- Second, we describe a new simulation approach that allows for significant reductions in the amount of time need to evaluate a decision on a particular design parameter. In our studies, we needed to simulate multiple data rates at multiple SNRs (signal-to-noise ratios) for an indoor Rayleigh fading channel model, and each simulation setting required 10,000 runs. We reduced the number of runs per setting to 100 by simulating only the worst-case channel scenario. The corresponding design parameters, chosen to ensure acceptable system performance under such worst-case scenario, are pessimistic. We then decide on the same design parameters based on 100-run simulations of the best-case channel scenario, which gives us the optimistic alternative. Having both pessimistic and optimistic sets of design parameters proved to be very useful in making final design decisions.

- Third, we identify (based on our simulation results) the required fixed-point word lengths to be used in typical receiver implementations for time-domain and frequency-domain signal processing. These fixed-point word lengths (allocated to represent floating-point numbers approximately in binary format) are, in fact, an example of a design parameter that we had to decide upon, with the objective to match the system performance exhibited in floating-point simulations. We found that the required fixed-point word lengths are strongly dependent on the data rate, which allows the receiver to scale its computational efforts accordingly.

The design recommendations presented in this thesis pertain to *channel-independent* receiver configurations that can be selected automatically by the system, based only on the information contained in the frame header. We do not address more advanced, *channel-dependent* receiver configurations that involve estimating and using more detailed information about the wireless channel state (e.g., SNR). The latter case is likely to improve system performance, but it would also require a higher computational effort related, for example, to more sophisticated channel estimation.

1.3. Related Work

There are numerous papers, such as [26, 27], that report various implementations of OFDM-based receivers, but they omit details on the actual design process used to arrive at the reported implementation. Our first contribution – a time-saving simulation methodology – aims at highlighting the issue of very long simulation times (which is particularly problematic when simulating fixed-point receivers) and providing a relatively simple solution for reducing simulation times.

The idea of using polar coordinates to process complex-valued signals is not new [41, 42], but at first (superficially) it appears to offer no significant advantages with respect to using Cartesian coordinates. For example, in the 54 Mbps data rate case (the highest rate specified by the IEEE 802.11a WLAN standard), this is indeed true, and consequently, the vast majority of reported implementations use Cartesian coordinates. However, from the computational scalability viewpoint, polar coordinates are a better choice, allowing for significant computational savings at lower data rates. To our knowledge, published research works have not investigated the merit of using polar coordinates in OFDM-based 802.11a-compliant receiver implementations. Information on the 802.11a standard specification can be found in Appendix B.

The current literature on OFDM-based receivers also lacks reports on fixed-point performance of physical-layer processing of received frames, which consider all of the following steps: synchronization, frequency offset compensation, channel estimation/equalization, fast Fourier transformation, phase tracking, and demodulation. Related publications, such as [26, 27], focus only on one or two steps (assuming ideal performance of the others), which provides an incomplete picture of overall performance of fixed-point receiver implementations. This thesis fills most of the gaps in the analysis of the sequence of typical computational steps of the OFDM-based receiver. The analysis does not include I/Q mismatch, sampling offset, and phase noise compensation [17-19].

The brief literature overview presented here serves only as a quick summary of related work in relation to our claimed contributions in general. More technical details and additional references are described in Chapters 4-6, where we discuss the merits and drawbacks of our specific design decisions (after Chapters 2-3 providing background information).

1.4. Thesis Organization

The rest of the thesis is organized as follows. Chapters 2 provides a basic background on the OFDM technology. Chapter 3 discusses critical challenges of synchronization, channel estimation, and phase tracking that need to be solved in practical implementations of OFDM-based receivers. Chapter 4 documents our reference floating-point (MATLAB) implementation of a typical OFDM-based 802.11a-compliant receiver that uses Cartesian coordinates. In Chapter 5 we describe our proposed receiver that uses polar coordinates and compare its floating-point performance against that of our reference Cartesian-coordinate receiver. Chapter 6 presents the simulation results and analysis of fixed-point performance of our proposed polar-coordinate receiver in comparison to its floating-point performance. In the same chapter, we also present our overall time-saving methodology used to arrive at the recommended values of our simulated fixed-point word lengths. Finally, Chapter 7 concludes the thesis with the summary, conclusions and suggestions for future work.

Chapter 2: OFDM Primer

Orthogonal Frequency Division Multiplexing, or OFDM, is a type of multicarrier modulation based on frequency division multiplexing (FDM) [1]. FDM divides the available spectrum into sets of channels with a certain bandwidth for each user. The available bandwidth is limited however, and increasing data rate without increasing bandwidth lowers the receiver's resistance to channel impairments, such as inter-symbol interference (ISI) and frequency selective fading. Multicarrier modulation overcomes channel impairments to some extent, by dividing a symbol stream into parallel substreams to be transferred over multiple subchannels [2]. OFDM subchannels are carried by subcarriers which are orthogonal to each other (see Fig. 2.1), which means that these subcarriers have zero cross-correlation and hence can be packed closer together without interference, as long as they are properly spaced.

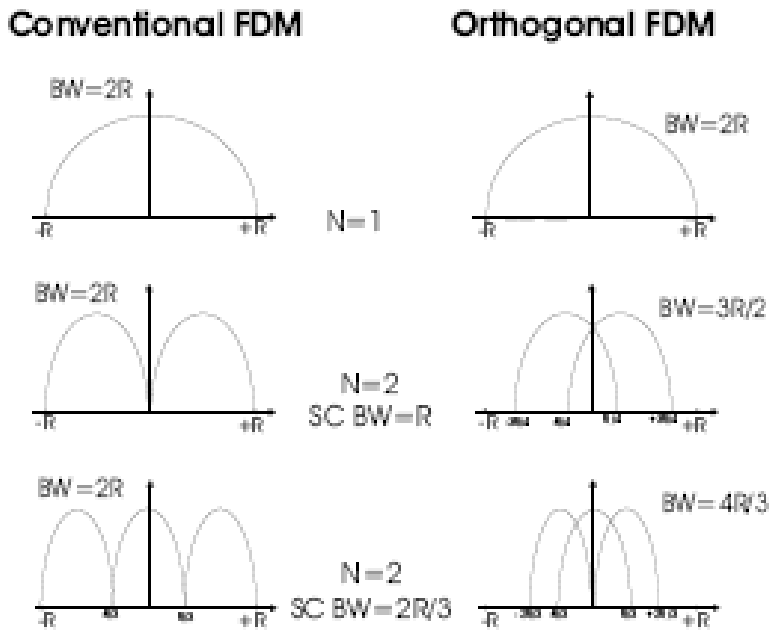


Figure 2.1: FDM vs OFDM [1]

2.1. OFDM Properties

There are several essential elements present in any OFDM-based system: Fast Fourier Transform (FFT), cyclic prefix, windowing, and orthogonal subcarriers. These are explained in more detail below.

2.1.1. FFT

Fourier transform derives spectral information from a given time-sampled signal, as shown in Eq. (2.1); while the inverse Fourier transform, given by Eq. (2.2), converts the frequency-domain information back to the time domain [5]:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt \qquad (2.1)$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft}\, df \qquad (2.2)$$

In a digital system working with discrete samples, the Discrete Fourier Transform (DFT) and its inverse IDFT are used, as given by Eq. (2.3)-(2.4), where *k* is the frequency-domain sample index and *n* is the time-domain sample index. Fast Fourier Transform (FFT) and its inverse IFFT are faster versions of DFT and IDFT, involving fewer multiplications but requiring the number of samples *N* be a power of two [5]:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N} \quad , \qquad k = 1,2 \dots, N \qquad (2.3)$$

$$x_n = \sum_{k=0}^{N-1} X_k e^{j2\pi kn/N} \quad , \qquad n = 1,2 \dots, N \qquad (2.4)$$

FFT and IFFT are at the heart of the OFDM modem. Each OFDM symbol is split into *N* parallel frequency-domain samples containing modulated data bits. By passing these samples through the *N*-tap IFFT, the transmitter generates a series of *N* time-domain samples, each containing information from all frequency-domain samples of an OFDM symbol. An OFDM symbol is formed when the *N* time samples are arranged back in a serial fashion. Fig. 2.2 shows how *N* = 4 subcarriers relate in time and frequency, while Fig. 2.3 shows how an OFDM symbol (shown in bold) is formed from *N* subcarriers. Once an OFDM symbol (i.e., a series of *N* time-domain samples) has arrived at the receiver, the latter passes time-domain samples through the *N*-tap FFT to retrieve the corresponding *N* subcarriers and then extracts data bits by demodulation. The value of *N* of an FFT/IFFT varies from one standard to another [3][4].

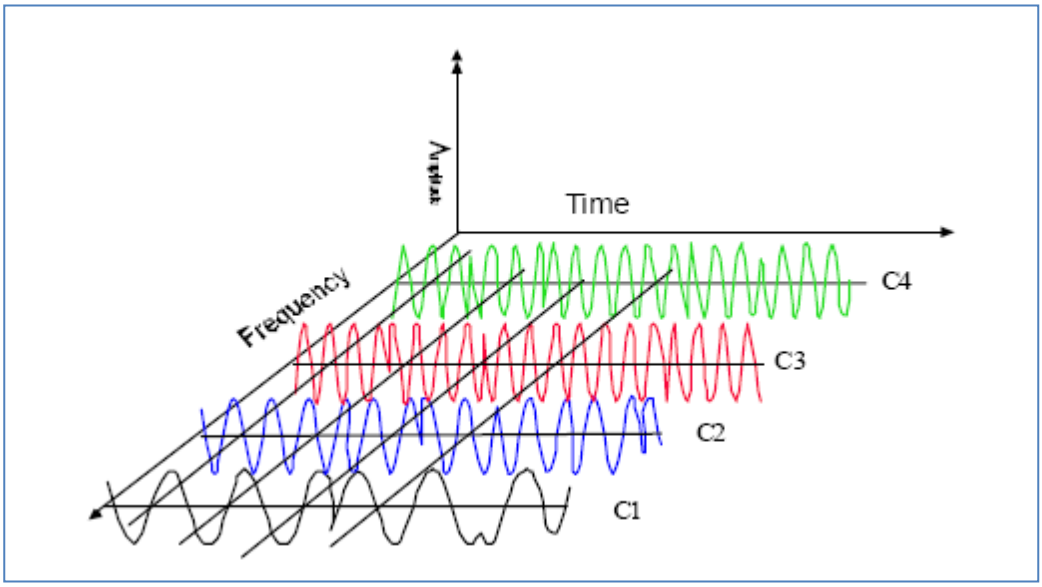


Figure 2.2: Time and Frequency Relationship among Multiple Subcarriers [3]

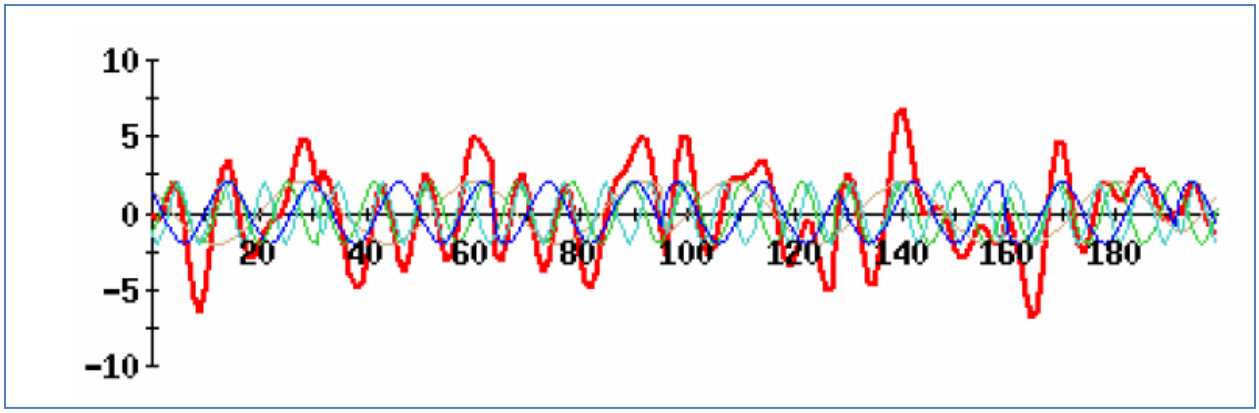


Figure 2.3: OFDM Symbol Obtained from Subcarriers Shown in Fig. 2.2 [3]

2.1.2. Cyclic Prefix

To deal with a multipath delay spread, OFDM relies on a guard interval placed at the beginning of each OFDM symbol. The guard interval is made wide enough to ensure that delayed paths from previous symbols do not overlap (i.e., not causing ISI) with new symbols. If the guard interval is left empty (no samples), the subcarrier orthogonality is compromised (i.e. each subcarrier characteristics are no longer independent of neighbouring subcarriers), causing inter-carrier interference (ICI) that can lead to complete loss of an OFDM symbol [4]. To avoid this problem, the guard interval is filled with a cyclic prefix made of copies of the last N_g time-domain samples of the OFDM symbol, as shown in Fig. 2.4. Using the cyclic prefix guarantees that the subcarrier orthogonality is maintained, as long as N_g is large enough to accommodate the maximum delay spread. The cost of using the cyclic prefix is a loss of signal energy on redundant information [1].

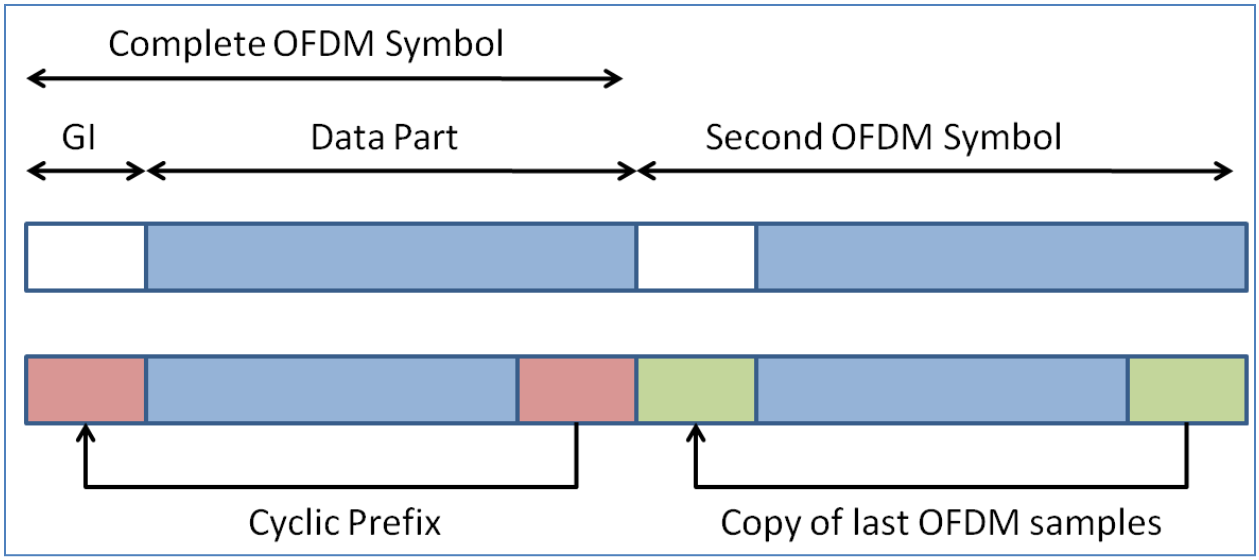


Figure 2.4: Guard Interval with Cyclic Prefix [1]

2.1.3. Windowing

When combining the series of OFDM symbols one after the other, sharp transitions may appear between adjacent OFDM symbols, as shown in Fig. 2.5. Such transitions (in the time domain) cause a slow decay in the frequency delay, which introduces unwanted out-of-band spectra [4].

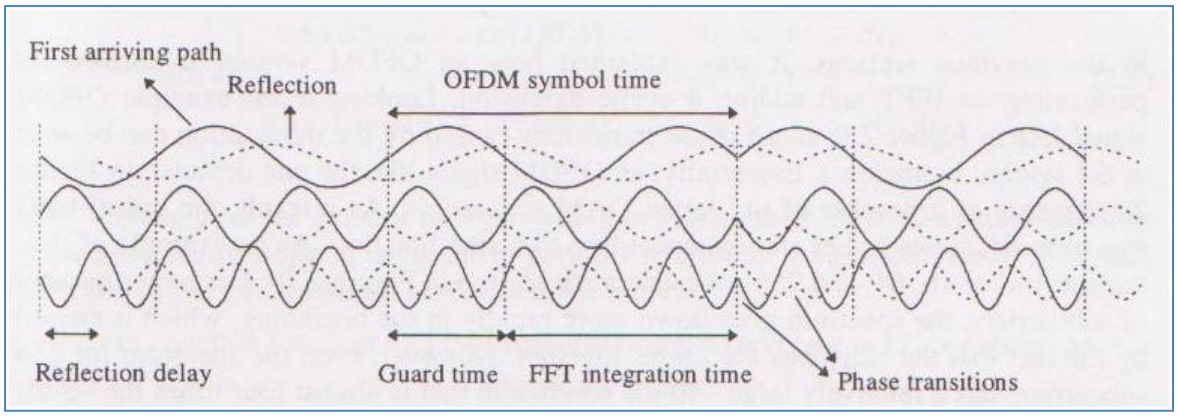


Figure 2.5: Sharp Transition between Adjacent OFDM Symbols [4]

Fig. 2.6 shows how increasing the number of subcarriers per OFDM symbol can help reduce the out-of-band spectra. However, even with a large number of subcarriers, the unwanted spectra can still be significant. To avoid this problem, a windowing function is used to smooth decaying of the first and last time-domain samples of an OFDM symbol to zero. There are a number of window designs available in literature. One common design is the raised cosine window shown in Eq. (2.4) [4].

$$w(t) = \begin{cases} 0.5 + 0.5\cos(\pi + t\pi/(\beta T_{SYM})) & 0 \le t \le \beta T_{SYM} \\ 1 & \beta T_{SYM} \le t \le T_{SYM} \\ 0.5 + 0.5\cos((t - T_{SYM})\pi/(\beta T_{SYM})) & T_{SYM} \le t \le (1+\beta) T_{SYM} \end{cases} \quad (2.4)$$

The variable β in Eq. (2.4) is the roll-off factor which defines the amount of sample overlap between adjacent OFDM symbols. The more samples overlap from adjacent OFDM symbols, the more out-of-band spectra are reduced. However, overlapping too many samples comes at the cost of reduced effective guard interval length [4].

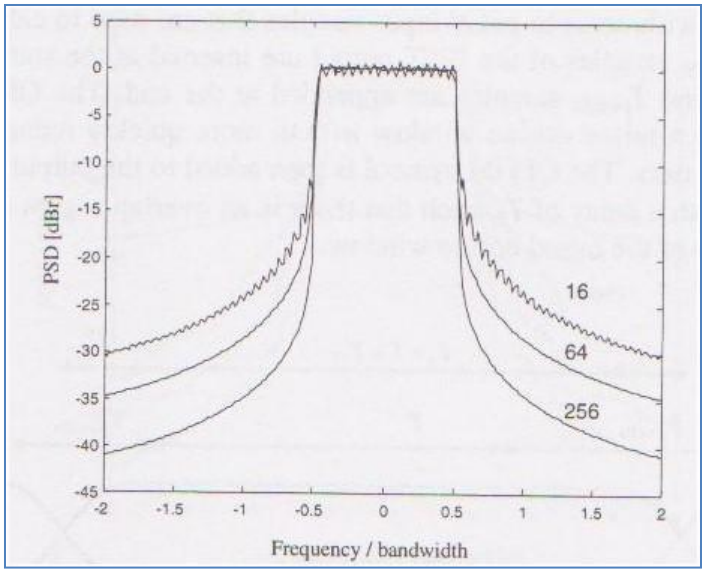


Figure 2.6: OFDM Out-of-Band Spectra [4]

2.1.4. Orthogonal Subcarriers

When designing OFDM subcarriers, three issues must be considered: the multipath delay spread, required data rate, and the available bandwidth. Knowing in advance the approximate delay spread of the propagation environment gives an idea of the guard interval (cyclic prefix) length T_{cp}. To reduce the guard interval's share of the signal energy, the OFDM symbol duration T_{SYM} is then chosen to be several times larger than the guard interval. Once appropriate symbol duration is chosen, the subcarrier spacing δf is set to $1/T_{SYM}$ [4].

Determining the number of needed subcarriers depends on the data rate and δf. Each subcarrier is capable of carrying a certain number of coded bits N_{CBPS}, which depends on the chosen modulation scheme and error-control code. The number of subcarriers N_s is then given by Eq. (2.5):

$$N_s = \frac{datarate \times (T_{SYM} + T_{cp})}{N_{CBPS}} \qquad (2.5)$$

Note that $N_s \times \delta f$ cannot exceed the available bandwidth. Another constraint on N_s is that the N-tap FFT/IFFT needs its N to be a power of two. It is preferable to have $N_s < N$ and insert zero subcarriers at the band edges to make N a power of two. These zero subcarriers are used for oversampling purposes to avoid aliasing. Finally, the sampling time T_s and overall bandwidth are re-adjusted to ensure an integer-valued number of samples for the FFT/IFFT to avoid ICI [4].

2.2. OFDM Receiver Challenges

OFDM receivers are generally more complex than transmitters. In addition to inverting operations carried out at the transmitter, the receiver must also compensate for distortions caused by channel effects as well as by hardware limitations. The methods used to perform distortion compensation are not specified in the WLAN standard [6].

There are three main sources of distortion. The first one is imprecise timing of OFDM symbols. The second one is related to the mismatch between the transmitter and receiver oscillators. Finally, the third source is channel impairments, such as fading and noise, which are discussed in detail in Appendix A. The first two sources are due to hardware limitations, as described next.

2.2.1. Symbol Timing Offset

When transmitted OFDM symbols arrive at the receiver, their exact starting point is not known. Proper timing synchronization at the receiver is crucial to achieving a correct alignment of each OFDM symbol within its FFT window. If not synchronized properly, subcarriers from different OFDM symbols may be present within the same FFT window, causing ISI and ICI [4].

Letting n_ϵ denote the number of offset samples within a single FFT window, the timing offset can be expressed as

$$\epsilon_t = n_\epsilon T_s \qquad (2.6)$$

Orthogonaility is maintained if all samples within the FFT window are from exactly one transmitted OFDM symbol, and this is precisely why the cyclic prefix is very important: it allows for robust timing synchronization [8]. As long as the symbol starting point (chosen by the receiver) is within the cyclic prefix, subcarrier orthogonality will be maintained. This requirement can be expressed as follows, for a given maximum delay spread τ_{MAX} [17]:

$$\frac{\tau_{MAX}}{T} - N_g < n_\epsilon < 0 \qquad (2.7)$$

If non-zero n_ϵ lies within the range given by Eq. (2.7), a slight phase rotation $\varphi_i = 2\pi\delta_i\epsilon_t$ is induced in the frequency domain at each subcarrier δ_i. Such phase rotations are easily compensated by a channel equalizer [4]. When n_ϵ is outside the bounds of Eq. (2.7), orthogonality is lost, and irreducible ISI and ICI arise [17].

2.2.2. Oscillator Imperfections: Carrier Frequency Offset (CFO)

Carrier frequency offset (CFO) is caused by the receiver local oscillator frequency not matching the carrier frequency of the transmitter. CFO is modeled as a phase rotation in time domain which increases with each sample. Given the frequency offset Δf, the normalized frequency offset is given by $\epsilon_f = \Delta f / T_{SYM} = \Delta f \cdot \delta f$ where δf is the subcarrier spacing. Assuming no timing offset and slow fading channel, the *n*-th sample of the received signal *r* becomes phase-rotated due to CFO and can be expressed as [9]:

$$r(n) = \left\{ \sum_i h_i(t) \cdot s(nT - \tau_i) \right\} e^{2\pi j n \epsilon_f / N} + \omega(n) \qquad (2.8)$$

In Eq. (2.8) *s(t)* represents the transmitted signal, h_i and τ_i characterize the *i*-th tap of the multipath channel, and $\omega(\tau)$ denotes AWGN noise. After the FFT, the frequency-domain effect of CFO is seen as a frequency shift and rotation of the subcarriers, plus ICI $I(k)$ for each subcarrier k, as given in Eq. (2.9) [9].

$$R(k) = S(k)H(k) \left(\frac{\sin \pi\epsilon_f}{N \sin\left(\frac{\pi\epsilon_f}{N}\right)} \right) e^{\frac{j\pi\epsilon_f(N-1)}{N}} + I(k) + W(k) \qquad (2.9)$$

Where, S(k) is the frequency domain representation of the transmitted signal s(t) and H(k) is the channel transfer function. CFO effects are the same for all subcarriers in one OFDM symbol. ICI will occur when CFO is large enough to cause a subcarrier shift larger than the subcarrier spacing. The degradation in SNR due to frequency offset is [4]:

$$D_{freq} \cong \frac{10}{3 \ln 10} (\pi\epsilon_f)^2 \frac{E_s}{N_o} \qquad (2.10)$$

The IEEE 802.11a standard specifies the frequency offset tolerance to be a maximum of ± 20 ppm (parts per million), which corresponds to $\pm$100KHz for the 5GHz center frequency.

2.2.3. Oscillator Imperfections: Sampling Frequency Offset (SFO)

Sampling frequency offset (SFO) is caused by non-identical sampling clock at the transmitter and receiver. In other words, SFO arises when the receiver sampling time T_s' is different from the transmitter sampling time T_s. The resulting offset is $\zeta = (T_s' - T_s)/T_s$ which contributes to the phase offset $\theta_o(nT_s') = (1 + \zeta)\epsilon_f nT_s$ and also causes a symbol window drift in time as shown in Fig. 2.7 [17].

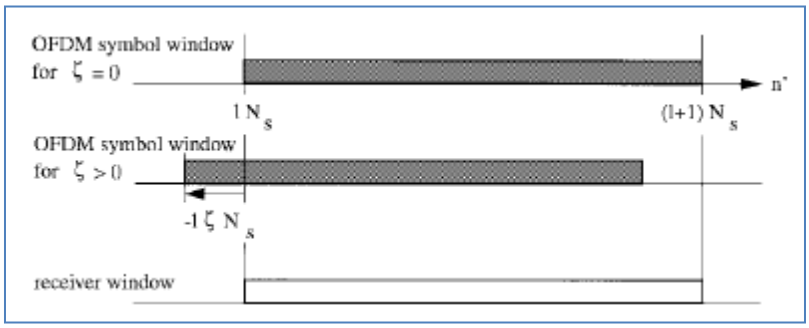


Figure 2.7: OFDM Symbol Window Drift [17]

Assuming no time synchronization errors, the received frequency-domain signal (after the FFT) is given by [17]:

$$R_l(k) = \left(e^{j\pi\varphi_k} \cdot e^{j2\pi(\frac{lN_s+N_g}{N})\varphi_k} \right) \cdot S(k)H(k) + I(k) + W(k) \qquad (2.11)$$

where *l* denotes the symbol number, and φ_k is the local frequency offset which consists of both the CFO and SFO:

$$\varphi_k = \Delta f + \zeta \cdot k \qquad (2.12)$$

It can be seen from Eq. (2.11) and (2.12) that CFO affects all subcarriers in an OFDM symbol in the same way, while SFO affects each subcarrier individually (also see Fig. 2.8). The phase increment from one OFDM symbol to the next can be derived from Eq. (2.11) as [17]:

$$\Delta\varphi_k = 2\pi\left(\frac{N+N_g}{N}\right)\varphi_k \tag{2.13}$$

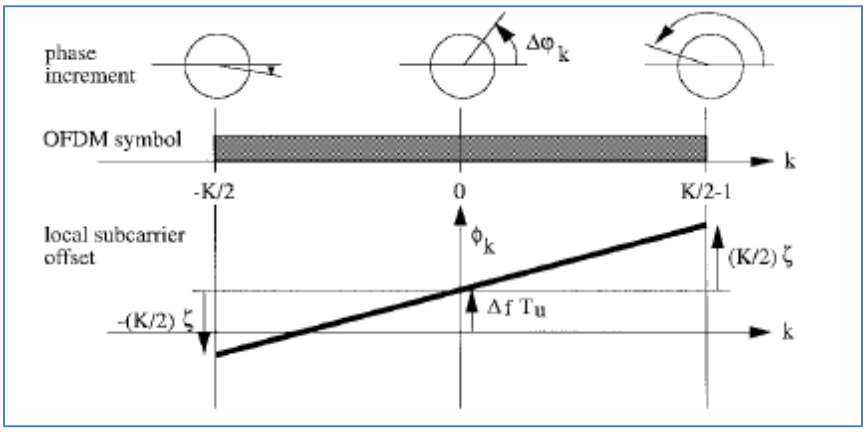


Figure 2.8: CFO and SFO effect on Subcarriers[17]

2.2.4. Oscillator Imperfections: Phase Noise

Phase noise is caused by the oscillator jitter that results in data not being modulated at exactly one frequency [29], and this may cause ICI [4]. Phase noise introduces a random phase rotation to the transmitted signal $r(n) = s(n) \cdot e^{j\varphi(n)}$ [19]. The phase noise is modelled as a discrete-time Wiener process which is generated by the summation of a white Gaussian process $\Delta\varphi(n)$ with each sample [18]:

$$\varphi(n) = \varphi(n-1) + \Delta\varphi(n) \tag{2.14}$$

$\Delta\varphi(n)$ has a zero mean and a variance which depends on the quality of the oscillator:

$$\sigma_{\Delta\varphi}^2 = 4\pi^2 f_c^2 c_q T_s \tag{2.15}$$

where the oscillator quality factor $c_q = \Delta f_{3dB}/(\pi f_c^2)$ is defined by the single-sided 3dB bandwidth of the Lorentzian spectrum Δf_{3dB} [29].

Although the phase rotations induced by phase noise are random, they are common to all subcarriers and are usually correlated from one OFDM symbol to the next [4]. In the frequency domain after FFT the received signal appears as [19]:

$$R(k) = S(k) + \frac{j}{N}\sum_{m=0}^{N-1} S(m) \sum_{n=0}^{N-1} \varphi(n) \cdot e^{j\left(\frac{2\pi}{N}\right)(m-k)n} \tag{2.16}$$

The severity of the error depends on the value of m: if $m = k$, a common phase rotation based on the average phase noise is seen; otherwise, ICI will be present [19].

2.3. OFDM Advantages and Disadvantages

There are a number of advantages of OFDM systems for wireless communications. First, OFDM is resistant to ISI and ICI, given a proper choice of the cyclic prefix. Second, subcarrier orthogonality allows OFDM to pack subcarrier bands closer together, as seen in Fig. 2.1, making it more spectrally efficient than conventional FDM. Orthogonality is also easier to maintain in OFDM systems than in TDMA or CDMA systems, while FFT/IFFT implementations are relatively simple and well-developed. OFDM is also resistant to frequency selective fading, where multipath fades affect a few subcarriers rather than destroy the entire band, as illustrated in Fig. 2.9 [1].

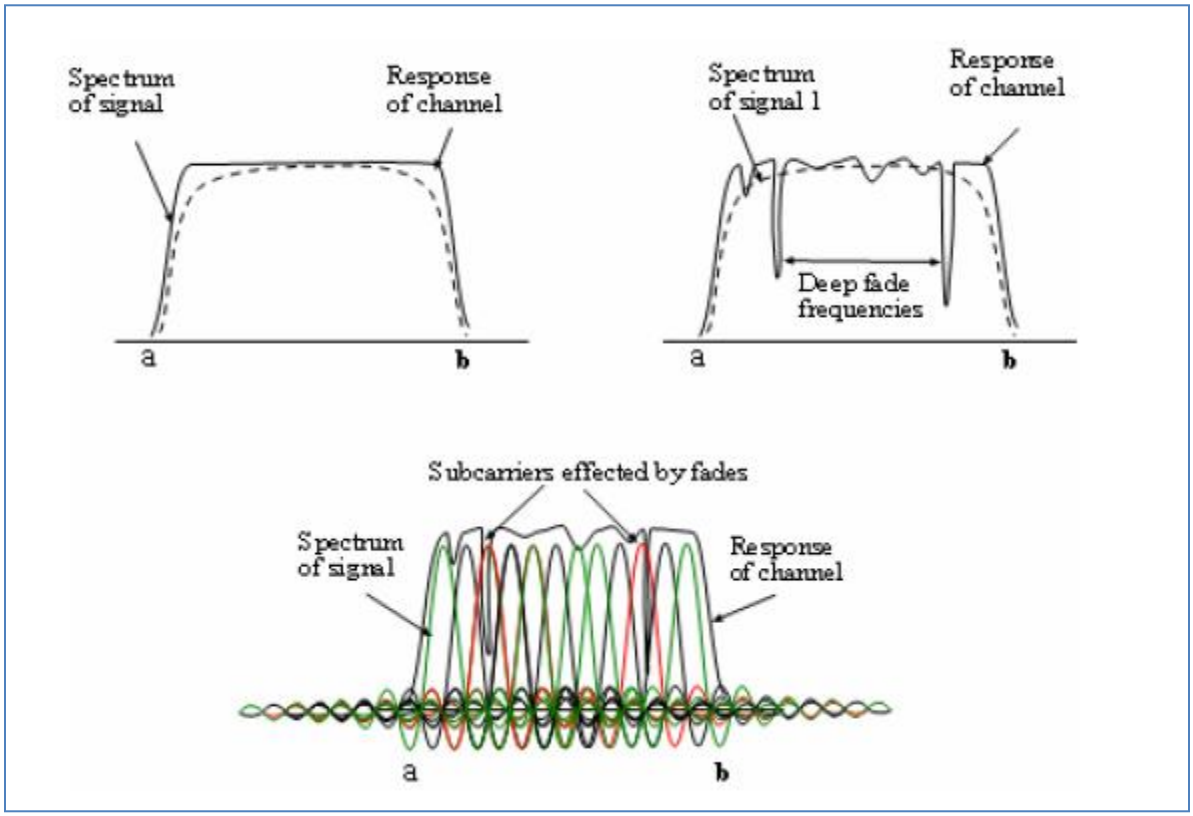


Figure 2.9: Frequency Selective Fading [3]

Although OFDM has many advantages, it is not without problems. The two most serious limitations are synchronization sensitivity and Peak-to-Average Power Ratio (PAPR). Due to the requirement to maintain subcarrier orthogonality, OFDM is sensitive to time and frequency synchronization errors [1, 2]. The most serious synchronization errors are related to symbol timing offset, carrier frequency offset, and sampling frequency offset, as discussed in the previous sections. The receiver must estimate and compensate for these errors either in the time domain or the frequency domain.

