Practical Implementation Considerations for Spectrally Agile Waveforms in Cognitive Radio

by

Kevin M. Bobrowski

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical and Computer Engineering
by

September 4 2009

APPROVED:

Professor Alexander M. Wyglinski, Advisor

Professor Xinming Huang

Professor Donald Richard Brown

Abstract

As the demand for bandwidth increases, the inefficient use of the spectrum becomes more apparent and limiting. Currently, secondary (unlicensed) users can not use sparsely occupied portions of radio spectrum that are not allocated to them. In prior research, a variant of Orthogonal Frequency Division Multiplex (OFDM) called Non-Contiguous OFDM (NC-OFDM) was found to be a suitable transmission technique for enabling Dynamic Spectrum Access, which allows for multiple secondary users to share the spectrum. This thesis presents an algorithm for the synchronization of NC-OFDM. Moreover, a hardware architecture is proposed for the synchronization, and a pruned FFT/IFFT core is designed.

At present, there has been minimal research into synchronization for NC-OFDM systems. As with any modulation scheme, synchronization is an important part for receiving the transmission successfully. The current synchronization scheme is simulated in variety of wireless channels to show that it can successfully communicate in the tested channels.

Additionally a hardware architecture is laid out for the practical implementation of the synchronization algorithm. Since NC-OFDM does not use all of the carriers for transmission, the FFT and IFFT can have their computations reduced. Since the FFT and IFFT are important parts to the receiver and the transmitter, a pruned FFT/IFFT in hardware makes the most sense to be able to reduced the computation. The number of butterfly computations is reduced at the expense of a large increase in resource usage.

Acknowledgements

I would like to thank my Mom for being with me from the very being of my education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The radio spectrum is a finite resource that is used by many. Application such as: Radar, Internet access, satellite and terrestrial television broadcasting, emergency services, and cellular telephone. The demand for more throughput for more users is always increasing while the usable spectrum is fixed. With the current system of applications receiving exclusive access to portions of spectrum, this leads to some portions being very congested and other used sparsely. This is an inefficient usage of the radio spectrum. These unused portions of spectrum are called *white spaces*. These white spaces are often already licensed to a specific application, but are used sparingly for actual transmission. Figure 1.1 shows a small portion of the radio spectrum.

One approach for resolving the spectrum problem is to allow for unlicensed (secondary) wireless access of licensed (primary) frequency bands. Currently, radio spectrum is divided into blocks in which only the primary entity is allowed to transmit in that particular frequency range. To enable greater access to the wireless spectrum, some researchers have proposed a *spectrum pooling* approach [34]. Using this method in conjunction with *orthogonal frequency division multiplexing* (OFDM), a transmitter is capable of using portions of the spectrum that are unoccupied, thus allowing for greater spectrally efficient utilization [33]. Spectrum pooling and *dynamic spectrum access* (DSA) are wireless concepts that can

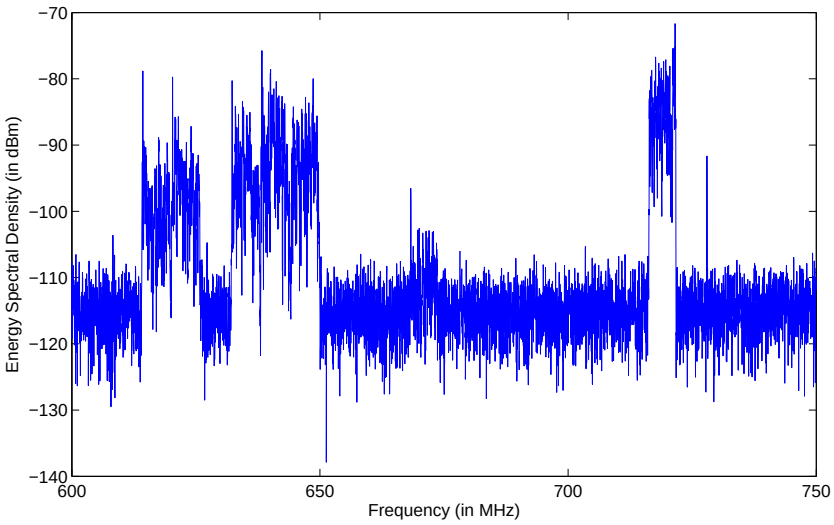
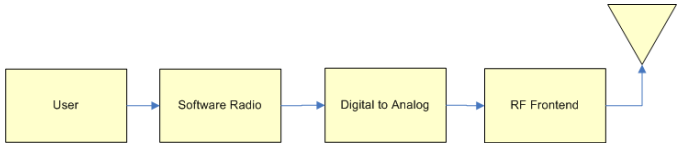


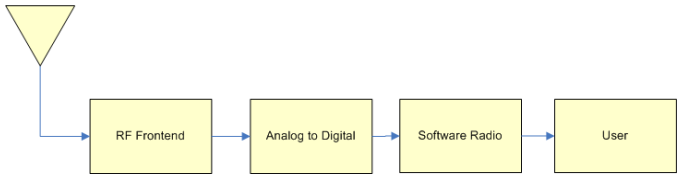
Figure 1.1: Portion of radio spectrum showing whitespaces. Location and time: Boston, 6/30/2009; Frequency Range: 600 MHz to 750 MHz; Resolution: 20 kHz

be enabled by *cognitive radios* [18]. Dynamic spectrum access involves radios that can dynamically change how much frequency bandwidth is used and where it is located within a given band. Spectrum pooling allows for an increase in spectrum usage efficiency by sharing a given frequency band by multiple radios. A cognitive radio is a radio system is aware of spectrum occupancy and current data bandwidth requirements, able to learn from past history, and can change its current operating parameters.

Software-defined radio (SDR) has been proposed by Mitola to have radios that can be reprogrammed to received and transmit on different wireless standards. The SDR is comprised of a software controlled processor, digital-to-analog converters (DACs), and analog-to-digital converters (ADCs). Figure 1.2 shows a simplified transmitter and receiver for a software defined radio.



(a) SDR Transmitter



(b) SDR Receiver

Figure 1.2: Example structures of a SDR receiver and transmitter

This type of radio has all processing done in software. The radio frequency front-end is a separate part that can also be controlled by the software [9]. One view of SDR is illustrated in Figure 1.2. This view of a SDR shows that the software handles all of the processing work to successfully send and receive a signal. The only specialized hardware is the RF frontend. In non-SDRs, most of the hardware is specialized to doing one specific task.

Although each radio communication standard is unique, many employ the same basic concepts and building blocks such as synchronization methods, channel equalization, and FFT/IFFT modules. Additionally, some of the building blocks, such as the FFT and IFFT, can be done more efficiently and faster in dedicated hardware. When a SDR starts to incorporate specialized configurable hardware, it is no longer a pure SDR. One proposed architecture was presented by Chamberlain for a SDR employ FPGAs [3]. Iacomacci proposed an architecture for a SDR platform using reconfigurable FPGAs [8].

A cognitive radio is an extension to SDR that is capable of adapting to changing conditions. The cognitive radio operation is based on specified goals, such as maximizing battery life, minimizing radio spectrum usage, bandwidth and power needed for transmission. The cognitive radio has some sense of autonomy and is capable of learning from past events. One of the performance goals for cognitive radio is to better utilize the radio spectrum [18].

Throughout the frequency spectrum, there exists several bands that are unoccupied in both frequency and time. Cognitive radio seeks to increase the efficiency of the wireless

usage by determining the locations of unused spectrum and transmitting in those bands. OFDM is well suited for such a task since it is comprised of multiple sub-carriers with each sub-carrier capable of being individually set to zero amplitude in the vicinity of another signal, *i.e.* a null sub-carrier. This is important in a cognitive radio since the unoccupied spectrum will probably not be a single contiguous block, but potentially be composed of multiple disjoint blocks. This variation of OFDM is referred to as non-contiguous OFDM (NC-OFDM) [22]. Since some of the sub-carriers will be nulled, not all the computations in the FFT and IFFT will be needed since several of the additions and multiplications involve zero values. In these cases, the computations can be eliminated entirely, resulting in the reduction of additions and multiplies, thus reducing the time needed to compute the FFT or IFFT.

The purpose of NC-OFDM is to be able to select which frequency ranges are used to transmit as required for DSA. Fig. 1.3 shows a small portion of sub-carriers in a larger NC-OFDM based transmission. The only carriers that are are used are the ones that are deemed not to interfere with the primary licensed user for the given spectrum.

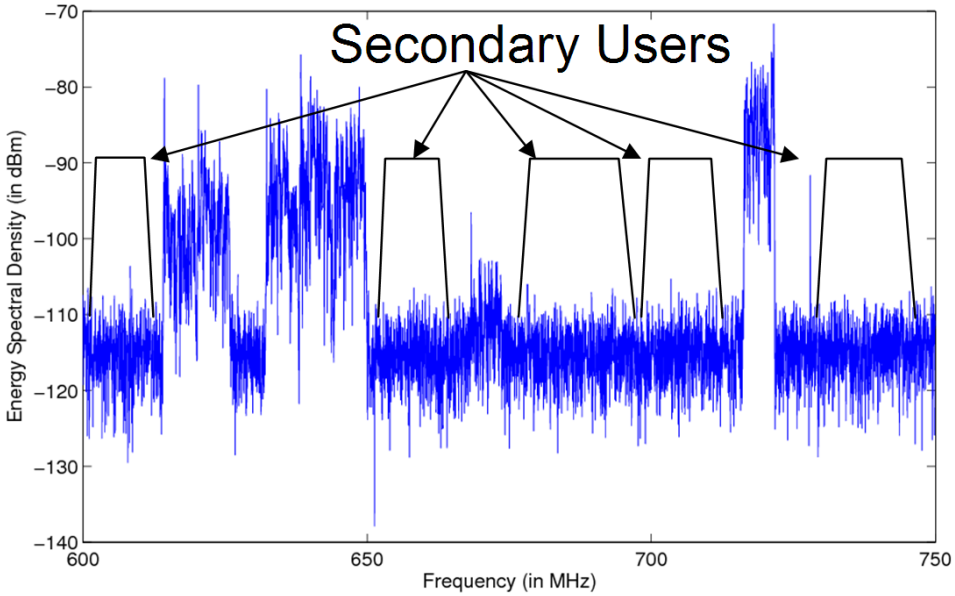
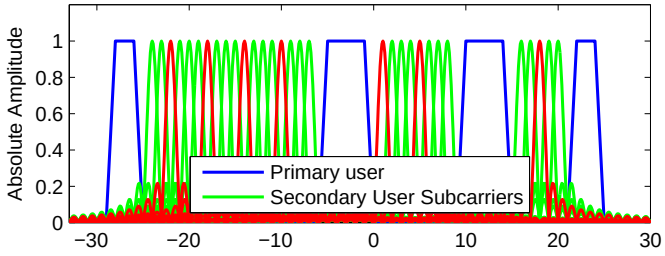


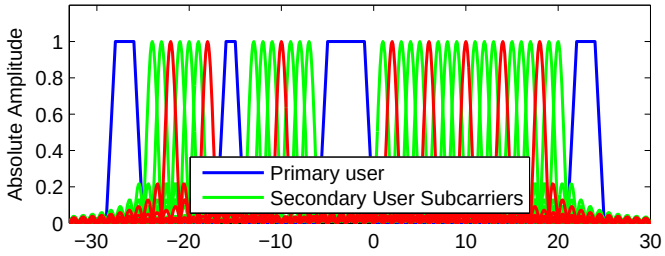
Figure 1.3: Primary users within a the spectrum of NC-OFDM transmission.

1.2 Current State-of-the-Art

At the University of Kansas, NC-OFDM was chosen over OFDM and NC-MC-CDMA for filling in the whitespaces in the radio spectrum. The biggest issue with NC-OFDM, was that no synchronization means existed. Since that decision, the issue of synchronization for NC-OFDM has not received much research attention. Synchronization can be broken into several parts, carriers frequency offset, sampling frequency offset, and symbol timing [28]. With respect to synchronization for NC-OFDM receiving, the problem of obtaining precise acquisition of a NC-OFDM data symbol in the presence of several other users was studied in [1]. This thesis does not address carrier frequency offset, sampling frequency offset, or pilot carrier arrangements within a data symbol. Since NC-OFDM can operate in time varying spectral availability channel, the carrier type must be able to change over time. Figure 1.4 shows how the carrier arrangement can change over time for NC-OFDM.



(a) Time instance T



(b) Time instance T+δ t

Figure 1.4: Varying pilot carrier location for NC-OFDM over time.

The sub-carriers of amplitude 1 are data carriers and sub-carriers of amplitude 1.5 are the pilot carriers. It can clearly seen that the pilot carriers used are different at the two

moments in time.

At the University of Kansas, it was noted that the FFT and the IFFT efficiency could be improved for a NC-OFDM transmitter and receiver [27]. In OFDM, the FFT and IFFT are used to efficiently demodulate and modulate the sub-carriers. For NC-OFDM, deactivated carriers can be treated as null carriers at both the transmitter and receiver and cancellation carriers can also be treated as null carriers in data symbols at the receiver. The null carriers have no effect on the output of IFFT or FFT and the cancellation carrier have no effect on the demodulated carriers from the FFT. In these cases, the FFT and IFFT can be pruned to remove the calculations that calculate information to these types of sub-carriers. Since FFT is tightly coupled with the synchronization of OFDM, and also NC-OFDM, pruning must be done in hardware [27]. A pruned FFT in hardware has been avoided mainly due to apparent irregularity of calculations and the apparent need to conditional statements, both of which are not conducive for direct hardware implementation. A group in the Netherlands has designed a hardware core for transform decomposition. Transform decomposition is similar to FFT pruning, with respect to removing calculations, it breaks a larger FFT is broken down into a series of smaller ones [32].

Since NC-OFDM was chosen as a possible candidate for enabling DSA for Cognitive radios, a flexible NC-OFDM transmitter and receiver are needed. As of yet, there have been no published papers on implementing a complete transceiver for NC-OFDM. Acharya *et al* looked at the data symbol timing acquisition, but not address pilot carriers or carrier frequency offset estimation. There has been some research into a flexible OFDM transmitter though by Interuniversity Micro Electronics Center (IMEC), a European research center. In particular, IMEC has proposed an architecture of a OFDM transmitter that can have 32-1024 carriers, variable cyclic prefix length, and multiple constellation mappings. The complete design has not been implemented in hardware, only the FFT core has been, which was not test at the time of publication [12]. Since OFDM is a now commonly used wireless transmission algorithm, most other research groups have implemented OFDM as part of a wireless standard. A few of the notable groups are the ITTC of Kansas University with their KUAR [17]; NICT of Japan and their Cognitive Radio Prototype [6]; Virginia Tech [15].

1.3 Research Objectives

The primary objective of this thesis is to create an algorithm and hardware architecture that enables cognitive radios to adequately fill in the unoccupied spectral gaps. At present, cognitive radios and other wireless platforms only transmit in contiguous blocks of unused spectrum.On the Other hand; this thesis investigates the feasibility of filling in multiple blocks that are separated by primary users. Specifically, this thesis focuses on the following:

- To develop a synchronization method for non-contiguous operation of OFDM in a wireless channel with primary users dispersed between the contiguous blocks of carriers. At present, OFDM wireless transmission schemes only transmit on a single, contiguous block of carriers that does not change with time. The synchronization algorithm is capable of allow carrier types to change over time and operate with additional signals dispersed between individual sub-blocks of sub-carriers.

- The Fast Fourier Transform (FFT) and its inverse (IFFT) are important functions in OFDM transceivers. In the case of NC-OFDM, the transceiver can benefit from pruning since not all carriers carry meaningful data. Since the FFT is closely tied to the operation of OFDM transceivers, pruning must be done in hardware. At present, no definitive set of approaches for FFT pruning in hardware exist. A pruned FFT and IFFT core is designed and integrated into a NC-OFDM transmitter.

- The hardware for a NC-OFDM is created to enable use of the NC-OFDM synchronization algorithm and the pruned FFT/IFFT cores. At present, OFDM transceivers do not operate dis-contiguously or utilize pruning in hardware. The transceiver will enable Cognitive Radios to better utilize spectrum that is sparsely occupied by existing users.

1.4 Thesis Contributions

This thesis presents novel work on the synchronization of NC-OFDM, a pruned FFT/IFFT core, and a hardware architecture for NC-OFDM. These three points form a realizable ar-

chitecture for an efficient NC-OFDM transmitter and receiver design. This thesis possesses the following three novel inter-related contributions:

- NC-OFDM synchronization algorithm that allows for non-contiguous operation operation of OFDM with dynamically changing carriers over time. This allows for a cognitive radio to take advantage of several smaller blocks of unused spectrum simultaneously. Since multiple blocks of spectrum can be used simultaneously, throughput is increased.