PAPR is caused when a number of independently modulated OFDM subcarriers are added together coherently to form the OFDM symbol. The power of the OFDM symbol in the time domain may experience spikes at different points, which may cause a reduction in RF power amplifier efficiency. Large PAPR also makes the design of Analog-to-Digital (ADC) and Digital-to-Analog (DAC) converters more complex [1, 4].

Many solutions to the PAPR problem have been investigated in the literature [4]. One method is to use signal distortion techniques, such as clipping, to remove the spikes in the time-domain as shown in Fig. 2.10. Clipping, however, causes the out-of-band spectra to increase (see Fig. 2.6), which in turn requires better windowing functions. Instead of clipping, special windowing functions can also be used. Another method is the use of proper scrambling to reduce the probability of high PAPR. Other methods include using special forward error coding, which excludes OFDM symbols that cause high PAPR [4].

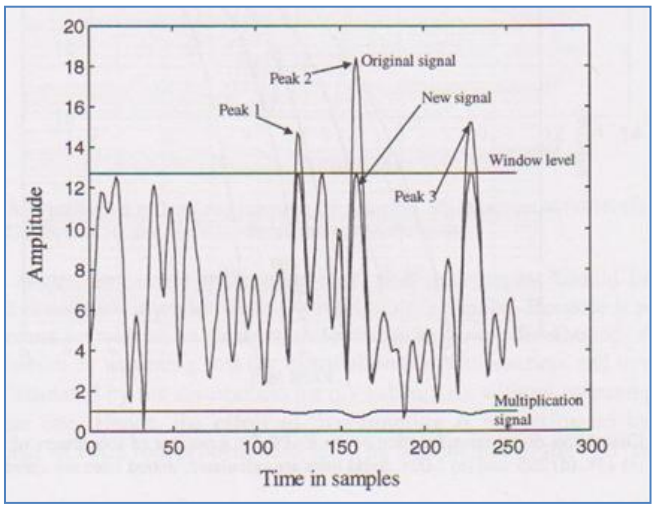


Figure 2.10: OFDM Power Spikes Causes of PAPR [4]

Chapter 3: Synchronization, Channel Estimation and Phase Tracking

Synchronization, channel estimation (with equalization), and phase tracking are the key computational tasks that must be performed by the OFDM-based receiver, in addition to inverting transmitter operations performed on the data. This chapter outlines common problems and solutions associated with the tasks in question.

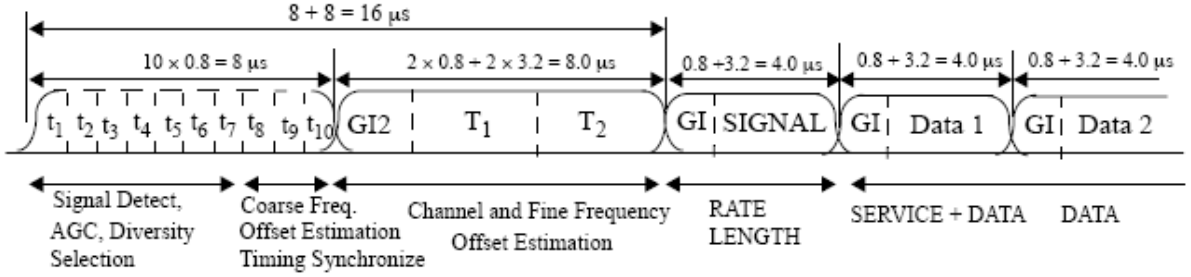
3.1. 802.11a Preamble

The importance of having a preamble for each OFDM frame was first studied in [8]. The idea is to transmit multiple copies of known symbols (sample sequences) before the data. The receiver uses these samples for synchronization (both the symbol timing and the carrier frequency). In 802.11a, the preamble is 16 microseconds long consists of two 8-microsecond parts: the short training sequence (STS) and long training sequence (LTS) [6].

The STS is constructed by first generating 53 subcarriers as shown below:

$$S_{-26,26} = \sqrt{(13/6)} \times \{0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 0,$$
$$0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0,0\}$$

Next, additional 11 zero subcarriers are inserted, and the 64-tap IFFT is used to generate the 64 time-domain STS samples. The resulting samples are treated as four short (0.8-microsecond long) OFDM symbols with 16 samples each. These are then copied to obtain the total of 10 short OFDM symbols t_1, t_2, …, t_{10} shown in the figure below [6].



The LTS is constructed by first generating 53 subcarriers as shown below:

$$L_{-26,26} = \{1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0,$$
$$1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1\}$$

Next, additional 11 zero subcarriers are inserted, and the 64-tap IFFT is used to generate the 64 time-domain LTS samples. The resulting OFDM symbol is extended by its copy, yielding the total of 128 samples, thus forming two symbols *T1* and *T2*. Then, the 32-sample cyclic prefix is appended to the beginning of LTS [6].

3.2. Synchronization

There are two basic methods of synchronization, using either preamble training symbols, or the cyclic prefix (blind synchronization). In practice, blind synchronization helps reduce overhead but its performance is worse than using training symbols. Blind synchronization is acceptable for circuit-switched application, but the accuracy needed for packet-switched applications makes preamble-based synchronization more suitable [4][8].

The preamble processing for receiver synchronization includes the following steps (see Fig. 3.1): automatic gain control (AGC), packet detection, time synchronization, and frequency synchronization. These steps are briefly summarized next.

Figure 3.1: Basic Synchronization Process

3.2.1. Automatic Gain Control (AGC)

As the demodulated received signal from the 5GHz band tends to have rapid variations in the incoming signal, the A/D converters can saturate causing synchronization difficulties and SNR degradation. AGC is used to modify the front-end gain until A/D conversion starts operating in the linear (non-saturated) region. The AGC process typically needs around 3-5 STS symbols to complete. Once AGC has stabilized the gain, the packet detector is enabled to monitor if a packet is present. More details on AGC are presented in [25, 30, 31].

3.2.2. Packet Detection

Most packet detectors in preamble-based synchronization circuits use the auto-correlation method due to its good performance and ease of implementation [4]. It was first introduced in [8]. The idea is to correlate the data with its copy using a certain window size. Letting *L* denote the length of a training symbol (16 for the STS and 64 for the LTS), the corresponding auto-correlation window size is *2L*, and the auto-correlation output is given by [8]:

$$P(d) = \sum_{m=0}^{L-1} \left(r^*(d+m) \cdot r(d+m+L) \right) \tag{3.1}$$

Where *r* is the received sequence of time-domain samples, and *d* is the first sample in the auto-correlation window. The later samples can be computed iteratively as:

$$P(d+1) = P(d) + r^*(d+L) - r^*(d) \cdot r(d+L) \tag{3.2}$$

The auto-correlation output is then normalized by the signal energy (which can also be computed iteratively) given by

$$R(d) = \sum_{m=0}^{L-1} |r(d+m+L)|^2 \tag{3.3}$$

to obtain the timing metric of interest:

$$M(d) = \frac{|P(d)|^2}{(R(d))^2} \tag{3.4}$$

The value of $M(d)$ is first compared to a certain rising threshold: if this threshold is crossed, the packet detector starts counting the number of samples from that point in time. The counter keeps incrementing with each newly received sample, as long as the value of M, which is updated on a sample-by-sample basis, remains above a certain falling threshold; otherwise, the counter is reset to 0 and waits until the rising threshold is crossed again. If the counter goes beyond a certain value, a packet is said to be detected [10]. Note that the timing metric $M(d)$ given by Eq. (3.4) produces a plateau-like curve over time when two successive training symbols fall within the auto-correlation window. Crossing of the rising threshold would indicate the potential start of the plateau, whereas the falling threshold would indicate the end of the plateau.

3.2.3. Coarse and Fine Time Synchronization

The challenge in timing synchronization is to find the optimum timing index at which the preamble ends, or equivalently, the first data OFDM symbol starts. Having a plateau-like behaviour of the timing metric makes it difficult to identify a precise timing index. Some research papers suggested the use of cross-correlation instead of auto-correlation [14, 15, 16]. It was found that cross-correlation estimates the timing index with less variance but at the cost of higher hardware complexity [10, 14]. To have the benefit of both methods, the authors of [15] have suggested a two-step process: coarse synchronization and fine synchronization. Coarse synchronization uses auto-correlation to find the approximate location of the timing index, while fine estimation uses cross-correlation to find a more precise timing index. In 802.11a, the preamble was in fact designed for such two-step synchronization, where the STS would be used for AGC, packet detection and coarse timing, while LTS would be used for fine timing synchronization.

In the original paper [8], two suggestions were made to find the best timing index. The first suggestion was to take the maximum metric in the plateau as the timing index. The second suggestion was to find two points (one on each side) of the plateau that were 90% of the maximum, and let the timing index be the midpoint between these two points. The analysis performed in [8] showed that the second method performed better.

In [11] the suggested timing index was found at the point *d* where *M*(*d*) drops to half its peak value. The same paper also suggests using only the real part of the auto-correlation *R*(*d*) to reduce hardware complexity:

$$R(d+1) = R(d) + Re\{r^*(d+L) \cdot r(d+2L) - r^*(d) \cdot r(d+L)\} \quad (3.5)$$

Other techniques, such as those based on the autocorrelation difference [12] and autocorrelation sum [13], were studied in [10] that showed that the approach from [11] was better practical choice for coarse time synchronization.

In fine time synchronization, a cross-correlator is used to correlate the received preamble samples *r* with the known original preamble *c*, producing the following metric:

$$\Lambda(d) = \sum_{m=0}^{L-1} c^*(m) \cdot r(d+m) \quad (3.6)$$

Once cross-correlation is carried out, a decision algorithm is used to determine the fine timing index. In [14] two algorithms are proposed: the first method simply identifies the maximum point of the cross-correlation, while the second method finds the point that exceeds a certain threshold. These methods can be expressed via Eq. (3.7) and (3.8) respectively.

$$d_{max} = arg \max_d(|\Lambda(d)|) \quad (3.7)$$

$$d_{xcp} = arg \min_d \big||\Lambda(d)| \geq th \times |\Lambda(d_{max})|\big| \quad (3.8)$$

In 802.11a, the cross-correlator works with the LTS using the window size of 64. The result is two peaks that identify the start of the first and second LTS symbols, as shown in Fig. 3.2.

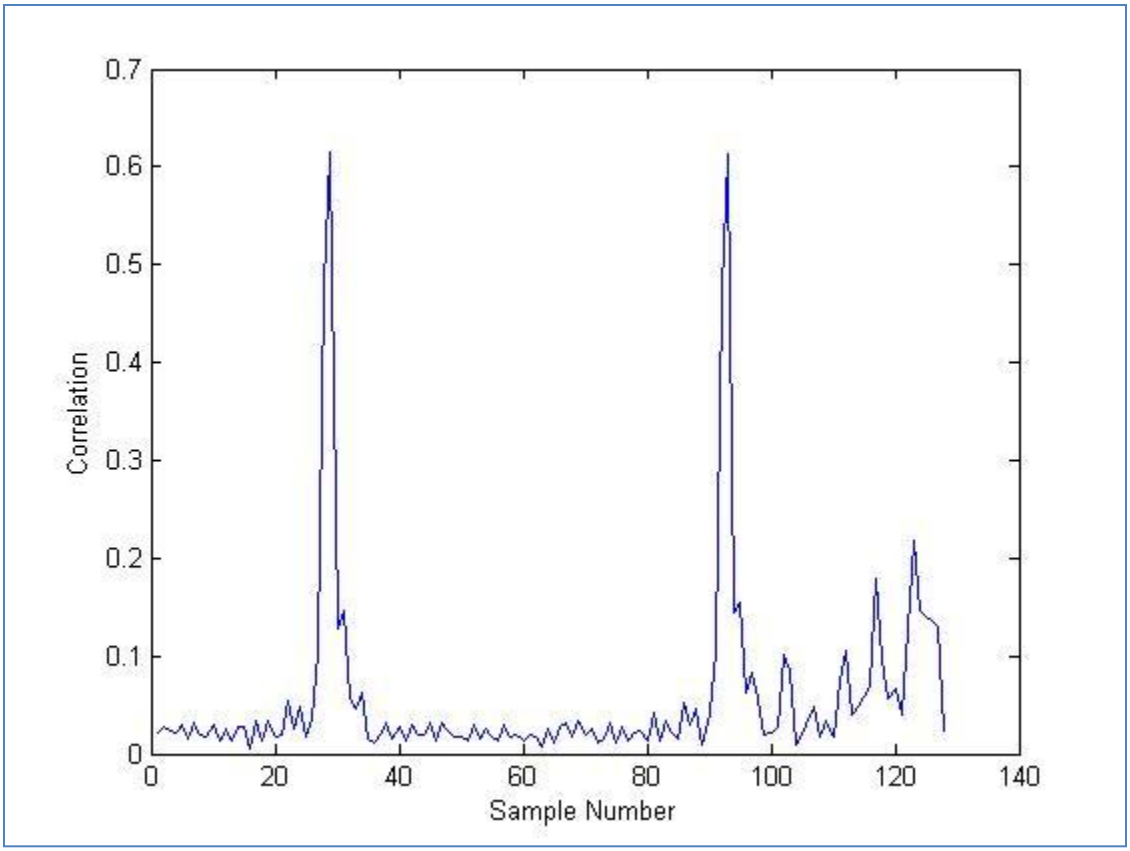


Figure 3.2: LTS Cross-Correlation Peaks

As the cross-correlator suffers from being more computationally complex than auto-correlation [14], the authors of [16] have proposed quantizing the preamble sample values to the power of 2. Such quantization allows expensive multiplication operations to be replaced with simple bit-shift operations. Tests reported in [16] have shown that the hardware could be reduced by 90% without affecting performance. Similarly, [10] have reported approximately 60% hardware savings and, in some cases, better performance than non-quantized cross-correlation.

3.2.4. Frequency Synchronization

In section (2.2.2.1) it was shown how CFO is represented as a phase rotation which accumulates with each sample in time domain. In his paper [9], Moose makes use of a repeated sequence(preamble) to make a Maximum Likelihood(MLE) estimate of the CFO present. By looking at Eq.(2.3) from a frequency perspective using FFT, the received complex envelope is expressed by[9]:

$$r(n) = \frac{1}{N}\left[\sum_{k=-K}^{K} X(k)H(k)e^{2\pi jn(k+e_f)/N}\right]; \qquad n = 0,1,\dots,2N-1 \qquad (3.9)$$

Now, assuming a two repeated symbols of length N, The FFT of the two symbols can be given by[9]:

$$R_1(k) = \sum_{n=0}^{N-1} r(n)e^{-\frac{2\pi jnk}{N}}; \qquad k = 0,1,\dots,N-1 \qquad (3.10)$$

$$R_2(k) = \sum_{n=N}^{2N-1} r(n)e^{-\frac{2\pi jnk}{N}}$$

$$= \sum_{n=0}^{N-1} r(n+N)e^{-\frac{2\pi jnk}{N}}; \qquad k = 0,1,\dots,N-1 \qquad (3.11)$$

From eq (3.9),

$$r(n+N) = r(n)e^{\frac{2\pi je_f}{N}} \rightarrow R_2(k) = R_1(k)e^{\frac{2\pi je_f}{N}} \qquad (3.12)$$

Eq. (3.12) shows that the phase difference between two consecutive symbols is a phase shift proportional to the CFO. Hence, the MLE of the CFO is taken as the angle between two consecutive symbols [9].

Knowing the coarse time index *d* from the auto-correlation, [8] suggests that the angle in question is the angle of the auto-correlation output at time *d*:

$$\varphi_o = \angle P(d) \qquad (3.13)$$

Note that the range and accuracy of the estimated CFO depends on the size of the training sequence used for auto-correlation [8, 9]. Given the sequence length *L*, the coarse frequency offset is [10]:

$$\Delta f = \frac{\varphi_o}{2\pi \times LT_s} \qquad (3.14)$$

Since the angle φ_o is between π and –π, the detectable frequency offset range is $-625kHz \leq \Delta f \leq 625kHz$ for *L* = 16 (STS) used for coarse time synchronization.

Given *L* =64 (LTS) used for fine time synchronization, the detectable frequency offset range becomes $-156.25kHz \leq \Delta f \leq 156.25kHz$. Hence, fine frequency synchronization is possible. It requires another auto-correlation to be performed on the LTS (using *L* =64), which will yield a more accurate estimates of ϕ_0 and then Δf based on Eq. (3.13) and (3.14) [10].

3.3. Channel Estimation

To compensate for multipath fading effects, the receiver must first estimate the channel transfer function (CTF), followed by channel equalization. Zero forcing channel equalization is simply multiplying received data by the CTF inverse [41]. A CTF example is shown in Fig. 3.3.

Channel estimation is usually performed in the frequency domain, where the received preamble is compared with the known preamble to identify the changes introduced in the channel. There are numerous published works on channel estimation techniques, the *least-squares* (LS) and *minimum mean-square error* (MMSE) methods being the most widely used and referenced. The latter provides better performance, while the former has less complexity [20]. It is possible to perform channel estimation not only in the frequency domain, but in the time domain as well, which was proposed in [32] that makes use of the preamble PN sequence. Using the method from [32] can also be beneficial in time-varying channels, but it would require a preamble at the beginning of each OFDM symbol (which is not the case in 802.11a).

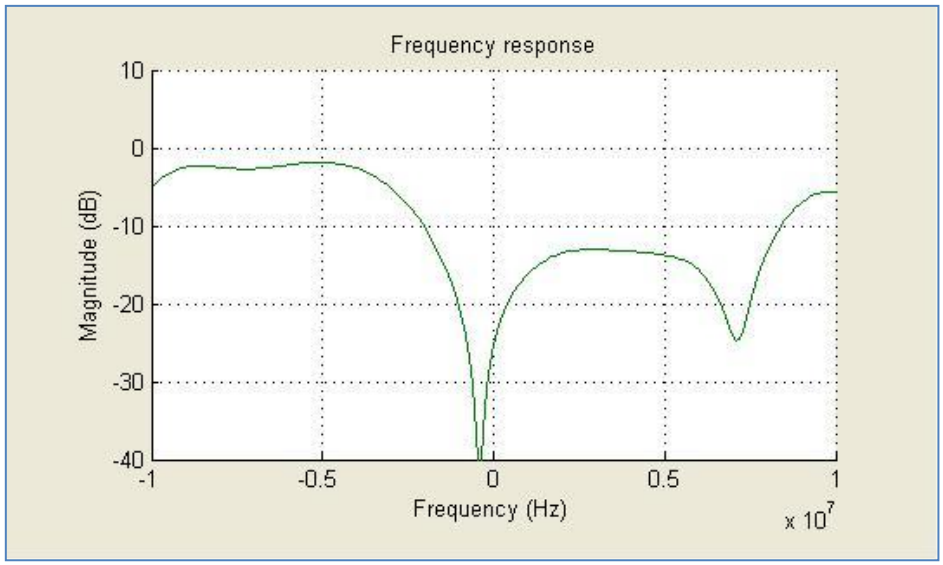


Figure 3.3: Multipath Channel Transfer Function Sample

3.3.1. LS Channel Estimation

The LS estimator [20], also known as zero-forcing estimator, is very simple: the CTF estimate H_{LS} is obtained by dividing (in the frequency domain) the received preamble Y by the known preamble X that consists of only 1's and –1's:

$$H_{LS} = \frac{Y}{X} \tag{3.15}$$

It works well at high SNR, however its performance at low SNR is severely limited. When SNR is low, the received preamble X's subcarriers that underwent deep fading are almost entirely corrupted by noise. Using such faded subcarriers in the denominator in Eq. (3.15) yields infinite CTF estimates. The MMSE estimator overcomes this problem.

3.3.2. MMSE Channel Estimation

The objective of the MMSE estimator is to compute the channel state information (CSI) h_{MMSE}, defined as [20]:

$$h_{MMSE} = \frac{R_{hY}}{R_{YY}} Y \tag{3.16}$$

where R_{hY} and R_{YY} are given by [20, 21]:

$$R_{hY} = E[hY^H] = R_{hh} F^H X^H \tag{3.17}$$

$$R_{YY} = E[YY^H] = XFR_{hh}F^H X^H + \sigma_n^2 I_n \tag{3.18}$$

where $R_{hh} = E[hh^H]$ is the channel auto-correlation matrix, σ_n^2 is the noise variance, and $F = [\frac{1}{\sqrt{K}} e^{-j2\pi nk/K}]$ is the FFT matrix which represents an FFT operation. The CTF, H_{MMSE} is then found by H_{MMSE} = Fh_{MMSE}.

From Eq. (3.16)-(3.18) it can be seen that MMSE estimates not only involve computationally intensive matrix inversions, but also require the knowledge of R_{hh} and σ_n^2 (which may not be readily available and must be estimated as well). Thus, in comparison with the LS estimator, the MMSE estimator is substantially more expensive.

A practical compromise between the LS and MMSE estimation approaches was proposed in [21], using the fact that most of the signals energy is concentrated in first few taps of the channel. This method first obtains an initial LS estimate of the CTF in the frequency domain, and then converts it into the time domain using IFFT. The SNR (and thus σ_n^2) is then estimated as the quotient of the first few channel taps over the remaining taps. Matrix R_{hh} is estimated using the LS method as well [21].

3.3.3. Channel Estimation in Time-Varying Environment

The estimation methods mentioned in 3.3.1 and 3.3.2 work with the preamble only, and the resulting channel estimates are updated only when the next preamble (i.e., the next frame) is received. As long the channel gains do not change significantly during an individual frame (lasting tens to thousands of microseconds in 802.11a), such an approach works well. If the channel is time-varying fast enough, the receiver must utilize the pilot subcarriers to adjust channel estimates within the frame duration [23]. In addition to fast changes in multipath gains, signals from a fast-moving transmitter (relative to the receiver) will experience the Doppler shift that may introduce a more severe change to the channel. Work [22] suggests a method which tracks Doppler shift as well as estimates the delay spread of the channel. In the context of the 802.11a standard designed primarily for the in-door office propagation environment, fast time-varying channels are not a likely scenario.

3.3.4. Channel Equalization

Given H_{LS} or H_{MMSE} after channel estimation, the receiver has quantitative information on signal distortions introduced by the channel. These distortions must be compensated for when processing OFDM data symbols arriving after the preamble. Zero-forcing (ZF) equalizer is commonly used in 802.11a due to its low complexity[42]. Given the revived OFDM symbols *Y*, the ZF-equalizer estimates the original OFDM symbols *X* as in Eq. (3.19). The CTF inverse is given by $C(n) = 1/H(n)$ and *n* represents the samples in one OFDM symbol, $1<n<64$.

$$X(n) = \frac{Y(n)}{H(n)} = C(n) \cdot Y(n) \qquad (3.19)$$

3.4. Phase Tracking

Phase tracking is necessary to compensate for the residual frequency offset (RFO). The RFO is the result of the CFO not being completely compensated in practical situations. The RFO is typically small, and its effects are usually negligible for frames containing less than 20 OFDM symbols. However, as the number of OFDM symbols in a frame increases, the RFO starts accumulating with each symbol to a point where the resulting phase rotations start causing significant errors [25][28]. It is sufficient to examine the pilot subcarriers to compensate for such phase rotations.

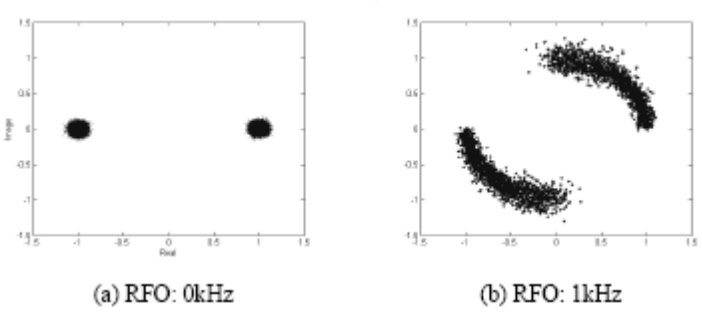


Figure 3.4: Effect of RFO [28]

3.4.1. Phase Compensation Process

Given the received pilots *p* and the known pilots *c* of OFDM symbol *l*, the phase rotation caused by the RFO is found by [25]:

$$\varphi_{RFO}(l) = \angle \sum_{k=1}^{N_p} \left(\gamma \cdot c_l(k) \cdot p_l(k)\right) \qquad (3.20)$$

where *k* is the pilot number, N_p is the total number of pilots (4 in 802.11a), and γ is either 1 or -1 depending on the PN sequence used by the transmitter. Alternatively [26],

$$\varphi_{RFO}(l) = \frac{1}{N_p} \sum_{k=1}^{N_p} \angle\left(\gamma \cdot c_l(k) \cdot p_l(k)\right) \qquad (3.21)$$

After the phase is found, each data subcarrier within the OFDM symbol in question is rotated to obtain its corrected value *D*, according to the following equation [26]:

$$D_l(k) = E(k) \cdot e^{-\varphi_{RFO}(l)} \qquad (3.22)$$

where $E_l(k)$ represents the *k*-th data subcarrier of the *l*-th OFDM symbol arriving from the channel equalizer.

3.4.2. Feedback-type Phase Tracking

In [26], the authors argue that performing both angle estimation and rotation can cause fatal delays in the receiver. For this reason, they propose a feedback-based compensator using the angle obtained from the previous symbol for rotation of the current symbol. Another idea was to separate rotation used for pilots and data to avoid unwanted changes to pilot values. In [27], the same authors state that the FFT delay is more significant than that of the angle calculation and rotation, and hence using the angle estimated from the current pilots poses no problems. Their tests confirmed that using the current pilots gives more accurate results than using those from the previous symbol.

In [28], the authors suggest that a better angle estimate can be found via the feedback from the Viterbi decoder. In this method, decoded data is re-encoded to obtain phase differences with respect to the data from the channel equalizer. The latter is then phase-compensated accordingly. It is clear that such a scheme would require symbol buffering and involve double decoding, both of which would increase the receiver hardware cost.

3.4.3. Compensating for Oscillator Imperfections

In addition to compensating for the RFO, the phase tracking circuit can be modified to handle other oscillator imperfections, such as the SFO and phase noise (see Chapter 2). The authors of [33] describe a feedback-based method compensating for both the RFO and the SFO. The estimate of the latter is updated every couple of OFDM symbols and relies on autocorrelation between pilots of successive OFDM symbols to track the changes. As for phase noise, [27] proposes the use of a modified feedback-based phase compensator from [26]. The use of Kalman filtering in [18] was shown to significantly improve the system performance affected by phase noise. In [24], the phase tracker complexity was reduced due to the use of a simplified state-space model to track the RFO as well as random phase error.

Chapter 4: Reference Receiver Design

This chapter provides a description of our reference 802.11a receiver implemented in MATLAB. This implementation includes all steps needed to reconstruct PSDU (see Appendix B), including synchronization, channel estimation (with equalization), and phase tracking. We have tried to simulate hardware based computations as much as possible, e.g., using CORDIC (see Appendix D). Unlike most published works that focus on specific parts of the receiver (assuming ideal performance of the remaining parts), our MATLAB implementation covers a complete digital signal processing path that starts from the time-domain unsynchronized samples coming out of the channel and ends with the decoded and descrambled data bits going into the MAC layer.

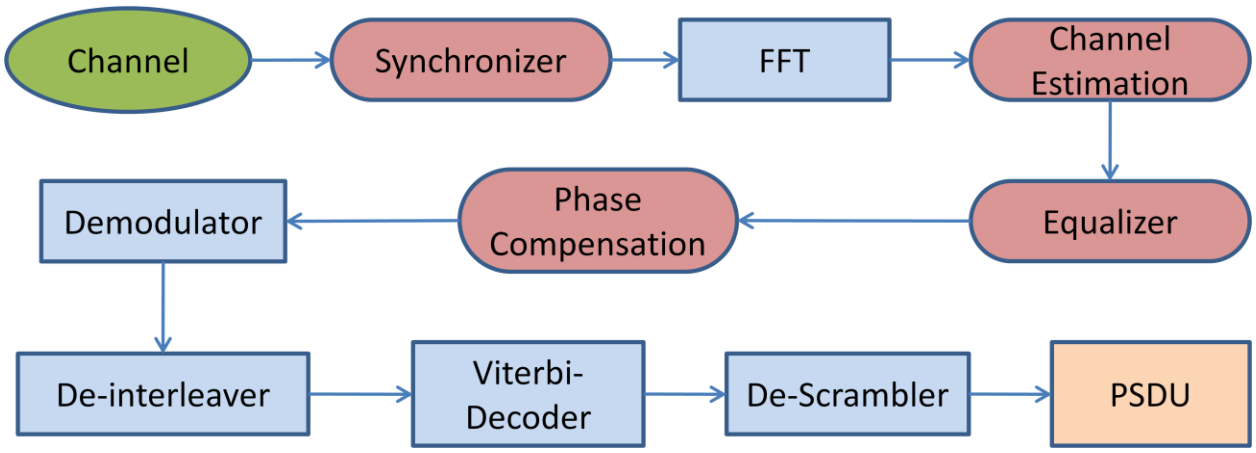


Figure 4.1: 802.11a Reference Receiver Block Diagram

The reference receiver block diagram is shown in Fig. 4.1, where the rectangular blocks represent the inverse of the transmitter operations, and the oval blocks represent additional tasks to be performed during reception (the 802.11a standard does not specify how these tasks are to be implemented). Fig. 4.2 shows a summary of the operations carried out by the Cartesian reference receiver in the frequency domain (past the FFT). Details of each of the oval block designs are presented subsequently.

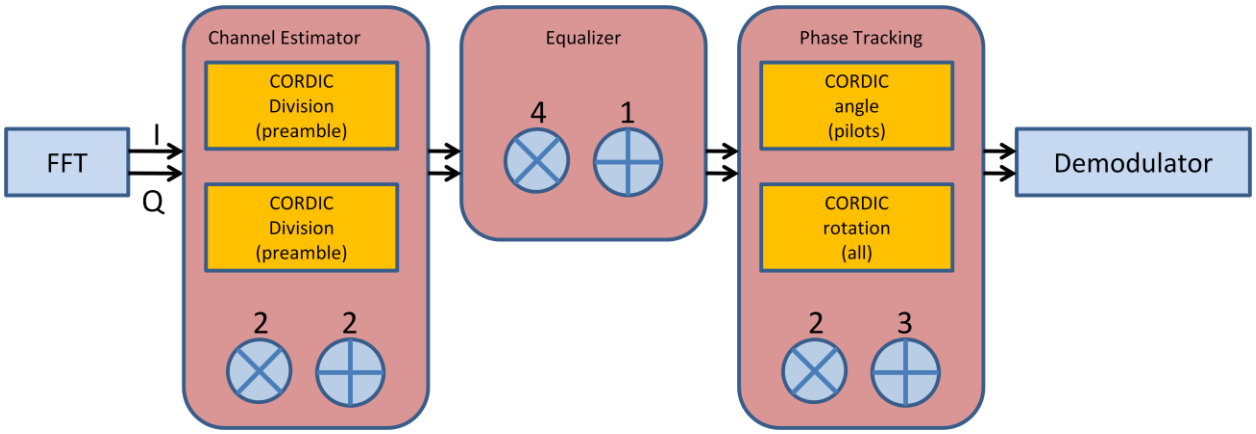


Figure 4.2: Frequency Domain Cartesian System Operation Summary

4.1. Synchronizer Design

Our synchronizer uses standard techniques already discussed in Chapter 3 and based mostly on [10]. It includes packet detection, coarse/fine time synchronization, and coarse/fine frequency synchronization.

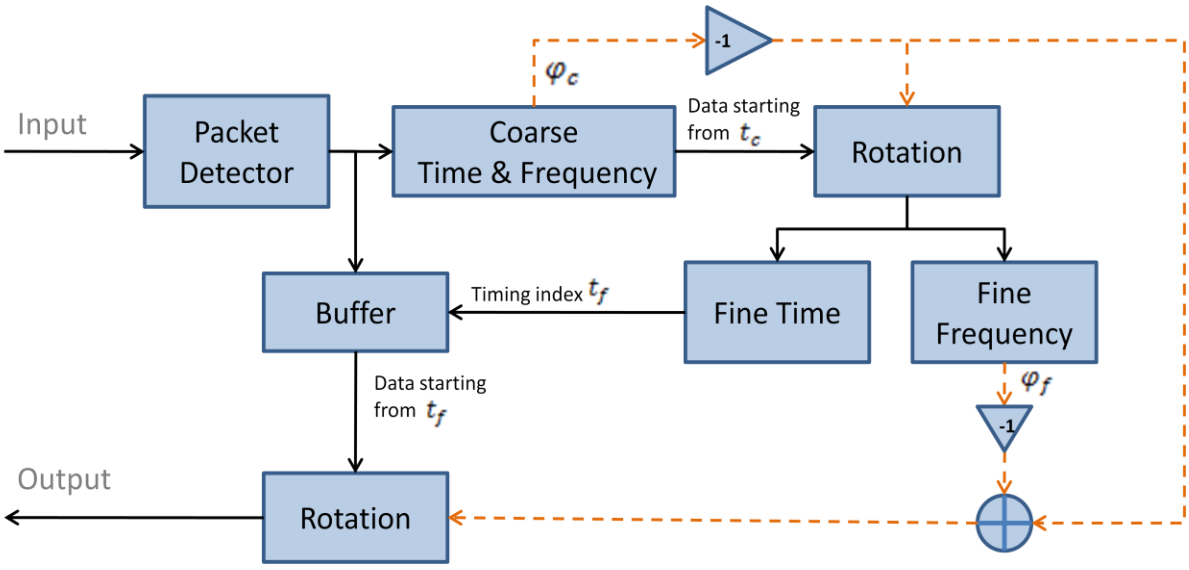


Figure 4.3: Synchronizer Block Diagram

4.1.1. Packet Detection

The packet detector is comprised of two parts: the auto-correlator of window size L = 16 and a sample counter. The corresponding timing metric $M(d)$ at each sample d, as per Eq. (3.4), is compared to the rising threshold Th_r. If $M(d)$ crosses and remains above Th_r for more than Th_{cnt} counted samples, a packet is said to be detected. We set Th_r = 0.25 and Th_{cnr} = 30. The packet detection block diagram is shown in Fig. 4.4.

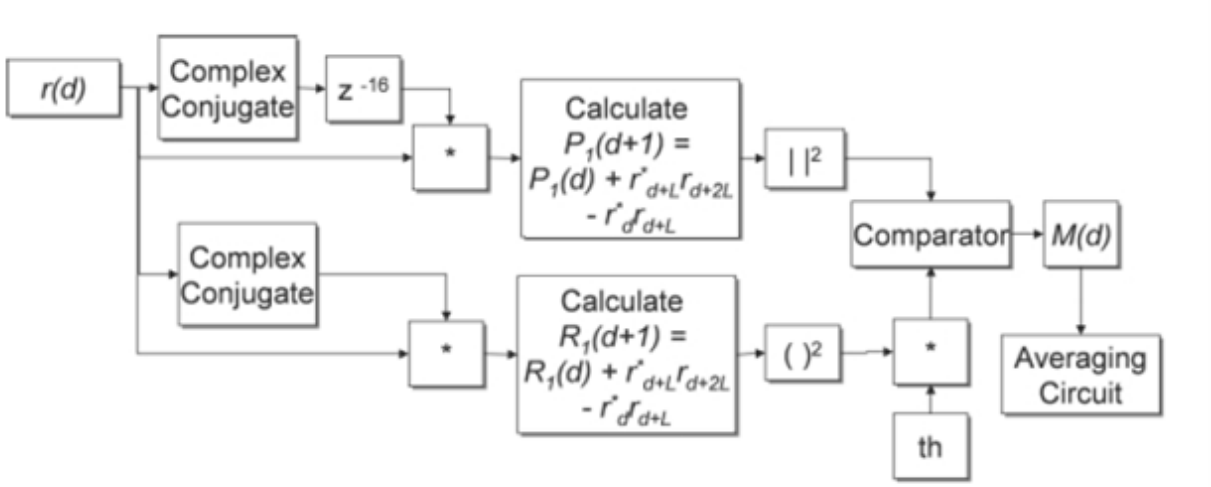


Figure 4.4: Packet Detector [10]

Recall that computing $M(d)$ involves a division operation per Eq. (3.4) and as implemented in [39]. To save hardware resources, [40] suggests replacing divisions with simple comparisons, which we use in our design as well:

$$|P(d)|^2 > Th_r \cdot (R(d))^2 \qquad (4.1)$$

Since our Th_r is a power of 2, the multiplication is reduced to a simple shift operation.

4.1.2. Coarse Synchronization

Once a packet is detected, the coarse time synchronization process is activated: the receiver keeps computing $M(d)$, but now it is checked against the falling threshold Th_f, which we set to 0.25 as well. Typically, Th_f will be crossed somewhere in the middle of the last STS symbol, hence the coarse timing point of interest is usually taken to be at $t_{coarse} = d + t_{adjust}$, where t_{adjust} is taken to be around 16 samples (one STS symbol). The coarse frequency offset is estimated by computing the phase of $P(d)$ at d where Th_f has been crossed. This phase is computed by CORDIC working in VM circular coordinates (see Appendix D), which is then normalized by the window size L. Since $L = 16$ is a power of 2, such normalization is a simple shift operation. The coarse synchronization block diagram is shown in Fig. 4.5.

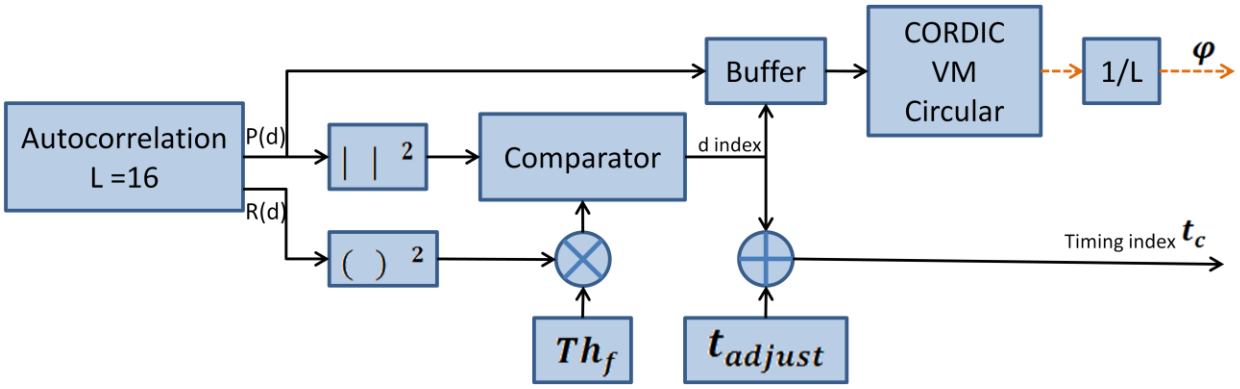


Figure 4.5: Coarse Synchronization Block Diagram

4.1.3. Fine Synchronization

After acquiring the coarse time and frequency estimate, the coarse frequency offset is compensated by phase rotations applied to the subsequently received samples of the LTS symbols. These rotations are performed by CORDIC working in RM circular coordinates (see Appendix D). To estimate fine frequency offset, the same steps as shown in Fig. 4.5 are performed, but using the LTS samples with $L = 64$, $Th_r = 0.25$, and $Th_f = 0.5$. The estimated fine phase (from the LTS auto-correlation) is then added to the estimated coarse phase (from the STS auto-correlation) to form the overall phase offset. This total offset is then used to rotate (using CORDIC) all the incoming data samples.

For fine time synchronization, a cross-correlation of the two LTS symbols is performed, which gives the related metric $\Lambda(d)$, as per Eq. (3.6). It will produce two peaks identifying the starting times of the first and the second LTS symbols. We take the time index of the first peak as the fine timing estimate and add $2L = 128$ to it, which gives us the starting point of the data samples.

4.2. Channel Estimation and Equalization

Our reference receiver uses the LS channel estimator, as per Eq. (3.15). Due to its simplicity, it is well-suited for hardware implementations [39, 41]. In order to compensate for channel effects, the data samples need to be multiplied by the CTF inverse $C = 1/H_{LS}$ or, equivalently, $C = X/Y$, where X is the known LTS (a PN sequence of 1 and –1), and Y is the received LTS. The division of a real number by a complex number requires two divisions:

$$z = \frac{1}{a+bi} = \frac{1}{a+bi} \times \frac{a-bi}{a-bi} = \frac{a-bi}{a^2-b^2} \qquad (4.2)$$

$$z = \frac{a}{a^2-b^2} - \frac{bi}{a^2-b^2} \qquad (4.3)$$

In order to perform these division operations, two CORDICs working in parallel can be used [39]. Hence, the channel equalization block diagram can be as shown in Fig. 4.6, which is what we use in our reference receiver.

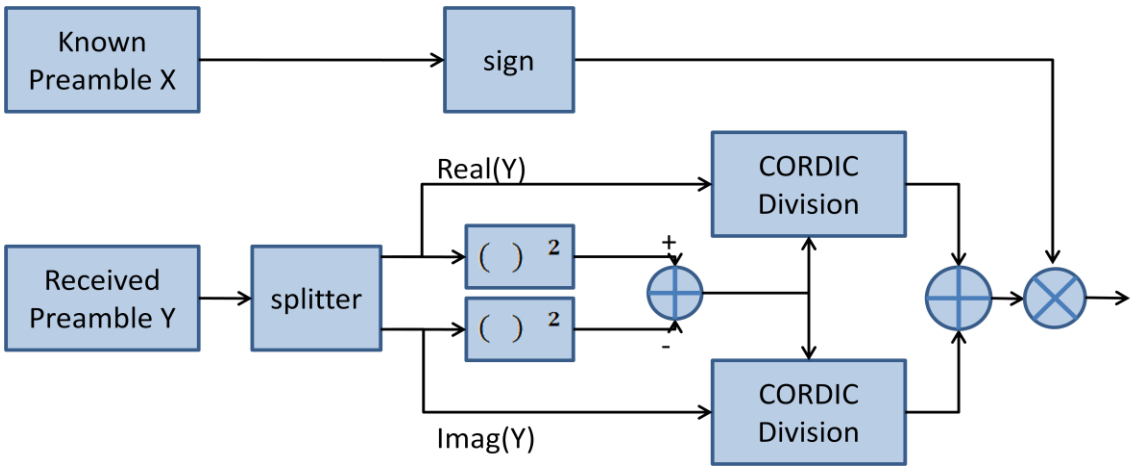


Figure 4.6: Channel Estimation with CORDIC Division

In [42] another approach is proposed, where the preamble is first converted from rectangular to polar coordinates (using another CORDIC), and then, one CORDIC performs phase rotation and the other performs division (both are working in parallel), as shown in Fig. 4.7. A similar approach is used in [41]; however, the phase rotations are delayed until phase tracking is also completed (see the next section). This phase is added to the channel equalized phase and only then all data is equalized via CORDIC rotation.

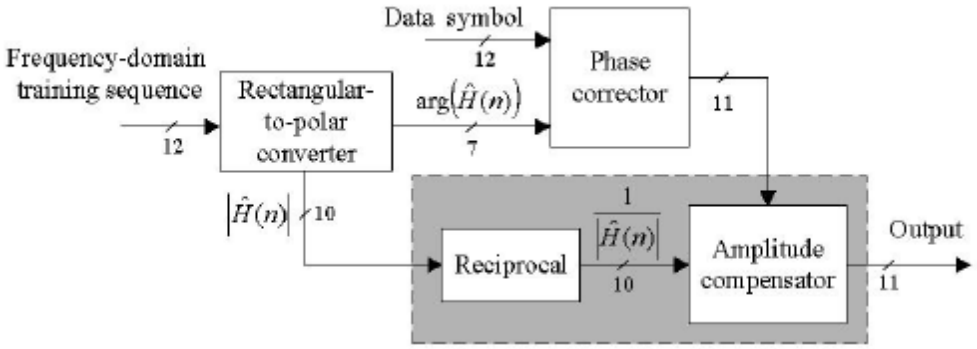


Figure 4.7: Channel Estimation using CORDIC Rotation [42]

4.3. Phase tracking

A typical phase tracker is shown in Fig. 4.8, which is taken from [41]. It does not take into account the 802.11a PN sequence applied to the pilots, and it performs a rotation by 180 degrees for samples that have a negative real value, which is done to keep the angle range of $\pm\pi/2$. In Appendix D, it is shown that CORDIC can be extended to work with the angle range of $\pm\pi$; hence, this extra rotation presented in [41] is not needed.

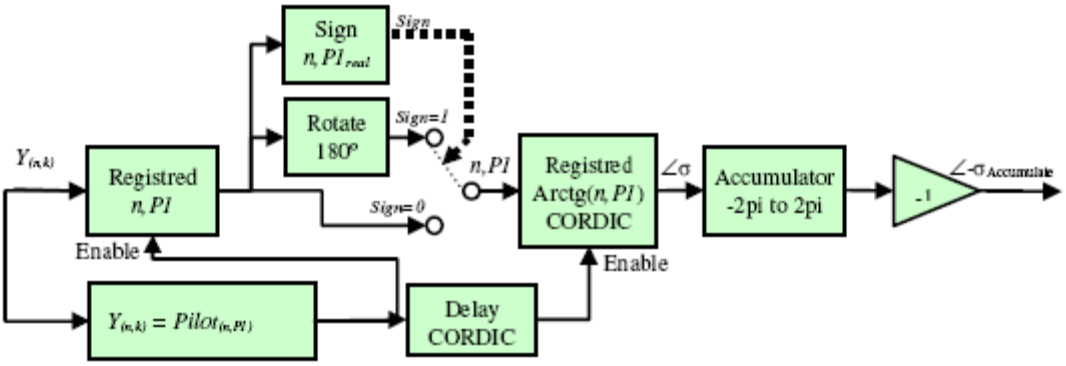


Figure 4.8: Phase Tracking used in [41]

Our phase tracking block diagram is shown in Fig. 4.9 and based on the discussion presented in Section 3.3. It requires one CORDIC to compute the phase difference between known and received pilots, and another CORDIC to perform phase rotations of data samples accordingly.

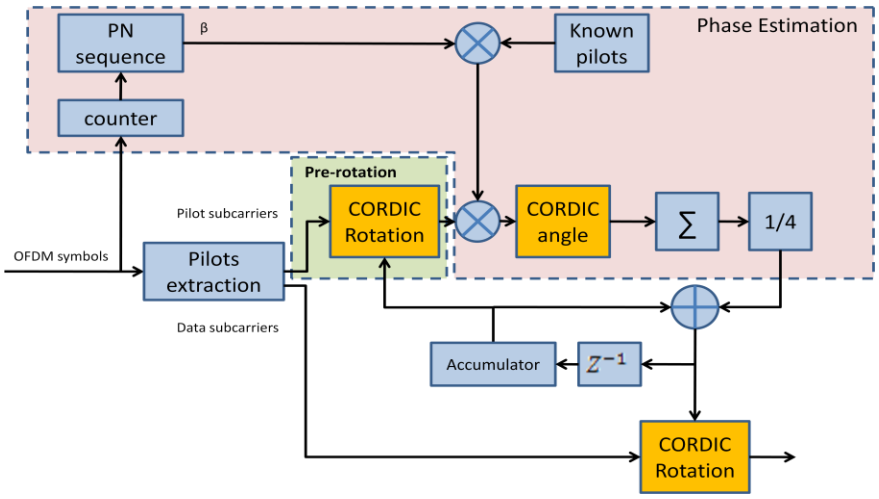


Figure 4.9: Phase Tracking Block Diagram

4.4. Fixed-Point Arithmetic Issues

In most published simulation results, the signal values are usually represented in the floating-point format. In practical systems, however, it is significantly less expensive to work with numbers in a fixed-point format. The problem with fixed-point numbers is a loss of precision and value saturation when word lengths are not chosen properly [45].

For any given number, the bit representation given by Eq. (4.4), where ◊ is the radix point and *B* is the number of bits. when the number of bits is infinite, infinite precision is achieved:

$$\hat{x}_B = b_0 b_1 \lozenge b_2 b_3 b_4 b_5 \cdots b_B \tag{4.4}$$

In practice, the number of bits is constrained, and hence the number value becomes quantized [45]:

$$x = X_m \left(-b_0 + \sum_{i=1}^{B} b_i 2^{-i} \right) \tag{4.5}$$

$$\hat{x} = Q_B[x] = X_m \left(-b_0 + \sum_{i=1}^{B} b_i 2^{-i} \right) = X_m \hat{x}_B \tag{4.6}$$

where b_i is the bit at the *i*-th position, and X_m is a scaling factor. It can also be seen that the quantization step size, or the smallest difference between two numbers, is given by [45]:

$$\Delta = X_m 2^{-B} \tag{4.7}$$

The corresponding quantization error bounds are given by Eq. (4.8) [45]:

$$e = Q_B[x] - x \quad \begin{cases} \frac{-\Delta}{2} < e \leq \frac{\Delta}{2} & for\ rounding \\ -\Delta < e \leq \Delta & for\ saturation \end{cases} \tag{4.8}$$

In addition to the quantization errors, finite world-length computations also give rise to overflow problems. There are two ways to handle overflow: either roll-over the number value to 0, or saturate it at its maximum, the later being a preferred choice [45]. To avoid overflow, one can either increase the number of bits (making hardware more expensive) or increase the scaling factor (making quantization errors greater).

In relation to CORDIC operations, [46] provides an error analysis and concludes that increasing the number of CORDIC iterations beyond the word length does not improve precision. Typically, as the number of CORDIC iterations increases, the mean square error (MSE) of the approximation is decreased. However, the overall MSE of the approximation is bounded by the quantization error, as shown in Fig. 4.10 taken from [46].

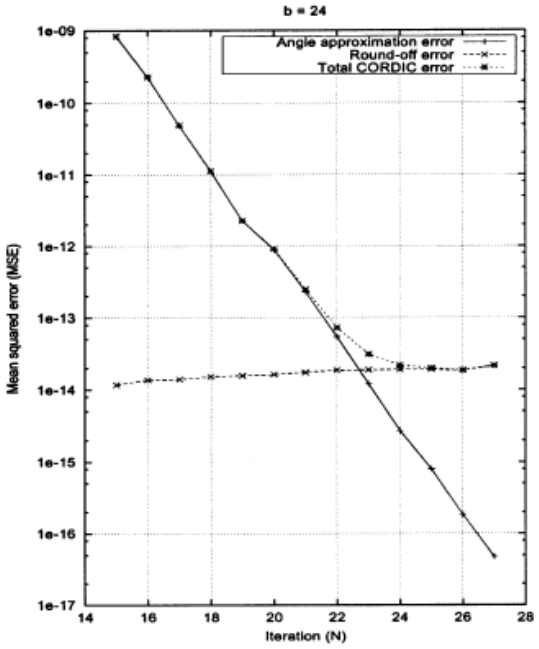


Figure 4.10: CORDIC Approximation Error [46]

Chapter 5: Proposed Receiver Design

Transmitted and received OFDM symbol samples in both the time domain and the frequency domain are complex-valued. Normally, each sample would be represented as two real-valued components: *in-phase* (real part) and *quadrature* (imaginary part), representing a complex number in the Cartesian coordinates. This means that the receiver would have to process two streams of numbers: the stream of in-phase components of received samples (we call it *I*-stream) and the stream of quadrature components of received samples (we call it *Q*-stream). Synchronization, FFT, channel equalization, and phase tracking would require calculations of both the *I*-stream and the *Q*-stream arriving at the demodulator input.

One of our main objectives is to allow the receiver to scale its computational effort according to the data rate requirements. The design presented in this chapter does not use the Cartesian coordinates in the frequency domain. We still use the *I*-stream and the *Q*-stream in the time domain, but after the conventional Cartesian FFT operation we convert all the samples into the polar coordinates, where each complex number is represented by its *magnitude* and *angle* components. Thus, in the frequency domain, we have the stream of magnitude components (we call it the *M*-stream) and the stream of angle components (we call it the *A*-stream). One immediate advantage of this approach is that for the 802.11a data rates of 6, 9, 12, and 18 Mbps the *M*-stream can be completely shut down, thus reducing the computational effort. (If working with the *I*-stream and *Q*-stream instead, a similar shut-down of one of the streams would have catastrophic effects on the receiver performance.)

While the idea of using polar coordinates is simple and intuitive, it appears to have been ignored in the literature on 802.11a-relevant receivers [39-44]. A possible culprit may be the usual focus of the other works on the receiver performance at high data rates, as opposed to our focus on the computational scalability from low to high data rates. We note that at the highest 802.11a data rates of 48 and 54 Mbps, the polar coordinates offer no practical advantage over Cartesian, hence there was no motivation to explore their use.

This chapter highlights both the advantages and disadvantages of using the polar coordinates, analyzes the extent of the computational scalability across the 802.11a data rates, and also discusses relevant fixed-point arithmetic issues. The material here covers only the frequency-domain processing, where we use the polar coordinates. The time-domain processing remains the same as in the reference receiver (using the Cartesian coordinates). The proposed changes to the reference receiver start after the FFT operation, where we first place a CORDIC to convert the frequency-domain *I*-stream and *Q*-stream into the *M*-stream and *A*-stream. The subsequent modifications to the channel estimator/equalizer, phase tracker, and demodulator are detailed next.

5.1. Channel Estimator and Equalizer

The block diagram of our proposed channel estimator and equalizer is shown in Fig. 5.1, where amplitudes and phases form the *M*-stream and *A*-stream respectively.

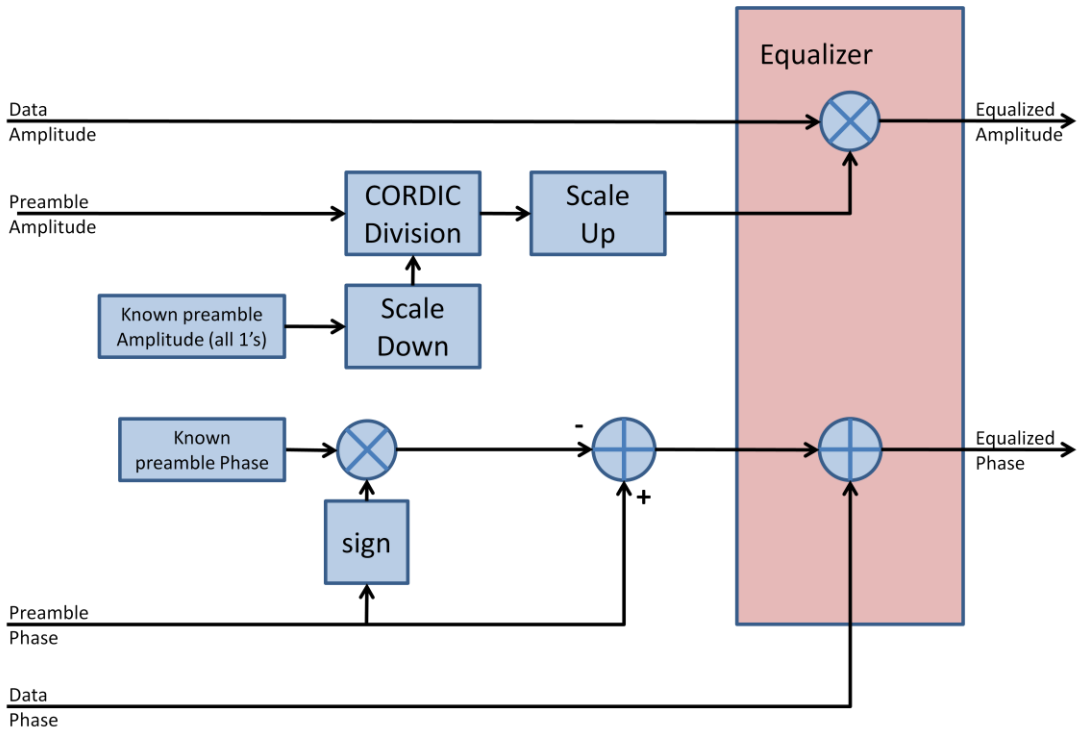


Figure 5.1: Proposed Channel Estimator and Equalizer

The magnitude of the known preamble p_k (always 1's) is first scaled down (divided) by the scaling factor S_c, followed by a CORDIC performing division by the magnitude of the received preamble p_r, whose output is then scaled up (multiplied) by S_c to obtain the magnitude equalization values $C = p_k/p_r$. Such scaling is needed due to the CORDIC output being bounded by 2. We use S_c = 16 (to allow for replacing divisions and multiplications by shifts), which can handle the CTF inverse magnitudes of up to 32. To obtain the phase equalization values, we calculate the difference between the known preamble phase (either π, or 0) and the received preamble phase according to Eq. (5.1):

$$\theta = p_r - sign(p_r) * p_k \qquad (5.1)$$

Given *C* and θ, channel equalization to be applied to the *M*-stream and the *A*-stream of data samples is trivial: magnitudes are multiplied by *C*, while θ is added to angles. This is in contrast to the conventional reference receiver that uses a CORDIC to equalize the data's *I*-stream and *Q*-stream.

5.2. Phase Tracker

The phase tracker is where the design is greatly simplified (compared to the Cartesian phase tracker in chapter 4), as the pilot and data samples are already in the polar coordinates. Phase estimations based on pilots and the corresponding phase rotations applied to the data samples are replaced with simple subtractions and additions of the *A*-stream numbers. Note that there is no need to process the *M*-stream numbers, and the two CORDICs of the conventional reference phase tracker are eliminated. The block diagram of our proposed phase tracker is shown in Fig. 5.2, where the phase range adjustment blocks are used to keep the angle within $\pm\pi$ according to Eq. (5.2). In addition to using the polar coordinates, it has two extra enhancements: pre-rotation and pilot masking.

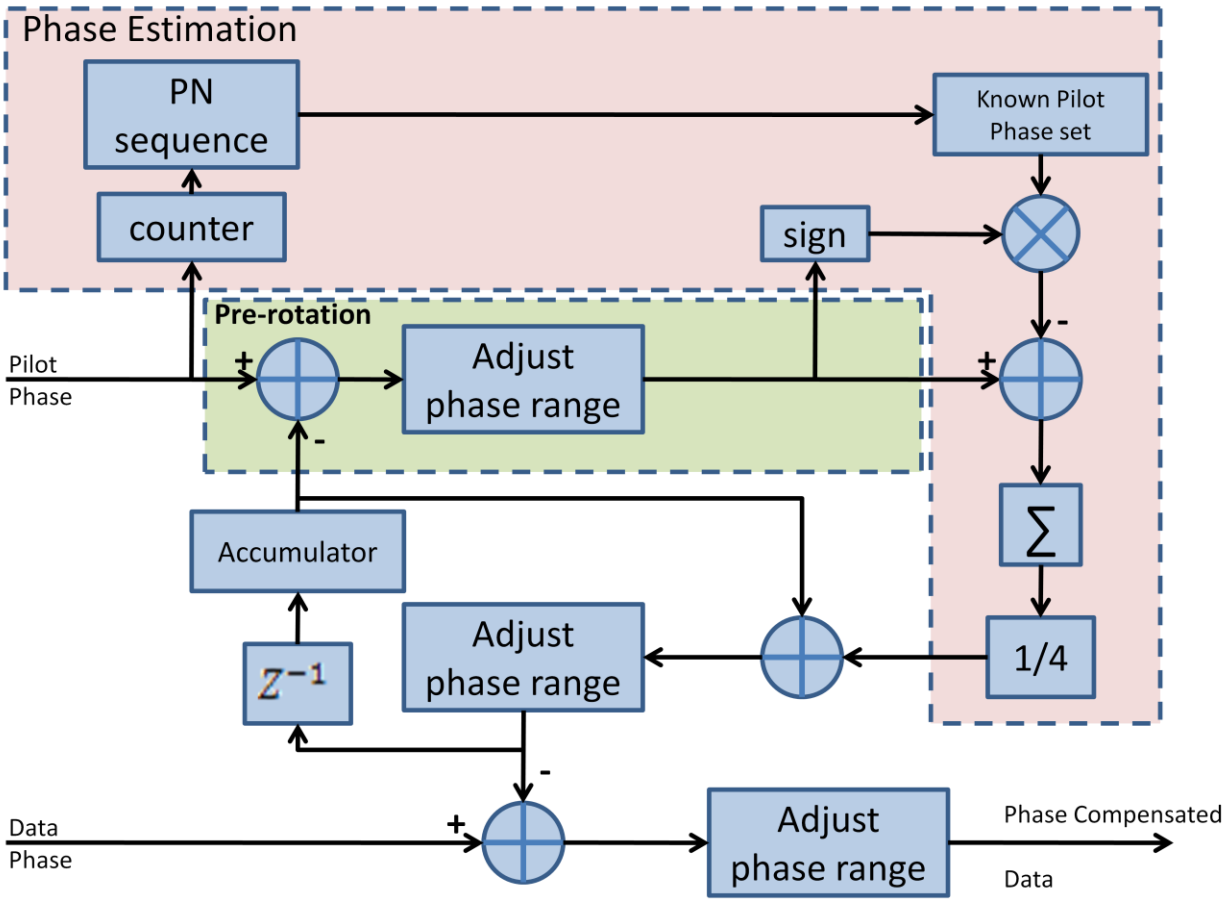


Figure 5.2: Proposed Phase Tracker

$$DATA_{adj} = \begin{cases} DATA_{in} - 2\pi, & DATA_{in} > \pi \\ DATA_{in} + 2\pi, & DATA_{in} < -\pi \\ DATA_{in}, & -\pi \leq DATA_{in} \leq \pi \end{cases} \quad (5.2)$$

5.2.1. Pre-rotation

During the phase tracking process, the RFO-induced phase offset increases with each OFDM symbol from zero to π and then follows a sawtooth-like pattern within $-\pi$ and $+\pi$, as illustrated in Fig. 5.3 showing the negated offset in question (to be added to the data phase). We introduce pre-rotation of the received pilots of the current OFDM symbol by a phase offset found for the previous OFDM symbol, in order to reduce the range of the current phase offset, as shown in Fig. 5.4. This pre-rotation is applied only to the pilots, while the data samples would be rotated by the previous plus currently measured phase offsets. It was observed that using pre-rotation resulted in slight improvements of the receiver performance. The cost of implementing pre-rotation is negligible when we use the polar coordinates, but it would require an additional CORDIC in the Cartesian case, i.e., its practical value in the conventional reference receiver would be very limited.

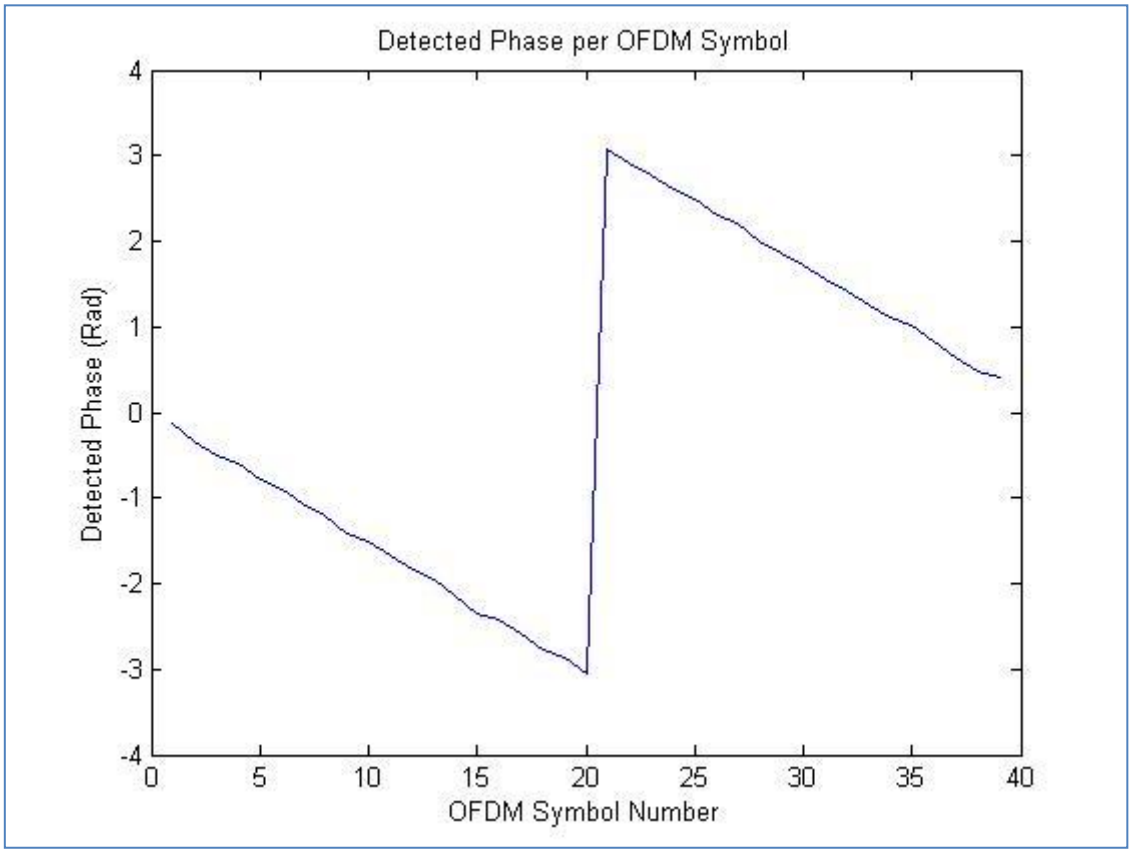


Figure 5.3: Detected Phase Offset without Pre-Rotation

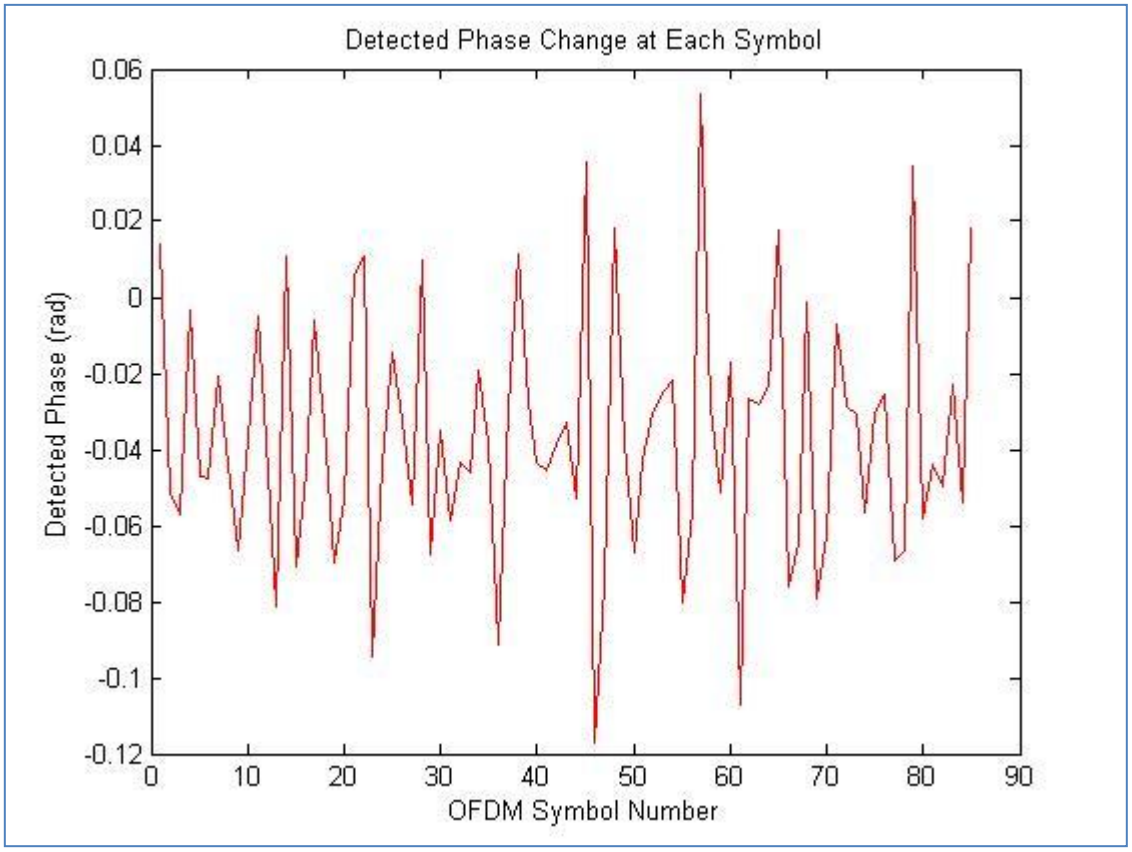


Figure 5.4: Detected Phase Offset with Pre-Rotation

5.2.2. Pilot Masking

In our simulations, it was noticed that in cases where significant fading occurred at the pilot subcarrier locations, the phase tracker was performing poorly: the estimated phase offset, taken as an average over four pilots (per OFDM symbol), was seriously corrupted by deeply faded pilots. The idea behind pilot masking is to identify and remove deeply faded pilots to keep the average phase offset estimate intact. Our proposed receiver relies on the channel estimation information to determine how severe the fading is at each of the four pilot locations: if the pilot power is 4 times lower than the expected value, then that pilot is to be discarded. We generate a 4-bit pilot masking vector, where 1's indicate good pilots, and 0's indicate damaged pilots. The phase tracker uses only the pilots that have 1's in the masking vector to compute the phase offset. Fig. 5.5 shows a simulation example illustrating the benefit of pilot masking.