- A pruned FFT/IFFT core was created to reduce computational load and execution time of a FFT/IFFT in hardware. In the case of NC-OFDM, the FFT/IFFT efficiency can be improved, while still being utilized as part of a transmitter or receiver. The pruned IFFT/FFT unit is able to bring the benefits of pruning to hardware based FFT/IFFT cores used in OFDM based transceivers.

- A NC-OFDM transmitter and receiver is designed in hardware to allow for cognitive radios to take advantage of the proposed NC-OFDM synchronization algorithm. This allows NC-OFDM to be more efficiently used in terms of speed and power consumption, as a general purpose processor is not used.

1.5 Thesis Outline

Synchronization is highly important in any OFDM, along with NC-OFDM, transmission. Any carrier frequency offset will degrade the performance of the communication system. A misalignment of the FFT window on the data symbols may also cause a degradation in system performance. In order to better understand the NC-OFDM receiver, a background in multi-carrier transmission fundamentals is presented in Chapter 2. From this, the NC-OFDM receiver can be presented in Chapter 3. A NC-OFDM transmitter and receiver is then designed in hardware on a Virtex-5 SX Field Gate Programmable Array (FPGA) in Chapter 4. The receiver is then tested in a simulated MATLAB environment in Chapter 5.

Chapter 2

Communication System Architecture

This thesis deals with multi-carrier modulation, particularly NC-OFDM [26]. Other multi carrier modulation schemes are briefly presented, such as frequency division multiplexing [7] and MC-CDMA [37]. NC-OFDM was chosen to be used as a modulation scheme to enable spectrum pooling because for better BER performance over NC-MC-CDMA and ability to operate in a non-contiguously, which OFDM can not. from [26]. Since this thesis proposes a portion of a physical layer for an NC-OFDM, an in-depth overview of OFDM is presented, particularly synchronization. The developed algorithm is later designed for implementation on a Virtex 5 SX FPGA. In this chapter a short overview of *field programmable gate arrays* (FPGAs) and the key features of the Virtex 5 SX is explained. FPGAs are ideal for SDR hardware prototyping and usage since they are reconfigurable.

2.0.1 Enabling DSA with NC-OFDM

At the University of Kansas, NC-OFDM, OFDM, and NC-MC-CDMA were compared with each other based on synchronization, spectrum agility, throughput, overhead, error robustness, and Peak-to-Average-Power Ratio (PAPR). These three modulation scheme were chosen for enabling cognitive radios to better fit the non-contiguous blocks of unused spectrum. A summary of these three modulation schemes is shown in Table 2.0.1.

Table 2.1: Performance Using Hard Decision Detection

Characteristics	MC-CDMA	OFDM	NC-OFDM
Synchronization		X	
Spectrum Agility	X		X
Throughput	X		X
Overhead		X	
Error Robustness		N/A	X
PAPR	X		

As can be seen from the table, MC-CDMA and NC-OFDM are better suited than OFDM for Cognitive Radios. NC-OFDM was chosen over MC-CDMA primarily for its error robustness as carriers are deactivated. In the case of MC-CDMA, information is spread across multiple carriers, as carriers are deactivated, information can be lost since multiple carrier share parts of the same data.

NC-OFDM also possesses several problems related to it, which are inherited from being an OFDM variant [25] . These problems are a high PAPR and relatively high side-lobes [38][24]. These problems reduce the maximum transmit power before non-linear distortion occurs and the usable spectrum in a given unoccupied piece of spectrum. Furthermore, a problem specific to NC-OFDM is that no known synchronization techniques existed at the time. This is because pilot carriers may exist in part of the spectrum that cannot be used. Synchronization is a very important part of any communication system, where improper synchronization can lead to an increased bit error rate (BER).

2.1 Programmable Logic

Programmable logic allows for the fast implementation of digital logic circuits. Before programmable logic was available, digital designs could only be implemented using discrete logic chips or *application specific integrated circuits* (ASICs). Both of these implementation methods would take much time to produce a working implementation, particularly ASICs. Early programmable logic chips were the *programmable logic array* (PLA) and the *programmable array logic* (PAL). PLAs and PALs have similar structures. The PLA consists of two arrays, a AND array and a OR array. Each array consists of intersecting wires that

can be shorted together through a fuse or transistor. The PAL is slightly simpler in design, possessing only a single AND array and a fixed OR function.

In order to improve the density of programmable digital logic circuit, several PALs or PLA can be built into a single chip and interconnected with a routing network. These devices are called *complex programmable logic devices* (CPLDs). CPLDs are typically able to be function upon startup, thus require no additional configuration memory. However, These have now been generally replaced by FPGA devices. FPGAs break away from the PLA and PAL basic building block and build upon configurable logic blocks interconnected by a routing network [40].

2.1.1 Field Programmable Gate Array Devices

A FPGA is a digital logic device that can be programmed by the user. Verilog and VHDL are two common hardware description languages that are used to program FPGAs. An FPGA consists primarily of a lattice of *configurable logic blocks* (CLB) that are interconnected with each other. The basic CLB consists of a *look up table* (LUT), D-flipflop, and a multiplexer. Figure 2.2 shows a simple FPGA structure with several CLBs and Figure 2.1 shows the internal structure of a CLB.

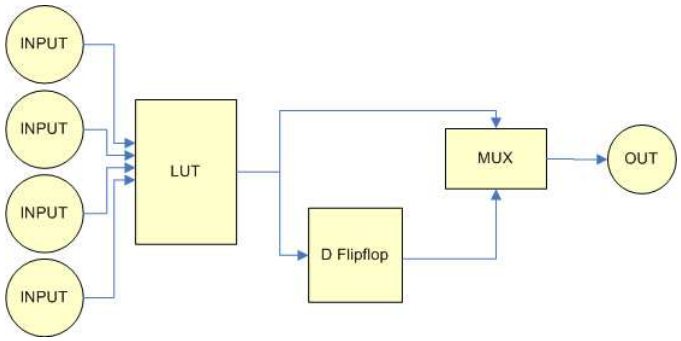


Figure 2.1: Configurable Logic Block.

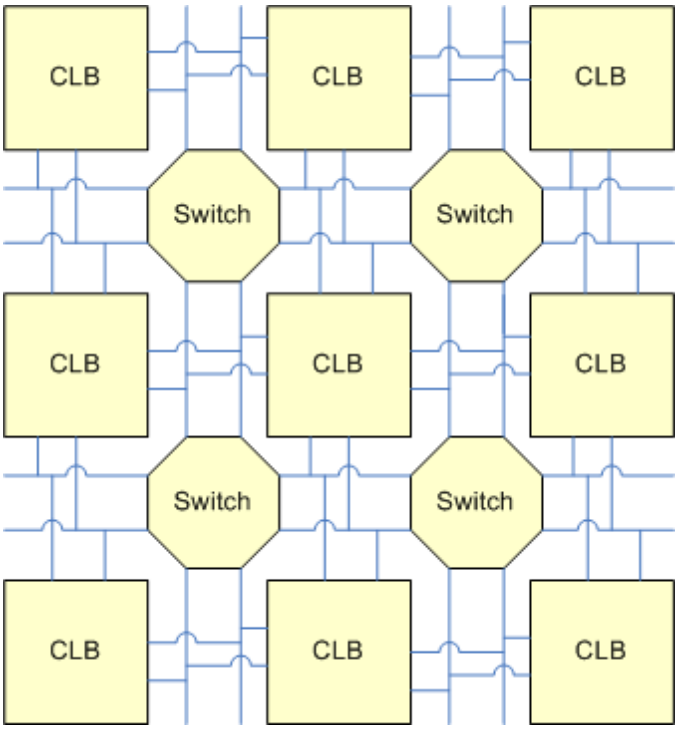


Figure 2.2: Small portion of networked CLBs in a FPGA.

The CLB is the basic building block of a simple FPGA. The CLB can be programmed by loading bits into the LUT and configuring the multiplexer to bypass the flip-flop to create a simple logic function. The routing fabric with the switches allow for taking the simple functions created by the CLBs to form a complex logic function such as an adder.

In digital circuit design, there are elements, such as wide multipliers, multiplexers, RAM, and adders, that are used throughout the design. These elements can be more efficiently, in speed and resource usage, as hard cores. In the more advanced FPGAs, processors, ethernet MACs, dedicated serial/deserial, PCIe MAC, and DSP slices, can be found. These hardcores, with appropriate software, can improve performance and simplify design of the user's project. [36]

Due to the nature of FPGAs, they can be used in many applications. Theses applications include prototyping, "glue" logic, reconfigurable computing, computation acceleration, and others.

2.1.2 Xilinx Virtex 5 SXT

The Xilinx Virtex 5 SXT is in the Virtex 5 FPGA family aimed at signal processing. Other members of the Virtex 5 family include the LX which is aimed high performance logic, the LXT which is aimed at high performance logic with serial connectivity, and the TXT which is aimed at high bandwidth applications. The SXT was chosen for its high DSP slice count, which would assist design of a the NC-OFDM transceiver. The DSP slice can be thought as an extension to embedded multipliers as seen in earlier FPGAs. The purpose of the embedded multiplier were to increase multiplication performance and free up logic element that would have been used for instantiating the multiplier. Since many DSP related functions also involve multiply-accumulate, the multiplier was extended to included addition and provide an accumulate feature. Figure 2.3 shows a simplified representation of the DSP48E.

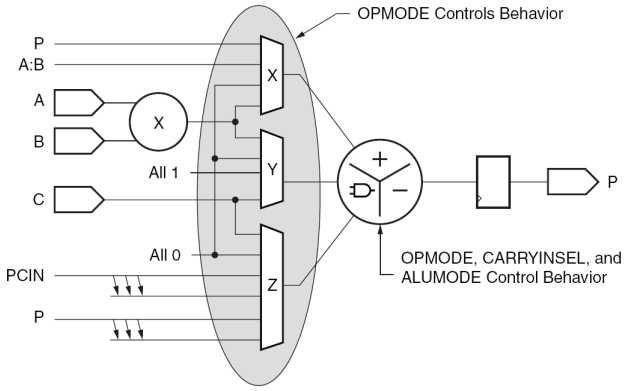


Figure 2.3: Simplified representation of DSP48e slice[35].

The SXT also contains PCIe, ethernet MAC, and RocketIO ^{TM}GTP Low-Power Transceivers. The ML506 Development board was chosen for its XC5VSX50T, USB ports, ethernet ports, PCIe connector, and GPIO. This would allow for multiple possible solutions for connecting to a computer or USRP [36].

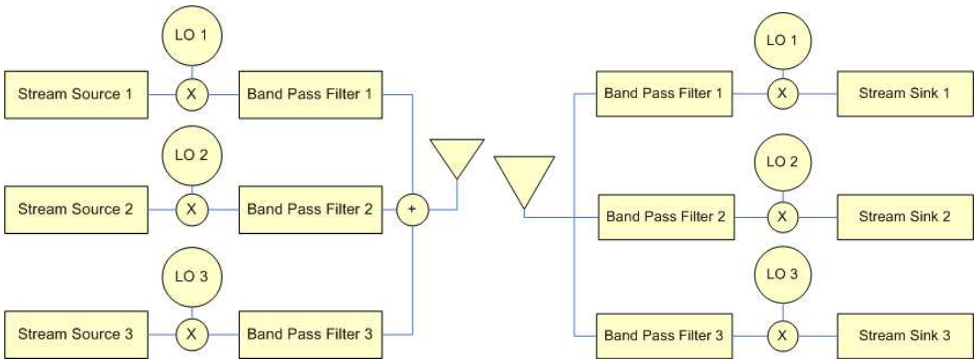
2.1.3 FPGA use in Software-Defined Radio

Software defined radio can benefit from a the flexibility of an FPGA. There are several common aspects between many communication standards such as IEEE 802.11a/g [21] [20], but most differ widely. This result in differing hardware configurations for different communications standards. Since an FPGA can be partially or fully reprogrammed a system containing FPGAs can be reconfigured for many different wireless communications standards. Another feature that FPGAs can offer is high performance through many parallel operations. Due to their nature, bitwise operations are also better performing on FPGAs.

2.2 Multi-carrier Transmission Fundamentals

2.2.1 Frequency Division Multiplexing

Frequency division multiplexing is a method of sending multiple independent signals simultaneously over different transmission center frequencies. Each of these independent signal can have their own modulation, such as M-ary QAM, M-ary PSK, FSK, voice, or others. Each of these signals are modulated onto its own unique sub-carrier. Once modulated, these signals are then summed together and passed to the *radio frequency* (RF) frontend in the case of wireless communication. On the receiver side, the RF-frontend receives the entire frequency channel. The sub channels are separated by parallel bandpass filters, one for each sub-carrier. Each individual sub-channel is demodulated back to baseband. From there each of the independent signals are demodulated to recover the individual data. Figure 2.4 shows the simplified structure of a FDM wireless communication system [7].



(a) FDM Transmitter. (b) FDM Receiver.

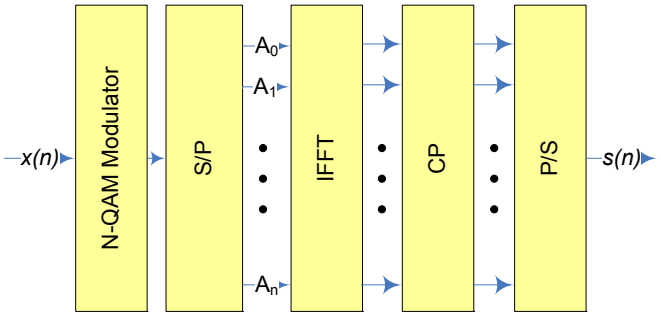
Figure 2.4: FDM Transceiver system

2.2.2 Orthogonal Frequency Division Multiplexing

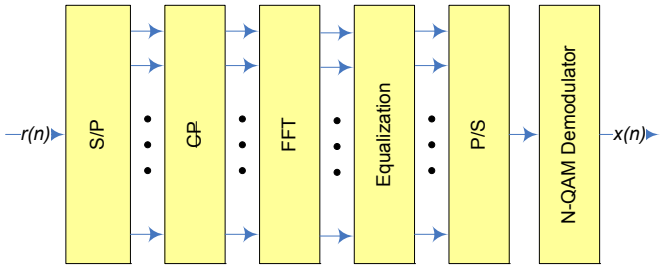
OFDM is similar to FDM, but uses the principle of orthogonality between sub-carriers on different frequencies to space the carrier closer together. Each carrier is individually modulated using M-ary *phase shift keying* (PSK) or M-ary *quadrature amplitude modulation* (QAM) to form a series of constellation points.

In OFDM, the N sub-carriers that can be individually modulated using either M-ary PSK or M-ary QAM transmission [23], depending on the quality of the transmission channel. The modulation used for the carriers depends on the the response of the channel and the noise of the channel. Typically each carrier is modulated with same constellation, but it is possible to modulate different carriers with different constellations as long as the receiver knows how each carrier is modulated [4]. In order to modulate a data stream, the data needs to be converted from a serial stream to several parallel data streams. Each of these streams are modulated into in-phase (I) and quadrature-phase (Q) portions. The resulting I and Q pairs can be modulated to a specific frequency by multiplying by cosine and sine sources, but a more efficient technique is to send all the I and Q pairs into a N-point IFFT. After the IFFT has been computed, a guard interval is appended to the NC-OFDM packet in the *cyclic prefix* (CP) block. This guard interval is used to mitigate *inter-symbol interference* (ISI) at the receiver. The resulting real data points are converted from the parallel streams back into a serial data stream, and sent to a RF-frontend. The receiver

operates in the reverse direction, with the data being sent through a FFT. The structure of a OFDM transceiver can be found in Figure 2.5. A_n are the individual sub-carriers.



(a) OFDM Transmitter.



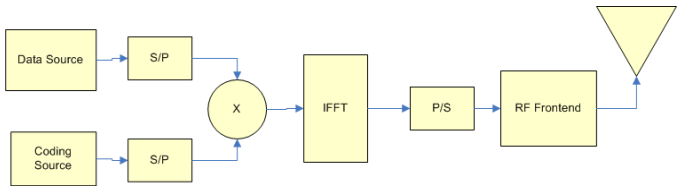
(b) OFDM Receiver.

Figure 2.5: OFDM Transceiver system

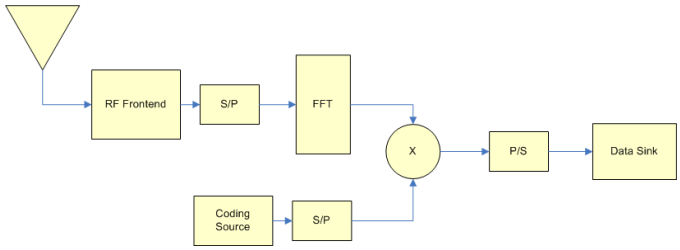
Since OFDM is comprised of sinusoids that are that are active during the symbol, each carrier in the frequency domain is a sinc pulse. Over the entire spectrum, there is only one carrier that has a non-zero value at a sampling point in the frequency domain, all others are making a zero crossing. This results in a high inter-carrier interference (ICI) when the carrier frequency offset between the transmitter and receiver since a since pulse rapidly changes when not at a zero crossing. Additionally, a sampling frequency offset at the ADCs and DACs can cause ICI that grows as the distance from zero hertz carrier increases. This was not investigated in this thesis.