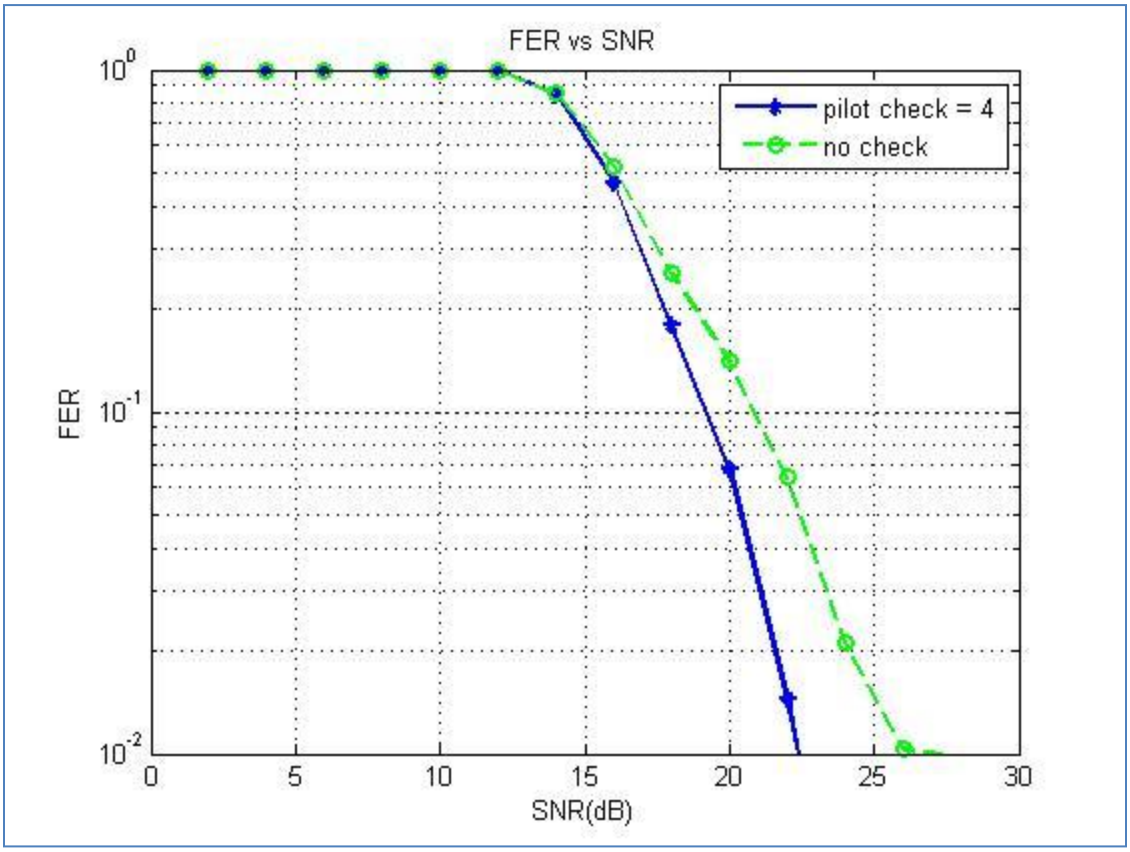


Figure 5.5: Example of Performance Improvement due to Pilot Masking

5.3. Demodulator

The required demodulation scheme in the 802.11a receivers depends on the data rate (see Appendix B), as shown in Table B-2. Given the *I* and *Q* components of a data sample, the demodulation task is essentially to find the closest constellation point to the equalized signal as per Fig. B.8. Our proposed receiver, however, works with the *M* (magnitude) and *A* (angle) components. It is possible to convert the *M*-stream and the *A*-stream to the *I*-stream and *Q*-stream (using an extra CORDIC), but this would largely eliminate the computational benefit of our proposed CORDIC-free phase tracker over the conventional CORDIC-based phase tracker. This sections describes our modified demodulator that avoids the conversion to the Cartesian coordinates when possible, or avoids using CORDIC if such conversions are needed.

5.3.1. BPSK and QPSK Demodulation

According to Fig. B.8, the BPSK and QPSK demodulation is possible using only the phase information. By looking at the *A*-stream numbers, our demodulator needs to check which half-plane (for BPSK) or which quadrant (for QPSK) the incoming angle belongs to. Note that the *M*-stream numbers are not required at all, hence the corresponding hardware can be completely shut down. Among eight 802.11a data rates, two are using BPSK (6 and 9 Mbps) and two are using QPSK (12 and 18 Mbps). Thus, in half of the data rate cases, we are able to eliminate frequency-domain processing of one stream altogether, which is not possible when using the Cartesian coordinates.

5.3.2. 16-QAM Demodulation

According to Fig. B.8, each 16-QAM constellation has four bits $b_0b_1b_2b_3$. In [7], the author shows how each bit can be represented by the region on the constellation. For Polar coordinates, bits b_0 and b_2 are easily found based on the angle information, as shown in Fig. 5.6. Bit b_0 = 1 if the sample angle lies between $-\pi/2$ and $+\pi/2$; 0 otherwise. Bit b_2 = 1 if the sample angle lies between 0 and π; 0 otherwise.

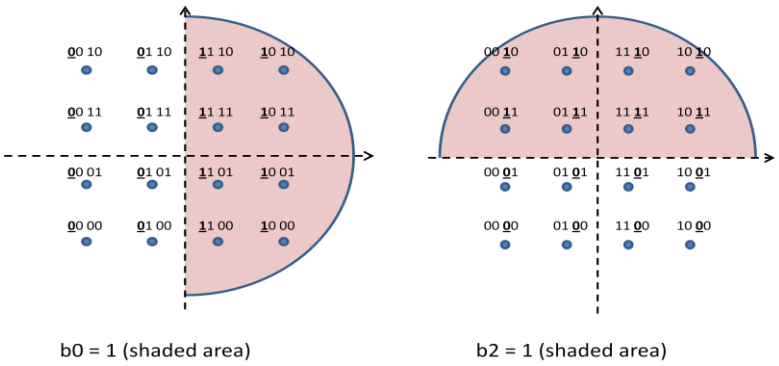


Figure 5.6: Constellation Regions for bits b0 and b2

Deciding on bits b_1 and b_3 is more complicated and requires the knowledge of the sample magnitude. Fig. 5.7 illustrates our proposed method using one specific quadrant as an example, where the angle ranges from 0 to $\pi/2$. This example quadrant is divided into four regions, numbered 1, 2, 3, 4 (shown in circles). Region pairs 1-2 and 3-4 are distinguished by the angle being $\theta \leq \frac{\pi}{4}$ or $\theta > \frac{\pi}{4}$, while region pairs 1-3 and 2-4 are distinguished by the magnitude being $\alpha \leq \sqrt{8}$ or $\alpha > \sqrt{8}$. In other words, knowing both the angle and the magnitude tells us which region to use. In region 1, bit b_3 = 1 for sure, but to decide on bit b_1 we need to know whether vertical boundary between the two lower constellations has been crossed (b_1 = 0) or not (b_1 = 1), which involves calculating $\alpha \cdot \text{Cos}(\theta)$. In region 2, bit b_1 = 0 for sure, but to decide on bit b_3 we need to know whether horizontal boundary between the two leftmost constellations has been crossed (b_3 = 0) or not (b_3 = 1), which involves calculating $\alpha \cdot \text{Sin}(\theta)$. Similarly, in regions 3 and 4, we need to calculate $\alpha \cdot \text{Cos}(\pi/2-\theta)$ and $\alpha \cdot \text{Sin}(\pi/2-\theta)$ respectively. Thus, to decide on unknown b_1 and b_3, we need to calculate either Sin or Cos (but never both) and perform one multiplication (by α). Using conventional *I*/*Q* demodulation (given a sample in the polar coordinates) would require calculating both Sin and Cos and performing two multiplications.

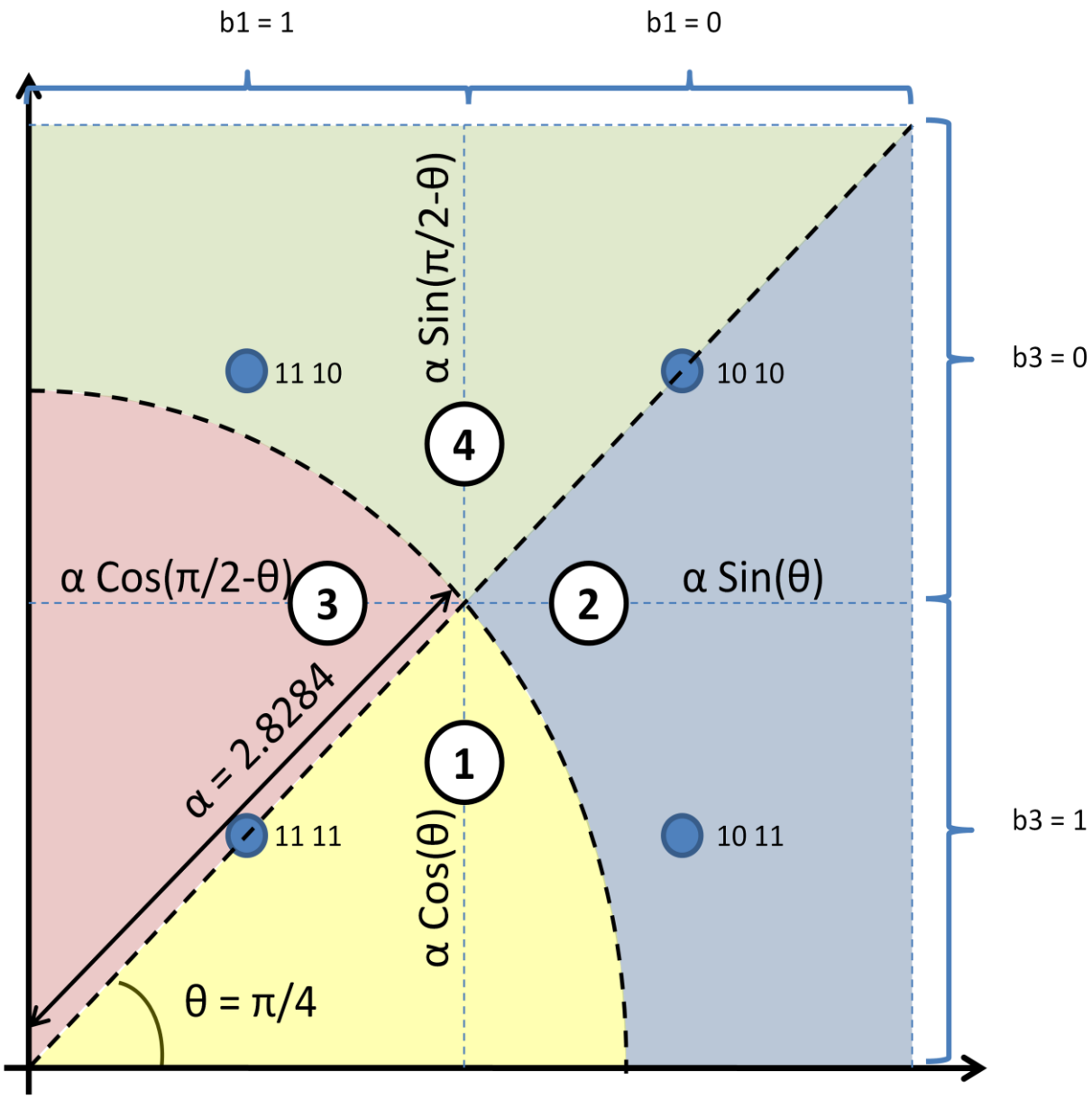
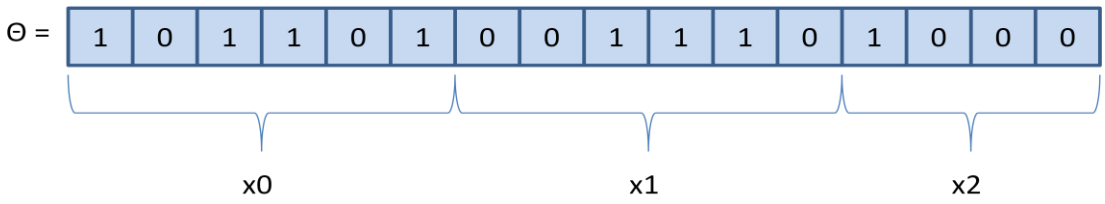


Figure 5.7: Example Quadrant Illustrating 16-QAM Demodulation

The problem is then how to calculate the Cos or Sin efficiently. Employing CORDIC for this would be power inefficient. Instead, we use a bipartite lookup table (LUT) method to obtain approximate values of Cos or Sin for a given angle. The bipartite LUT method [48], splits the angle's word length into three *k*-bit subwords x_0, x_1, and x_2, as shown below for the 16-bit angle example (k = 6). The value of k is chosen as to divide the angle word length into subwords of equal length of even bits when possible.



Next, two functions *A* and *B* are defined as follows

$$A(x_0, x_1) = \cos(x_0 + 2^{-k}x_1 + 2^{-2k-1})$$
$$B(x_0, x_2) = -(2^{-k}x_2 - 2^{-2k-1})\sin(x_0 + 2^{-k-1}) \quad (5.3)$$

Two LUTs are used to store pre-computed values of functions *A* and *B*. For our proposed receiver, the maximum word length of the angle fractional part was set to 16 bits, with x_0, x_1 and x_2 made 6-, 6-, and 4-bit long respectively. Including the sign bit, *A* is stored with the precision of 20 bits, and *B* is stored with the precision of 8 bits, giving the overall LUT memory size of 11 Kb ($20 \cdot 2^{6+6} + 8 \cdot 2^{6+4}$). It is important to note that the bipartite LUT method works well only for $\theta \epsilon [0, \pi/4]$. Hence, in Fig. 5.7, if $\theta > \pi/4$, the new angle is made as $\theta_{new} = \pi/2 - \theta$ (with interchanging Cos and Sin), to comply with the LUT range.

When the angle belongs to the quadrant other than the one shown in Fig. 5.7, the demodulation process follows the same steps as discussed above: two bits will be determined by the quadrant itself (see Fig. 5.8), while the others will be decided on by analysing four regions of interest, after transforming the angle according to Eq. (5.4).

$$\theta_{new} = \begin{cases} |\theta| & for \quad \theta \leq \frac{\pi}{2} \\ \pi - |\theta| & for \quad \theta > \frac{\pi}{2} \end{cases} \quad (5.4)$$

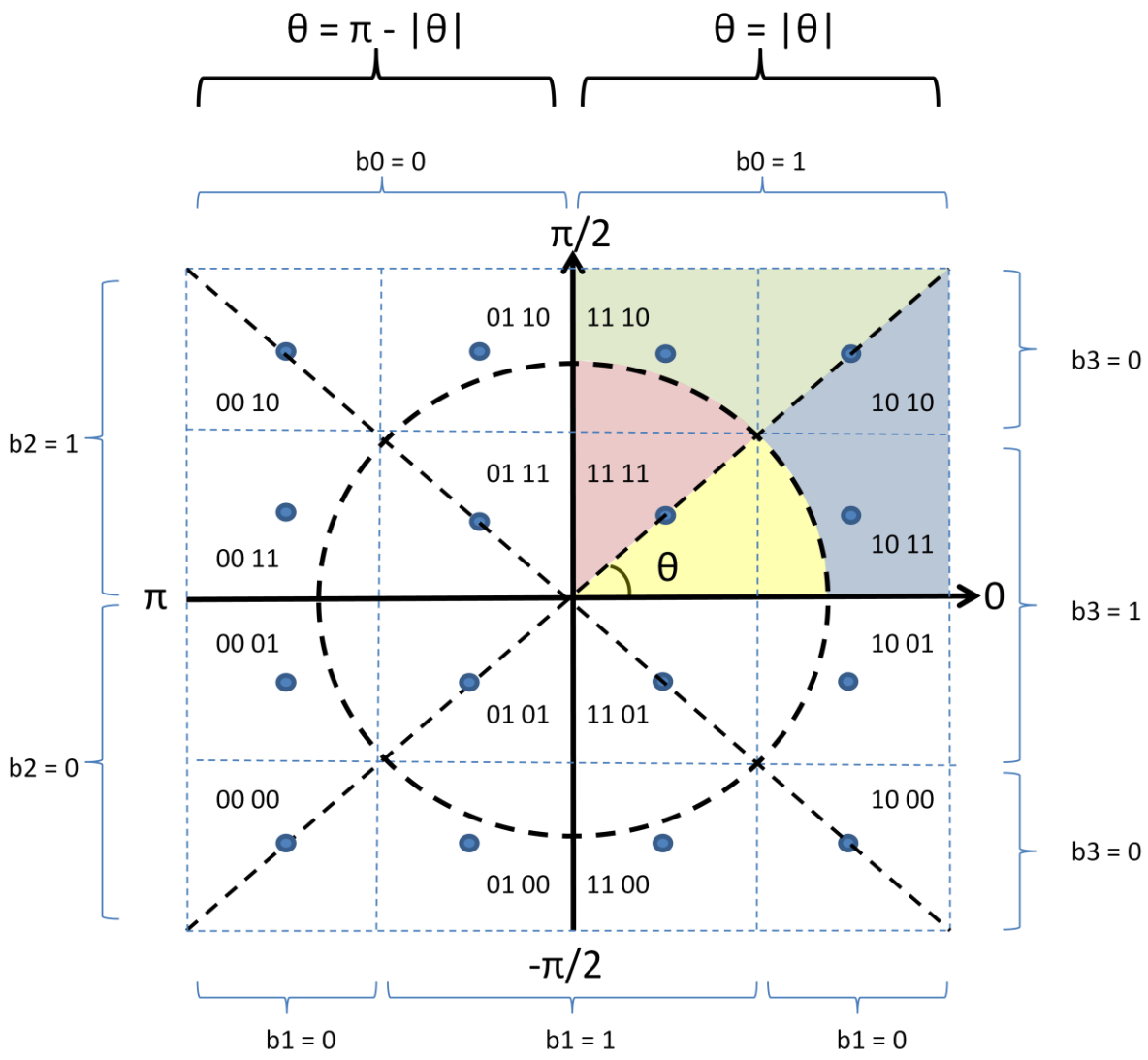


Figure 5.8: General Case of 16-QAM Demodulation

5.3.3. 64-QAM Demodulation

In the case of the 64-QAM demodulation, we need to convert the angle and the magnitude into the Cartesian form, which involves calculating both Cos and Sin and performing two multiplications. Our proposed receiver employs the same LUT-based method as in the 16-QAM case to approximate Cos and Sin. To maintain the angle within the range from 0 to π/4, each quadrant is divided into two regions, as shown in Fig. 5.9. The in-phase (*X*) and quadrature (*Y*) are calculated according to Eq. (5.5) and (5.6):

$$X = \begin{cases} \alpha \cdot \cos(|\theta|) & for\ |\theta| \leq \frac{\pi}{4} \\ \alpha \cdot \sin(\pi - |\theta|) & for\ |\theta| > \frac{\pi}{4} \end{cases} \quad (5.5)$$

$$Y = \begin{cases} \alpha \cdot \sin(|\theta|) & for\ |\theta| \leq \frac{\pi}{4} \\ \alpha \cdot \cos(\pi - |\theta|) & for\ |\theta| > \frac{\pi}{4} \end{cases} \quad (5.6)$$

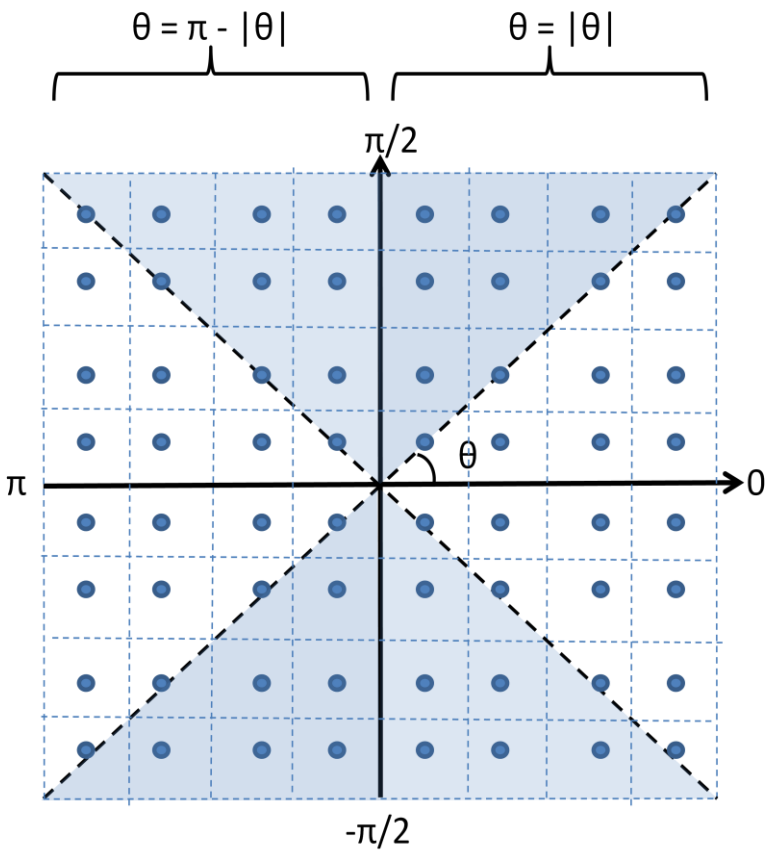


Figure 5.9: 64-QAM Demodulation

Note that our proposed 64-QAM demodulation performs a full polar-to-Cartesian conversion, but does not use CORDIC. Since 16-QAM demodulation already uses the LUTs, they are simply re-used for 64-QAM as well. Instead of one LUT lookup and one multiplication in the case of 16-QAM (used for 24 and 36 Mbps), there will be two LUT lookups and two multiplications performed in the case of 64-QAM (used for 48 and 54 Mbps), i.e., the demodulator's computational effort will scale according to the data rate.

5.4. Design Summary

The overall proposed design is presented in Fig. 5.10, will be contrasted with the frequency-domain part of the reference (conventional) receiver shown in Fig. 4.2. It can be seen that at the cost of placing a CORDIC at the beginning (immediately after FFT) leads to substantial gains in the subsequent stages where CORDIC usage is reduced or eliminated altogether, while some operations (such as complex multiplication in the channel equalizer) are simplified.

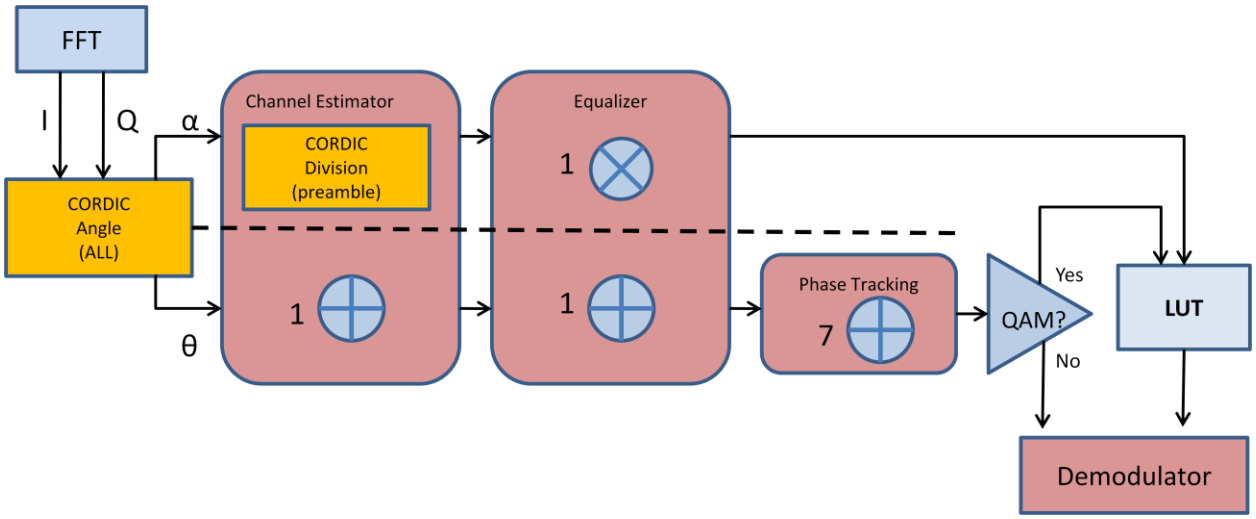


Figure 5.10: Overall Proposed Design in Frequency Domain

Table 5-1 presents the overall operations being used in both Cartesian and Polar receiver designs. The total number of times each type of operation is used is presented with N_{SYM} indicating the number OFDM symbols per packet. In terms of multiplication operations, the Polar system gives around 75% savings for BPSK and QPSK Modulations, around 57% savings for 16-QAM and around 40% savings for 64-QAM. In terms of CORDIC operation use, the Polar system gives around 10% savings for all data rates. It is also seen that there is an increase in the number of addition operations, however, addition operations are less computationally expensive than multiplication and CORDIC operations.

Table 5-1: Operations Summary

Operation Type	Cartesian	Polar BPSK/QPSK	Polar 16-QAM	Polar 64-QAM
LUT Access	0	0	$48N_{SYM}$	$96N_{SYM}$
Multiplication	$128+260N_{SYM}$	$64N_{SYM}$ (~75% less)	$112N_{SYM}$ (~57% less)	$160N_{SYM}$ (~40% less)
Addition	$128+128N_{SYM}$	$64+180N_{SYM}$	$64+180N_{SYM}$	$64+228N_{SYM}$
CORDIC	$104+58N_{SYM}$	$52N_{SYM}$	$52+52N_{SYM}$	$52+52N_{SYM}$

As mentioned earlier in this chapter, working with the polar coordinates facilitates scalability of the receiver computational effort in relation to the data rate requirements. For the data rates of 6, 9, 12, and 18 Mbps (using BPSK and QPSK) the *M*-stream (sample magnitudes) can be completely shut down, as only the *A*-stream (sample angles) is needed. For the data rates of 24 and 36 Mbps (using 16-QAM), the receiver must increase its computational effort by processing the *M*-stream, and for each sample to be demodulated, it must perform one LUT read and one multiplication. For the data rates of 48 and 54 Mbps (using 64-QAM), the receiver must increase its computational effort again: for each sample to be demodulated, it must now perform two LUT reads and two multiplications.

One block that benefits the most from using the polar coordinates is the phase tracker. It no longer requires one CORDIC to estimate the phase offset and the other CORDIC to rotate the data samples accordingly. These operations are reduced to simple subtractions and additions of the *A*-stream numbers. Another added benefit is that any fixed-point approximation errors arising during phase offset estimation from the *A*-stream do not propagate into the *M*-stream. In fact, the fixed-point word length for these two streams may be different. In the case of the Cartesian *I*-stream and *Q*-stream, both are used to estimate the phase offset, and both are affected by the subsequent rotation.

There are two disadvantages of using the polar coordinates, however. First, the additive noise in the Cartesian *I*-stream and *Q*-stream becomes nonlinearly embedded noise within the polar *M*-stream and *A*-stream. Nevertheless, as we show in the next chapter, the receiver performance is not seriously affected. Second, as we do not perform a full polar-to-Cartesian conversion during BPSK, QPSK and 16-QAM demodulation, the use of a soft-decision Viterbi decoder [31] is not possible. Such decoder typically offers a 3-dB gain over hard decoders, but it is more computationally expensive needing more hardware than its classic hard-decision variant. Our proposed receiver uses a hard-input Viterbi decoder. Nevertheless, in the case of 64-QAM, our receiver does not prevent the choice of performing soft-decision decoding, and this is arguably the case where the data bits are most vulnerable to errors and soft-decision decoding is most valuable (in comparison to the cases of BPSK, QPSK, and 16-QAM).

Chapter 6: Performance Analysis

This chapter presents simulation results used to evaluate our proposed receiver design and also describes our methodology for evaluating design decisions pertaining to fixed-point word precisions at different data rates. Our MATLAB simulation setup is documented in Appendix C. The results presented here include the receiver FER/BER performance at the minimum 802.11a-specified data rate of 6 Mbps and at the maximum 802.11a-specified data rate of 54 Mbps. The BER and FER are performance metrics that are typically used when evaluating communication systems. BER is defined as the measured probability of having a received data bit in error while FER is the measured probability of having a received frame (or packet) in error.

The reported simulation tests demonstrate several important points. First, we show that our reference receiver from Chapter 4 is valid, by comparing its performance (see Fig. 6.1) to the performance of reference designs reported in published literature. Second, we show that performance of our proposed polar receiver using floating point arithmetic matches that of the Cartesian reference receiver, thus demonstrating that the use of polar coordinates instead of Cartesian does not inherently hinder the receiver's performance. Third, we show that the floating-point performance of our polar receiver can be matched with fixed-point computations (involving CORDIC when needed) and that the required fixed-point precision must be greater for higher data rates, thus demonstrating the possibility of scaling the receiver computational effort depending on the data rate requirements. Finally, we show how to evaluate the design recommendations regarding the fixed-point precision, by explicitly stating our time-saving evaluation procedure for our design decisions.

6.1. Proposed Receiver vs. Reference Receiver vs. Published Works

Before we start comparing our proposed receiver against the conventional reference receiver, we need to be sure that our reference receiver is correct. This is done by comparing the FER performance of our receiver with reported performance of other receivers in the literature. Unfortunately, none of the studied papers that deal with 802.11a-compiant receivers reports performance evaluations of a complete implementation. The works dealing primarily with the receiver hardware [39-44] discuss hardware costs (e.g., number of gates and flip-flops) with no mention of BER/FER curves. On the other hand, the works dealing primarily with the receiver theory [22-28] discuss BER/FER curves in relation to only some specific block(s), assuming ideal performance of the others. Our reference receiver includes implementation of the entire 802.11a receiver including synchronization, channel estimation/equalization, and phase tracking, thus capturing all non-ideal behaviours arising in practice during receiver signal processing. Consequently, it is expected that the BER/FER curves produced by our reference receiver will be slightly worse that those published in the literature.

The analysis in [24] and [26], used a similar simulation setup in terms of channel model as well as packet format. [24] reports performance curves for the data rates of 12Mb/s and 48 Mb/s, while [26] reports those for 6Mb/s and 54Mb/s. Fig. 6.1 compares the FER curves produced by our reference receiver with those reproduced from [24] and [26]. One can see that the performance curves for the proposed receiver are shifted to the right by 2-4 dB in relation to the published curves, while the slopes in both cases are essentially the same. The slopes of the curves represent how well a system handles certain environments at different SNR values. If the slopes differ this would indicate a different test environment or different coding performance. For this reason, slope similarities are the most important indicator that our reference receiver is valid. The ~3dB shift can be due to the likely use of the soft-decision decoder and the assumption of ideal synchronization and/or channel estimation in the previously published results.

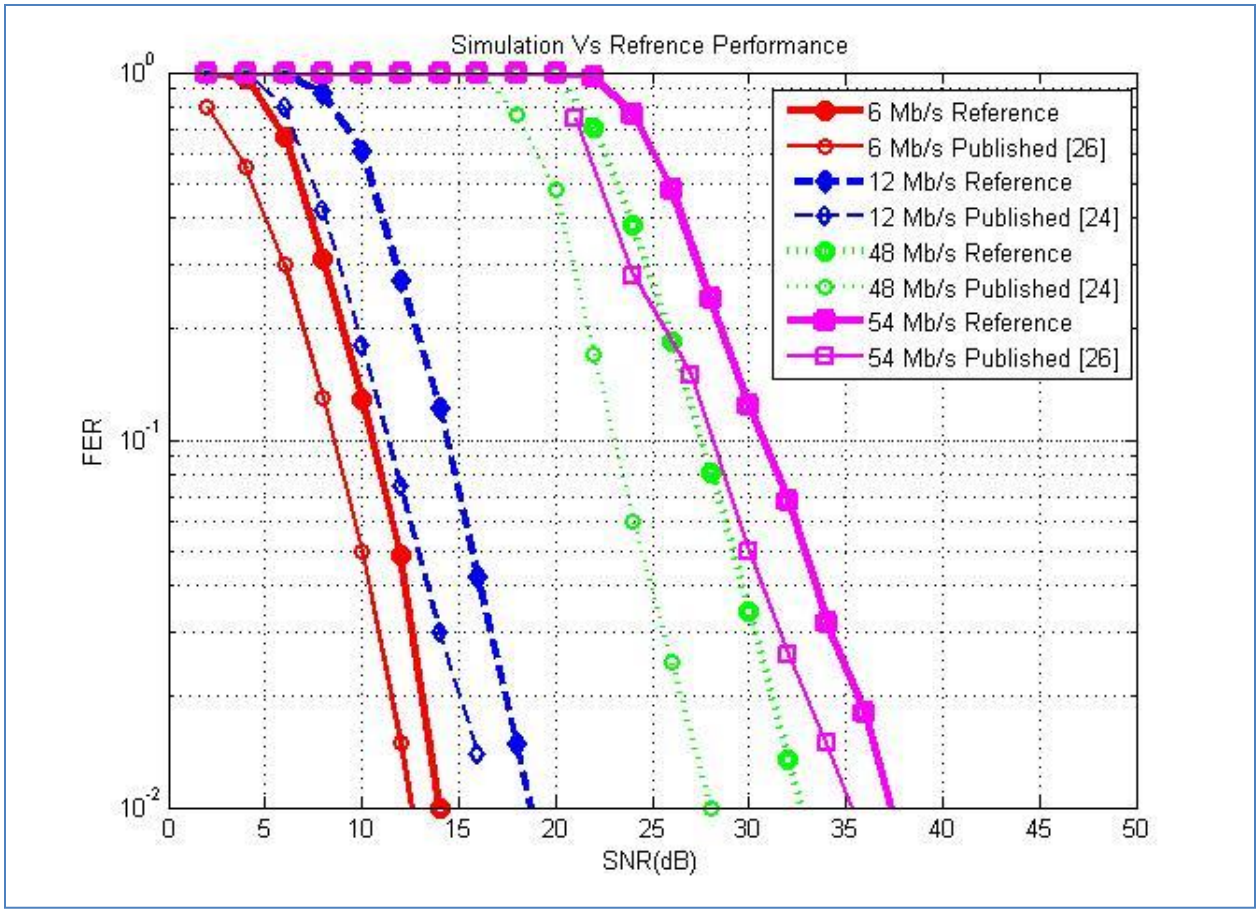


Figure 6.1: Reference Receiver Performance vs. Published Works [24, 26]

Next, we want to compare the reference (Cartesian) receiver's performance to that of our proposed (polar) receiver using only floating-point calculations. The corresponding FER curves are shown in Fig. 6.2, which demonstrates that the use of polar coordinates provides nearly identical performance to the use of Cartesian coordinates.

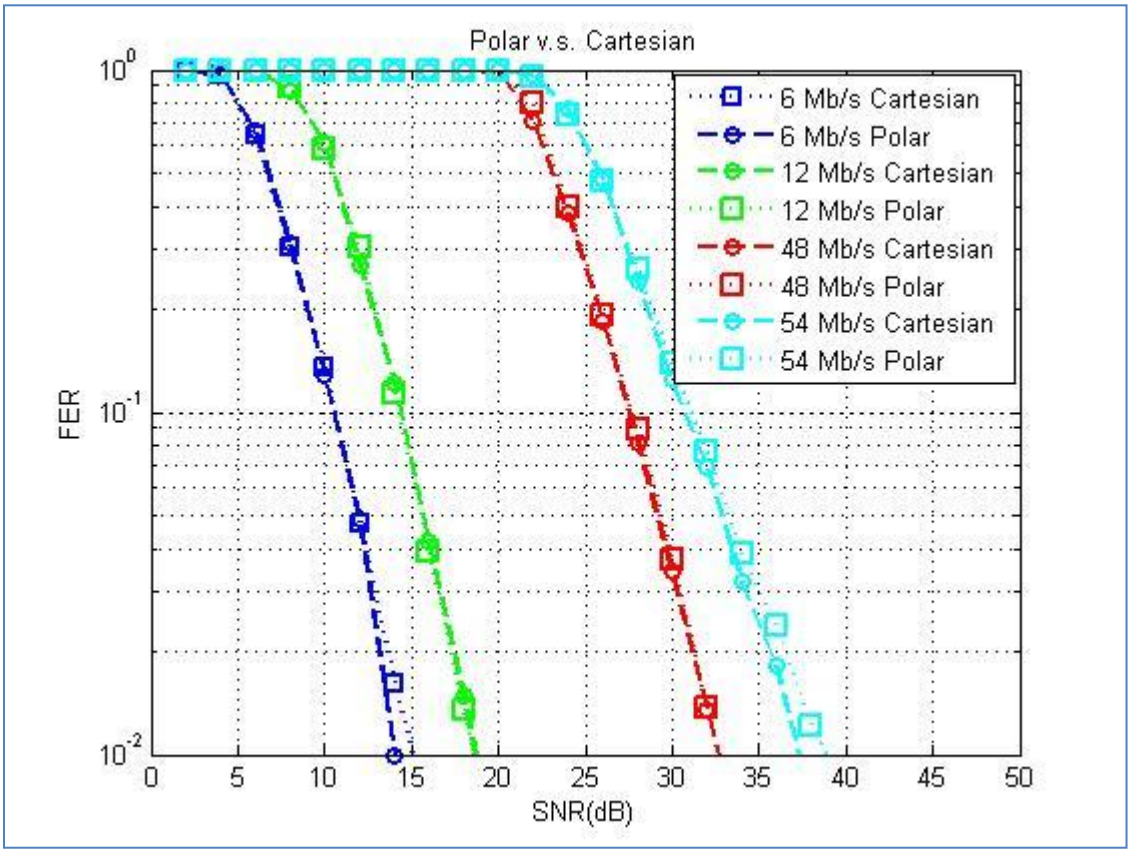


Figure 6.2: Floating-Point Performance of Proposed (Polar) Receiver vs. Reference (Cartesian) Receiver

6.2. Fixed-Point Test Methodology

When preparing the test environment in MATLAB it became apparent that running fixed-point simulations required considerably more time than floating-point simulations (days vs. hours). The use of C-MEX files and multi-core execution parallelization was beneficial in reducing the simulation time, but still insufficient to yield simulation runs of reasonable length. For each data rate, we needed to simulate several fixed-point precision settings. For each precision setting, we needed to simulate several SNR values to obtain the FER performance curves. For each SNR value, we needed to process 10,000 channel instances to have a good statistical average. We let the limiting FER value to be 0.01, which corresponds to detecting at least 100 (among 10,000) cases where a frame is in error.

Before presenting our design methodology that avoids long simulation times when searching for appropriate fixed-point precision setting, we introduce the following quantities of interest:

- N_{SNR} – the number of random samples taken per given (mean) SNR value;
- $N_{multipath}$ – the number of independent random channel instances generated as per the given channel model (see Appendix C);
- N_{bad} – the number of "bad" channel samples considered;
- N_{good} – the number of "good" channel samples considered;
- FER_{limit} – the limiting (smallest) FER value under consideration;
- SNR_{limit} – the smallest simulated SNR value at which FER < FER_{limit};

During our floating-point simulations, we set N_{SNR} = 1 and $N_{multipath}$ = 10,000, which yields $N_{SNR} \times N_{multipath}$ = 10,000 runs per (mean) SNR value. Averaging these 10,000 cases gives us one point on a FER curve. Increasing N_{SNR} will increase the simulation time, but the effect on the performance curves will be miniscule: it is the channel instance randomness per simulated multipath model (not SNR randomness per simulated mean value) that determines the BER performance. Setting $N_{multipath}$ = 10,000 provides a sufficient coverage of the channel sampling space, and decreasing $N_{multipath}$ would compromise the reliability of the performance curves in our case, especially near our FER_{limit} = 0.01.

While evaluating various fixed-point design decisions, simulations need to be of short duration to explore as many design alternatives as possible within a limited design time frame. One way to reduce the simulation time, while maintaining the validity of the performance-related conclusions, is to first run a number of short simulations to narrow down the required fixed-point word lengths. Once the word lengths of interest are identified the longer, 10,000 channel simulation can be run. The 5-step procedure described below shows the procedure used to identify the fixed point word length using shorter simulation runs.

Step 1: We start with performing simulations with floating-point arithmetic, setting N_{SNR} = 1 and $N_{multipath}$ = 10,000, to obtain the mean floating-point FER curve for a specific data rate for each mean SNR level. From this curve, we find SNR_{limit}, i.e., the lowest simulated SNR value at which the mean FER drops below FER_{limit} = 0.01. For example, SNR_{limit} is 15 dB for 6 Mbps and 39 dB for 54 Mbps as seen in Fig. 6.2.

Step 2: From the 10,000 channel instances, two groups of channels are classified: "bad" channels corresponding to the N_{bad} worst performing channels in terms of FER, while "good" channels corresponding to the N_{good} best performing channels in terms of FER. We set N_{bad} = N_{good} = 100, or 1% of all $N_{multipath}$.

To identify the “bad” channels, a search procedure is used to find the channel instances that produce the worst FER performance curves. This is done by first noting the FER values for each generated channel instance at SNR_{limit} . The N_{bad} channels with the worst FER values at SNR_{limit} are chosen. If the number of channel instances with non-zero FER values, denoted as $N_{temp,}$ at SNR_{limit} is less than N_{bad}, the remaining channels are identified by removing the collected N_{temp} channels from the $N_{multipath}$ channel pool ($N_{multipath}$–N_{temp}) and repeating the procedure for the previous SNR value. If a case arises where more than N_{bad} channel instances have similar FER values, these channels are stored and compared with each other at a lower SNR value, the channels with worse FER values at this lower SNR value are selected. This is repeated until exactly N_{bad} worst channels are found. For the “good” channels, the procedure used is similar to the “bad” channel case but instead of looking for the worst performing FER curves, the best performing curves are identified. We evaluate our system using the identified "good" and "bad" channel instances exclusively, i.e., the multipath model is no longer randomly generated.

Step 3: We take the N_{bad} “bad” channel instances, set $N_{SNR}^{*} = 100$, and repeat the floating-point simulations. We now consider each “bad” channel instance individually and, for each simulated SNR value, we average over N_{SNR}^{*} cases to obtain one FER points. Thus, we re-generate the N_{bad} individual FER curves but with more precision. Note that the only source of randomness now is the AWGN at each SNR sampling (per given mean value). Among these new FER curves we pick the one with the greatest number of frames in error at SNR_{limit}. We designate the corresponding “bad” channel instance as the “worst-channel case”. We also repeat this step for our N_{good} “good” channel instances, in order to identify the “best-channel case” with the least number of frames in error. All of our subsequent fixed-point simulations will be limited to these two channel instances.

Step 4: Our design objective is to decide on the number of bits for the integer part and the fractional part in the fixed-point format, such that the resulting fixed-point performance closely matches the floating-point performance up to SNR_{limit}. We evaluate each design decision for the channel instance identified as being the worst in Step 3. A FER curve is generated for a simulated receiver using fixed-point arithmetic, where 100 independent simulation runs are generated for each SNR point for the identified channel instance. This curve is then compared with the floating-point worst-channel FER curve obtained in Step 3. The fixed-point simulations are run with different numbers of fixed-point bits. This is to identify the lowest number of fixed-point bits denoted as S_{wc}, needed to generate a FER curve closely matching that of the floating point curve for SNR values up to SNR_{limit}.. We repeat this process for the best-channel case as well, thus obtaining an alternative fixed-point setting, denoted by S_{bc}. The difference between S_{wc} and S_{bc} gives us the precision range for the data rate under investigation, which can serve as an empirical measure on how potentially pessimistic a design using S_{wc} bits is.

Step 5: To ensure that the average-case fixed-point performance closely matches the average-case floating-point performance for SNR up to SNR_{limit}, verification simulations are performed. A total of $N_{multipath}$ = 10,000 independent channel instances are generated and the fixed-point receiver using S_{wc} bits is simulated. Since, S_{wc} bits is sufficient for the worst case channel, it is expected that the S_{wc} word length is also sufficient for the average 10,000 channel case without degradation in the overall FER performance (compared to floating-point). We repeat such simulations with the fixed-point setting S_{bc} as well. Since, S_{bc} bits are only good for a best case scenario channel, it is expected that S_{bc} word length not be sufficient for the10,000 channel pool which contains worse channel instances. Hence, a degradation in the overall 10,000 channel average FER performance (compared to floating-point) is expected.

Our 5-step procedure described above is much faster than performing average-case fixed-point simulations for all fixed-point design choices under evaluation. Let $T_{fl\text{-}pt}$ and $T_{fx\text{-}pt}$ denote, respectively, the floating-point and fixed-point simulation times per SNR value (i.e., for one point on the FER curve). Also, let N_{design} denote the number of design choices to be evaluated, and N_{FER} denote the number of FER curve points simulated, up to SNR_{limit}. Then, the usual average-case-based approach will yield the following simulation time:

$$T = T_{fl\text{-}pt} \times N_{FER} \times N_{SNR} \times N_{multipath} + N_{design} \times T_{fx\text{-}pt} \times N_{FER} \times N_{SNR} \times N_{multipath}$$

$$T = (10{,}000) \times N_{FER} \times (T_{fl\text{-}pt} + N_{design} \times T_{fx\text{-}pt}).$$

Our 5-step approach, however, will yield

$$T^{*} = T_{fl\text{-}pt} \times N_{FER} \times N_{SNR} \times N_{multipath} + T_{fl\text{-}pt} \times N_{FER} \times N_{SNR}^{*} \times (N_{bad} + N_{good}) + N_{design} \times T_{fx\text{-}pt} \times N_{FER} \times N_{SNR}^{*} \times (1+1) + 2 \times T_{fx\text{-}pt} \times N_{FER} \times N_{SNR} \times N_{multipath}$$

$$T^{*} = 30{,}000 \times T_{fl\text{-}pt} \times N_{FER} + (200 \times N_{design} + 20{,}000) \times T_{fx\text{-}pt} \times N_{FER}.$$

Thus, we require additional 20,000 fast floating-point simulation runs, but we eliminate $(9{,}800 \times N_{design} - 20{,}000)$ slow fixed-point simulation runs. Clearly, N_{design} must be at least 3, and if $N_{design} = 5$, our approach is guaranteed to be much faster, even if $T_{fx\text{-}pt} = T_{fl\text{-}pt}$, (i.e., even if fixed-point simulations become as fast as floating-point simulations).

The results of applying our 5-step procedure are presented in the next section. Note that for FER_{limit} will have the corresponding BER_{limit}, and any design decisions can be based on the fixed-point vs. floating-point BER performance comparisons, which would offer a more detailed picture.

6.3. Proposed Receiver Performance with Fixed-Point Computations

In this section, we present the fixed-point performance of our proposed polar receiver for the data rates of 6,12,24 and 54 Mbps. For each data rate, we show the BER and FER curves averaged over all channel instances with the two fixed-point precision settings: one found from the worst-channel analysis, and the other found from the best-channel analysis. We also show the BER curves for the worst channel and the best channel for multiple choices of the fixed-point precision, to justify the settings used for generating our aforementioned average BER/FER curves.

The curve labels are in the format *TD:U/V–FD:X/Y*, where *TD:U/V* means that in the time-domain each number has the *U*-bit integer part and *V*-bit fractional part, *FD:X/Y* means that in the frequency-domain each number has the *X*-bit integer part and *Y*-bit fractional part. All fixed-point performance curves are accompanied by the corresponding floating-point performance curves. Our objective was to match the latter as closely as possible. A final note is that in all cases the FFT (between TD and FD) word length was kept constant at 24 bits (5 integer part bits, 19 fractional part bits) to avoid any precision loss and overflow within FFT interior calculations.

6.3.1. Performance at 6 Mbps (BPSK)

Fig. 6.3 shows the fixed-point performance for the best-channel case at 6 Mbps. The required time-domain word length is 14 bits, with the 5-bit integer part and the 9-bit fractional part. The required frequency-domain word length, however, is only 9 bits, with the 5-bit integer part and the 4-bit fractional part. Removing another bit from the word length would significantly degrade the performance, as indicated in Fig. 6.3.

On the other hand, Fig. 6.4 shows the fixed-point performance for the worst-channel case at 6 Mbps. The required time-domain and frequency-domain word length is 15 bits, with the 5-bit integer part and the 10-bit fractional part. Removing another bit from the word length would slightly degrade the performance, while adding another bit to the word length would not yield any performance improvements, as indicated in Fig. 6.4.

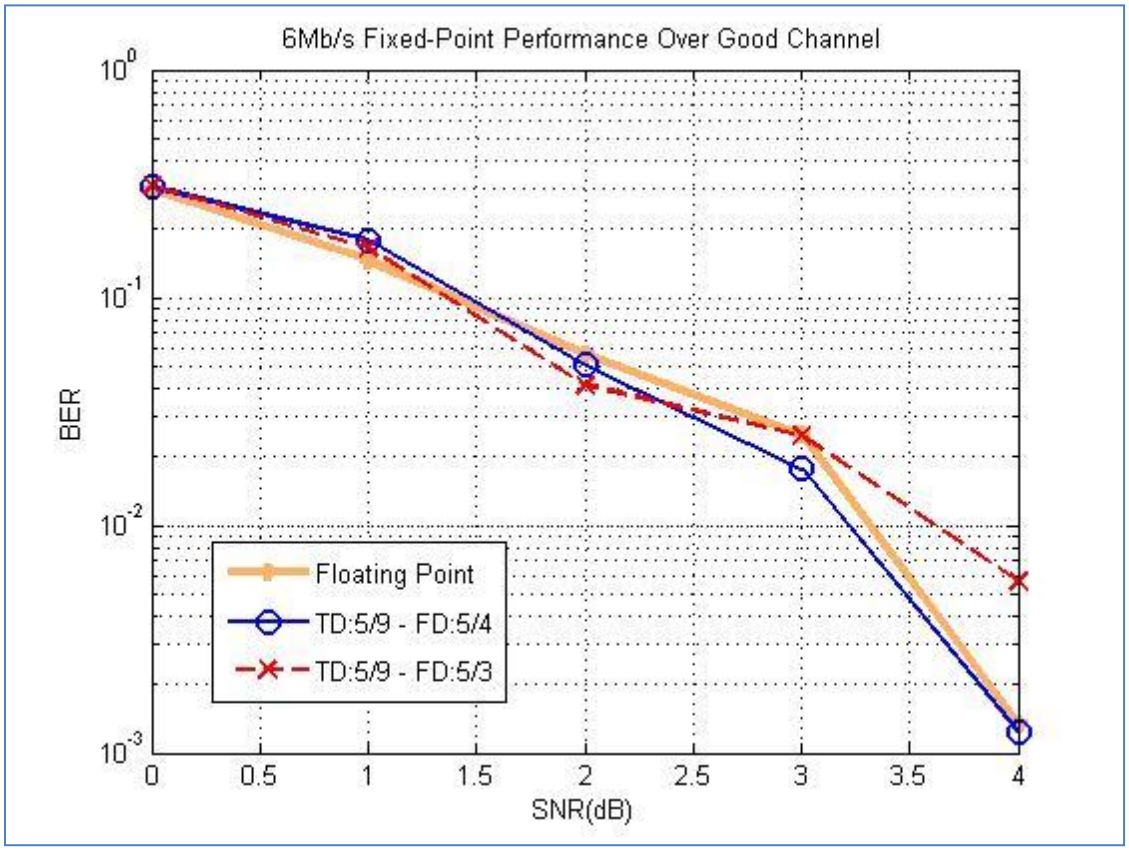


Figure 6.3: 6 Mbps Best-Channel Case

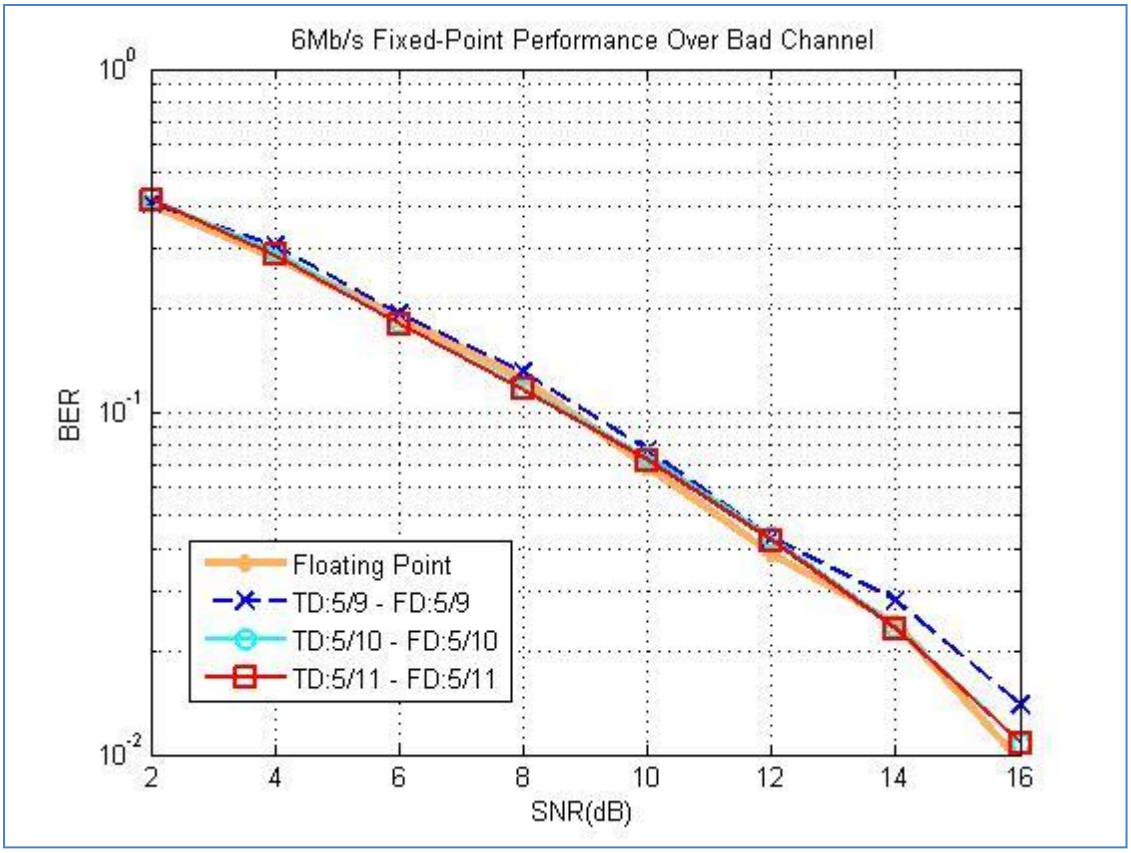


Figure 6.4: 6 Mbps Worst-Channel Case

Our final choice for the fixed-point setting is 5 bits for the integer part and 10 bits for the fractional part, in both the time and the frequency domains. The average-case performance (over all 10,000 channel instances) is shown in Fig. 6.5 (FER) and Fig. 6.6 (BER). The BER curve indicates that our fixed-point setting, based on the worst channel, results in a performance very close to the average-case floating-point performance. The alternative setting based on the best-channel, saving 1 bit in the time domain and 6 bits in the frequency domain, does not work well, but interestingly, this can be seen only from the BER curve in Fig. 6.6 (both settings show very similar FER curves in Fig. 6.5). If we let the FER be the only performance measure, then the best-channel setting is also acceptable, which suggests that at such a relatively low data rate (with the associated relatively low SNR), the frame recovery is not very sensitive to the quantization noise. As we will see later in the thesis, the higher SNR needed for the higher data rate makes the level of quantization noise more critical for good receiver performance.

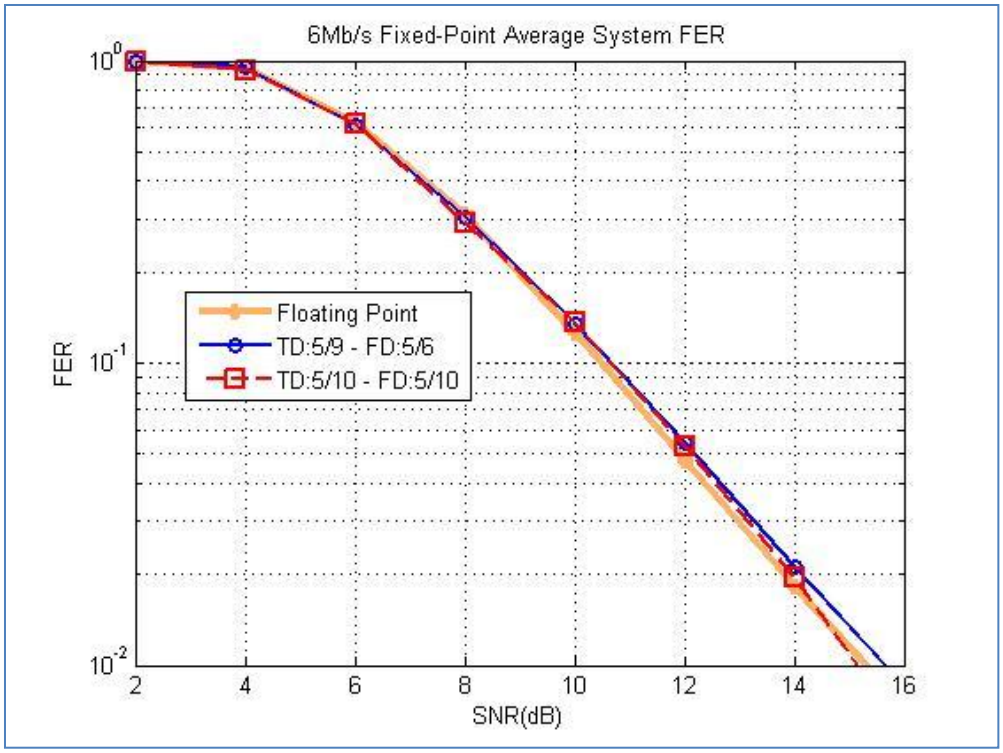


Figure 6.5: 6 Mbps Average Case (FER)

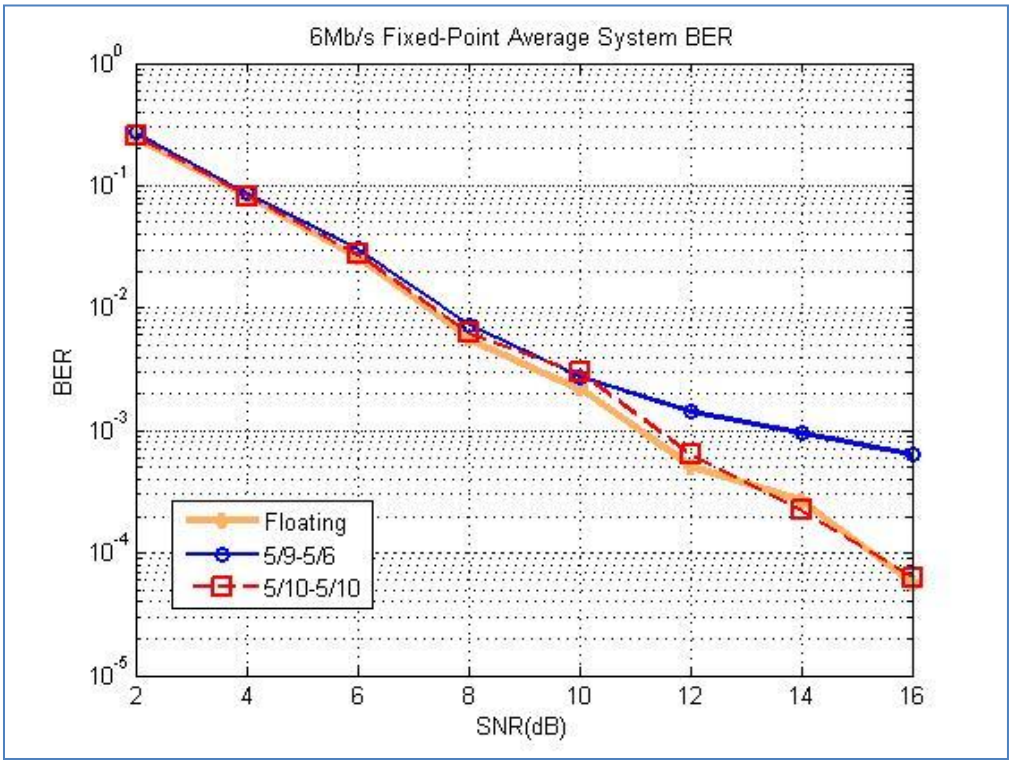


Figure 6.6: 6 Mbps Average Case (BER)

6.3.2. Performance at 12 Mbps (QPSK)

Fig. 6.7 shows the fixed-point performance for the best-channel case at 12 Mbps. The required time-domain word length is 14 bits, with the 5-bit integer part and the 9-bit fractional part. The required frequency-domain word length, however, is only 10 bits, with the 5-bit integer part and the 5-bit fractional part. Removing another bit from the word lengths slightly degrades the performance, while adding another bit to the word lengths does not yield any performance improvements, as indicated in Fig. 6.7.

Fig. 6.8 shows the fixed-point performance for the worst-channel case at 12 Mbps. The required time-domain and frequency-domain word length is 16 bits, with the 5-bit integer part and the 11-bit fractional part. Reducing another bit from the word lengths slightly degrades the performance. For a more accurate picture, both the 16 bit and 15 bit cases were simulated later using all 10,000 channel instances as shown in Fig. 6.9 and 6.10.

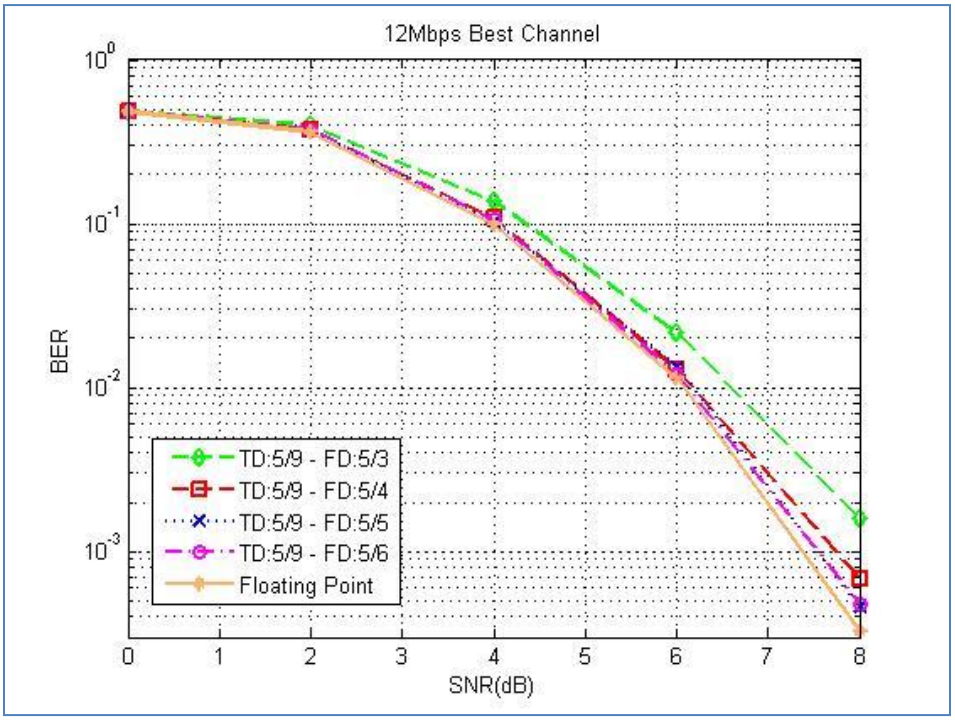


Figure 6.7: 12 Mbps Best-Channel Case

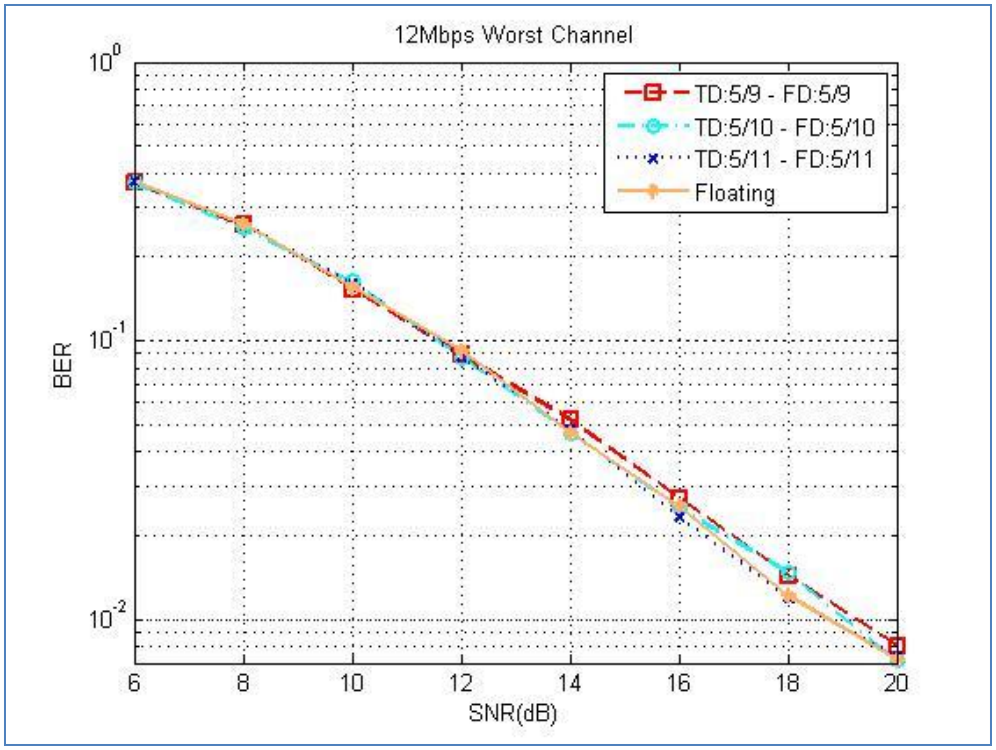


Figure 6.8: 12 Mbps Worst-Channel Case

Based on our investigations the final choice for the fixed-point setting is 5 bits for the integer part and 11 bits for the fractional part, in both the time and the frequency domains, based on the worst channel. The average-case performance (over all 10,000 channel instances) is shown in Fig. 6.9 (FER) and Fig. 6.10 (BER). The BER curve indicates that our fixed-point setting, based on the worst channel, results in a performance very close to the average-case floating-point performance. The alternative setting based on the best-channel, saving 5 bits in the frequency domain, does not work well, as expected. On the other hand, adding an extra bit to the world length does not improve performance appreciably. If we restrict our attention only to the SNR value at which FER drops below 0.1 (standard-specified threshold), then the difference between the best-channel and worst-channel settings is negligible.