2.2.3 Multi Carrier - Code Division Multiplexing Access

Multi carrier - code division multiplexing access (MC-CDMA) is a mixture of *code division multiplexing access* (CDMA) and OFDM. In MC-CDMA, separate data channels are spread over multiple sub-carriers according to a spreading code. Figure 2.6 shows the basic structure of a MC-CDMA transmitter and receiver.



(a) MC-CDMA transmitter



(b) MC-CDMA receiver

Figure 2.6: MC-CDMA transmitter and receiver

MC-CDMA is very similar in structure to that of OFDM. In both, the IFFT and FFT are used to modulate and demodulate a set of sub-carriers. This is where the similarities end. On the transmitter side, multiple data streams are coded across multiple carriers through the used of the their corresponding code. On the receiver side, the received signal is transformed back into the frequency domain [37]. From there, the data packet is multiplied by the appropriate code to obtain the original data. In the case of *non-contiguous* MC-CDMA (NC-MC-CDMA), some of these carrier are deactivated [26].

2.3 Orthogonal Frequency Division Multiplexing in Depth

In this thesis, a variation of OFDM is used called non-contiguous OFDM. In order to best understand the techniques used in NC-OFDM, specifically synchronization, OFDM

must be explained in more detail.

2.3.1 Synchronization

Synchronization is critical for obtaining the best performance from an OFDM communication system. There are two main parts in OFDM synchronization: carrier frequency offset and symbol timing. As can be seen previously, the data carriers appear as sinc pulses in the frequency domain. Sampling at the correct frequencies allows for only one tone in the OFDM transmission to be read. Since a sinc pulse rapidly decreases from its peak and rises in magnitude from a zero crossing, deviations in carrier offsets can quickly result in a degradation in system performance. Figure 2.8 shows the effect of sampling at the incorrect frequency. It can be seen when a frequency offset is present, the other sub-carriers are at a non-zero value, thus adding to the sub-carrier that is being sampled and corrupting the sampling.

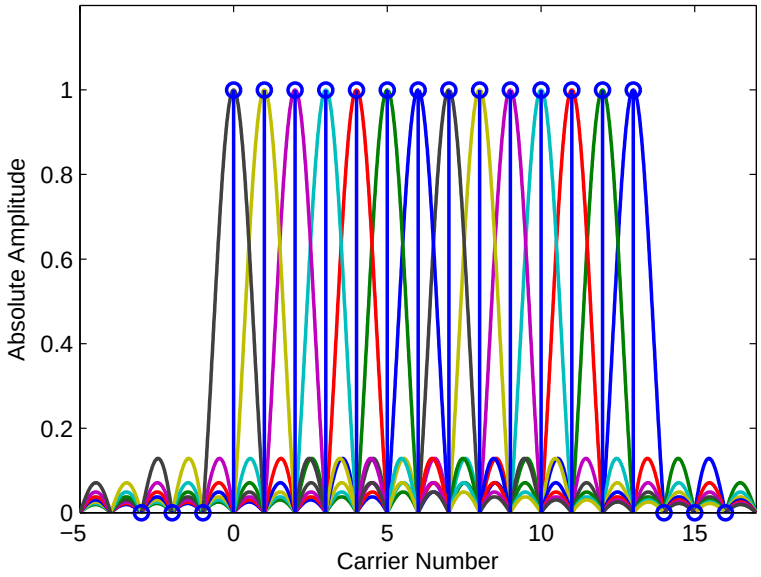


Figure 2.7: Sampling points of OFDM transmission in frequency domain with no CFO.

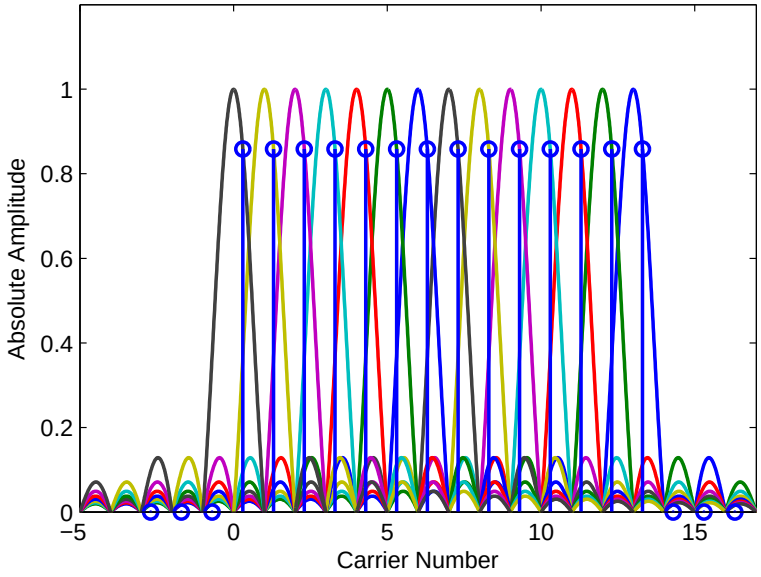


Figure 2.8: Sampling points of OFDM transmission in frequency domain with a CFO.

The symbol timing portion of synchronization deals with sampling instant for the FFT of a set of received data. An incorrect window of data will distort the output from the FFT since a portion of the symbol will be missing. The degradation of the data is increased further if the channel is a fading or multi-path channel.

Symbol Timing

The *cyclic prefix* (CP) is found in nearly all OFDM communication systems. The purpose of the CP is to assist in symbol timing and to mitigate the effects of a multi-path channel. The length of the CP is chosen to ensure that it is longer than the length of the longest path in a multi path channel. A size of one forth of full symbol length and the last samples of the transmitted symbol are appended to the beginning of the symbol, such is in the case of 802.11a [21]. In several communication systems, a cyclic postfix is also used. Note that a cyclic postfix can also be employed which is comprised of a portion the last several samples of the symbol appended to the beginning of the symbol [11]. Figure 2.9 shows an example structure of a several OFDM data symbols with and option *guard interval* (GI).

Figure 2.9: OFDM data symbols with guard intervals and cyclic prefix.

Since the CP is identical to the last section of samples in a OFDM symbol, it can to assist in finding the proper window to sample for the FFT. [28] This is one method of symbol timing that is used. Equation (2.1), (2.2), and (2.3) were derived by Schmidl and Cox and are used to find the start of a symbol [28]:

$$P(d) = \sum_{m=0}^{L-1} r^*_{d+m} r_{d+m+L}, \tag{2.1}$$

$$, R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2 \tag{2.2}$$

$$M(d) = \frac{|P(d)|^2}{R(d)^2}, \tag{2.3}$$

where *P(d)* is the autocorrelation of the received data stream *r*, *R(d)* is the energy of the received data, L is length of the data symbol without the CP or cyclic postfix which is N, the size of the FFT, and $M(d)$ is used to find the start of the OFDM symbol. Specifically Equation (2.3) will rise to a plateau when there is no noise in the channel. The end of this plateau is the precise start of the symbol. Figure 2.10 shows this plateau with the ideal sampling point marked.

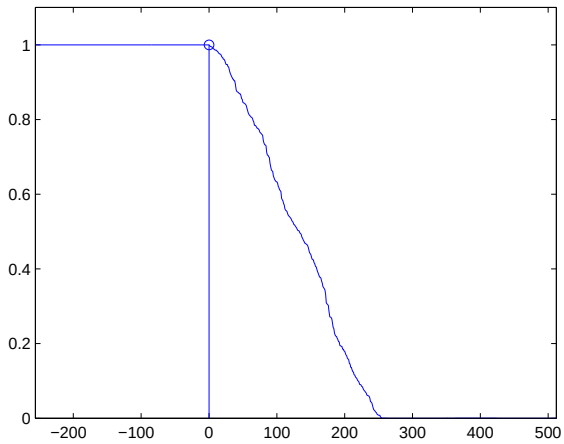


Figure 2.10: Example plateau showing the start of a data symbol

In the presence of noise or other channel effects, the end of the plateau may not be as well defined, but may have several points that appear to be end of the plateau or may drop off early compared to the actual start of the symbol. This interferes with the finding of the true symbol timing instant. In this case, it is best to ensure sampling is earlier than the correct instant since the a late sampling will be corrupted because only a portion of the symbol is captured. An early sampling will contain part of the CP and the symbol, but no irrecoverable corruption. The only effect will be a phase rotation that increases as the index from the DC carrier increases. For this reason, it is better to sample early for the FFT of the data symbol as opposed to sample late, since the data still can be recovered in the channel estimation portion of the receiver.

2.3.2 Carrier Frequency offset

Carrier allocation

In OFDM based communication systems, the carriers are in predetermined states that cannot change. The carriers in the preambles are in fixed locations and the carriers in the data symbols all have fixed states. The states for these carriers are data, pilot, and null.

Carrier Frequency Offset

Carrier frequency offset can be broken into two portions as seen in Equation 2.4.

$$f_{\text{offset}} = \frac{\phi}{\pi * T} + \frac{2 * z}{T} \tag{2.4}$$

As can be seen in this equation, there are two terms, a fine carrier frequency offset and a coarse carrier frequency offset. In this equation, ϕ is the phase rotation from the timing metric used with the short preamble, z is an integer amount determined from the coarse CFO estimate and T is the preamble length. The phase rotation is caused by the the CFO. Coarse frequency offset is an integer offset of the inter-carrier frequency spacing and the fine frequency offset is offsets that are offsets are smaller than the carrier spacing. The first term is the fine offset and the second term is the coarse offset. In order to correct for these two portions of the carrier frequency offset, the short preamble is used to estimate the fine carrier offset and the long preamble is used to estimate the coarse offset [19].

Equation (2.5) shows the equation for calculating the non-normalized offset metric. Equation (2.6) is used to create the normalization metric which is used with Equation (2.5) as shown in Equation (2.7) :

$$P(d) = \sum_{m=0}^{L-1} r^*_{d+m} r_{d+m+L}, \tag{2.5}$$

$$R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2, \tag{2.6}$$

$$M(d) = \frac{|P(d)|^2|}{R(d)^2}. \tag{2.7}$$

These equations are nearly identical to equations (2.1), (2.2), and (2.3) used to find the start of the data symbols, but in this case L is the length one of the individual symbols in the short preamble. Typically there is four or eight smaller symbols in the short preamble. From Equation 2.7, the fine offset can be estimated by using Equation 2.8,

$$f_{offsetfractional} = \frac{\arctan P(d)}{\pi * T} \tag{2.8}$$

where T is the length in time of symbol.

Since the fine carrier frequency can only estimate the non-integer portion of the carrier frequency offset, the coarse frequency offset estimation must also be performed. Schmidl and Cox proposed a method correcting the integer portion by sending a specially designed OFDM symbol, or preamble. The only active carriers are the even carriers, with the DC-term zero. The symbol is comprised of a cyclic prefix, and two identical OFDM symbols. The receiver takes the FFT of this special OFDM symbol. Using the demodulated symbol, an estimate can be formed using the maximal likelihood function on $B(g)$, defined in Equation 2.9.

$$B(g) = \frac{|\sum_{k \in X} x^*_{1,k+2g} v^*_k x_{2,k+2g}|^2}{2\sum_{k \in X} |x^2_{2,k}|^2}. \tag{2.9}$$

The long preamble is composed of two identical symbols concatenated with each other. The only active carriers are the even carriers. These FFT's of these two halves are $x_{1,k}$ and $x_{2,k}$, and v_k is the original symbol without any carrier offset effects. The set X is the set of possible even frequency offsets. This approach can be a lengthy process as vectors need to be shifted and several vector multiplication need to be done for each shift. A simplification can be made by assuming that the carrier offset can only span a small portion of the possible offsets. This assumption can be extended further to possibly eliminate the long preamble entirely if the short preamble will cover all expected frequency offsets. This may be the case in which the a carrier frequency offset estimation has recently occurred [28].

2.3.3 Channel Estimation

Channel estimation is very important to the receiver. Most wireless channels do not have a flat frequency response since the radio waves will reflect off objects in the environment, resulting in a slightly different length path from the direct line-of-sight (LOS) path, and thus yielding a delayed response. Consequently, this creates a non-flat channel. In order to correct for having a non-ideal channel, pilot carriers are often used to estimate the channel. The channel estimation and correction process may be done in the time or frequency domain, or a combination of both. Additionally, if the data symbol acquisition does not trigger at

the correct sampling instant, the carriers will be rotated proportionally to their distance to the DC term [13].

2.3.4 Effects of Poor Synchronization

Poor synchronization can have many effects, some of which are more devastating than others and some are correctable. When a sampling frequency offset exists, the spacing between the sampling points in the frequency domain changes. This results in sampling at the non-ideal points along the sync pulses. This results in an increasing offset between the ideal sampling point for a particular sub-carrier to the actual sampling point as the frequency increases or decreases from 0 Hz in baseband. This results in the sub-carriers being corrupted with the appearance of worsening noise as the frequency moves away from 0 Hz. The apparent noise is caused from interference from the other carriers. A carrier frequency offset has a similar effect, but since the sampling points have the correct spacing and are only shifted a constant amount of hertz, every carrier has the appearance of an increase amount of noise.

The data symbol acquisition is another portion that can produce various effects when not perfectly timed. In the case of sampling too early, the sub-carriers are rotated proportionately to their distance to the 0 Hz sub-carrier. This also assumes that the sampling instance of the data symbol is within the cyclic prefix and the last reflection in a multi-path environment has arrived. In the case of sampling too late, the sub-carriers are again rotated. If there is not a cyclic postfix, the sub-carriers will have the appearance of an increased amount of noise.

2.3.5 Complete Transmitter and Receiver Structure for OFDM

Transmitter Structure

The transmitter for an OFDM modem is relatively simple compared to the receiver since it lacks synchronization. Figure 2.11 shows the structure of a typical OFDM modem.

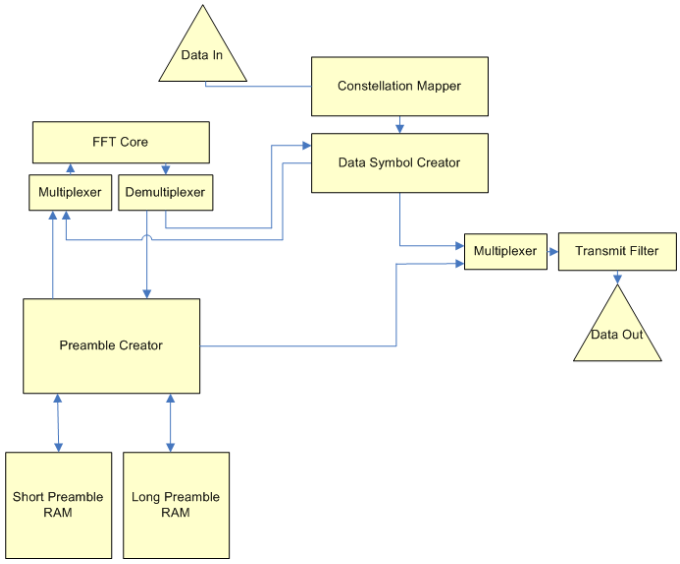


Figure 2.11: Basic OFDM transmitter

The FFT core is used to initially create the long and short preambles and then is use to modulate the individual carriers that make up an OFDM symbol. The short preamble is sent out to the RF-frontend which is then followed by the long preamble. Once this is done, data symbols can be sent. The contents of the initial data symbols may not be data, but could be for control. In some cases, a transmit filter is present to reduce the effects of the non-idealities of the RF-frontend, such as non-linear gain or a non-flat frequency response.

Receiver Structure

The receiver possesses the additional complexity of supporting the synchronization and channel equalization units. These units are critical for correctly receiving an OFDM transmission burst. Figure 2.12 shows the structure of a basic OFDM receiver.

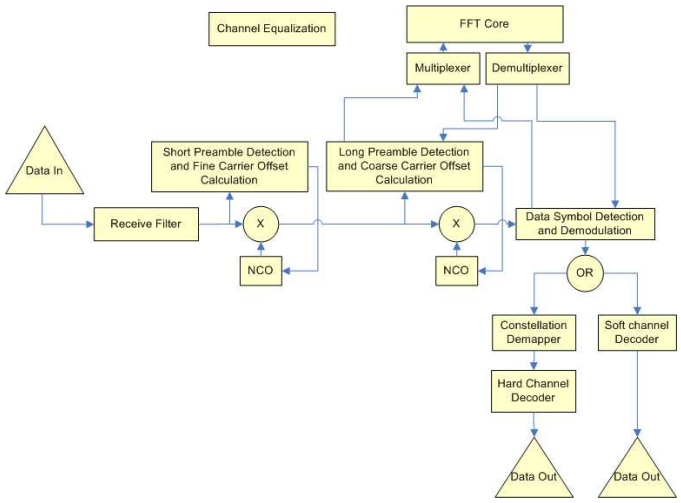


Figure 2.12: Basic OFDM receiver

The RF frontend feeds directly into the receive filter, which has a similar purpose to transmit filter. The data stream from the receive filter is then sent the synchronization units. The fine frequency offset estimation and correction has two purposes: to listen for the start of a transmission, and to correct for the fine frequency offset. The coarse frequency estimator is then used estimate and correct for the coarse frequency offset. At this point, the carrier frequency offset is minimal. The data stream then goes to the data symbol detector and demodulator. Since multiple techniques exist for frequency domain [13] and time domain [29] channel estimation and correction, the channel equalizer is left unconnected. Once the channel has been compensated, the individual carriers can be sent to constellation de-mapper or soft channel decoder, depending on the channel coding used.