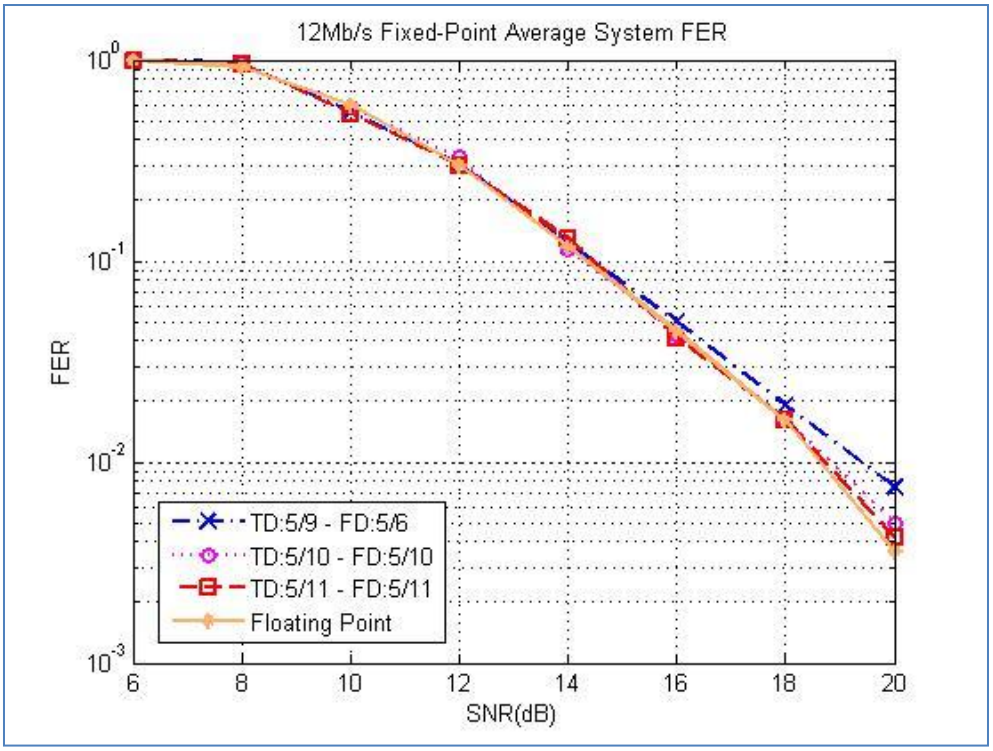


Figure 6.9: 12 Mbps Average Case (FER)

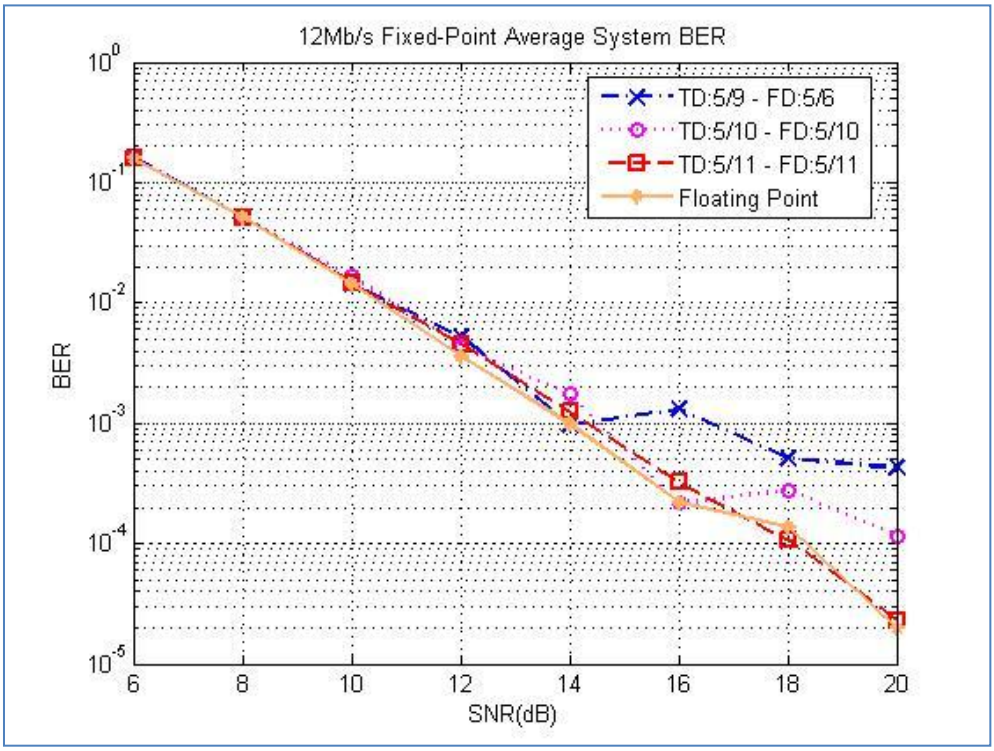


Figure 6.10: 12 Mbps Average Case (BER)

6.3.3. Performance at 24 Mbps (16-QAM)

Fig. 6.11 shows the fixed-point performance for the best-channel case at 24 Mbps. The required time-domain and frequency-domain word length is 14 bits, with the 5-bit integer part and the 9-bit fractional part. Removing another bit from the word lengths degrades the performance.

Fig. 6.12 shows the fixed-point performance for the worst-channel case at 24 Mbps. The required time-domain and frequency-domain word length is 16 bits, with the 5-bit integer part and the 11-bit fractional part. Reducing another bit from the word lengths slightly degrades the performance. For a more accurate picture, both the 16 bit and 15 bit cases were simulated later using all 10,000 channel instances as shown in Fig. 6.13 and 6.14.

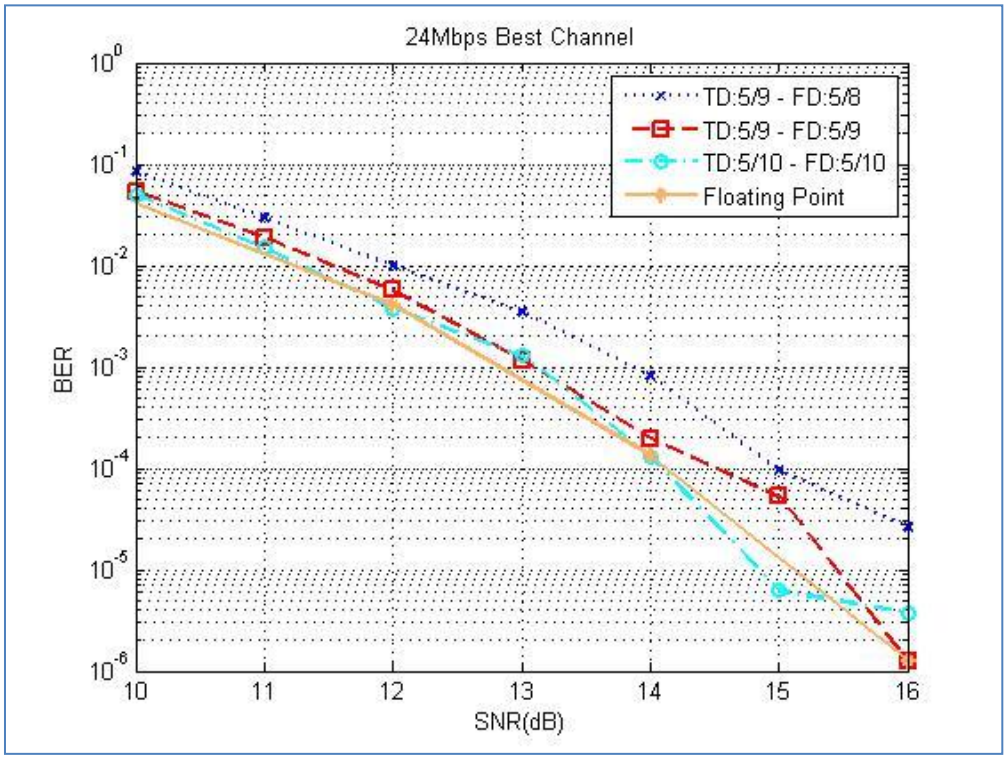


Figure 6.11: 24 Mbps Best-Channel Case

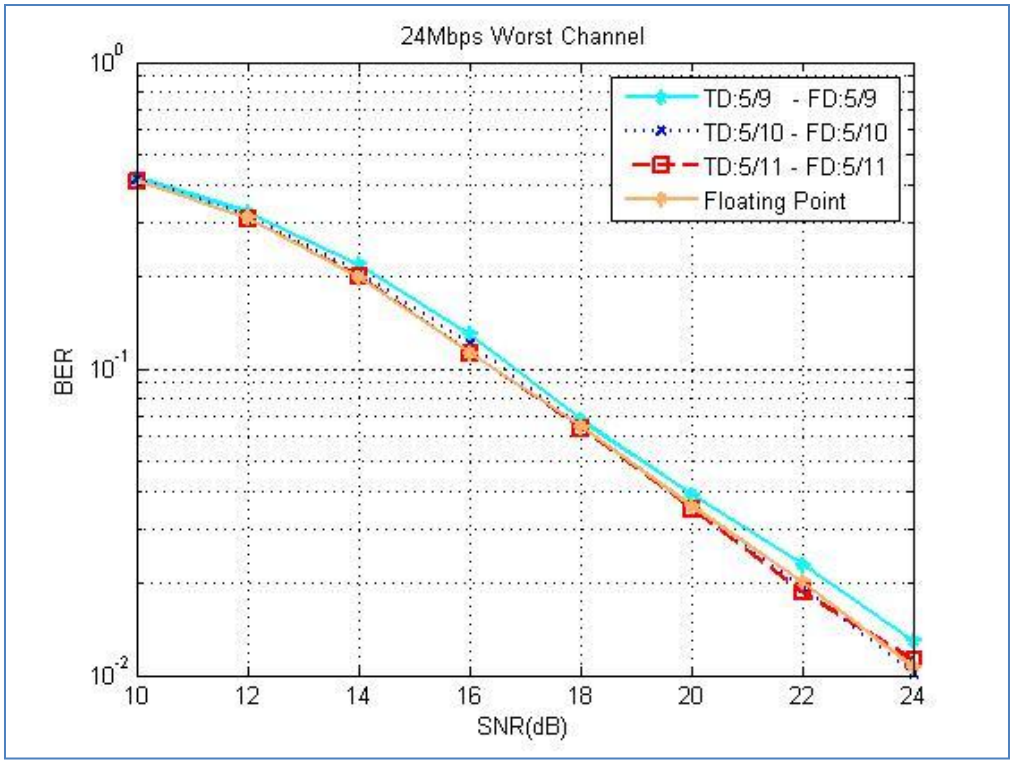


Figure 6.12: 24 Mbps Worst-Channel Case

Based on our investigations the final choice for the fixed-point setting is 5 bits for the integer part and 11 bits for the fractional part, in both the time and the frequency domains, based on the worst channel. The average-case performance (over all 10,000 channel instances) is shown in Fig. 6.13 (FER) and Fig. 6.14 (BER). Both the FER and BER curves indicate that this fixed-point setting results in the performance very close to the average-case floating-point performance. The alternative setting based on the best-channel, 2 bits in the frequency domain, does not work well, as expected. On the other hand, adding an extra bit to the world length does not improve performance appreciably. If we restrict our attention only to the SNR value at which FER drops below 0.1 (standard-specified threshold), then the difference between the best-channel and worst-channel settings is ~1 dB (~19 dB vs. ~20 dB). Whether such a difference is tolerable (i.e., the best-channel setting is acceptable) would depend on the user requirements.

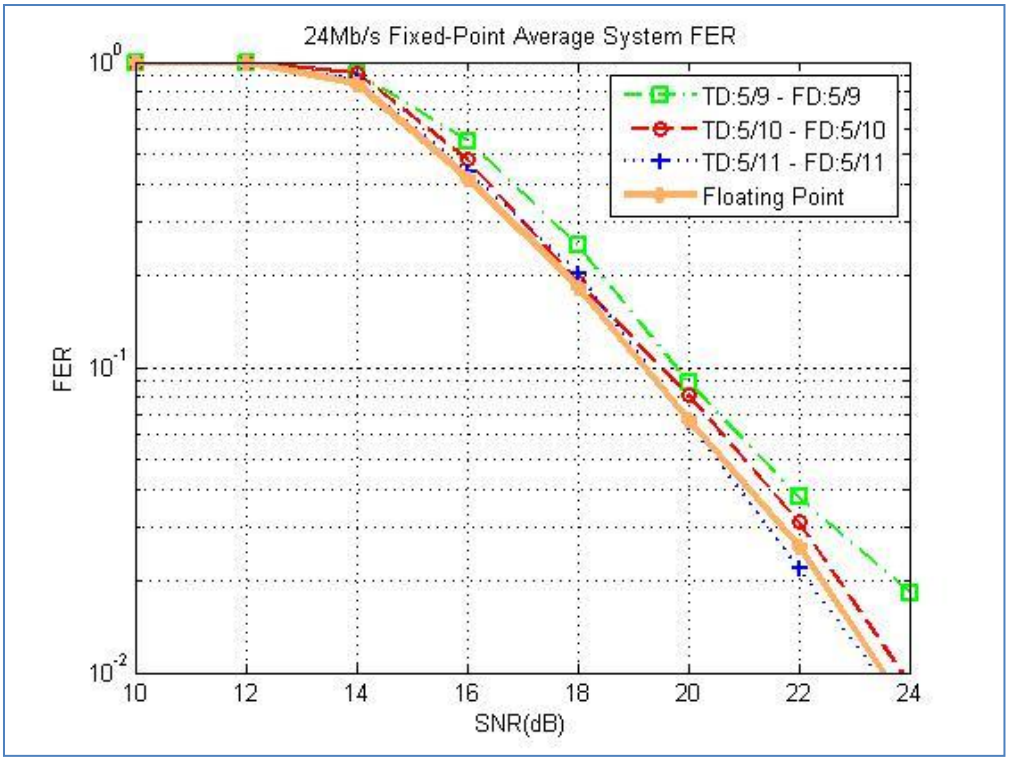


Figure 6.13: 24 Mbps Average Case (FER)

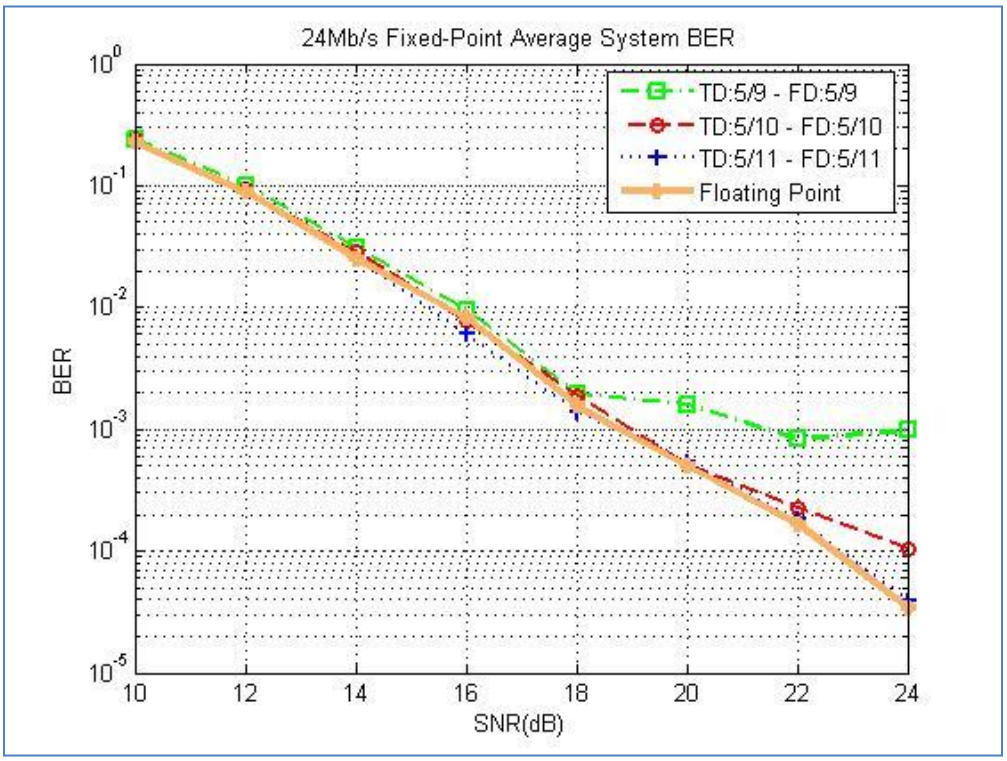


Figure 6.14: 24 Mbps Average Case (BER)

6.3.4. Performance at 54 Mbps (64-QAM)

Fig. 6.15 shows the fixed-point performance for the best-channel case at 54 Mbps. The required time-domain word length is 16 bits, with the 5-bit integer part and the 11-bit fractional part. The required frequency-domain binary word structure used for computation is the same. Removing another bit from these fractional and integer part lengths significantly degrades the performance, while adding another bit to the word length does not yield any performance improvements, as indicated in Fig. 6.15.

Fig. 6.16 shows the fixed-point performance for the worst-channel case at 54 Mbps. The required time-domain and frequency-domain word length is 20 bits, with the 6-bit integer part and the 14-bit fractional part. Reducing the set bit lengths significantly degrades the performance, while adding another bit to the word length does not yield any performance improvements, as indicated in Fig. 6.16.

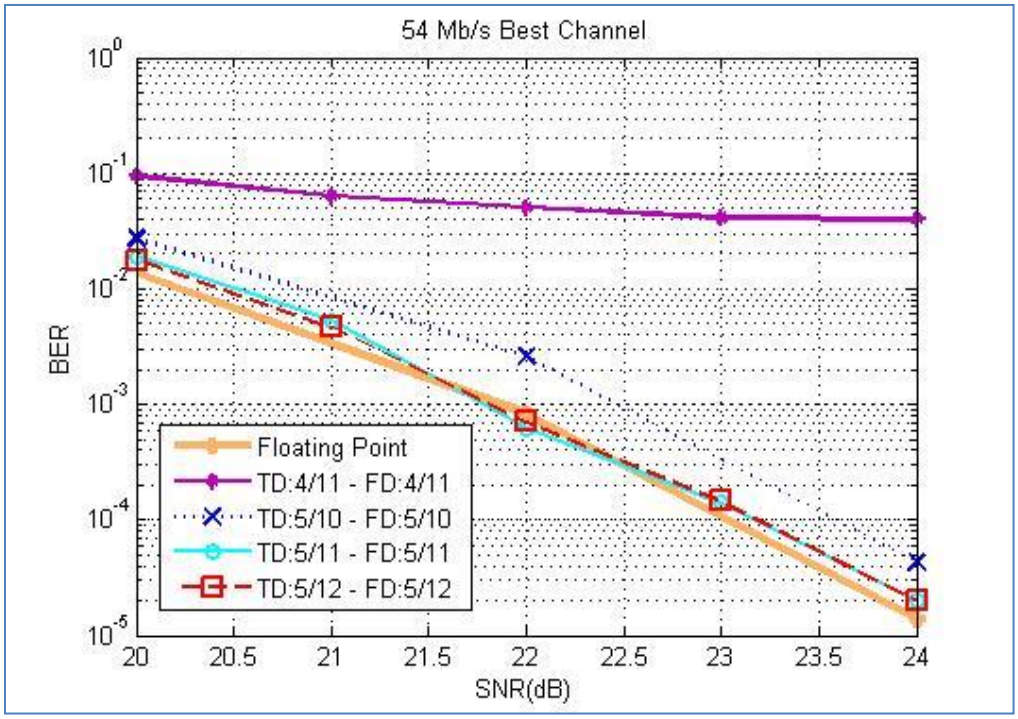


Figure 6.15: 54 Mbps Best-Channel Case

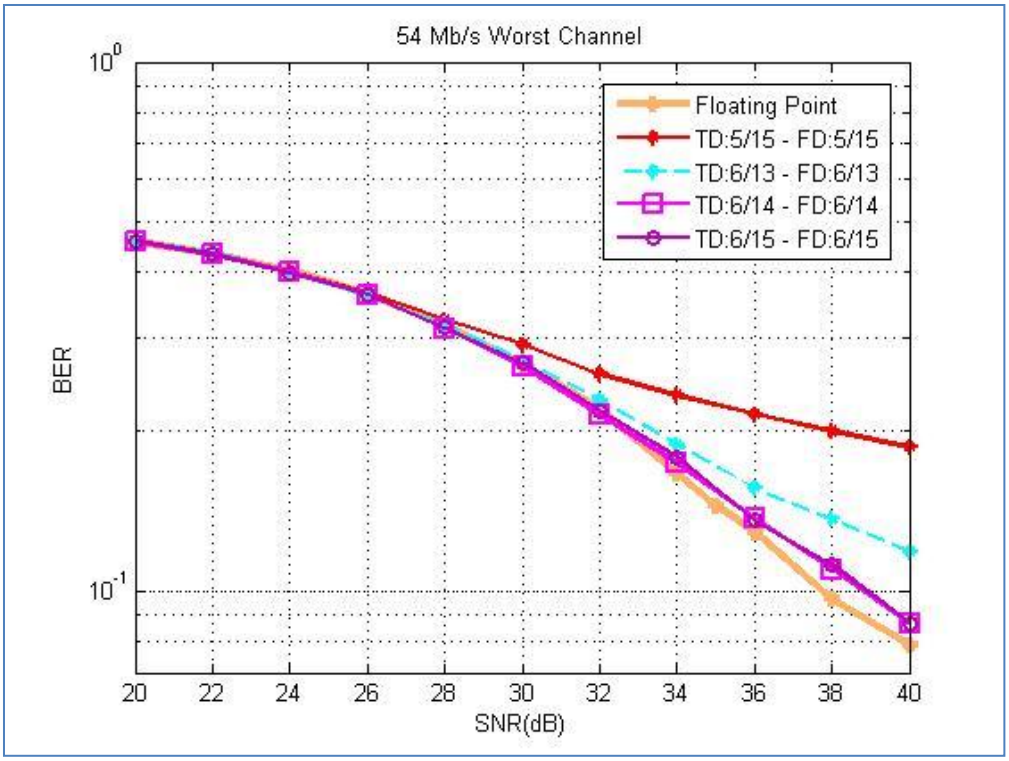


Figure 6.16: 54 Mbps Worst-Channel Case

Based on our investigations the final choice for the fixed-point setting is 6 bits for the integer part and 14 bits for the fractional part, in both the time and the frequency domains, based on the worst channel. The average-case performance (over all 10,000 channel instances) is shown in Fig. 6.17 (FER) and Fig. 6.18 (BER). Both the FER and BER curves indicate that this fixed-point setting results in the performance very close to the average-case floating-point performance. The alternative setting based on the best-channel, saving 1 bit in the time domain and 3 bits in the frequency domain, does not work well, as expected. On the other hand, adding an extra bit to the world length does not improve performance appreciably. If we restrict our attention only to the SNR value at which FER drops below 0.1 (standard-specified threshold), then the difference between the best-channel and worst-channel settings is ~1 dB (~31 dB vs. ~32 dB). Whether such a difference is tolerable (i.e., the best-channel setting is acceptable) would depend on the user requirements.

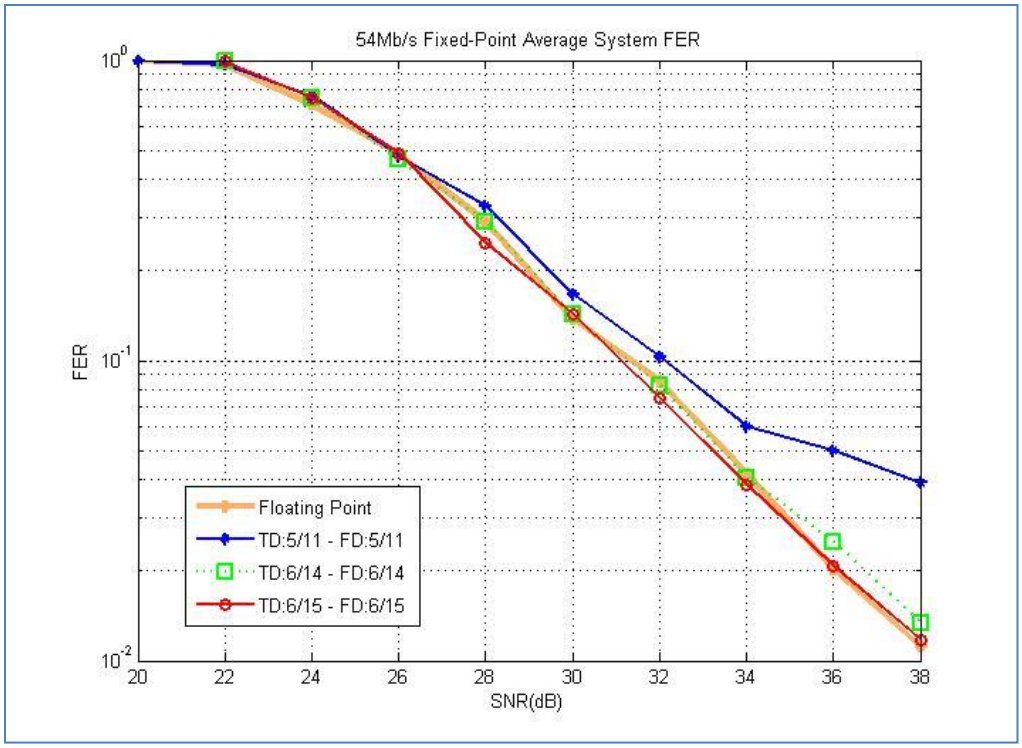


Figure 6.17: 54 Mbps Average Case (FER)

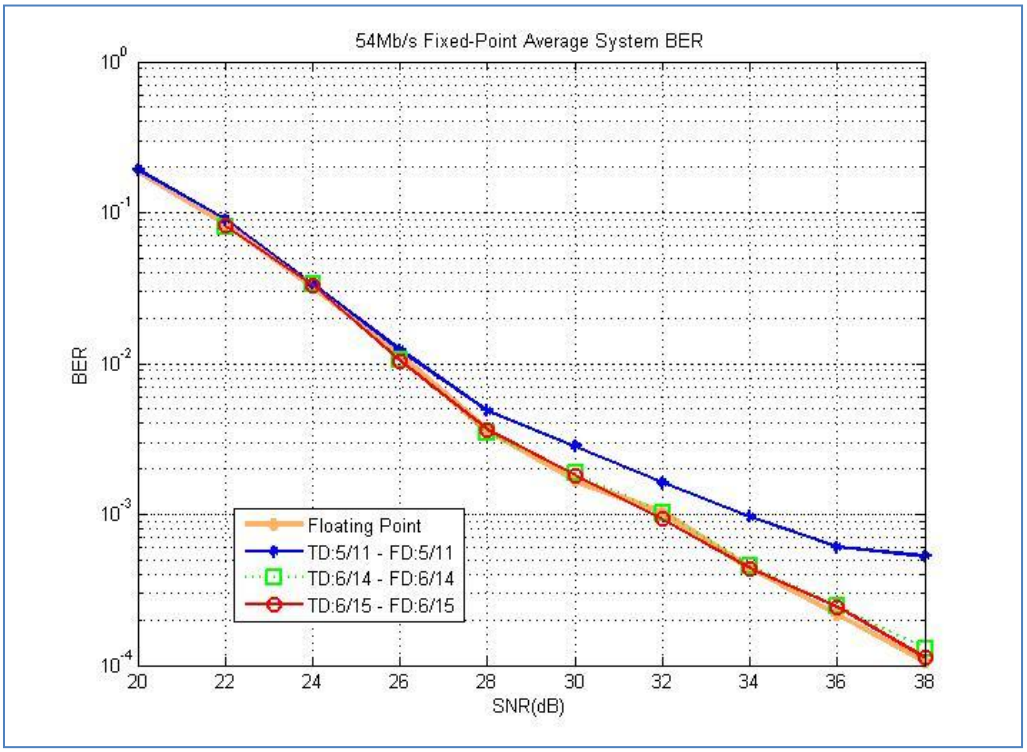


Figure 6.18: 54 Mbps Average Case (BER)

6.4. Cartesian System Performance with Fixed-Point Computations

Tests similar to those presented in 6.3 were also conducted for the reference Cartesian system. The results indicated that in most cases, Cartesian system required the same number of bits as the Polar system. However, an interesting note was that under best channel conditions and at a data rate of 6Mbs, Cartesian system required 8 bits in the fractional part which is double what was needed in the Polar system.

Further investigation identified that the reason Cartesian required more bits was the division operation performed in the channel estimator. The results of the division are too small for the CORDIC to generate with the few bits provided, causing zero values as output. Due to these zero values, the BPSK demodulator is not be able to identify the correct constellation symbol. For the polar system, the division operation is not performed on the phase stream and hence this problem does not occur. The demodulated symbols in both Polar and Cartesian case are presented in Fig. 6.11 and Fig 6.12 respectively. Fig 6.13 shows the Demodulated Symbols for the Cartesian system when 8 bits are used for the fractional part.

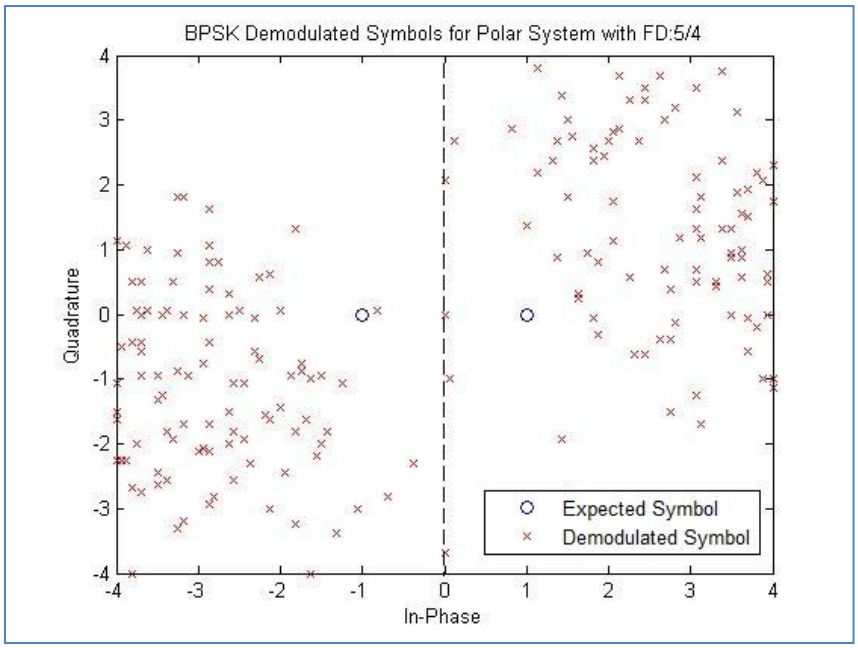


Figure 6.19: Demodulated BPSK Symbols for Polar System with 9 Bit Word Length

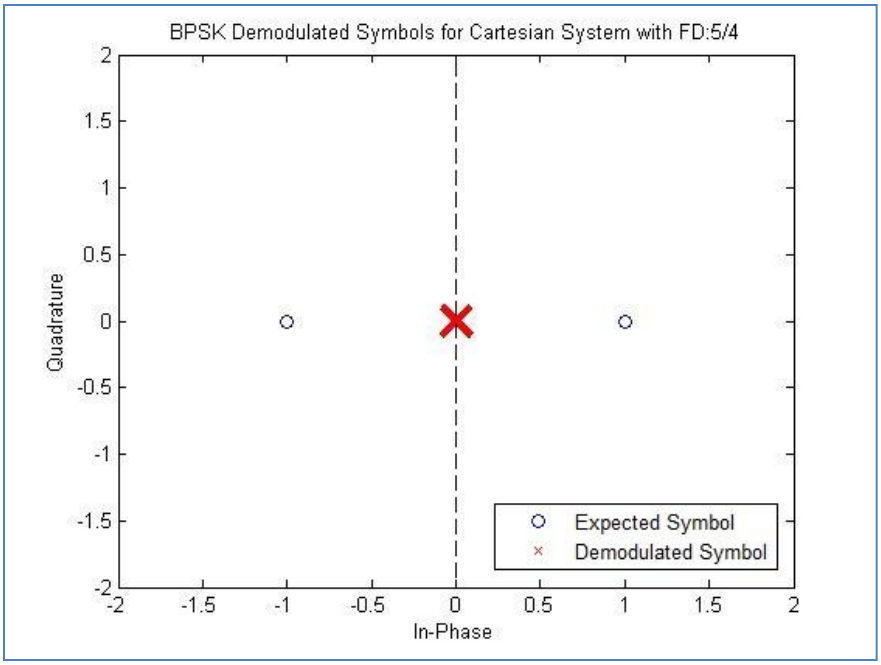


Figure 6.20: Demodulated BPSK Symbols for Cartesian System with 9 Bit Word Length

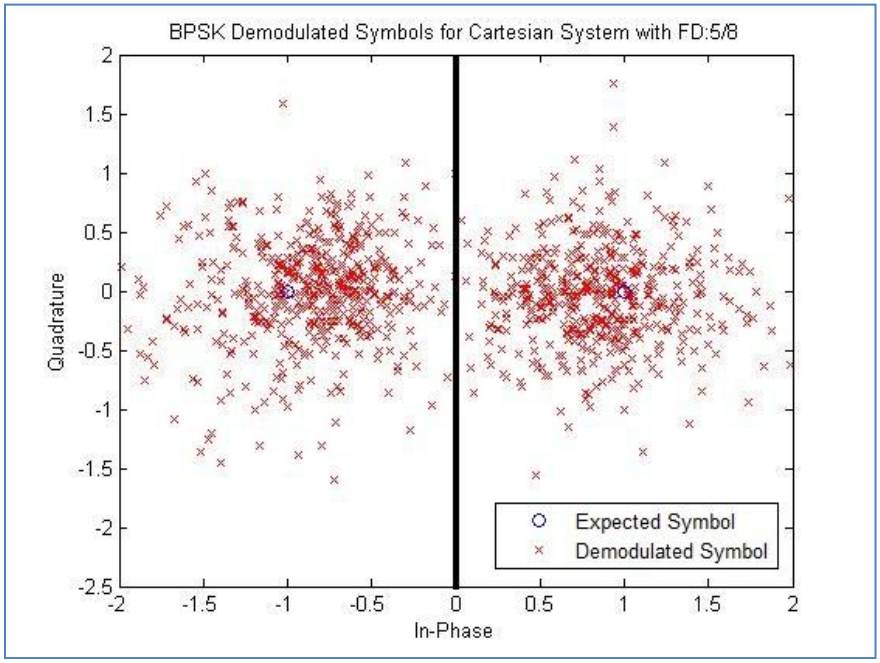


Figure 6.21: Demodulated BPSK Symbols for Cartesian System with 13 Bit Word Length

6.5. Summary

Tables 6-1 through 6-4 summarize the number of bits required by our polar receiver in the frequency domain, to adequately match the floating-point performance for the best-channel and the worst-channel cases.

Table 6-1: Fixed-Point Settings (Frequency Domain) for the Polar Receiver at 6 Mbps

System	Channel	Bits needed for Amplitude	Bits needed for phase
6Mb/s	Good channel	0	9 bits (5/4)
	Bad channel	0	15 bits (5/10)
	Overall Average	0	15 bits (5/10)

Table 6-2: Fixed-Point Settings (Frequency Domain) for the Polar Receiver at 12 Mbps

System	Channel	Bits needed for Amplitude	Bits needed for phase
12Mb/s	Good channel	0	10 bits (5/5)
	Bad channel	0	16 bits (5/11)
	Overall Average	0	16 bits (5/11)

Table 6-3: Fixed-Point Settings (Frequency Domain) for the Polar Receiver at 24 Mbps

System	Channel	Bits needed for Amplitude	Bits needed for phase
24Mb/s	Good channel	14 bits (5/9)	14 bits (5/9)
	Bad channel	16 bits (5/11)	16 bits (5/11)
	Overall Average	16 bits (5/11)	16 bits (5/11)

Table 6-4: Fixed-Point Settings (Frequency Domain) for the Polar Receiver at 54 Mbps

System	Channel	Bits needed for Amplitude	Bits needed for phase
54Mb/s	Good channel	16 bits (5/11)	16 bits (5/11)
	Bad channel	20 bits (6/14)	20 bits (6/14)
	Overall Average	20 bits (6/14)	20 bits (6/14)

These tables clearly expose the potential for computational effort scalability of our proposed receiver with respect to the data rate requirements. We have already mentioned in the previous chapter that at 6 -Mbps and 12-Mbps, one processing stream (sample magnitudes) can be completely shut down. It was interesting to note that for data rates ranging between 6-Mbps and 24-Mbps a word length of 16 bits is sufficient. Additionally, the fixed-point precision can be reduced by 20% in comparison to the setting for the 54-Mbps data rate (16 bits vs. 20 bits). This reduces not only the amount of switching in the receiver hardware (thus, reducing dynamic power consumption), but also the number of word-length-dependent CORDIC iterations (thus, reducing the processing latency).

A possible data-rate-scalable implementation scenario would be as follows. By default the receiver is configured to work with the 16-bit fixed-point precision. Once a frame is detected and synchronized, the receiver processes the first OFDM symbol, which is the SIGNAL header carrying the data rate information. After the frame's data rate becomes known, the receiver switches to the data-rate-specific precision (e.g., 20 bits for 54 Mbps). Such precision settings would be stored in an 8-byte ROM, for example, with each byte encoding the lengths of the integer and fractional parts for one of the eight 802.11a data rates.

It is important to note that there is still room for improvement. In the results provided, the fixed point word lengths chosen are what are required by the system as a whole (from FFT output to demodulator). Individual blocks such as the channel estimator & phase compensator may require different precision from one another hence further investigation is possible. Another improvement possibility is to set the word lengths separately for both phase and amplitude since they go through different streams.

As a final remark, we note that the precision range is significant, from 9 bits (the best channel at 6 Mbps) to 20 bits (the worst channel at 54 Mbps), which invites further investigations into appropriate precision adjustments based on not only the data rate, but also the channel condition (some limited initial attempts were reported in [47]). The precision-related decisions in such cases would involve assessing the channel estimation information. As reported in the previous section, for a given data rate, the difference between the best-channel and the worst-channel settings is 0-1 bit in the time domain and 2-6 bits in the frequency domain.

Chapter 7: Concluding Remarks

We have studied implementation-related issues in OFDM-based digital receivers, using the IEEE 802.11a WLAN standard as a specific wireless technology. Our first objective was to demonstrate the possibility of scaling the receiver computational complexity in relation to variable data rate requirements (ranging from 6 to 54 Mbps). During our fixed-point simulation studies, we found that simulations times were prohibitively long. Our second objective was to find a faster way of evaluating various fixed-point precision alternatives. Section 7.1 of this chapter summarizes how and to what extent our objectives have been met, while Section 7.2 discusses the limitations of this work and outlines some of the future research efforts needed.

7.1. Summary of Contributions

Our first contribution is the design and analysis of the receiver working in polar coordinates in the frequency domain, as opposed to the conventional Cartesian approach. Using polar coordinates facilitates computational scalability in relation to the data rate requirements. For the data rates of 6, 9, 12, and 18 Mbps, demodulation can be done using only the sample angles, i.e., the hardware that processes sample magnitudes can be completely shut down. For the data rates of 24 and 36 Mbps, the receiver must process sample magnitudes, and for each sample to be demodulated, it must perform one LUT read and one multiplication. For the data rates of 48 and 54 Mbps, the receiver must perform an additional LUT read and an additional multiplication.

Our second contribution is the 5-step procedure (see Chapter 6) that allows for significant reductions in the simulation time needed to evaluate multiple fixed-point precision settings. In our studies, we would have needed to perform 10,000 simulation runs (corresponding to 10,000 random channel instances) per SNR value, for each fixed-point precision setting under evaluation, for each data rate under investigation. Using our procedure instead, we introduced additional 20,000 fast floating-point simulation runs, but eliminated ($9{,}800 \times N_{design} - 20{,}000$) slow fixed-point simulation runs, where N_{design} is the number of precision settings tried. If N_{design} is no less than 5, our approach is guaranteed to be much faster.

Our third contribution is the evaluation and recommendations on the fixed-point precision in the cases of 6 Mbps and 54 Mbps, the minimum and the maximum data rates from 802.11a. We found that using 15 bits (5 bits for the integer part, 10 bits for the fractional part), in both the time domain and the frequency domain, was sufficient to closely match the average-case floating-point FER/BER performance for 6 Mbps. For 12Mbps and 24Mbps a word length of 16 bits (5 bits for the integer part, 11 bits for the fractional part) was sufficient. As for 54 Mbps, we found that using 20 bits (6 bits for the integer part, 14 bits for the fractional part), in both the time domain and the frequency domain, was sufficient to closely match the average-case floating-point FER/BER performance.

7.2. Future Work

The work presented in this thesis has several limitations that may motivate further research. Some of our recommendations for future work are presented below:

- **Soft Decoding:** Our proposed receiver was designed and evaluated using hard decoding. One important extension would be to introduce soft decoders, but it will require a non-trivial generation of soft-bits using polar coordinates. Using soft decoding has the benefit of generally improving the BER performance by 3dB.
- **Phase Noise and SFO:** In our studies we considered only the CFO (used in the channel model), as it is more severe and common in WLAN propagation scenarios than other oscillator imperfections. It would be of interest to extend and evaluate our polar receiver with respect to the SFO and phase noise compensation.
- **Channel-Dependent Precision:** Our work has exposed the fixed-point precision dependency on the data rate. We have also seen that there is a difference between the best-channel and the worst-channel settings, i.e., the channel quality is an obvious factor affecting the receiver performance. It may be valuable to investigate practical techniques for channel quality estimation (including both the multipath fade and the SNR), and then relax the worst-channel setting (i.e., switch to a lower precision) when appropriate.

- **SNR Estimation:** Estimation of the SNR of received signals (e.g., see [21]) can be useful for soft-bit detection, more accurate channel estimation, and finer tuning of the fixed-point precision setting.
- **Extension to 802.11n:** The relatively new 802.11n standard retains many features of the 802.11a standard (for backward compatibility), but provides data rates of up to 300 Mbps using a wider bandwidth of 40 MHz and MIMO (Multiple Input Multiple Output) antennas. It would be of interest to adapt our polar receiver to this standard and evaluate its error-rate performance, as well as its potential for computational scalability.

Bibliography

[1] M.Rahman, S.Das and F.Fitzek. (2005). "OFDM Based WLAN Systems". Center for TeleInFrastruktur (CTiF), Aalborg University. Denmark.Technical report R-04-1002: v1.2

[2] A. Goldsmith, Wireless Communications. New York: Cambridge Univ. Press, 2005

[3] C. Langton, "Orthogonal Frequency Division Multiplexing (OFDM) Tutorial", 2004. [online] Available: http://www.complextoreal.com/chapters/ofdm2.pdf [Accessed: Aug. 16, 2009]

[4] R. van Nee and R. Prasad, OFDM Wireless Multimedia Communications. Boston, MA: Artech House, 2000

[5] R.Ziemer, W.Tranter and D.Fannin, Signals & Systems Continuos and Discrete 4th Edition. New Jersey: Prentice-Hall, 1998.

[6] IEEE Standard 802.11a-1999. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5GHz Band. IEEE, September 1999

[7] Hen-Geul Yeh; Hong Seok Seo; , "Low complexity demodulator for M-ary QAM," Wireless Telecommunications Symposium, 2007. WTS 2007 , vol., no., pp.1-6, 26-28 April 2007

[8] Schmidl, T.M.; Cox, D.C.; , "Robust frequency and timing synchronization for OFDM," Communications, IEEE Transactions on , vol.45, no.12, pp.1613-1621, Dec 1997

[9] Moose, P.H.; , "A technique for orthogonal frequency division multiplexing frequency offset correction," Communications, IEEE Transactions on , vol.42, no.10, pp.2908-2914, Oct 1994

[10] Pierri, J. (2007) Design and Implementation of an OFDM WLAN Synchronizer. M.A.Sc Thesis. University of Waterloo.

[11] Jianhua Liu; Jian Li; , "Parameter estimation and error reduction for OFDM-based WLANs," Mobile Computing, IEEE Transactions on , vol.3, no.2, pp. 152- 163, April-June 2004

[12] K.Wang, M. Faulkner, J. Singh, and I. Tolochko. "Timing Synchronization for 802.11a WLANs under Multipath Channels." In Proc. ATNAC 2003, 2004.

[13] Fort, A.; Eberle, W.; , "Synchronization and AGC proposal for IEEE 802.11 a burst OFDM systems," Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE , vol.3, no., pp. 1335- 1338 vol.3, 1-5 Dec. 2003

[14] Fort, A.; Weijers, J.-W.; Derudder, V.; Eberle, W.; Bourdoux, A.; , "A performance and complexity comparison of auto-correlation and cross-correlation for OFDM burst synchronization," Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on , vol.2, no., pp. II- 341-4 vol.2, 6-10 April 2003

[15] Taekyu Kim; Sin-Chong Park; , "A New Symbol Timing and Frequency Synchronization Design for OFDM-based WLAN Systems," Advanced Communication Technology, The 9th International Conference on , vol.3, no., pp.1669-1672, 12-14 Feb. 2007

[16] Taehyeun Ha; Seongjoo Lee; Jaseok Jim; , "Low-complexity correlation system for timing synchronization in IEEE802.11a wireless LANs," Radio and Wireless Conference, 2003. RAWCON '03. Proceedings , vol., no., pp. 51- 54, 10-13 Aug. 2003

[17] Speth, M.; Fechtel, S.A.; Fock, G.; Meyr, H.; , "Optimum receiver design for wireless broad-band systems using OFDM. I," Communications, IEEE Transactions on , vol.47, no.11, pp.1668-1677, Nov 1999

[18] Bittner, S.; Frotzscher, A.; Fettweis, G.; Deng, E.; , "Oscillator Phase Noise compensation using Kalman tracking," Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on , vol., no., pp.2529-2532, 19-24 April 2009

[19] Garcia Armada, A.; , "Understanding the effects of phase noise in orthogonal frequency division multiplexing (OFDM)," Broadcasting, IEEE Transactions on , vol.47, no.2, pp.153-159, Jun 2001

[20] van de Beek, J.-J.; Edfors, O.; Sandell, M.; Wilson, S.K.; Borjesson, P.O.; , "On channel estimation in OFDM systems," Vehicular Technology Conference, 1995 IEEE 45th , vol.2, no., pp.815-819 vol.2, 25-28 Jul 1995

[21] Jie Ma; Hua Yu; Shouyin Liu; , "The MMSE Channel Estimation Based on DFT for OFDM System," Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on , vol., no., pp.1-4, 24-26 Sept. 2009

[22] Jeong, Kyowon; Lee, Jungwoo; , "Low Complexity Channel Tracking for Adaptive MMSE Channel Estimation in OFDM," Information Sciences and Systems, 2007. CISS '07. 41st Annual Conference on , vol., no., pp.548-552, 14-16 March 2007

[23] Colieri, S.; Ergen, M.; Puri, A.; Bahai A; , "A study of channel estimation in OFDM systems," Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th , vol.2, no., pp. 894- 898 vol.2, 2002

[24] Wonchae Kim; Cox, D.C.; , "Residual Frequency Offset and Phase Compensation for OFDM Systems," Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th , vol., no., pp.2209-2213, Sept. 30 2007-Oct. 3 2007

[25] Jimenez, V.P.G.; Garcia, M.J.F.-G.; Serrano, F.J.G.; Armada, A.G.; , "Design and implementation of synchronization and AGC for OFDM-based WLAN receivers," Consumer Electronics, IEEE Transactions on , vol.50, no.4, pp. 1016- 1025, Nov. 2004

[26] Akita, K.; Sakata, R.; Sato, K.; , "A phase compensation scheme using feedback control for IEEE 802.11a receiver," Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th , vol.7, no., pp. 4789- 4793 Vol. 7, 26-29 Sept. 2004

[27] Sakata, R.; Akita, K.; Sato, K.; , "Real-time Phase Tracking Method for IEEE 802.11a/g/n Receiver under Phase Noise Condition," Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd , vol.4, no., pp.1951-1955, 7-10 May 2006

[28] Jo, J.; Hyung-Woo Kim; Dong-Seog Han; , "Residual frequency offset compensation for IEEE 802.11a," Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th , vol.3, no., pp. 2201- 2204 Vol. 3, 26-29 Sept. 2004

[29] Demir, A.; Mehrotra, A.; Roychowdhury, J.; , "Phase noise in oscillators: a unifying theory and numerical methods for characterization," Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on , vol.47, no.5, pp.655-674, May 2000

[30] Xiao-Xin Zhang; Yuping Zhao; , "Joint Estimation for Both AGC and DC Based on Distribution Function for OFDM Systems," Communications, 2008. ICC '08. IEEE International Conference on , vol., no., pp.1312-1316, 19-23 May 2008

[31] Hagenauer, J.; Hoeher, P.; , "A Viterbi algorithm with soft-decision outputs and its applications ," Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM '89., IEEE , vol., no., pp.1680-1686 vol.3, 27-30 Nov 1989

[32] Y.Chen; W.Lin; J.Lin;,"Channel Estimation Technique with Assistance of PN-Coded Training Sequences for Wireless OFDM Communications" IWCMC'07, 12-16 Aug. 2007

[33] Sourour, E.; El-Ghoroury, H.; McNeill, D.; , "Frequency offset estimation and correction in the IEEE 802.11a WLAN," Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th , vol.7, no., pp. 4923- 4927 Vol. 7, 26-29 Sept. 2004

[34] Volder, Jack E.; , "The CORDIC Trigonometric Computing Technique," Electronic Computers, IRE Transactions on , vol.EC-8, no.3, pp.330-334, Sept. 1959

[35] R. Andraka, “A survey of CORDIC algorithms for FPGAs,” in FPGA ‘98, Proceedings of the 1998 ACM/SIGDA sixth international symposium on field programmable gate arrays, Monterey, CA, Feb. 1998, pp. 191–200.

[36] F. Angartia, M. J. Canet, T. Sansaloni, A. Perez-Pascual and J. Valls, "Efficient Mapping of CORDIC Algorithm for OFDM-Based WLAN", Springer Science, Journal of Signal Processing Systems 52, 181–191, 2008

[37] Tran, A.T.; Truong, D.N.; Baas, B.M.; , "A complete real-time 802.11a baseband receiver implemented on an array of programmable processors," Signals, Systems and Computers, 2008 42nd Asilomar Conference on , vol., no., pp.165-170, 26-29 Oct. 2008

[38] Rodrigues, T.K.; Swartzlander, E.E.; , "Adaptive CORDIC: Using Parallel Angle Recoding to Accelerate Rotations," Computers, IEEE Transactions on , vol.59, no.4, pp.522-531, April 2010

[39] Dick, C.; Harris, F.; , "FPGA implementation of an OFDM PHY," Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on , vol.1, no., pp. 905- 909 Vol.1, 9-12 Nov. 2003

[40] Canet, M.J.; Vicedo, F.; Almenar, V.; Valls, J.; De Lima, E.R.; , "A common FPGA based synchronizer architecture for Hiperlan/2 and IEEE 802.11a WLAN systems," Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on , vol.1, no., pp. 531- 535 Vol.1, 5-8 Sept. 2004

[41] Serra, M.; Marti, P.; Carrabina, J.; , "Implementation of a channel equalizer for OFDM wireless LANs," Rapid System Prototyping, 2004. Proceedings. 15th IEEE International Workshop on , vol., no., pp. 232- 238, 28-30 June 2004

[42] Wei Zhong; Zhigang Mao; , "Design and VLSI Architecture of a Channel Equalizer Based on Adaptive Modulation for IEEE 802.11a WLAN," Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on , vol., no., pp.1699-1702, 4-7 Dec. 2006

[43] Canet, M.J.; Vicedo, F.; Almenar, V.; Valls, J.; , "FPGA implementation of an IF transceiver for OFDM-based WLAN," Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on , vol., no., pp. 227- 232, 13-15 Oct. 2004

[44] Garcia, J.; Cumplido, R.; , "On the design of an FPGA-based OFDM modulator for IEEE 802.11a," Electrical and Electronics Engineering, 2005 2nd International Conference on , vol., no., pp. 114- 117, 7-9 Sept. 2005

[45] A.Oppenheim, R.Schafer and J.Buck, Discrete-Time Signal Processing 2nd Edition. New Jersey: Prentice-Hall, 1999

[46] Sang Yoon Park; Nam Ik Cho; , "Fixed-point error analysis of CORDIC processor based on the variance propagation formula," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.51, no.3, pp. 573- 584, March 2004

[47] Yoshizawa, S.; Miyanaga, Y.; , "Use of a Variable Wordlength Technique in an OFDM Receiver to Reduce Energy Dissipation," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.55, no.9, pp.2848-2859, Oct. 2008

[48] J. Muller, Elementary Functions: Algorithms and Implementation 2nd Edition. Boston; Birkhauser, 2006

[49] J. Medbo and P. Schramm, 'Channel Models for HIPERLAN/2," ETSI/BRAN document, no.3ERI085B (1998)

Appendix A: Wireless Channels

There are several phenomena present in wireless channels that introduce challenges not faced in wired communication links. The main issues experienced in wireless channels are path loss, shadowing, Doppler shift and multipath. To design modems capable of working in a wireless environment, different characteristics of the channel need to be understood and modelled accurately to provide a basis for designers to work with.

A.1 Signal Model

In communication systems, all signals are modulated sinusoids that can be represented by the in-phase (I) and quadrature (Q) components. Channels introduce random variations in the I/Q values, i.e., the received signal will be different from the transmitted signal. For simplicity of analysis, mathematical models of signals use complex notation [2], as illustrated by Eq. (A.1) showing a basic transmitted signal *s(t)*:

$$s(t) = R\{u(t)e^{j2\pi f_c t}\} \quad \text{(A.1)}$$

$$= R\{u(t)\}\cos(2\pi f_c t) - I\{u(t)\}\sin(2\pi f_c t) \quad \text{(A.2)}$$

$$= s_I(t)\cos(2\pi f_c t) - s_Q(t)\sin(2\pi f_c t) \quad \text{(A.3)}$$

The signal *u(t)* is a complex envelope equivalent low-pass signal of *s(t)*. The exponential term in Eq. (A.1) modulates *u(t)* by the carrier frequency f_c. From Eq.(A.3) it can be seen that *u(t)* represents the I and Q parts of *s(t)* as a complex number: $u(t) = s_I(t) + js_Q(t)$. Similarly, the received signal is represented as:

$$r(t) = R\{(u(t) * h(t))e^{j2\pi f_c t}\} \quad \text{(A.4)}$$

Where *h(t)* is the equivalent channel impulse response and $(*)$ is the convolution operation [2].

A.2 Path Loss

Path loss defines the amount of power lost over distance in a wireless medium. Path loss is affected mainly by the carrier frequency's wavelength λ, antenna gain $G_l$and signal propagation distance *d*. A general representation of the effect of path loss on the received signal is given in Eq.(A.5).

$$r(t) = R\left\{\frac{\lambda\sqrt{G_l}e^{-j2\pi d/\lambda}}{4\pi d}u(t)e^{j2\pi f_c t}\right\} \quad \text{(A.5)}$$

The power lost due to path loss is given by:

$$\frac{P_r}{P_t} = \left[\frac{\lambda\sqrt{G_l}}{4\pi d}\right]^2 \quad \text{(A.6)}$$

Eq. (A.6) shows that the power is inversely proportional to the square of the distance. There are many different models available for path loss depending on the scenario being investigated [2].

Path loss is mainly dealt with via different antenna designs that aim to provide better gain. Different antenna designs provide different beam patterns and gains depending on the needed application and the used frequency range.

A.3 Shadowing

Shadowing is a general term used to determine the effect of objects (trees, buildings, mountains, etc.) on a signals path. Signals may be blocked, reflected or diffracted depending on the objects materials and shapes. Shadowing can have varying effect from one area to another, and the resulting power loss can be modelled as a log-normal distribution as in Eq.(A.7) [2].

$$p(\psi) = \frac{\xi}{\sqrt{2\pi}\sigma_{\psi dB}\psi} exp\left[-\frac{(10\log_{10}\psi-\mu_{\psi dB})^2}{2\sigma_{\psi dB}^2}\right], \psi > 0, \quad \text{(A.7)}$$

Where, $\psi = P_r/P_t$, $\xi = 10/\ln 10$, $\mu_{\psi dB}$ is the mean of ψ, and $\sigma_{\psi dB}^2$ is the standard deviation of ψ. The values of $\mu_{\psi dB}$ are derived from empirical measurements or analytical models and depend on the propagation environment (indoor, urban, forests, mountains, etc.) [2].

A.4 Doppler Shift

Doppler shift is a phenomenon that causes a slight shift to the perceived frequencies at the receiver. This shift is commonly called the Doppler frequency f_D. There are three parameters that affect the Doppler frequency: the carrier frequency f_c, the angle between transmitter and receiver θ, and the velocity of the moving object v [2].

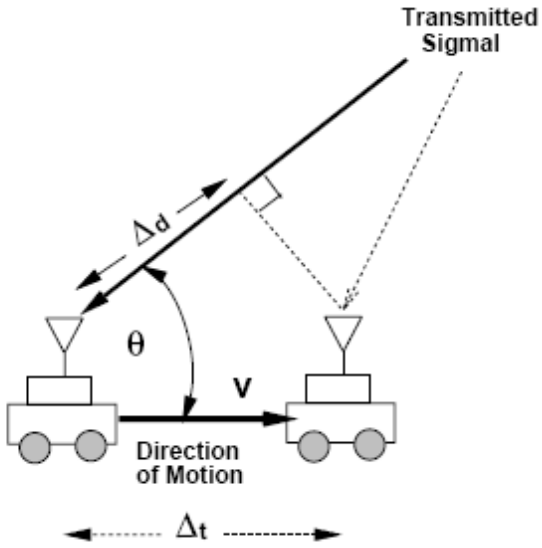


Figure A.1: Doppler Shift[2]

Fig. A.1 shows the change of the moving receiver location with time. The change in distance is $\Delta d = v\Delta t cos\theta/\lambda$, where Δt is the change in time and λ=c/f_c (c is the speed of light and f_c is the carrier frequency). This change in distance affects the phase of arrival of transmitted signals by $\Delta\varphi = 2\pi v\Delta t cos\theta/\lambda$. The Doppler frequency is hence given by

$$f_D = \frac{1}{2\pi}\frac{\Delta\varphi}{\Delta t} = \frac{v\cdot\cos\theta}{\lambda} \qquad \text{(A.8)}$$

If the receiver is moving towards the transmitter, f_D is positive; if the receiver is moving away from transmitter, f_D is negative [2].

A.5 Multipath

In a typical wireless channel, the receiver receives multiple copies of the originally sent signal. Each copy is a result of signal propagations passing through different paths, each experiencing different path loss and shadowing effects (see Fig. A.2). At the receiver, paths add up either constructively or destructively, depending on the phase of arrival. Path delays can also cause Inter Symbol Interference (ISI) due to paths from previous symbols arriving at times allocated for next signal.

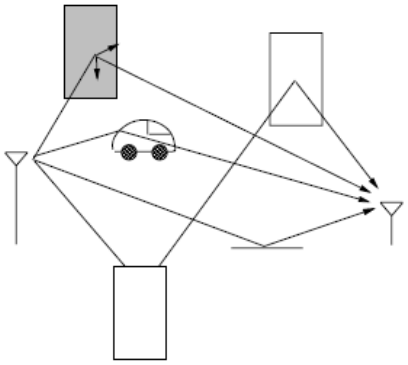


Figure A.2: Multipath [2]

Fig. A.3 shows how path loss, shadowing and multipath affect the transmitted signal power as the distance increases. It can be seen that path loss and shadowing have slow varying effects on the channel, whereas multipath has fast varying effects. This fast varying nature of multipath is a serious concern, as small variations in the channel can have very different multipath effects [2].

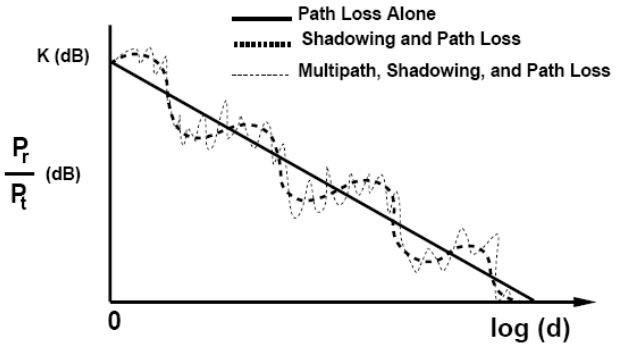


Figure A.3: Effect of Path Loss, Shadowing and Multipath on Power Loss [2]

There are two key multipath attributes – the delay spread and the coherence bandwidth – that help determine how multipath compares to the symbol period and the signal bandwidth. Another important factor is whether a multipath channel is stationary or changing with time. These are briefly described next.

A.5.1 Delay Spread

Each path in a multipath channel carries a copy of the transmitted signal arriving with a delay τ_i from the first received path. The delay spread T_{ds} is generally defined as the maximum time delay between the first and last received paths. Delay spread is also sometimes defined as the time delay between the mean delay $\bar{\tau}$ and the last path delay [2].

Fig. A.4 shows an example of two types of multipath, resolvable and non-resolvable. Two paths are resolvable if the delay between them $\Delta\tau$ is greater than transmitted symbol period T_{SYM}. If $\Delta\tau$ is smaller than T_{SYM} the paths are non-resolvable and appear as a single path with phases and amplitudes added to each other. Such addition of paths causes fading variations in the received signal strength [2].

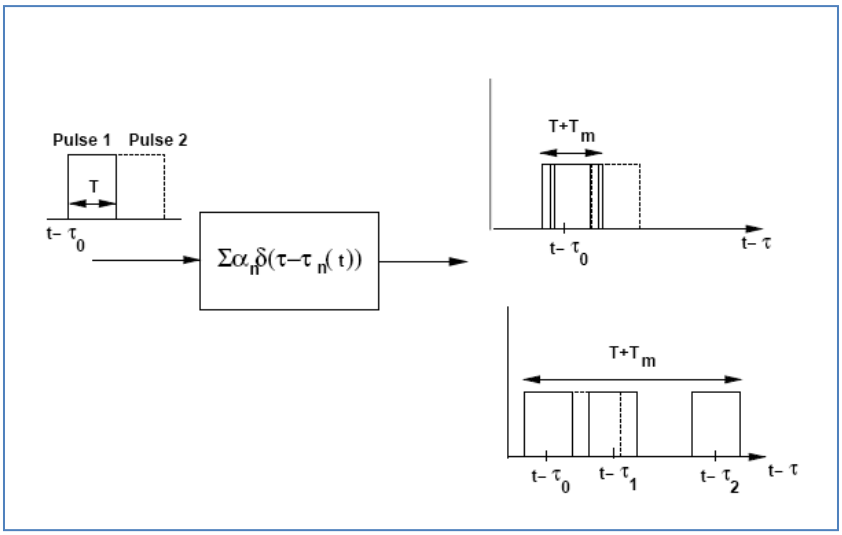


Figure A.4: Resolvable and Non-Resolvable Multipath [2]

At first glance one may think that resolvable paths are preferred due to their slower variations. However, if the delay spread T_{ds} becomes greater than the transmitted symbol period T_{SYM} the delay spread is said to be significant. A significant delay spread can lead to multipath signals overlapping with the next transmitted symbol, causing signal distortion and ISI [2].

A.5.2 Coherence Bandwidth

The Coherence Bandwidth is defined as the inverse of the delay spread $B_c = 1/T_{ds}$. If B_c is greater than the signal bandwidth (narrowband channel), the resulting fading is correlated within the band and is hence named *flat fading*. On the other hand, if B_c is less than the signal bandwidth (wideband channel), the fading differs at different regions of the band and is dubbed *frequency-selective fading*.

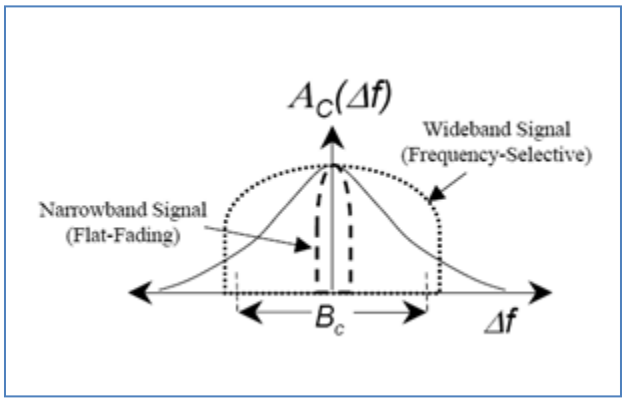


Figure A.5: Coherence Bandwidth [2]

A.5.3 Time Varying Channels

A multipath channel is said to be time varying (non-stationary) when the multipath environment is changing with time. Changes in a multipath environment can be caused by the receiver moving into an area with different multipath properties such as gains, phases and number of paths. Time varying channels introduce random delay spreads that require sophisticated tracking at the receiver [2].

A.5.4 Multipath Model

Putting it all together, a multipath channel impulse response for a time variant channel can be modeled as follows [2]:

$$h(\tau, t) = \sum_{n=0}^{N(t)} \alpha_n(t) e^{-j\emptyset_n(t)}\, \delta\big(\tau - \tau_n(t)\big) \qquad (A.9)$$

where *n* denotes the path number, *N(t)* is the total number of paths at time *t*, α_n is the path gain, and $\emptyset_n$ the path phase. The latter is expressed as $\emptyset_n(t) = 2\pi f_c \tau_n(t) - \emptyset_{Dn}$, where $\emptyset_{Dn}$ is the Doppler phase shift due to the Doppler frequency f_D. For time invariant channels the time index *t* is a fixed value, i.e., the variables in questions are not changing in time. From Eq. (A.4) and Eq. (A.9), the received signal with multipath can be modeled as:

$$r(t) = R\left\{\left[\sum_{n=0}^{N(t)} \alpha_n(t) e^{-j\emptyset_n(t)}\, u\big(\tau - \tau_n(t)\big)\right] e^{j2\pi f_c t}\right\} \qquad (A.10)$$

The amplitudes and phases of the added paths are random variables. The path gain α can be modeled as a Gaussian random process that depends on the path loss and shadowing of each added path. The path phase $\emptyset$ is also a Gaussian random process (independent of α) that depends on the path delays and the Doppler shift. The I and Q parts of the received signal are [2]:

$$r_I(t) = \sum_{n=1}^{N(t)} \alpha_n(t) \cos \emptyset_n(t) \qquad (A.11)$$

$$r_Q(t) = \sum_{n=1}^{N(t)} \alpha_n(t) \sin \emptyset_n(t) \qquad (A.12)$$

Since α and $\emptyset$ are independent Gaussian processes, the I and Q parts of the received signal are said to be jointly Gaussian. The envelope of the received signal given by Eq. (A.13) is Rayleigh distributed: it can be modeled as a Rayleigh process as per Eq.(A.14), where σ^2 is the variance of both r_I and r_Q [2].

$$z(t) = |r(t)| = \sqrt{r_I^2(t) + r_Q^2(t)} \qquad (A.13)$$

$$p_Z(z) = \frac{z}{\sigma^2} exp[-z^2/P_r] = \frac{z}{\sigma^2} exp[-z^2/(2\sigma^2)], \quad z \geq 0, \qquad (A.14)$$

A Rayleigh distributed envelope is an accurate model when all paths exhibit delays and gains changing in a uniform fashion. The phase is uniformly distributed in the range (-π,π) in this case. This holds true when no line of sight (LOS) path exists. If a LOS path is present, it would exhibit higher power and less delay, yielding a different mean than other paths. For this reason, the envelope of a multipath signal with LOS is more accurately modeled as a Rician process as per Eq.(A.1.5).

$$p_Z(z) = \frac{z}{\sigma^2} exp\left[\frac{-(z^2 + s^2)}{2\sigma^2}\right] I_0(\frac{zs}{\sigma^2}), \quad z \geq 0, \qquad (A.15)$$

where s^2 represents the power of the LOS path and $2\sigma^2$ is the average power of the non-LOS paths. Other models, such as those based on the Nakagami distribution, can be found in [2].

Appendix B: IEEE 802.11a Primer

IEEE 802.11 (WiFi) is a family of standards for wireless Local Area Networks (WLAN). There are several versions of this standard: 802.11b, 802.11g, 802.11a, and the newly released 802.11n. This thesis focuses on the OFDM-based 802.11a standard that is well-suited for office indoor propagation environment. It uses the 5GHz band with the bandwidth of 20 MHz, requires 64 samples per OFDM symbol (64-tap FFT/IFFT), and achieves the maximum data rate of 54Mbits/s. Table B-1 specifies the key design parameters for an IEEE 802.11a-compliant system, while Table B-2 specifies a range of supported data rates, based on the chosen coding and modulation schemes [6].

Table B-1: 802.11a Specifications [6]

Parameter	Value
N_{SD}: Number of data subcarriers	48
N_{SP}: Number of pilot subcarriers	4
N_{ST}: Number of subcarriers, total	52 ($N_{SD} + N_{SP}$)
Δ_F: Subcarrier frequency spacing	0.3125 MHz (=20 MHz/64)
T_{FFT}: IFFT/FFT period	3.2 μs ($1/\Delta_F$)
$T_{PREAMBLE}$: PLCP preamble duration	16 μs ($T_{SHORT} + T_{LONG}$)
T_{SIGNAL}: Duration of the SIGNAL BPSK-OFDM symbol	4.0 μs ($T_{GI} + T_{FFT}$)
T_{GI}: GI duration	0.8 μs ($T_{FFT}/4$)
T_{GI2}: Training symbol GI duration	1.6 μs ($T_{FFT}/2$)
T_{SYM}: Symbol interval	4 μs ($T_{GI} + T_{FFT}$)
T_{SHORT}: Short training sequence duration	8 μs ($10 \times T_{FFT}/4$)
T_{LONG}: Long training sequence duration	8 μs ($T_{GI2} + 2 \times T_{FFT}$)

Table B-2: 802.11a Data Rates [6]

Data rate (Mbits/s)	Modulation	Coding rate (R)	Coded bits per subcarrier (N_{BPSC})	Coded bits per OFDM symbol (N_{CBPS})	Data bits per OFDM symbol (N_{DBPS})
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	3/4	6	288	216

B.1 Frame Format

The 802.11a physical layer (PHY) is divided into two parts; Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD) system. PLCP simplifies the interface between the Medium Access Control (MAC) layer and the PMD. PLCP maps PHY Service Data Units (PSDU) into appropriate frame format and then encodes the frame into OFDM symbols for transmission. PMD deals with handling the transmission of the OFDM symbols over the 5GHz channel [6].