2.4 Fast Fourier Transform

At the University of Kansas, it was noted that the FFT and the IFFT efficiency could be improved for a NC-OFDM transmitter and receiver. In an OFDM based transmission scheme, the FFT and IFFT are used to demodulate and modulate the sub-carriers. For NC-OFDM, carriers maybe deactivated, thus set to zero. These deactivated carriers may interfere with the primary user or carry no meaningful information after demodulation.

These null (deactivated) carriers have no effect on the output of IFFT or do not provide meaningful information after the FFT. In these cases, the FFT and IFFT can be pruned to remove the calculation that don't carry relevant information through to later stages. Since FFT is tightly coupled with the synchronization of OFDM, and also NC-OFDM, pruning must be done in hardware [27].

2.4.1 FFT Pruning

The FFT and IFFT are efficient methods of computing the *discrete Fourier transform* (DFT) and the *inverse discrete Fourier transform* (IDFT). The DFT operates by taking a set of time domain points and transforming them into the frequency domain. Similarly, IDFT takes a set of frequency domain points and transforms them into the time domain. The mathematical formulas describing the DFT and IDFT operating are [23]:

$$X_k = \sum_{n=0}^{N-1} x_n e^{(\frac{-2\pi j}{N} kn)}, k = 0, ..., N-1, \tag{2.10}$$

$$x_k = \frac{1}{N} * \sum_{n=0}^{N-1} X_k e^{(\frac{2\pi j}{N} kn)}, n = 0, ..., N-1, \tag{2.11}$$

where X_k is the kth frequency term, x_k is the kth time domain term, and $0N$ is the number of points in the DFT or IDFT. The IDFT and DFT are computationally complex process with order $O(N^2)$ complexity. [5] The FFT and IFFT reduces this complexity by repeatedly splitting the summation, for the DFT or IDFT into an even half and an odd half until the computation consists of just two points. This approach for computing the DFT is called a radix-2 FFT [5]. The FFT has a complexity of $O(N * log(N))$, which is significantly lower relative to a conventional DFT.

Since the FFT and IFFT are computationally similar in structure, most pruning research has been done on the FFT, although it can can also be applied to the IFFT. Only algorithms that prune from the input side or from the output side are considered since only the input to the IFFT will contain nulls sub-carriers and only the output of the FFT will contain null sub-carriers and complex conjugates, which also can be eliminated. Figure 2.13 shows the structure of a pruned 8-point FFT in radix-2. In this case, the first and fifth inputs to

the FFT are both zero while the others are non-zero. Since the first and fifth inputs form the inputs to the first butterfly in the first stage, this butterfly is not needed and can be pruned. This is the only butterflies that has both input term zero, and thus pruned.

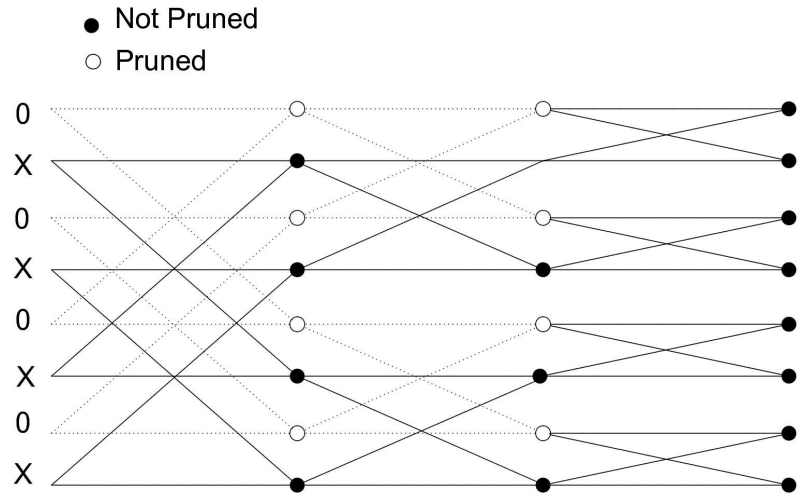


Figure 2.13: Pruned butterfly structure. An 'X' denotes a used sub-carrier and a '0' denotes a unused sub-carrier.

One method proposed by Alves *et al.* uses the generation of an M matrix [2]. This algorithm uses a radix-2 Cooley Tukey implementation of the FFT [5]. This matrix is a table listing of whether a node needs to be calculated or not. The columns of this matrix correspond to the stages of a FFT and the elements in the columns correspond to the nodes. A node represent by a single calculation in the FFT or IFFT algorithm. The resulting matrix is comprised of ones and zeros corresponding to whether a node needs to be calculated or not. This algorithm is also extended to perform both input and output pruning. In this case, some of the inputs to the FFT are zero and not all the outputs are needed. This can result in a further decrease in the number of computations needed to perform the FFT. For example, the FFT pruning algorithm of Alves *et al.* will generate the following matrix for the FFT:

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{2.12}$$

Algorithm 1 Generate M_i (from [2])

1: $M_{\text{itemp}} = [ivector, zeros(N, M)]$;
2: **for** $i = 1$ to M **do**
3: **for** $j = 0$ to $2^M - 1$ **do**
4: $pair(1) = j + 1$
5: $pair(2) = mod(((N/(2^i)) + j), (2 * N/(2^i))) + floor((j/(2^{M-i+1}))) * floor(N/(2^{i-1})) + 1$
6: $x = M_{\text{itemp}}(pair(1), i)$
7: $y = M_{\text{itemp}}(pair(2), i)$
8: $M_{\text{itemp}}(j + 1, i + 1) = x|y$
9: **end for**
10: **end for**
11: **return** $M_{\text{i}} = M_{\text{itemp}}[2 : M, 2^M]$

Another pruning method is proposed by Rajbanshi *et al.* [27], which builds upon the algorithm proposed in [2]. The resulting matrix needs to be indexed through during the computation of the FFT. The code that implements the FFT is modified to have conditional statements. These conditional statements result in the slowing demand of the total execution time, which yields a longer execution time for a simple FFT implementation. In Algorithm 2 which was proposed by Rajbanshi *et al.* [27], a new matrix called M_{index} is created which removes the conditional statements. The top row of the M_{index} matrix is an

integer value for the number of nodes that need to be calculated for that particular stage. In the column directly below each of these numbers as seen in Equation (2.13), the nodes that need to be calculated are listed in order. The FFT calculating portion know how many nodes need to be calculated and can loop for the precise amount of times needed [27]. The Algorithm 2 generates another matrix, M_{index}, which yields:

$$M_{\text{index}} = \begin{pmatrix} 6 & 8 & 8 \\ 2 & 1 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 3 \\ 6 & 4 & 4 \\ 7 & 5 & 5 \\ 8 & 6 & 6 \\ 0 & 7 & 7 \\ 0 & 8 & 8 \end{pmatrix} \tag{2.13}$$

Algorithm 2 Generate M_{index} (from [27]

1: $[n, m] = size(M)$
2: $y = zeros(n, m)$
3: **for** $j = 1$ to m **do**
4: $\quad te = find(M(:, j))$
5: $\quad y(1 : length(te), j) = te$
6: **end for**
7: $y = [sum(M, 1); zeros(1, m); y]$
8: $j = [1 : m]$
9: $y(1, :) = (n - y(1, :))./2j$
10: **return** $M_{\text{index}} = y$

2.4.2 FFT Hardware Implementations

Since the FFT and IFFT are built upon the butterfly operation, all of which are independent from each other in a given stage, the hardware implementation of the FFT can provide a significant speed up. There are multiple hardware based FFT architecture. These architecture are: parallel and serial. Additionally, these may be pipelined.

In the parallel FFT hardware architecture, there are multiple FFT butterfly in a single cluster working in parallel on the same set of data [39]. This can be extend be a parallel pipelined architecture. In this case, there are mutliple clusters of FFT butterflies, all of which are working on different data. These clusters are arranged in a serial fashion and the data is passed from one cluster for partial processing to the next [14]. In the case of a serial FFT hardware architecture, the butterfly units are arranged in a single line, connected by buffers. A pair of data points are fed into the begining of the pipeline. The butterflies, in this case, operate at the same rate as the data is being fed in [10]. The have been other proposed FFT hardware architectures, which are variations of the above, which have their own benefits and tradeoffs.

2.5 Chapter Summary

From this chapter, an overview of cognitive radio and prior research into SDR hardware platform was presented in this chapter. These groups have used FPGAs, ASICs, DSPs, and computers. The work of this thesis is aimed to assist in enabling cognitive radio performing spectrum pooling by presenting the architecture for the physical layer of a NC-OFDM transceiver. Orthogonal Frequency Division Multiplexing was given special emphasis, particularly with respect to synchronization. Understanding how OFDM is structured and its synchronization process is important in understanding the proposed NC-OFDM structure and synchronization techniques since the techniques are similar to that of OFDM.

Chapter 3

Non-Contiguous Transmission Enabling Dynamic Spectrum Access

This chapter presents the the NC-OFDM synchronization algorithm. As will be seen in this chapter, the existing techniques used for OFDM synchronization do not need to be drastically changed. The changes that do need to be made are presented in this chapter

3.1 Non-contiguous Orthogonal Frequency Division Multiplexing

In non-contiguous OFDM (NC-OFDM), contiguous bands of sub-carriers cannot be used because their use would interfere with the primary user of a given portion of the radio spectrum, thus those carriers are left unused. Figure 3.1 shows a set of carriers within an NC-OFDM transmission deactivated to minimize interference with the primary user.

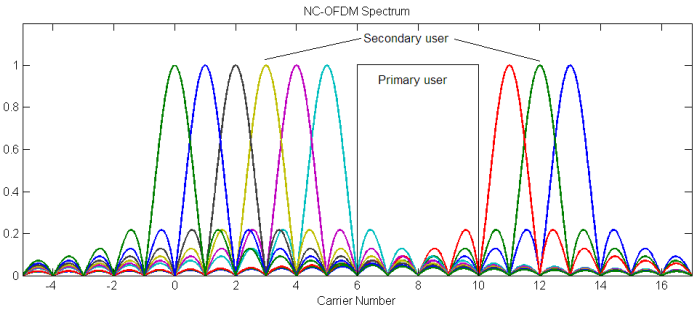


Figure 3.1: Sub-carriers of an OFDM transmission interfering with primary user and a NC-OFDM with much reduced interference

From this example figure, it can be seen that six through ten would cause a significant amount of interference with the primary user. Although the NC-OFDM does bleed over in the spectrum used by primary user, this can be be minimized through various techniques.

The deactivated carriers creates several problems that are not seen in wireless OFDM communication systems. The first problem is that there are several unwanted signals dispersed throughout the transmission band. This interference may cause a large error in carrier frequency offset estimation and incorrect data symbol timing. In nearly all OFDM transmission systems, the short preamble carriers, long preamble carriers, and pilot carriers in the data symbols are at fixed locations. In NC-OFDM transmission, those carriers could interfere with the primary user, which would be an unacceptable case. In addition to this, the locations of the primary users is expected to change over time, thus the usable carrier for pilots and the preambles will also change over time.

3.2 Primary User Filter

In order to filter out the primary user, a complex FIR filter is used. The primary user filter is comprised of multiple bandpass filters, one for each block of carriers. Each bandpass filter is first created as a low pass filter of the appropriate bandwidth for that block and then modulated with a complex sinusoid to be centered at the correct frequency. The low-pass filter is an approximation of an ideal low pass filter. Equation 3.1 shows the equation for an ideal low pass filter and Equation 3.2 shows an approximation of an ideal filter.

$$h = \sum_{n=-\infty}^{\infty} BW * sinc(BW * N) * n_N \tag{3.1}$$

$$h = \sum_{n=-x}^{x} BW * sinc(BW * N) * n_N \tag{3.2}$$

where $h(t)$ is the impulse response of the filter, BW is the bandwidth of the filter, and x is an arbitrary approximation factor.

Once each of the bandpass filters are created, the individual filters are summed to together to form one complex FIR filter. The next step is use a Hamming window to increase the flatness of the filter response in the bandpass sections. The Hamming window coefficients are defined by:

$$w(n) = .54 - .46 * cos\left(\frac{2 * \pi * n}{N - 1}\right), \tag{3.3}$$

where N is the length of the window, n is the individual coefficient term number, and $w(n)$ is the coefficient itself [31].

3.3 Synchronization

As stated in the previous chapter, there are two parts within an OFDM transmission that need to be synchronized: the carrier frequency offset and symbol timing offset. Once the primary user has been removed, the existing techniques can be used with some modification.

3.3.1 Symbol Timing

In the presented OFDM symbol timing algorithm by Schmidl and Cox, the output of the normalized decision function has a plateau. The ideal sampling instant is at the very end of this plateau. Figure 3.2 shows an example output of this function with the ideal sampling point marked.

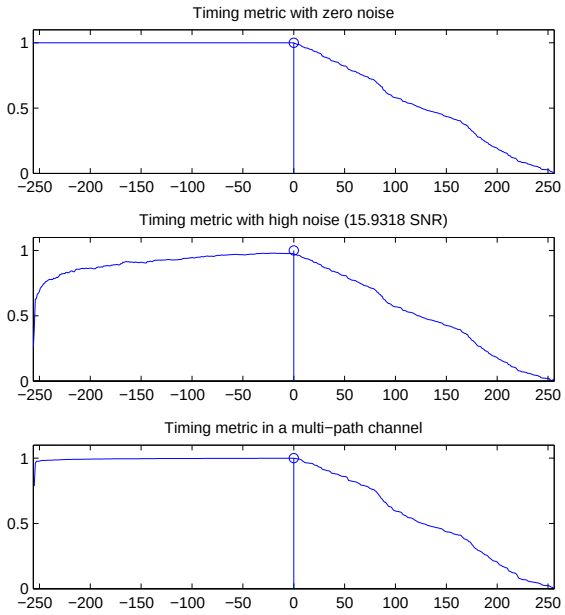


Figure 3.2: Plot showing timing metric in various channels

In the presence of little noise or no multi-path, the end of the plateau is well defined. In the case of noise of a multi-path environment, this end in the plateau is not as clear. It was found that by modifying the normalization function, the normalized function would output a peak at the ideal sampling instant. Thus the new normalization function is defined as:

$$R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2 + r_{d+m} \tag{3.4}$$

Figures 3.3 show the output of the decision function with a the proposed normalization function.

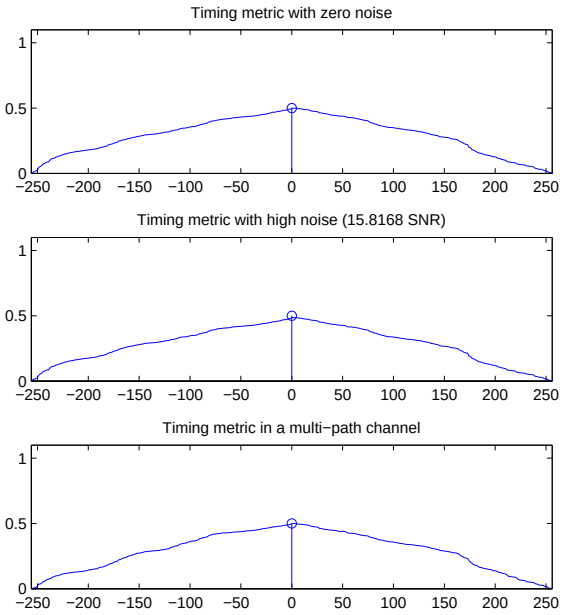


Figure 3.3: Plot showing timing metric in various channels

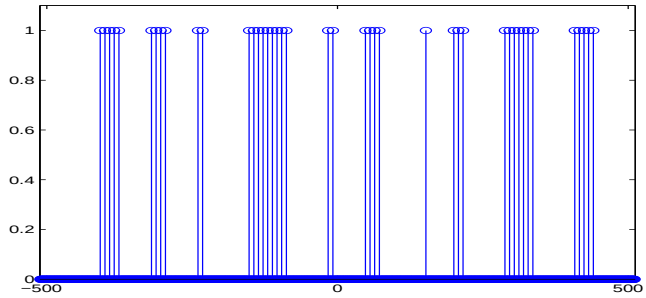
As can be seen, the function forms one peak. Detecting the location of the maximum is better defined than that of the the ending of the plateau in the presence of noise.