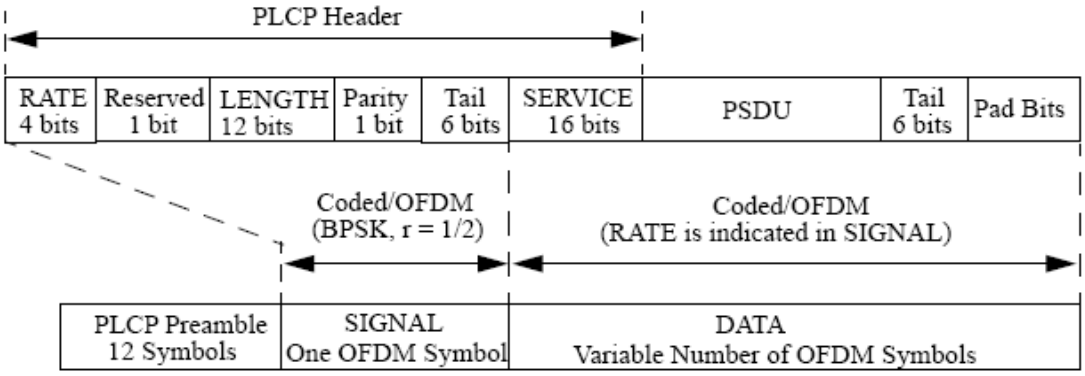


Figure B.1: 802.11a PHY [6]

Each PLCP frame has three fields: the preamble, SIGNAL, and DATA. The preamble consists of a known repeated sequence used for synchronization and channel estimation tasks at the receiver. More information about the Preamble is shown in section 3.1. The contents of the SIGNAL and DATA fields are discussed next.

B.1.1 SIGNAL Field

The SIGNAL field of the frame contains 23 bits which are encoded to form one OFDM symbol. The bit allocations are shown in Fig. B.2, forming the RATE, R (Reserved), LENGTH, P (Parity), and SIGNAL TAIL fields.

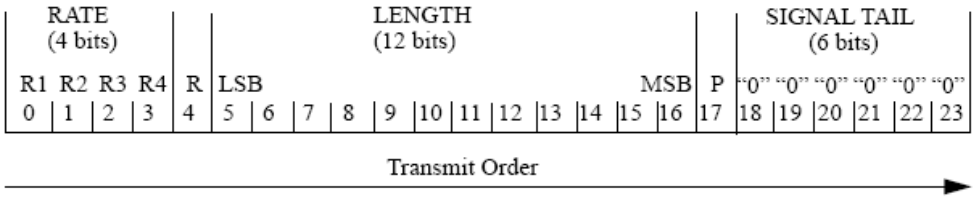


Figure B.2: SIGNAL Field [6]

The RATE field contains 4 bits R1-R4 identifying the data rate to be used for transmission. Based on the RATE information, other parameters – such as the coding rate, number of data bits per OFDM symbol N_{DBPS}, number of coded bits per OFDM symbol N_{CBPS}, and number of coded bits per subcarrier N_{BPSC} – can be calculated according to Table B-3 [6]:

Table B-2: RATE Bits [6]

Rate (Mbits/s)	R1–R4
6	1101
9	1111
12	0101
18	0111
24	1001
36	1011
48	0001
54	0011

The LENGTH field contains 12 bits which indicate the number of octets to be transmitted in the PSDU. The range of octets permitted by the standard is 1-4095. The Reserved bit (R) is set to zero by default. The Parity bit (P) is used to check the even parity of SIGNAL bits 0-16. SIGNAL TAIL contains 6 bits set to zero to restore convolution encoder to the zero state [6].

B.1.2 DATA Field

The DATA field carries the SERVICE, PSDU, Tail and Pad bits. The PSDU contains the payload to be sent over the channel. The SERVICE field contains 16 bits: the first 6 bits are set to zero to synchronize the descrambler at the receiver, and the other 10 bits are reserved for future use (set to zero by default). The 6 Tail bits are set to zero to reset the convolution encoder to the zero state, which helps improve the decoder's performance at the receiver [6].

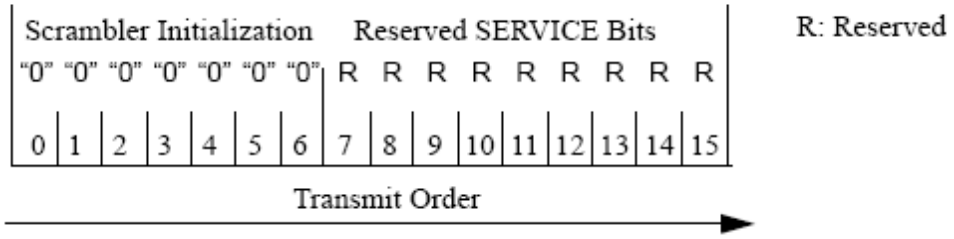


Figure B.3: SERVICE Field [6]

Depending on the amount of data to be transmitted in the PSDU, the total number of subcarriers needed may or may not amount to an integer number of OFDM symbols. For this reason, Pad bits are added to ensure that the number of OFDM symbols is an integer. The number of padding bits N_{PAD} needed depends on the data rate and is calculated as follows [6]:

$$N_{SYM} = Ceiling((16 + 8 \times LENGTH + 6)/N_{DBPS})$$

$$N_{DATA} = N_{SYM} \times N_{DBPS} \quad \text{(B.1)}$$

$$N_{PAD} = N_{DATA} - (16 + 8 \times LENGTH + 6)$$

B.2 OFDM Symbols

The PLCP layer prepares the OFDM symbols as shown in Fig. B.4. After preparing the PLCP frame, the frame is scrambled and then encoded at the 1/2 rate, followed by the puncturer that removes certain bits to provide other coding rates of 3/4 and 2/3. Next, the data is interleaved (to avoid burst errors) and modulated into either BPSK, or QPSK, or 16-QAM, or 64-QAM constellations, in accordance with the coding rate (see Table B-2). The output samples from the modulator are grouped into the sets of 48 (data subcarriers), and then 4 pilots and 12 zero subcarriers are added to form a 64-subcarrier OFDM symbol. The IFFT takes each group of 64 subcarriers and produces 64 time-domain samples per OFDM symbol. Each symbol is then appended with a cyclic prefix of 16 samples, so that an 80-sample time interval hosts one OFDM symbol. The window function is applied next, and finally, the preamble is added.

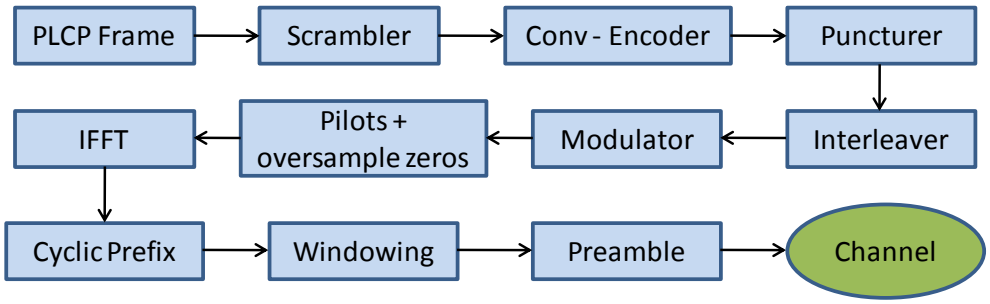


Figure B.4: OFDM Symbol Construction in PHY

B.2.1 Scrambler

When transmitting data over a communication channel it is desirable not to have a long stream of all 1's or all 0's. The scrambler helps avoid this scenario. The 802.11a standard prescribes a length-127 frame synchronous scrambler with generator polynomial $S(x) = x^7 + x^4 + 1$, shown in Fig. B.5 [6].

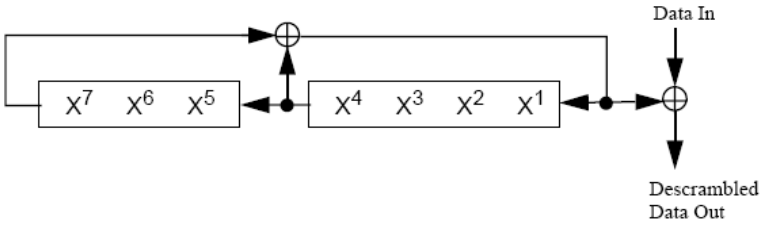


Figure B.5: Scrambler [6]

It is important to note that (1) the SIGNAL field is not scrambled at all, (2) the scrambled tail bits in the DATA field are overwritten with six zeros to return the convolution encoder to the zero state, and (3) the receiver uses the same scrambler design for descrambling [6].

B.2.2 Convolutional Encoder

For forward error correction coding, 802.11a prescribes a rate-1/2 convolutional encoder that uses the standard $g_0 = 133_8$ and $g_1 = 171_8$ generator polynomials. Fig. B.6 shows the encoder in question, where bit A is produced before bit B in the output sequence [6].

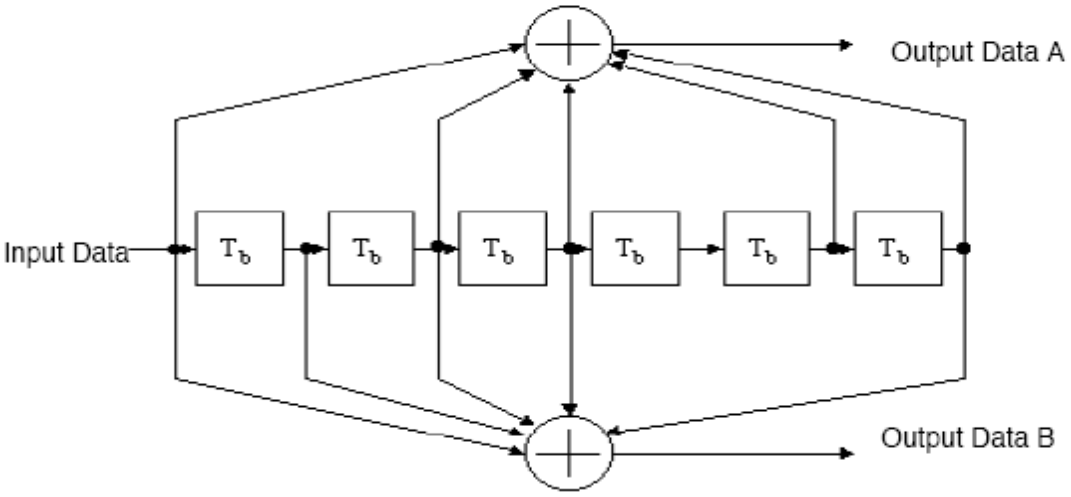


Figure B.6: Rate 1/2 Convolutional Encoder [6]

B.2.3 Puncturer

To provide higher coding rates, the puncturer omits bits in a certain pattern at the transmitter. This procedure reduces the number of transmitted bits therefore increasing the coding rate [6]. The puncture patterns for rate 2/3 and 3/4 are shown in Fig. B.7. For the 1/2-rate encoding, no puncturing is performed. It is important to note that the SIGNAL symbol always uses the 1/2-rate encoding. The 802.11a standard does not specify a required decoder; however, a Viterbi decoder is recommended [6].

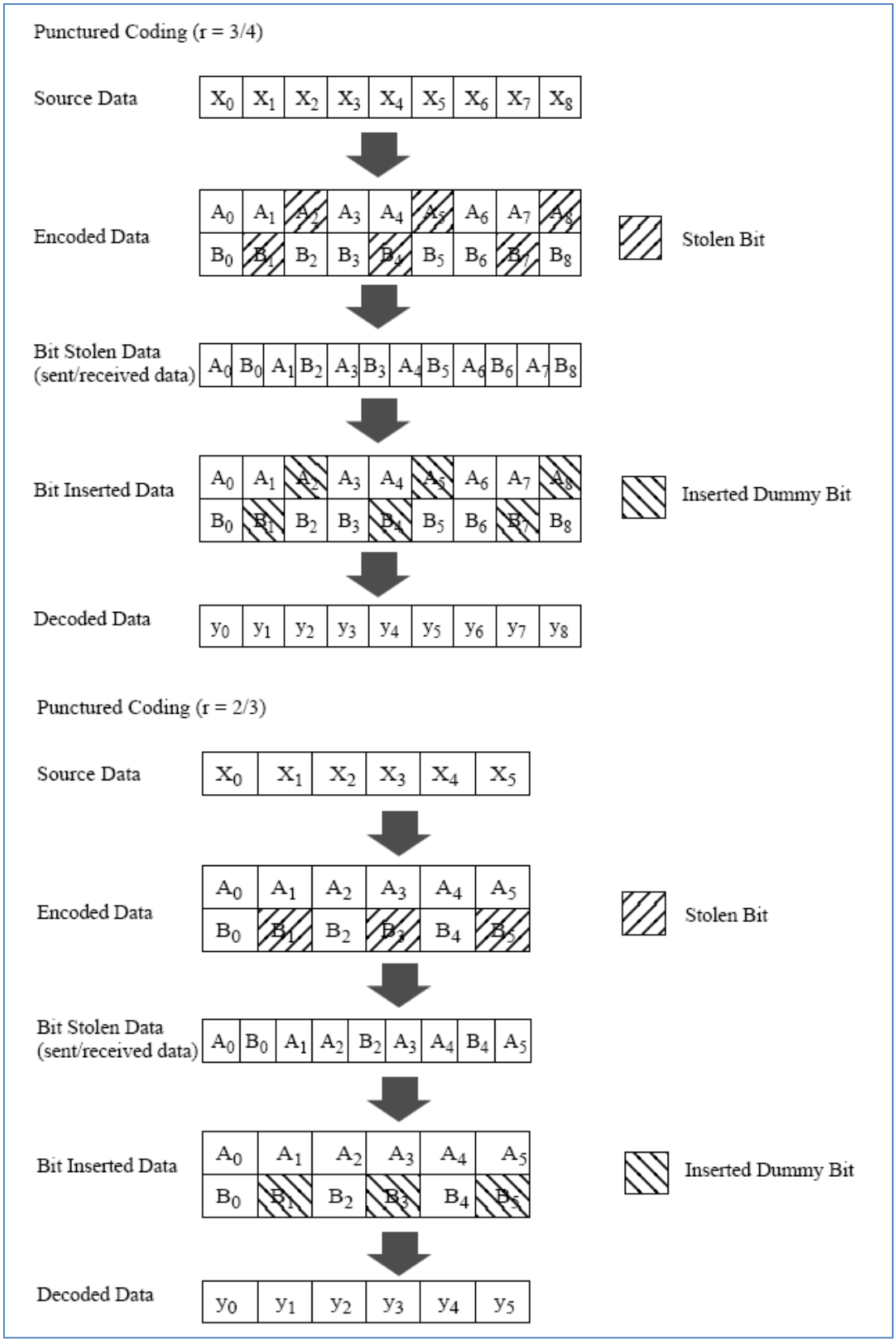


Figure B.7: Puncturing Patterns [6]

B.2.4 Interleaver

Interleaving is used to avoid burst errors. It is done by re-ordering the bits before transmission, so that encoded bits are placed in different locations. If a burst error were to occur, the restored original bit stream would have these errors scattered apart, which improves decoding performance. In 802.11a, the block size of the interleaver is equal to the number of bits per OFDM symbol, and the interleaving process is done by two permutations [6]. The first permutation is given in Eq. (B.2), which is followed by the second permutation per Eq. (B.3):

$$i = (N_{CBPS}/16)(k \bmod 16) + floor(k/16) \qquad k = 0,1,\dots,N_{CBPS}-1 \qquad (B.2)$$

$$j = s \times floor\left(\frac{i}{s}\right) + \left(i + N_{CBPS} - floor\left(16 \times \frac{i}{N_{CBPS}}\right)\right) mod\ s \quad i = 0,1,\dots,N_{CBPS}-1 \qquad (B.3)$$

$$\mathrm{s} = \max\left(\frac{N_{BPSC}}{2,1}\right) \qquad (B.4)$$

where *k* is the index of the coded bit before the first permutation, *i* is the index after the first permutation, and *j* is the index after the second permutation.

In the receiver, the de-interleaver performs the inverse operation, according to Eq. (B.5) for the first permutation and Eq. (B.6) for the second permutation:

$$i = s \times floor(j/s) + \left(j + N_{CBPS} - floor(16 \times j/N_{CBPS})\right) mod\ s \qquad j = 0,1,\dots,N_{CBPS}-1 \qquad (B.5)$$

$$k = 16 \times i - (N_{CBPS}-1) floor(16 \times i/N_{CBPS}) \qquad i = 0,1,\dots,N_{CBPS}-1 \qquad (B.6)$$

where *j* is the index of the original received bit before the first permutation, *i* is the index after the first permutation, and *k* is the index after the second permutation [6].

B.2.5 Modulator

After interleaving, the bits are modulated into the OFDM subcarriers. Depending on the modulation scheme, groups of 1, 2, 4 or 6 grey-coded bits are mapped to constellations as shown in Fig. B.8. Each constellation point is represented by a complex number representing in-phase *I* and quadrature *Q* components of the corresponding subcarrier. At the receiver, given the *I* and *Q* components of the received samples, the demodulator will reconstruct the modulated data bits accordingly. To insure same average power for all subcarriers for all modulation types, the *I* and *Q* values are normalized by the factor K_{MOD} given in Table B-4.

Table B-3: Normalization Factor [6]

Modulation	K_{MOD}
BPSK	1
QPSK	1/√2
16-QAM	1/√10
64-QAM	1/√42

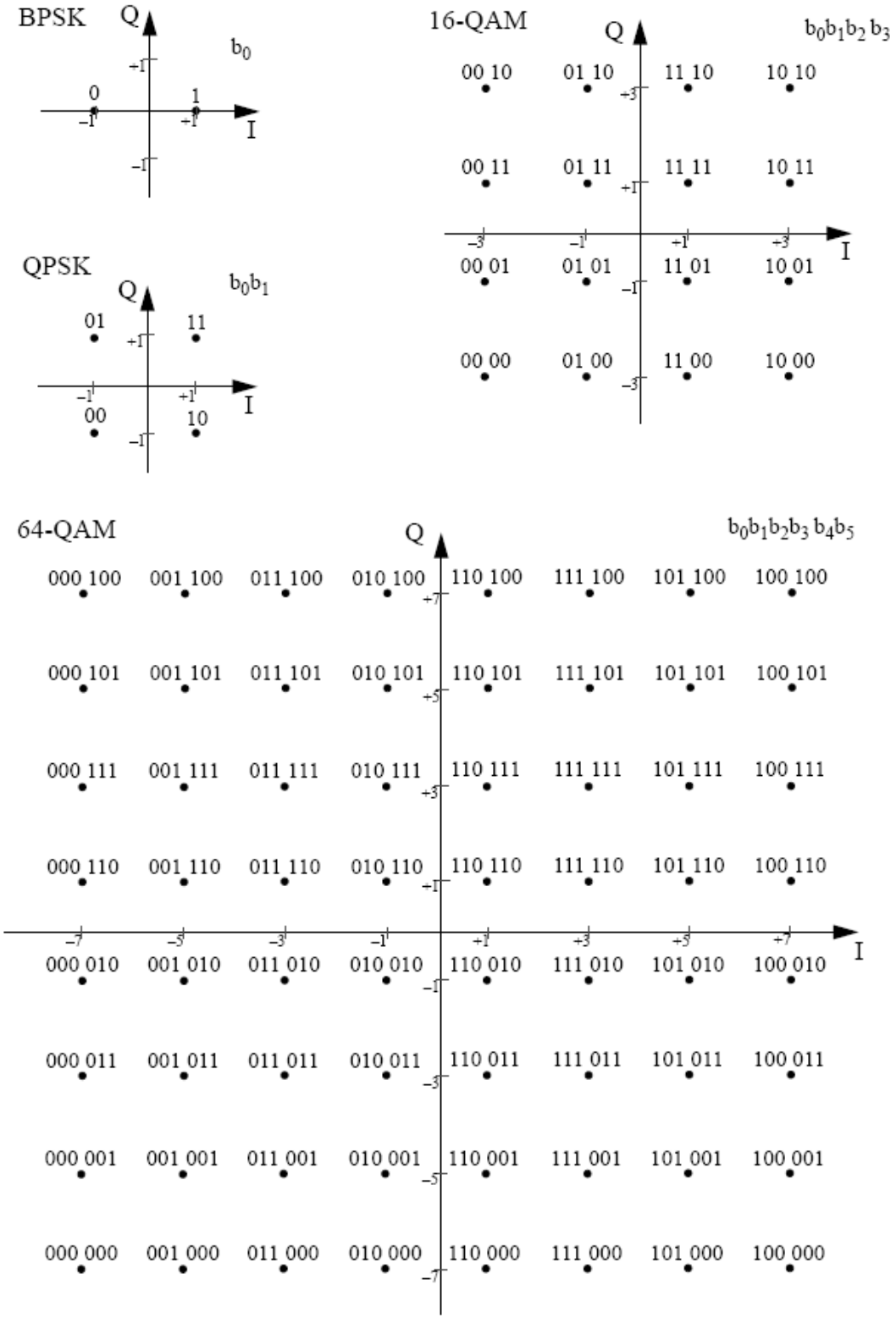


Figure B.8: Modulation Constellations [6]

B.2.6 Pilots and Zero Subcarriers

To help the receiver identify channel effects on the transmitted data, known pilot subcarriers are inserted in certain locations within each OFDM symbol. In 802.11a, four pilot subcarriers are added to the 48 data subcarriers produced by the modulator. Letting the resulting 52 subcarriers occupy frequency bins numbered from -26 to 26, the pilots are found in positions -21, -7, 7 and 21 [6]. These pilot subcarriers are always BPSK-modulated and their reference values are (1,1,1,-1). For each OFDM symbol, these reference values are multiplied by the PN sequence shown below [6]:

$$p_{0..126v} = \{1,1,1,1,\ -1,-1,-1,1,\ -1,-1,-1,-1,\ 1,1,-1,1,\ -1,-1,1,1,\ -1,1,1,-1,\ 1,1,1,1,\ 1,1,-1,1,$$
$$1,1,-1,1,\ 1,-1,-1,1,\ 1,1,-1,1,\ -1,-1,-1,1,\ -1,1,-1,-1,\ 1,-1,-1,1,\ 1,1,1,1,\ -1,-1,1,1,$$
$$-1,-1,1,-1,\ 1,-1,1,1,\ -1,-1,-1,1,\ 1,-1,-1,-1,\ -1,1,-1,-1,\ 1,-1,1,1,\ 1,1,-1,1,\ -1,1,-1,1,$$
$$-1,-1,-1,-1,\ -1,1,-1,1,\ 1,-1,1,-1,\ 1,1,1,-1,\ -1,1,-1,-1,\ -1,1,1,1,\ -1,-1,-1,-1,\ -1,-1,-1\}$$

At position 0, a zero (Null) subcarrier is inserted. To complete the 64-tap FFT/IFFT window size, 11 more zero subcarriers are appended: 6 in positions -32, -31, …, -27 and 5 zero subcarriers in positions 27, 28, …, 31.

B.2.7 FFT/IFFT

The 802.11a-specified arrangement of the subcarriers at the IFFT input is shown in Fig. B.9. Subcarriers 1 to 26 are mapped to the same numbered inputs of IFFT; subcarriers -26 to -1 are mapped to inputs 38 to 63; the rest of the inputs are the zero (Null) subcarriers. At the receiver, the FFT reconstructs the subcarriers in the same order and then re-arranges them in their proper positions before demodulation.

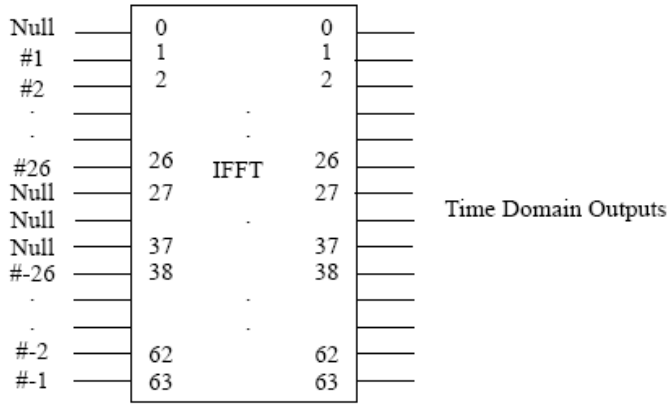


Figure B.9: 64-IFFT [6]

B.2.8 Cyclic Prefix and Windowing

In 802.11a, the cyclic prefix is taken to be 25% of the OFDM symbol. Hence, the last 16 samples of each OFDM symbol are copied to the back to form 80-sample 4-microsecond symbol interval. Note that the maximum delay spread must not exceed 0.8 microseconds to avoid ISI. The recommended windowing function is given by Eq. (B.7):

$$w_T[n] = w_T(nT_s) = \begin{Bmatrix} 1 & 1 \le n \le 79 \\ 0.5 & 0,80 \\ 0 & otherwise \end{Bmatrix} \qquad \text{(B.7)}$$

The first and last sample of adjacent OFDM symbols are then overlapped (added) with neighbouring OFDM symbols to provide smooth transitions.

Appendix C: Simulation Setup

The complete system, including data processing at both the transmitter and the receiver, was built from scratch in MATLAB, save for a few built-in functions that will be detailed as they are mentioned. For our system simulations two environments were prepared: floating-point and fixed-point. The floating-point environment was prepared using a straightforward programming of the equations describing each processing block of the transmitter, channel, and receiver. In the fixed-point environment (receiver only), the MATLAB fixed-point library was used to perform fixed-point arithmetic operations and the CORDIC algorithm (see Appendix C) was used to calculate trigonometric functions.

C.1 Transceiver Model

Each block in the transmitter shown in Fig. B.4 was implemented in MATLAB to perform the pre-channel data processing. The Convolutional Encoder and IFFT blocks used built-in MATLAB functions, while all the other blocks were coded (as specified by the 802.11a standard) specifically for this work. Each block in the receiver shown in Fig. 4.1 was implemented in MATLAB to perform the post-channel data processing. The FFT and Viterbi Decoder used built-in MATLAB functions, while all the other blocks (including CORDICs) were coded specifically for this work. We used hard-decision Viterbi decoding with the trace-back length of 96. Fig. C.1 shows an example illustrating the effect of having different trace-back lengths, where 96 is identified as the best setting, with a slight improvement over 64.

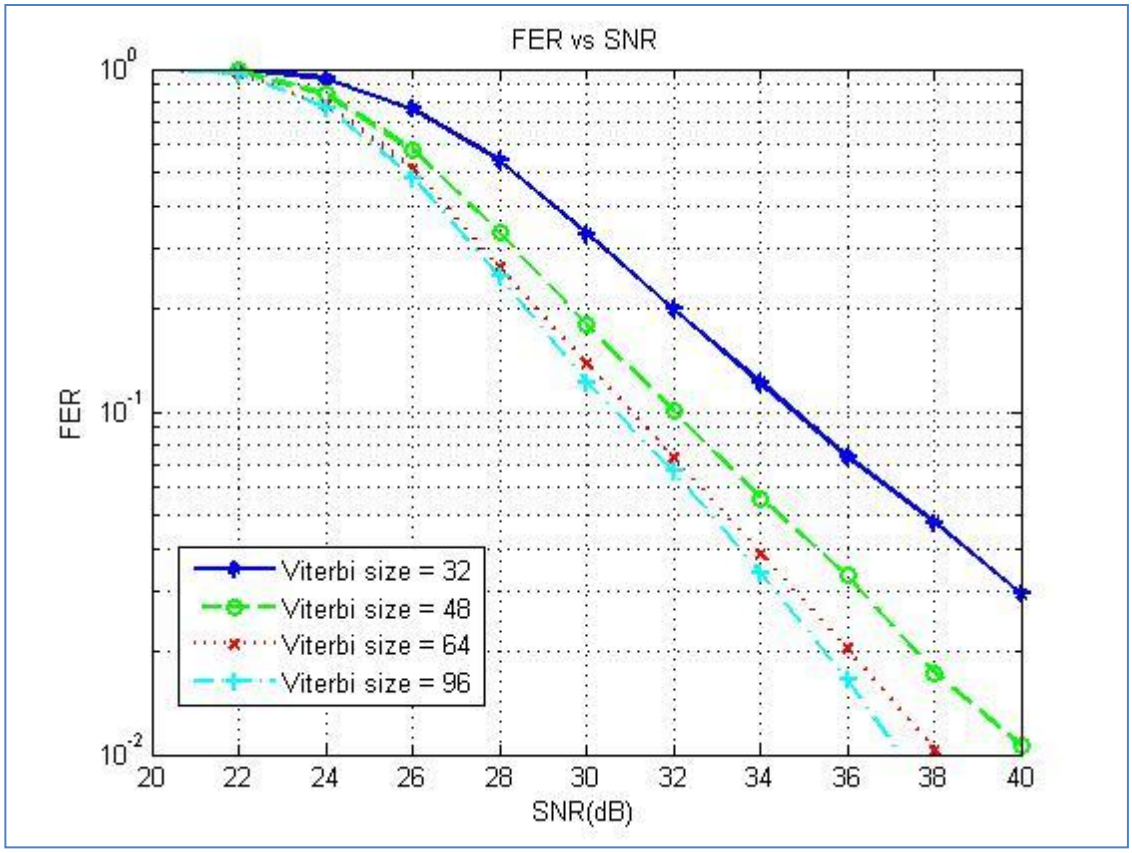


Figure C.1: Effect of Viterbi Trace Back Length

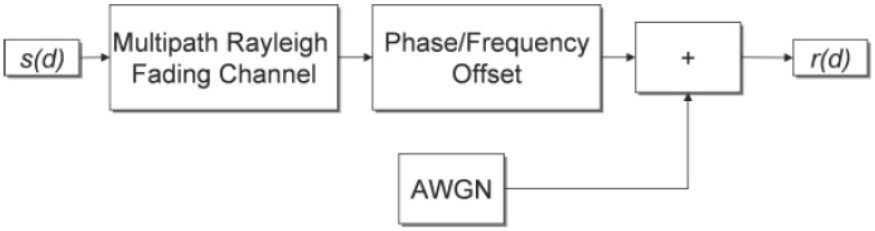


Figure C.2: Channel Model

C.2 Channel Model

The simulated channel model was coded to include three effects: the CFO, Rayleigh multipath fading and AWGN noise (see Fig. C.2). For the Rayleigh multipath , we used the built-in MATLAB function that requires four inputs: Doppler frequency, sampling frequency, path gains, and path delays. For the AWGN noise, we used the built-in MATLAB function as well, defining SNR relative to the measured input signal power. We kept our channel settings in line with those mentioned in the research literature. The indoor channel model that we simulated is from [49], as shown in table (B-1), which is widely referenced and used in numerous papers. Assuming a stationary environment (e.g., office WLAN), the Doppler frequency was set to zero. Our CFO was set to 50 KHz, which is also a typical setting. More details on our simulation methodology is provided in Chapter 6.

Table C-1: 50ns Delay Spread Multipath Channel Model (Indoor Office Environment)[49]

Path Number	Path Delay (ns)	Path Gain (dB)
1	0	0.0
2	10	-0.9
3	20	-1.7
4	30	-2.6
5	40	-3.5
6	50	-4.3
7	60	-5.2
8	70	-6.1
9	80	-6.9
10	90	-7.8
11	110	-4.7
12	140	-7.3
13	170	-9.9
14	200	-12.5
15	240	-13.7
16	290	-18.0
17	340	-22.4
18	390	-26.7

Appendix D: CORDIC

The receiver data processing involves angle calculation, vector rotation, and division operations. These operations are trivial to implement in hardware. In 1959, Volder [34] presented the Coordinate Rotation Digital Computer (CORDIC) algorithm that performs vector rotations as a series of shift-and-add operations. The algorithm was later extended to perform linear (Cartesian) to circular (polar) coordinate conversions, division, as well as other operations [35].

D.1 CORDIC Operation

Depending on its control configuration, CORDIC can work in one of three coordinate systems: Linear, Circular, or Hyperbolic. CORDIC can also be set to two modes of operation: Vectoring Mode (VM) and Rotation Mode (RM). A good survey of CORDIC configurations is presented in [35]. For an OFDM-based receiver the hyperbolic coordinate system is not needed [36].

The CORDIC inputs are three data vectors X, Y, Z and control signals specifying the mode (VM or RM) and the coordinate system (Linear or Circular). The general form of the CORDIC algorithm [36] is given by:

$$\begin{aligned} X_{i+1} &= X_i - m d_i 2^{-i} Y_i \\ Y_{i+1} &= Y_i + m d_i 2^{-i} X_i \\ Z_{i+1} &= Z_i - d_i \alpha_i \\ d_i &= \begin{cases} sign(Z_i), & for\ RM \\ -sign(Y_i), & for\ VM \end{cases} \end{aligned} \qquad (D.1)$$

where *i* is the iteration number, *d* is defined depending on the mode (RM or VM), while *m* and *alpha* are defined depending on the coordinate system:

Table D-1: CORDIC Control Parameter Values

Coordinates	m	α
Linear	0	2^{-i}
Circular	1	$tan^{-1}(2^{-i})$

For CORDIC Linear functions, the output is limited to ±2. For CORDIC Circular functions, the output is limited to ±π/2, which can be extended to ±π, the following pre-iteration is added [36]:

$$\begin{aligned} X_0 &= -d_{-1}Y_{-1} \\ Y_0 &= d_{-1}X_{-1} \\ Z_0 &= Z_{-1} - d_{-1}\alpha_{-1}, \quad \alpha_{-1} = \frac{\pi}{2} \\ d_0 &= \begin{cases} sign(Z_{-1}), & for\ RM \\ -sign(Y_{-1}), & for\ VM \end{cases} \end{aligned} \qquad (D.2)$$

When using circular coordinates, each CORDIC iteration produces a gain A_n as shown in Eq. (C.3). As the number of iterations approaches infinity, A_n approaches 1.647. To compensate for this, the CORDIC result is multiplied by 1/1.647 = 0.607 to produce the final answer.

$$A_n = \prod_n \sqrt{1 + 2^{-2i}} \qquad (D.3)$$

Note that the step size α and the scaling factor A_n are constants that depend only on the number of iterations. It is suggested that these values be pre-calculated and stored in a look-up table, rather than computed when needed [36].

D.2 CORDIC Challenges

Although the CORDIC algorithm can handle complex operations, it entails a considerable hardware footprint and a delay overhead. In an analysis performed in [37], it was shown that in an OFDM system, the CORDIC operations make up the most part of the receiver delay, due to the large number of iterations performed on each data sample.

One approach to reduce timing delays is suggested in [37]: it relies on several CORDICs working in parallel. It was found in [37] that the use of seven CORDIC processors in parallel resulted in optimal performance and also helped reduce the required clock frequency.

Another approach is to use a variable step size for different iterations, giving rise to the adaptive CORDIC algorithm [38]. This algorithm adaptively reduces the number of iterations needed by using only those iteration steps that approach the result without alternating around it. Although this study shows around 50% reduction in number of iterations, faster cycles are needed to make future predictions of the required iterations.

Another CORDIC-related problem that needs to be addressed is the required hardware footprint, e.g., in terms of FPGA resource usage. In [36] a compact hardware design of a variable CORDIC is constructed to perform the operations required for OFDM (angle calculation, vector rotation, and division). The proposed design gives a savings of 28% in terms of resource usage and a throughput increase of 64%.