3.3.2 Carrier Frequency offset

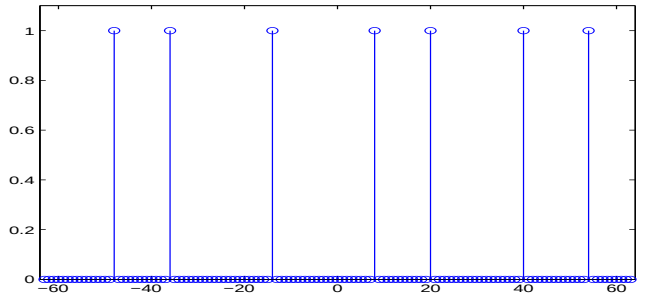
Carrier Allocation

In the case of OFDM, the carriers usage in the short preamble, long preamble, and data symbols symbols are fixed and never change from a predetermined function. This means that once a carrier type has been determine, such as a pilot carrier in the data symbol, the carrier will always be that type. In NC-OFDM, this approach would not work since some sections of the spectrum are unusable and a fixed carrier type approach would interference with the primary users by transmitting in their spectrum. This means that any carrier can be used as a null, data, pilot, preamble carrier, or allocated for cancellation carrier use. Since carrier availability will be given, this information can be used to determine a single possible configuration for the carriers.

The allocation of the carriers for the short and long preamble correspond to the same frequency. This simplifies the selection of carriers to use. The constraints for selecting carriers is that the carrier must in lie with in a block of carriers used for transmission and to be the furtherest away from the edges of the block. The last constraint is for the reduction side-lobes that exist in all OFDM transmissions. Figure 3.4 shows the possible carriers for use in the short preamble of a NC-OFDM transmission and the carriers to be used in the actual transmission of the preamble.



(a) Possible carriers for preambles



(b) Used carriers for preambles

Figure 3.4: Plots showing which carriers could be used and which carriers cannot be used in the preambles

The allocation of the data, pilot, and possible cancellation carriers is similar. The basic structure of a usable carrier block of within the NC-OFDM transmission spectrum starts with the outermost carriers. In this case, three carriers on either side are allocated for the cancellation carriers. From there, the next two carriers are data carriers followed by a pilot carrier. This results in the outermost six carriers on either side being allocated and

the middle to be determined. The middle carriers are allocated to data and pilot carriers such that the maximum distance between the pilot carriers is five carriers. This can result in some spacings between adjacent pilot carriers to be different from the spacing between another set of adjacent pilot carriers. This creates a dense pilot carrier arrangement, but can be changed depending on the response of the channel. A fairly dense pilot carrier arrangement is needed to ensure the a proper estimate of the channel can be made and phase rotations caused by an early data symbol acquisition trigger. Figure 3.5 shows an example allocation of the carriers within the data symbols.

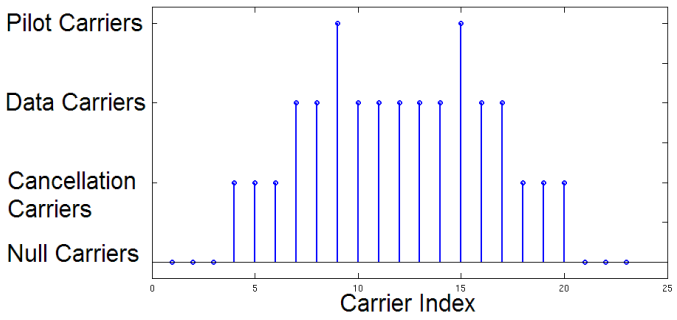


Figure 3.5: Example carrier allocation for a block of carriers

Fine Carrier Frequency Offset

The fine carrier frequency offset is very similar to the proposed method by Schmidl and Cox [28]. The autocorrelation function and the normalization function from equations (2.1), (2.2), and (2.3) has been changed to the following expressions:

$$P(d) = \sum_{m=0}^{L-1} r^*_{d+m} r_{d+m+2*L}, \tag{3.5}$$

$$R(d) = \sum_{m=0}^{L-1} |r_{d+m}|^2 + \sum_{m=0}^{L-1} |r_{d+m+2*L}|^2, \tag{3.6}$$

$$M(d) = \frac{|P(d)|^2|}{R(d)^2}, \tag{3.7}$$

In these equations, *P(d)* is the autocorrelation of the received data stream, *r*, *R(d)* is the energy of the received data, *L* is length of the data symbol without the CP or cyclic

postfix which is N, the size of the FFT.

This allows for a more accurate estimate on the start of the short preamble, which signifies that a burst of data will soon follow. The proper detection of the short preamble is very important as too late or too early may cause incorrect estimates on the carrier frequency offset. It is also possible that the start of the NC-OFDM is missed entirely. Additionally, the presence of noise, multiple transmission paths, and reduced primary user input from the primary user filter have a wide range of effects on the how $M(d)$ appears graphically. The combination of these three can create false plateaus, or the true plateau resulting from the short preamble can be be widely varying. From this, an algorithm is derived which can be seen in Algorithm 3.

Algorithm 3 Algorithm for detection of the preamble

1: $count_{down} = 0$
2: $count = 0$
3: **for** $d = 1$ to $length(M(d))$ **do**
4: **if** $M(d) > threshold$ AND $R(d) > threshold_{norm})$ OR $count > 0$ **then**
5: $count = count + 1$
6: **end if**
7: **if** $M(d) < threshold$ AND $count > 0$ **then**
8: $count_{down} = count_{down} + 1$
9: **else**
10: $count_{down} = 0$
11: **end if**
12: **if** $count_{down} > count_{maxdown}$ OR $M(d) < threshold_{low}$ **then**
13: $count = 0$
14: $count_{down} = 0$
15: **end if**
16: **if** $not(triggered)$ AND $count > count_{max}$ **then**
17: $start = j - count$
18: $triggered = 1$
19: $break$
20: **end if**
21: **end for**
22: **return** $start$

The $count_{max}$, $count_{maxdown}$, $threshold$, and $threshold_{low}$ are set before hand, depending on the size of the FFT/IFFT. The algorithm above allows $M(d)$ to dip below $threshold$ and still allow $count$ to increment. The variable $count$ is used to determine whether or not a plateau has been reached, since a plateau will stay above a certain level for some time. The $count$ is reset should either $M(d)$ dips below $threshold_{low}$ or stays below $threshold$ for more than $count_{maxdown}$.

Coarse Carrier Frequency Offset

The coarse carrier frequency offset estimation has been modified from what Schmidl and Cox proposed. The maximum likelihood estimator employed in OFDM is fairly complex, the received carrier locations is shifted to every possible location and it is seen which fits best to that of what is expected [28]. This is a mathematically intense approach since it require multiple vector multiplications and every possible configuration is tried in the worse case. If an assumption on the maximum possible frequency offset is made, all possible combinations do no need to be attempted. The proposed solution uses a sparse allocation of carriers for the long preamble. The proposed method of coarse carrier frequency offset estimation is similar with respect to taking the FFT of a specially designed long preamble, but differs from that point onwards. The first step is to take the FFT of the long preamble. All of the carriers of the spectrum are compared to a threshold to determine if a carrier is present or not. The next step is to form a weighting vector that when multiplied by the guessed received carrier locations vector gives the number for frequency bins to shift by multiplied by number of carriers in the guessed received carrier locations vector. This value is divided by the number of carriers detected in the long preamble and then multiplied by the carrier spacing of the symbol in hertz. Figure 3.6 shows an example spectrum of the long preamble, presence vector, expected carrier locations, and weighting vector.

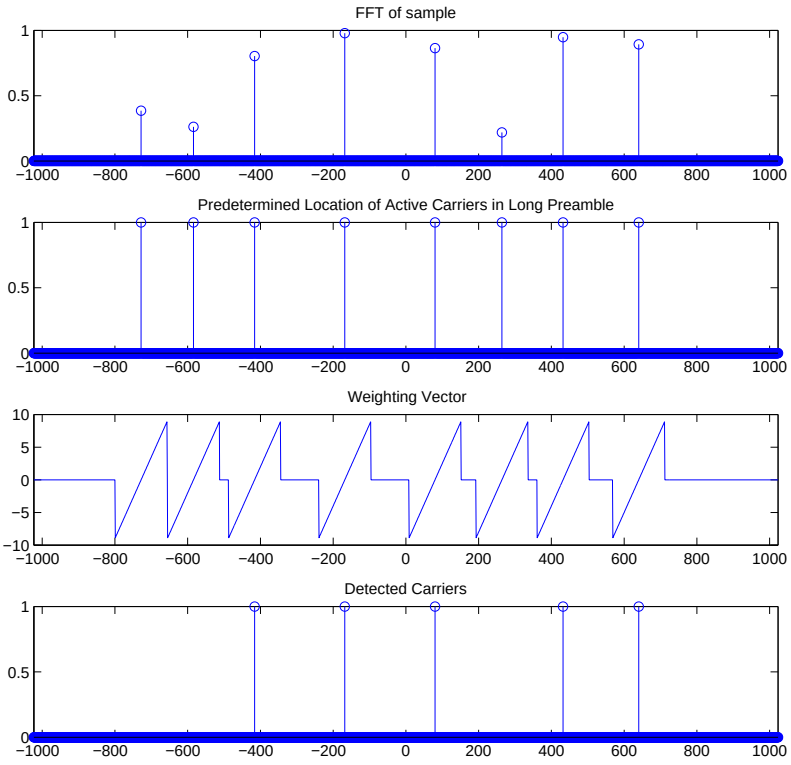


Figure 3.6: Graphs showing several graphs and their relation between each other.

As can be seen in the above figure, not all of the carriers are detected in the preamble. This is not a problem since weighting factor is divided by the number of detected carriers, not the number of expected carriers.

3.4 Channel Estimation

The channel estimation is an important aspect of an OFDM transmission system. The channel estimation is able to assist in the correction for a non-flat transmission channel, but also an early triggering of the data symbol sampling window. An early triggering of the sampling window does not cause a large increase in errors, but just a phase rotation of the carriers directly proportionate to their distance from the DC term. Currently, a simple

approach is taken to equalize the channel. This consists of finding the transfer function in the frequency domain based on the noisy pilot carriers. The transfer function is linearly interpolated to estimate the response between the pilot carriers. From there, the carriers can be compensated for any channel response. Additionally, a correction factor in the phase is applied before hand to account for the split paths. The primary user filter creates addition distortion, which can be avoided entirely by sampling from a carrier frequency offset corrected data path that bypasses the primary user filter.

3.5 Complete Transmitter and Receiver Structure for NC-OFDM

The NC-OFDM transmitter is very similar to that of the OFDM transmitter,which is seen in Figure 3.7. The primary difference is the IFFT core with pruning. The preamble creator creates both the short and long preambles and saves them for future use as it is not expected that the carriers able to be use will change between every transmission. The Data Symbol Creator takes the data stream of constellation mapped data, forms a vector containing the data, pilot carriers, and cancellations carriers. The vector is then converted to the time domain by the IFFT core with pruning. The transmission filter is optional, but can be used to correct for non-linear RF-front-end.

Since the spectrum that is being used may not change over short periods of time, the Preamble Creator will store the long and short preambles. This unit, upon command, send the short preamble to the transmit filter and then the long preamble. The Preamble Creator also needs access to the pruned IFFT core since a frequency domain vector is passed to it with the values of the carriers used for the preambles. In this case, it makes sense to use a pruned IFFT as the data input will be very sparse. The data symbol creator takes in raw data points from the data source. The Data Symbol Creator does not modulate the carriers as this is will be done before hand because each carrier may have a different modulation scheme, shifted constellation points, or possibly a cancellation carrier. Once the frequency domain data is received, the data is sent to the pruned IFFT core and then sent back, as a time domain signal, back to the Data Symbol Creator. This unit will then send the data

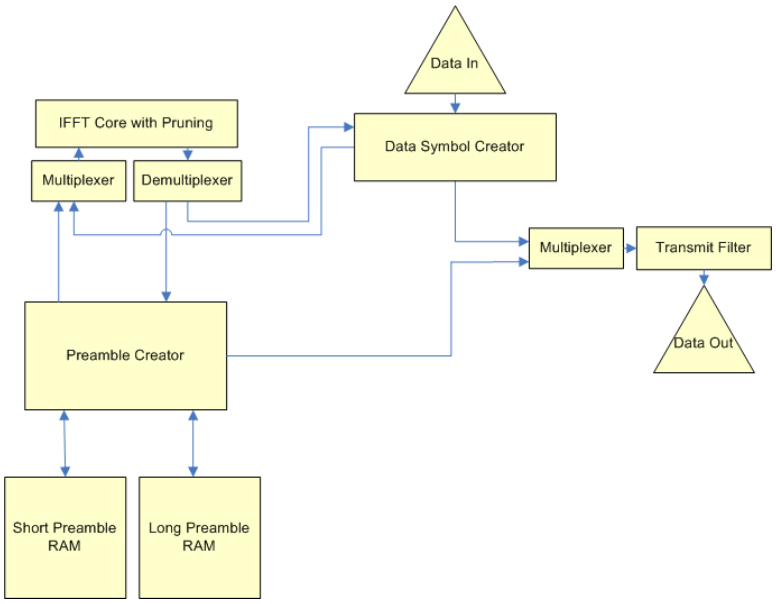


Figure 3.7: NC-OFDM transmitter.

out to the transmit filter with the cyclic prefix first.

The NC-OFDM receiver has different structure from that of the OFDM receiver, as seen in Figure 3.8. The Primary User Filter is a new feature to the NC-OFDM filter. The FFT core is now a FFT core with pruning. Additionally, another data path of received samples is created. As said earlier, this allows for a less distorted version of the received data.

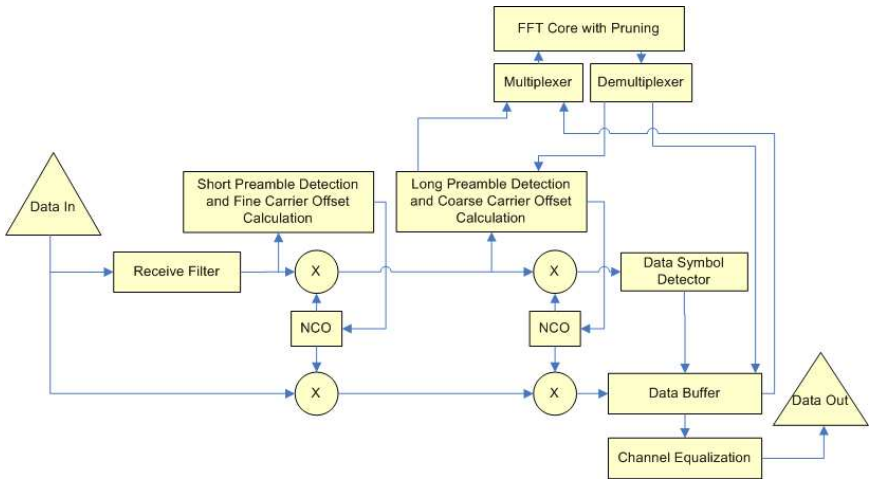


Figure 3.8: NC-OFDM receiver.

The data flow is fairly simple in the receiver. As can be seen in the diagram, there are two parallel data paths coming from the RF frontend. The upper path is used to correct for the CFO and determine the when a data symbol has arrived. The Short Preamble Detection and Fine Carrier Frequency Offset Estimator is constantly checking to see when a NC-OFDM symbol packet is being transmitted. Once this has been detected, the NCO is set to the appropriate frequency and the long preamble can be detected by the Long Preamble Detection and Coarse Carrier Frequency Offset Estimator. This unit utilized the pruned FFT core, but for a full FFT when no assumptions are made. Once an estimate is determined, the second NCO can be set to the appropriate frequency. At this point the CFO is minimized. The Data Symbol Detector can search for the starts of the data symbols as they arrive. The lower path bypasses the receive filter and feed into a buffer. This buffer is locked and the data is sent to the pruned FFT core to be transformed back into the frequency domain. This is then passed to the channel equalizer and then to a carrier demodulator.

3.6 Chapter Summary

NC-OFDM was chosen to allow for secondary unlicensed users to use spectrum that would otherwise be left unused. Spectrum exclusivity is not acceptable when there are increasing demands on needing more bandwidth. NC-OFDM is able to operate in among multiple primary users that are licensed for a given piece of spectrum while minimizing interference with them. A synchronization scheme is need for all communication systems, this chapter proposes one for NC-OFDM. A complete NC-OFDM synchronization algorithm (CFO estimate, data symbol acquisition, and carrier arrangement) has not been proposed before.

Chapter 4

Hardware Architecture for NC-OFDM Transceiver

In the previous chapter, an algorithm was proposed for synchronization of NC-OFDM. This chapter takes that a step further to propose an architecture that allows it to be realized in hardware. Also in this chapter, the pruned IFFT/FFT core architecture is presented.

4.1 FFT/IFFT Pruning in Programmable Logic

In NC-OFDM, the FFT and IFFT are important parts of the transmitter and receiver. The IFFT is used to create the long and short preambles and the data symbols of the transmitter. The FFT is used to find the coarse frequency offset and demodulate the data symbols into the individual carriers. Since not every carrier used, the FFT and IFFT can be optimized to improve execution time and power consumption. This is known as FFT/IFFT pruning [16].

Since a pruned FFT only calculates a subset of the outputs, a pruned FFT should not be performed on the long preamble. This is due to the fact that the exact location of carriers are not known due to a possible frequency offset. Pruning can occur if assumptions on maximum possible frequency offset are made. In the case of data symbols, a pruned FFT should be done. There will be carriers that are not used or carry no useful information,

such as null carriers and cancellation carriers. Cancellation carriers are a special type of carrier used to reduced the side-lobe levels of OFDM and NC-OFDM transmissions [38].

4.1.1 Proposed FFT and IFFT Pruning Algorithm

Since the FFT and IFFT are highly integrated into the transmitter and receiver, a hardware pruned FFT/IFFT algorithm and core are developed. Nearly all research into FFT and IFFT pruning has been done into just software solutions. The main reason for this is that many algorithm use conditional statements, which are not efficiently implemented in hardware. Nevertheless, several recently proposed algorithms do not employ conditional statements [27]. In the paper by Singh *et al.* [30], an algorithm is proposed that can be broken up into stages that can be realized in pipelined hardware architecture. The proposed algorithm works with simple bit-wise logical operations to determine the paths needed to computed from the input of an FFT to the output.

The proposed algorithm, shown in Algorithm 4, uses a technique that is based on the algorithm proposed by [2], which was then further worked upon in [27]. The proposed algorithm makes a simplification by only removing butterflies in the FFT or IFFT that have both inputs that are zero. This results in a regular an algorithm that is simple to implement.

The first stage, consists of lines 2 through 8, the proposed algorithm is to generates a M_i matrix. This matrix lists all computation nodes that need to be calculated and those that do not need to be listed. Since only full butterflies are presently being eliminated, this matrix is then simplified into another matrix that lists which butterflies need to be computed. After this matrix is created, an approach similar to the one presented in [27] is applied. The end result is a matrix with a top row listing how many stages need to be computed for each stage and the columns under the top row listing which butterflies need to be computed called the M_{index} matrix. The final matrix produced by the proposed algorithm is given by:

$$M_{\mathrm{i}} = \begin{pmatrix} 3 & 4 & 4 \\ 2 & 1 & 0 \\ 3 & 2 & 0 \\ 4 & 3 & 0 \\ 0 & 4 & 0 \end{pmatrix} \tag{4.1}$$

where the top row denotes the number of butterflies to be done in a particular stage and the column directly beneath the numbers in the top row denote which butterfly needs to be computed.

This is the algorithm that generates a M_i matrix that that has the same structure of the M_i matrix as seen in algorithm 4. The for loops defined from lines 2 through 8 compute a matrix that the algorithm proposed in [2], which can be seen in Figure 2.12. The outer loop, defined in the nested for loops on lines 12 through 24, iterates through each of the columns of the M_{inew} matrix and starts a counter. The inner loop, from lines 15 to 19, cycle through the rows of a single column which adds to the counter and fills in the appropriate elements in the M_{index} matrix. The if statement can be realized as a conditional move such that there is no branch penalty.

When computing the IFFT, the M_{index} matrix is used as a guide. An outer loop iterates a pre-established number of times, which was determined in the generation of the M_{index} matrix, to go through the first stages of the IFFT. An inner loop iterates around the butterfly calculation portion. The number of times this loop iterates is determined by indexing into the first row of the M_{index} matrix. After the software portion of the IFFT is done, it can be sent to a hardware implementation of the IFFT to finish the IFFT.

Algorithm 5 uses the M_{index} matrix to calculate the FFT. The layout is similar to that of a typical implementation of a FFT with two nested loops indexing through each of the points in the butterfly structure of a FFT. In the proposed algorithm, the for loop defined on lines 5 through 20 index through the M_{index} matrix, with the outer loop indexing through the columns and the inner indexing through the rows of a particular column. The lines from 7 to 10 generate from the M_{index} matrix the pair of points to be calculated and lines 11 through 18 perform the butterfly calculation.

Algorithm 4 Proposed FFT Pruning Algorithm

1: $M_{\text{inew}} = zeros(2^{(M-1)}, M)$
2: **for** $i = 1$ to M **do**
3: **for** $j = 0$ to $2^{(M-1)} - 1$ **do**
4: $pair(1) = 2^{(M+1-i)} * floor(j/2^{(M-i)}) + mod(j, 2^{(M-i)}) + 1$
5: $pair(2) = pair[1] + 2^{(M-i)}$
6: $M_{\text{inew}}(j+1, i) = M_{\text{index}}(pair[1], i) | M_i(pair[2], i)$
7: **end for**
8: **end for**
9: $[n, m] = size(M_i)$
10: $M_{\text{index}} = zeros(2^{(M-1)} + 1, M)$
11: $iter = 0$
12: **for** $j = 1$ to M **do**
13: $index = 2$
14: $M_{\text{index}}(1, j) = 0$
15: **for** $i = 1$ to $2^{(M-1)}$ **do**
16: $M_{\text{index}}(index, j) = i$
17: $index = index + M_{\text{inew}}(i, j)$
18: $M_{\text{index}}(1, j) = M_{\text{index}}(1, j) + M_{\text{inew}}((i, j)$
19: **end for**
20: **end for**
21: **return** M_{index}

Algorithm 5 Proposed Pruned FFT Algorithm

$W = zeros(2^M, 1);$
for $i = 0$ to $2^M - 1$ **do**
  $W(i+1) = exp(\frac{-2*pi*J}{N} * i)$
end for
for $g = 1$ to $iter - 1$ **do**
  for $h = 0$ to $M_{\text{index}}(1, g) - 1$ **do**
    $I = g$
    $J = M_{\text{index}}(h + 2, g) - 1$
    $pair(1) = 2^{(M+1-I)} * floor(J/2^{(M-I)}) + mod(J, 2^{(M-I)}) + 1$
    $pair(2) = pair(1) + 2^{(M-I)}$
    $W_{\text{index}} = mod(J * 2^{(I-1)}, N/2) + 1$
    $WW = W(W_{\text{index}})$
    $x(1) = X(pair(1))$
    $x(2) = X(pair(2))$
    $y(1) = 1/2 * (x0 + x1)$
    $y(2) = 1/2 * (x0 - x1) * WW$
    $X(pair(1)) = y(1)$
    $X(pair(2)) = y(2)$
  end for
end for

4.1.2 FFT Pruning Realization In Programmable Logic

Since the FFT and IFFT are very similar, the individual pruned FFT and IFFT cores are mostly identical. The limits of operation for the IFFT and FFT are 4096 points maximum and 32 points minimum. The FFT and IFFT are simple units that only have one butterfly unit operating at 400 MHz. This puts limits the size of the FFT to be 4096, assuming a 64 MHz DAC rate. This ensures that the FFT and IFFT can operate in real-time. In order for the FFT and IFFT unit to operate, the addresses of the data pair has to be calculated, which comes from the M_{index} matrix. Storing this matrix is impractical since it would require 295,056 bits or 16 BRAMs in a Xilinx Virtex 5, which is while practical, is a large amount of BRAMs to commit to. The Mi_{new} matrix could be dynamically generated for a pruned input FFT or IFFT, but this would not work for output pruning. This means that the Mi_{new} needs to be stored. The storage of a complete stage of M_{index} would be ideal, but would require a significant amount of memory, thus portions of a stage of M_{index} are calculated. The basic flow of data through the FFT and IFFT consists of loading an initialization vector in to the *ivec table*, which is then used to create the Mi_{new} matrix. From there, the portions of the M_{index} matrix are calculated, which is then fed to the address calculator. The address calculator calculated the data addresses for the data to be fed into the butterfly unit. Figure 4.1 shows the overall structure of the pruned FFT core.

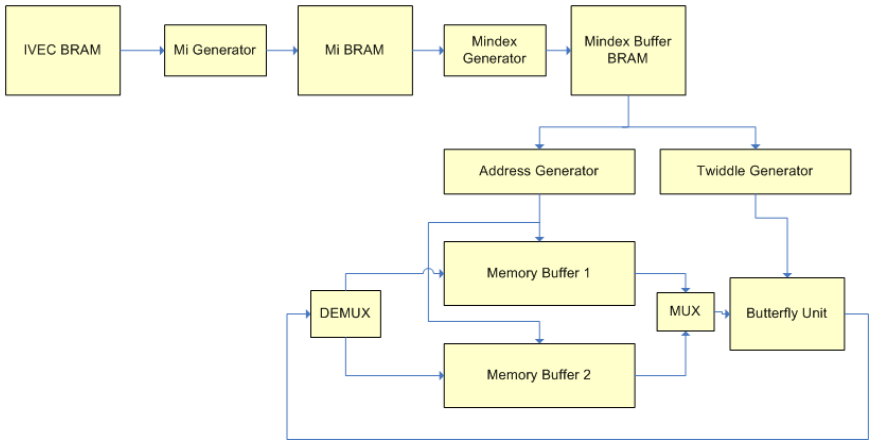


Figure 4.1: Overall structure of the pruned FFT/IFFT core

The the *ivec BRAM* stores an initialization vector that marks which butterflies are needed in the beginning for the IFFT and the end for the FFT, since only a subset of the outputs of the FFT are needed and only a subset of the input to the IFFT are needed. This table stores 8 initialization vectors. This is done since it was expected that there would be fairly small set of carrier allocations that are commonly used and loading new initialization vectors would not have to be every time a block of carriers are activated or deactivated in a NC-OFDM transmission. The first 12 bits determine the size of the FFT or IFFT the vector is for, the rest of the 2048 bits are the IVEC. As with the IVEC in the algorithm, the IVEC stored in memory is an array of zeros and ones corresponding to the first set of butterflies for the IFFT or the last set of butterflies for the FFT. A one signifies that the butterfly must be calculated and a zero signifies the butterfly is not relavent and should not be calculated. All of the initialization vectors are the same size to simplify the indexing through the *IVEC table* by the *MI generator*. Any unused locations are treated as don't care states.From this table, the *MI generator* takes a single initialization vector and generates partial Mi_{new} matrix.

Table 4.1: Performance Using Hard Decision Detection

Ivec (stage 1)	Stage 2	Stage 3	Stage 4
0	1	1	1
0	0	0	1
0	0	1	1
0	0	0	1
1	1	1	1
0	0	0	1
0	0	1	1
0	0	0	1

A '1' signifies that a butterfly has non-zero data for inputs and thus needs to be calculated and a '0' denotes the inputs to that particular butterfly are zero and do not need to be calculated.

It should be noted that that this unit does not generate a full Mi_{new} matrix. When the Mi_{new} matrix is examined, there is redundancy in through out each stage except for the first in the case of the FFT and the last in the case of the IFFT. Table 4.1.2 shows an

example of this based on a small 16 point IFFT Mi_new matrix. By definition, the first stage is the 8 bit initialization vector. The second stage is comprised of two 4 bit vectors that are identical, the third stage is four 2 bit vectors, and eight 1 bit vectors. This is redundant information that does not need to be calculated multiple times. The Mi_{new} matrix BRAM is capable of storing a complete Mi_{new} matrix. The final two stages are assumed to contain no pruned butterflies to simplify design and based on experimentation, which has shown that the pruning quickly diminishes after the first several stages unless very few carriers are used. As the generation of Mi_{new} progresses, the calculated vector is halved until it reaches a length of 32 bits, thus the lower limit of 32 on the FFT and IFFT size. The final stages all generate 32 bit vectors, which was dictated on how portions of Mi_{index} is generated.

The butterfly must be fed with data every cycle, except at the end of a stage within the FFT or IFFT. This requires the generation of M_{index} to be complete, or at least appear to be from the perspective of the butterfly address generation unit. In a software implementation of this algorithm, M_{index} can be generate one term at a time. This can not be done in hardware since it is likely there will long string of zeros, which would result in a stall in the butterfly pipeline. This means that portions of M_{index} must be calculated in parallel. The first step in calculating the M_{index} matrix is to read in two 32 bit chunks of data broken into eight 8 bit chunks. These 8 bit chunks are fed into a look up table to determine a how many butterflies are used in the section and which butterflies need to be calculated. The structure is similar to that of M_{index}. A sample input to the look up table and the corresponding output is:

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 6 \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 0 \\ 0 \end{pmatrix} \tag{4.2}$$

Figure 4.2 shows the structure of this portion of the pruned FFT/IFFT unit.

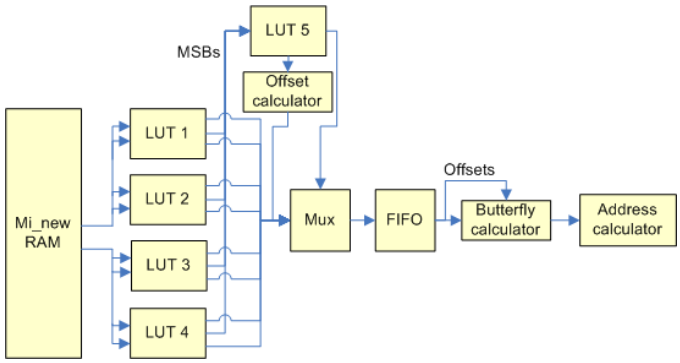


Figure 4.2: Partial M_{index} calculation unit

There are four dual port look up tables that are used to calculate eight 9x3-bit vectors. A look up table approach was used to provide a simple, fast, and low latency conversion method of converting a portion of M_{inew} to a portion of M_{index}. It is expected that there will be times in which the vector is comprised of all zeros. This would create a stall on the butterfly unit. To remove these, an additional look up table that is identical to the other four is used to assist in removing vectors that are all zero. The non-zero vectors are fed into a buffer. From this buffer, the vector is read into the first part of generating the data addresses for the butterfly unit. The first bin is used as a limit to how many bins needs cycled through. The output is the sum of the previous first bins and eight times the number

of zero vectors removed. The output is the number of the butterfly in a particular stage that need to be calculated. This is used as input to the address calculation:

$$\text{address}_1 = 2^{log_2(N)-stage+1} * floor\left(\frac{butterfly}{2^{log_2(N)-stage}} + mod(butterfly, 2^{log_2(N)-stage}\right), \quad (4.3)$$

$$\text{address}_2 = \text{address}_1 + 2^{log_2(N)-stage}. \quad (4.4)$$

The butterfly unit is a simple unit comprised of two complex adds, one complex multiplication, and a hardwired bit shift in the case of IFFT. The latency of the adder is 2 cycles and the multiplier is 4 cycles, for a total latency of 6 cycles. Figure 4.3 shows a detailed representation of the complex adder. Figure 4.4 shows the structure of the complex multiplier. Figures 4.5 and 4.6 show the the overall structures of the FFT and IFFT butterfly units respectively.

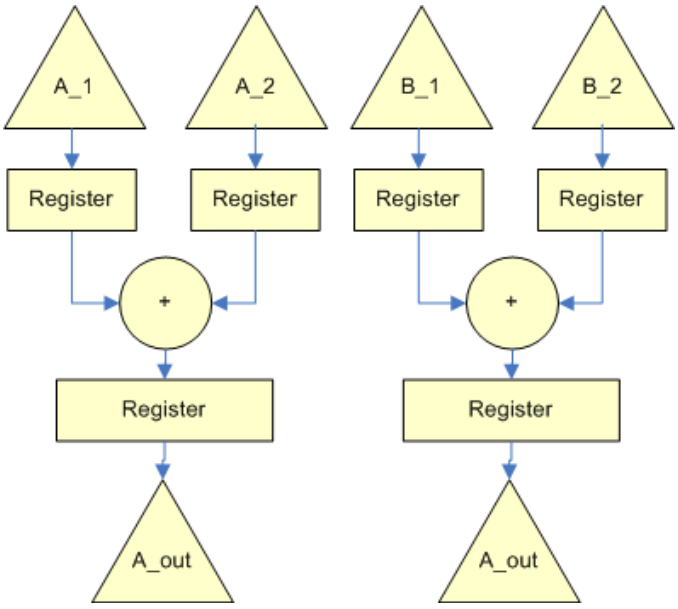


Figure 4.3: Complex adder.

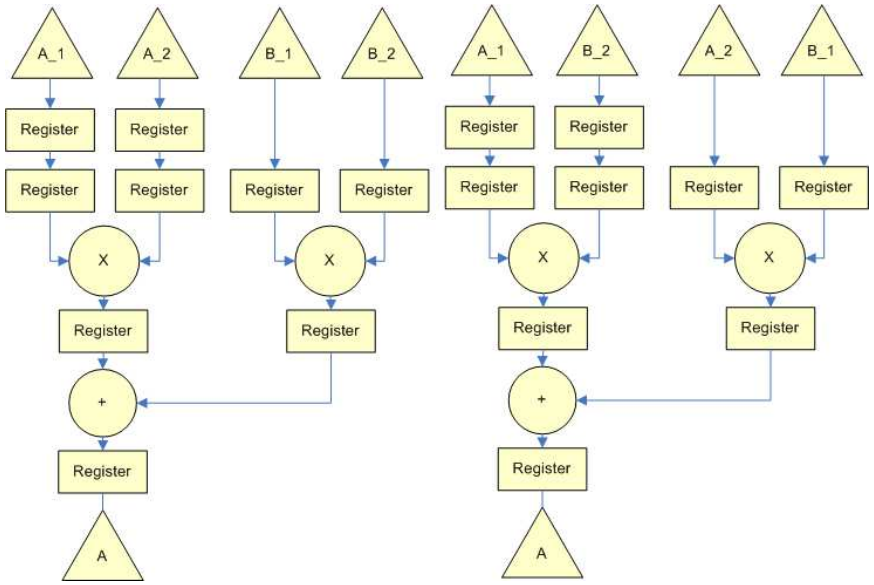


Figure 4.4: Complex multiplier.

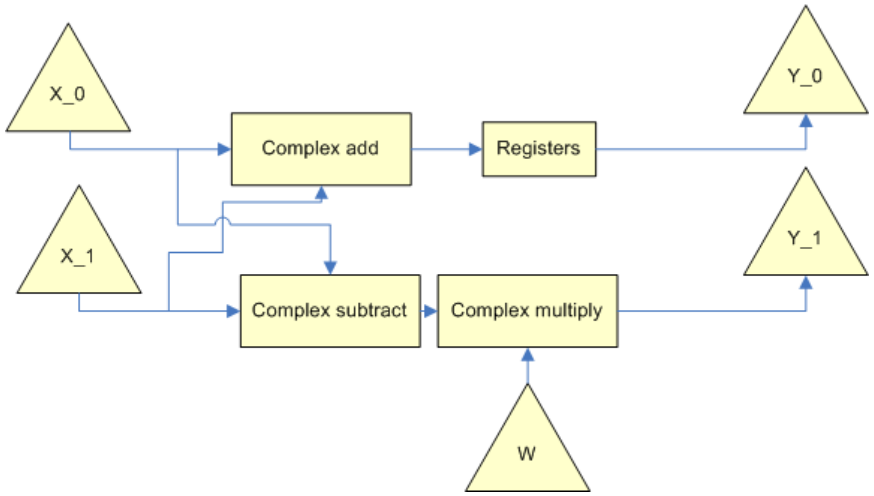


Figure 4.5: FFT butterfly unit.

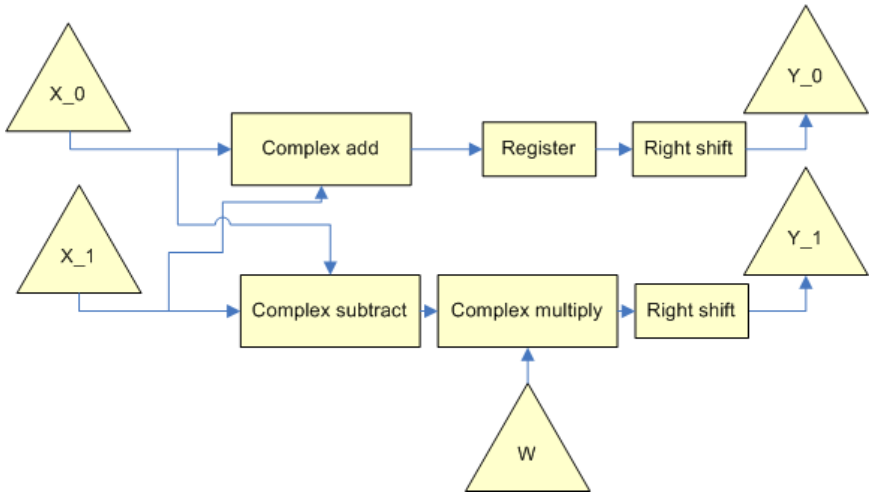


Figure 4.6: IFFT butterfly unit.

The butterfly unit is connected to two dual port BRAMs. For one stage, the first BRAM has the data from the previous stage and the second BRAM is used to store the newly calculated data. In the next stage, the roles of the BRAMs are reversed, the second BRAMs is now used as the input to the butterfly unit and the first stores the output of the butterfly unit. In order the keep output addressing simple, the addresses are fed into a delay line of 6 cycles. ¡check on cycles¿ This can be seen in the lower part of figure 4.1.

4.2 Complex Finite Impulse Response Filter

The receiver requires a complex FIR filter to filter out the primary user. Should a filter be needed on the transmitter side, a complex FIR filter would be needed. There are multiple methods of realizing a FIR filter in hard of an FPGA. The three basic approaches are: sequential FIR filters, semi-parallel FIR filters, and parallel FIR filters. Figure 4.7 shows the structure for a sequential filter. Figure 4.8 shows the structure for a semi-parallel filter. Figure 4.9 shows the structure of a parallel FIR filter.

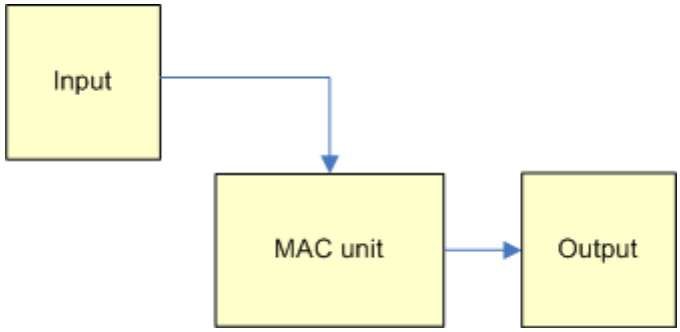


Figure 4.7: FIR sequential filter structure

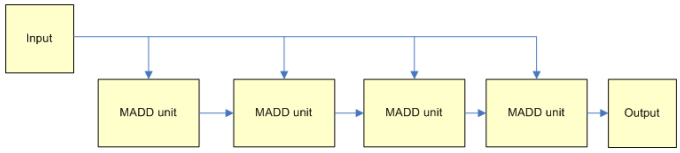


Figure 4.8: FIR parrallel filter structure.

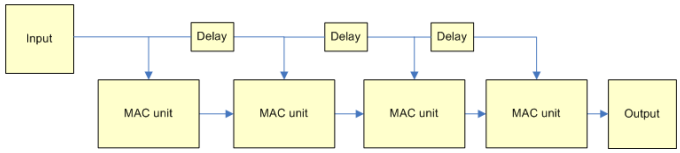


Figure 4.9: FIR semi-parrallel filter structure.

From the MATLAB simulations, higher order complex FIR filter results in a tighter drop off in the transition band and have a flatter response. Both of these qualities are important for the primary user filter. This results in a higher resource usage in the FPGA. This made the semi-parallel filter structure the best choice considering resource usage and a reasonable target frequency. Each unit will perform six FIR tap calculations per a sample clock tick. Since the sampling clock is running at 64 MHz, the target frequency is 384 MHz. This effectively puts the DSP slice usage at one sixth that of the filter order rounded up to the nearest integer. When look at the coefficients of the complex FIR, it can be seen that it is complex symmetric around the middle tap.

As can be seen in the above table, the real components of a sample FIR filter are

conjugate symmetric about the twenty fifth term. This allows for a further optimization of the complex semi-parallel FIR filter. The semi-parallel approach can be combined with a symmetric systolic FIR filter to exploit this conjugate symmetry. Figure 4.10 shows the structure of a symmetric systolic FIR filter.

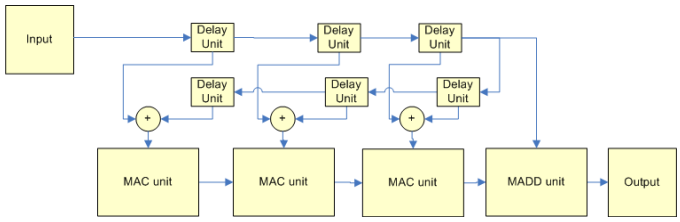


Figure 4.10: Symmetric FIR filter structure

Equation (4.5) shows the original equation for the FIR filter. Equation (4.6) shows a rearranged form of the Equation (4.5). In these equations, y[] is the output, x[] is the input, and b[] is the filter coefficients

$$y[n] = \sum_{i=0}^{2*m} x[n-i] * b[i] \tag{4.5}$$

$$y[n] = x[n-m] * b[m] + \sum_{i=0}^{m-1} (x[n-i] + x[2*m-n+i]) * b[i] \tag{4.6}$$

In the case of the complex FIR filter, the first equation is shown in Equation 4.7 and the second rearranged equation is shown in Equation 4.8.

$$y[n] = \sum_{i=0}^{2*m} (real(x[n-i]) + imag(x[n-i])) * (real(b[i]) + imag(b[i])) \tag{4.7}$$

$$\begin{aligned} y[n] = {} & (real(x[n-m]) + imag(x[n-m])) * (real(b[m]) + imag(b[m])) + \\ & \sum_{i=0}^{m-1} ((real(x[n-i]) + real(x[2*m-n+i])) * real(b[i]) + \\ & (imag(x[n-i]) - imag(x[2*m-n+i])) * imag(b[i])) + \\ & ((real(x[n-i]) - real(x[2*m-n+i])) * imag(b[i]) + \\ & (imag(x[n-i]) + imag(x[2*m-n+i])) * real(b[i])) * j \end{aligned}$$

By doing this optimization, a one complex multiplication is saved and one complex add is no longer needed to be done. The trade off is the that pre-adders to the DSP slices are needed. The resource usage for one FIR unit is four pre-adders and four DSP slices. Figure 4.12 shows structure of the basic FIR filter sub-unit. The middle term is unique from the the coefficients. Figure 4.11 shows the structure of the middle FIR filter sub-unit. Figure 4.12 shows the complete complex FIR sub-unit with coefficient RAM and delay RAMs of the sample data. Figure 4.13 shows the complete complex FIR middle sub unit.

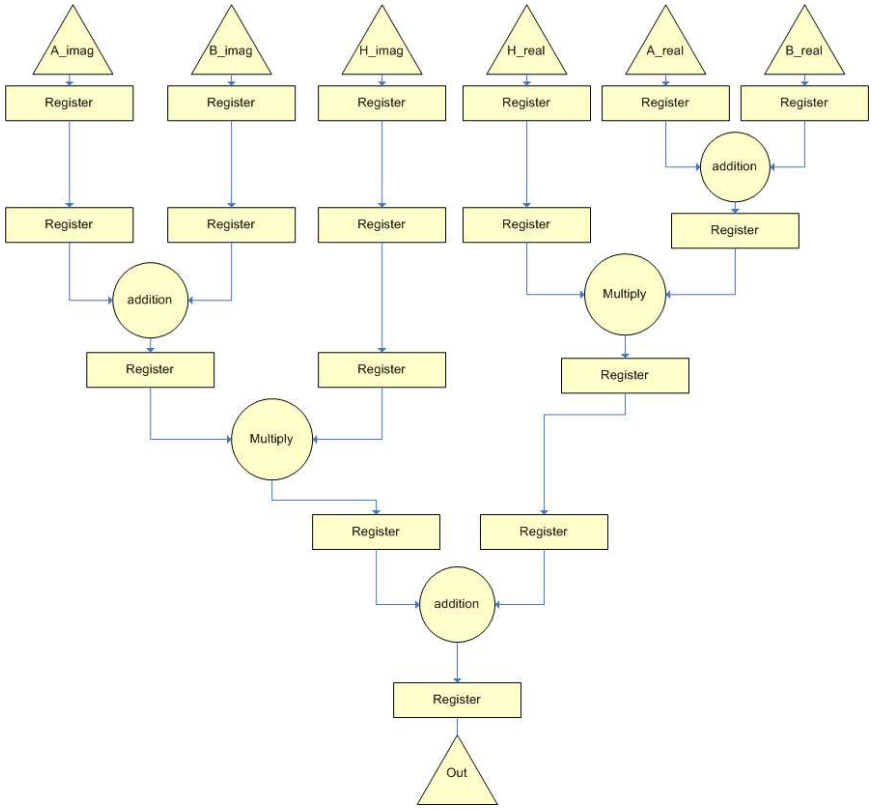


Figure 4.11: Structure of complex FIR block with registers.

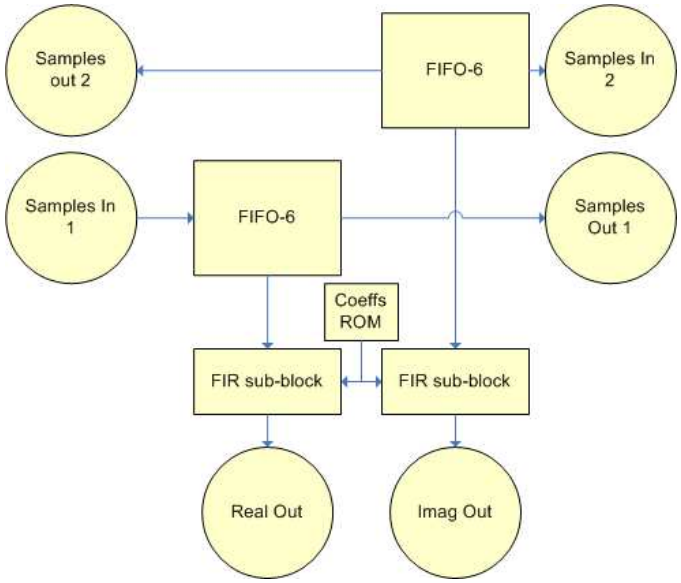


Figure 4.12: Basic complex FIR with BRAMs.

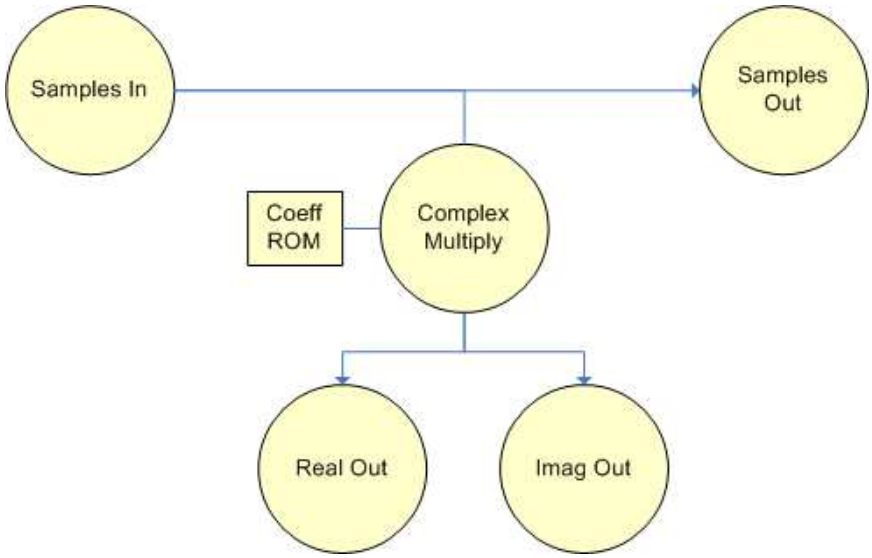


Figure 4.13: Structure of middle complex FIR block with registers.

With these two basic units, any size complex or real coefficient symmetric FIR filter can be realized. Since the basic FIR sub-unit does six tap calculations per a sample clock cycle and conjugate symmetry is exploited, an optimal tap number is 12*n+1, where n is a

positive integer. A non-optimal order would leave zeros on the ends, which are treated the same as a non-zero number.

4.3 Transmission Side

The transmission side of the NC-OFDM is much simpler than the receiver since it lack the synchronization components and the channel equalizer. The basic units of the transmitter are the data symbol creator, pruned IFFT unit, preamble creator, and the transmission filter. Figure 4.14 shows the overall structure for the transmitter.

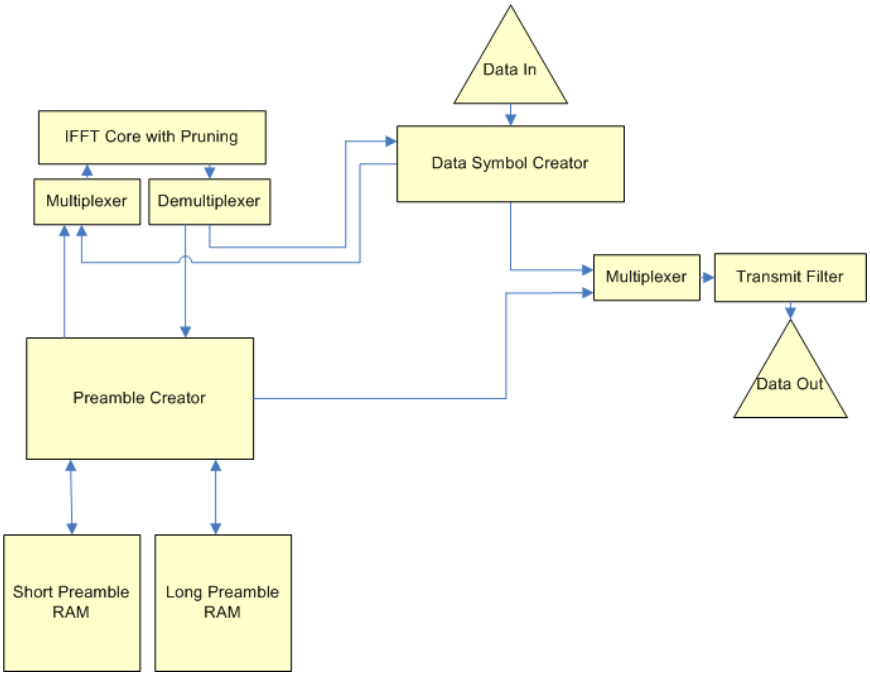


Figure 4.14: NC-OFDM transmitter

At the start of a transmission, the computer, or other controller, tells the preamble creator to send the short and the long preambles out. Once this has been done, the data symbols can be transmitted. The data symbol creator accepts raw constellation points, it does not know the difference between data, pilot, null, or cancellation carriers. The pruned IFFT core is able to differentiate between null and non-null carriers. The constellation points are chosen in software. This is different from other OFDM transmitter, which would

have hardware that maps coded data to constellation points. This was not done since there are cancellation carriers that can be any value and other techniques may shift the constellation points to improve other performance metrics of the transmission. A modified approach could have been done in which the data carriers are mapped by a special look up table. The data symbol send the data to the pruned IFFT core, waits for the data to be transform and receives it back from the pruned IFFT core. The data symbol creator then starts sending the data to the transmit filter starting with the cyclic prefix followed by the complete data symbol.

4.3.1 Preamble Creation

Since it is expected that the usable spectrum will not change between every transmission burst, the preambles should be stored, since there is only one possible carrier allocation for the carriers. Figure 4.15 shows the structure of the preamble creator core.

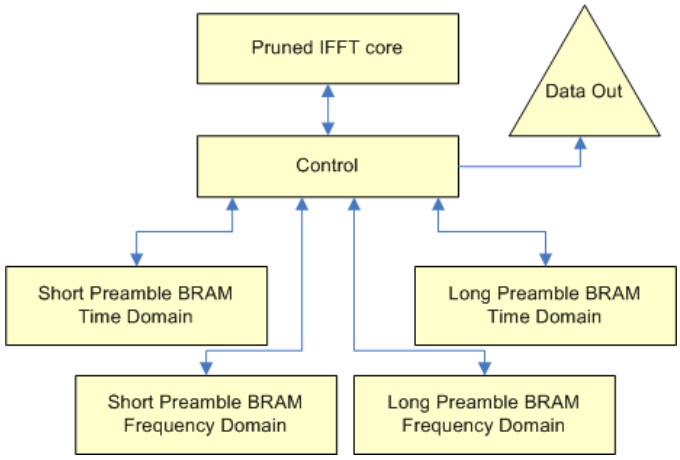


Figure 4.15: Preamble Creator

When a new preamble is needed to be created, the raw carrier information is loaded in the short and long preamble BRAMs. Once the data load is complete, the data is sent to pruned IFFT core to be transformed into the time domain. In this case, the IFFT is a pruned IFFT. The data is held in the two BRAMs until unstructured to begin a NC-OFDM transmission burst.

4.3.2 Data Symbol Creation

The data symbol creator is another simple unit. Figure 4.16 shows the structure for the data symbol creator.

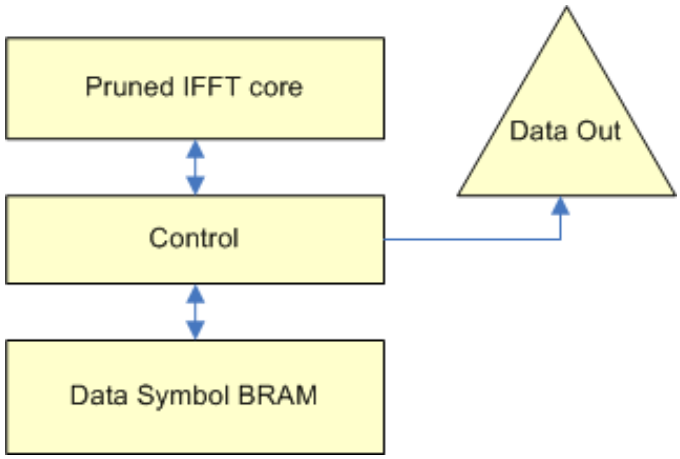


Figure 4.16: Data Symbol Creator

The data is loaded into the BRAM. Once the load is complete, the data is sent to the pruned IFFT core to perform a pruned IFFT. The data is read back into the BRAM. The next step is to send the cyclic prefix then the data.

4.3.3 Transmission Filter

It is expected that there will be a transmission filter, and thus has been shown in the overall NC-OFDM transmitter structure. The transmission filter is capable of reducing the effects of a non-flat frequency response of the RF-frontend. The transmission filter could also be used for other purposes such as side-lobe reduction.

4.4 Receiver Side

4.4.1 Receive filter

The receive filter is an important part of the receiver. This filter reduces the presence of the primary user, but also can be combined with channel filter or a filter to correct

for a non-flat radio frequency fronted transfer function, provided it does not disturb the conjugate symmetry. Figure 4.17 shows the complete structure of the receive filter.

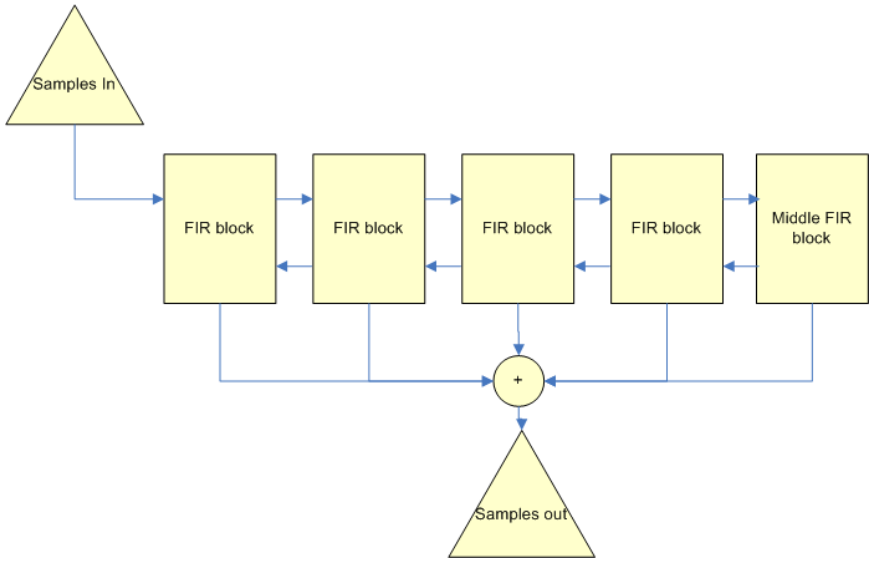


Figure 4.17: Receiver filter

4.4.2 Fine Carrier Frequency Offset correction

The short preamble has two purposes, the first is to provide an initial estimate of the fine carrier frequency offset and then to signify that a transmission is being sent. Figure 4.18 shows the entire structure of the fine CFO estimator.

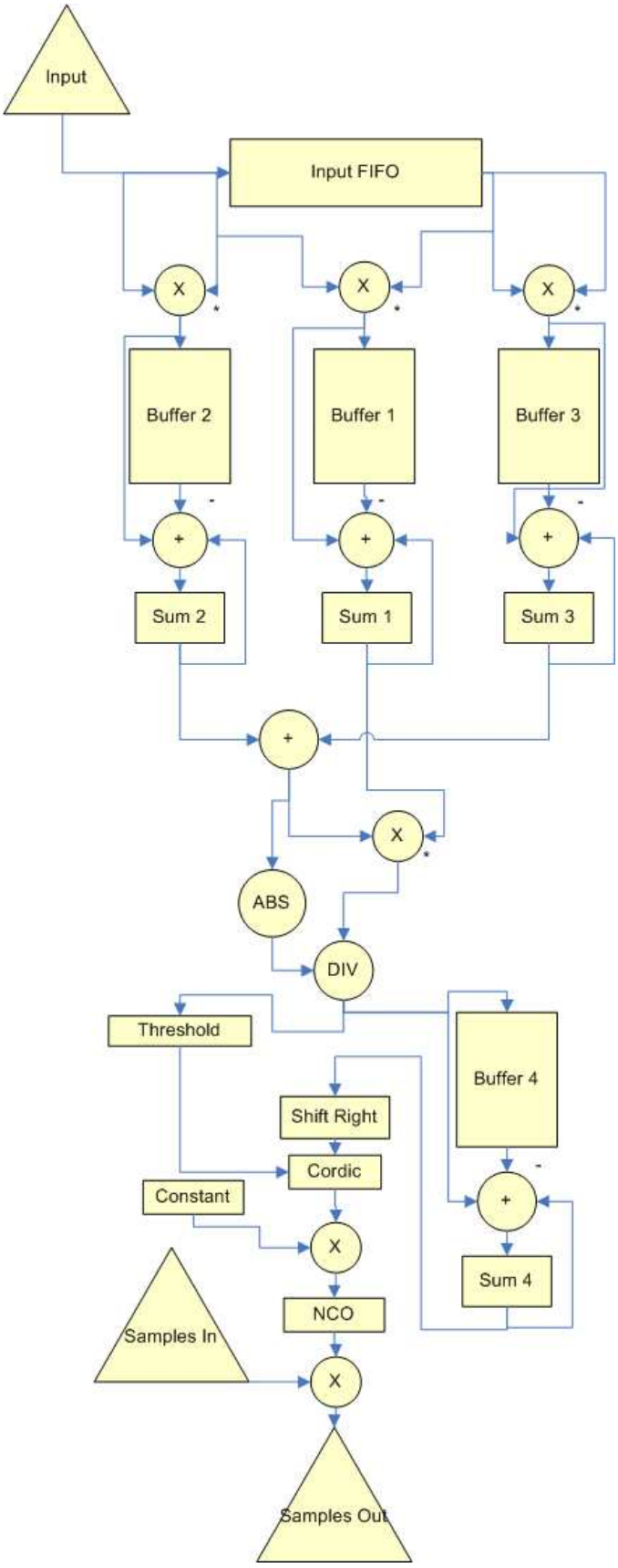


Figure 4.18: Fine carrier frequency offset estimator and mixer

The first portion of this unit is comprised of three similar configuration of a complex multiplier, buffer, and complex accumulator centered around a buffer of the input data. In order to efficiently calculate the autocorrelation and the energy terms, Equation (3.5) can be modified to become:

$$P(d) = P(d-1) + r_d^* r_{d+L} - r_{d+L}^* r_{d+2L}. \tag{4.8}$$

This concept is used to implement the three units that create the ¡numerator¿ and the gain correction term. These units are fed to a divider to created a normalized function that can be used to estimate the fine CFO. A complex division is implement as one complex multiply, two real multiplies, and two real divisions.

The normalized output is continually check to see if it is about a predefined threshold and being fed into a accumulator that contains the sum of the most recent N/4 samples. Once the threshold has been reached, the a flag is set to signify the start of transmission burst. Once a transmission burst has been detected, the accumulator is sampled N/4 samples after. The output of the core is shifted left log2(N/4) bits ¡need special math stuff¿ to form an average of the previous N/4 samples of the normalized output. The plateau that is characteristic to the normalized output is only flat when there is no noise. Since there will be noise in any communication, an average of the output is taken to form a more accurate estimate of the fine CFO.

Once the average has been taken, the estimate is fed into a cordic core to take the arctangent to find the angle of the complex number. This is then multiplied by a constant loaded from the computer, which is calculated using Equation (2.8). The output is the fine CFO estimate. This is loaded into the NCO which is the local oscillator to a complex mixer.

4.4.3 Coarse Carrier Frequency Offset Correction and Data Symbol Detection

The coarse CFO and data symbol detection units are combined together to optimize resource usage. The detection of the cyclic prefix signifies the start of the long preamble and the start of a data symbol. The arrival new data symbols can be used provide a tracking

estimate of the CFO. It is expected the CFO will change slightly over time, more so during the startup of the receiver or transmitter radio. The tracking of the CFO during the a transmission burst should be done in the case of long transmissions and large number of carriers. Figure 4.19 shows the complete structure of the combined coarse CFO estimator and data symbol detector.

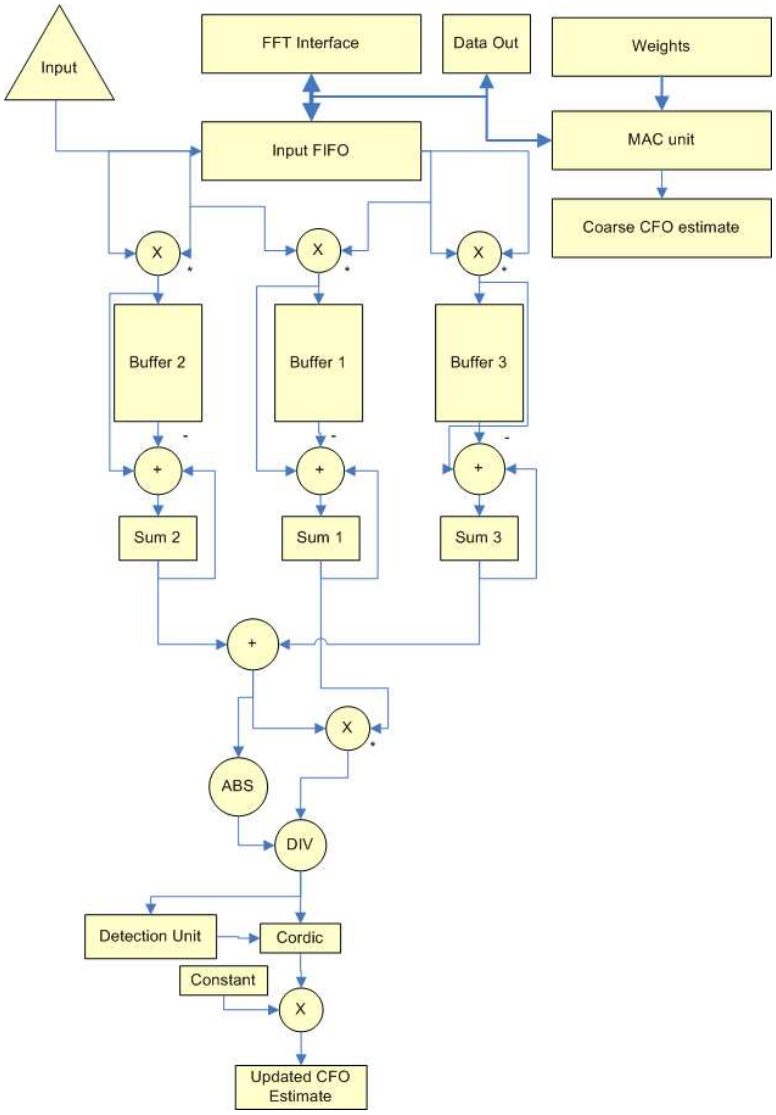


Figure 4.19: Coarse frequency offset estimation and CP detection

The the structure is similar to that of the fine frequency offset since it also relies on

autocorrelation, but differs since the raw data is still important. In order to detect a symbol, the CP needs to be detected, since it is the same as the end of the symbol without noise or a time varying channel. The detection of the highest peak is critical to estimating the start of the OFDM symbol. From the MATLAB simulations, there can potentially be local peaks caused by noise. In order to detect the highest peak, a simple offset and maximum value comparator is used. The exact value is not known, but is known to occur above .9. Once the autocorrelation function has exceeded this set value, the comparator value can be initialized and the offset initialized to zero. The next sample of the autocorrelation is compared to the previous sample. If the value is greater than stored value, the stored value is updated and the offset value is set to zero, else the offset is incremented by one. This process continues until the autocorrelation has dropped below the threshold of .9. This ensures that a the highest peak was detected and not a local peak. The offset is used to tell the start location of the symbol within the buffer. After this is done, the data is symbol is sent to the FFT and IFFT core for demodulation

In the case of the long preamble, the demodulated symbol is then used to estimate the coarse carrier frequency offset. This done by a direct implementation of the method described in the the previous chapter. For the case of data symbols, the demodulated symbol is fed to a channel equalizer and then to channel decoding.

4.5 Chapter Overview

Along with Chapter 3, a complete NC-OFDM synchronization algorithm and transceiver pair is presented. The transmitter and the receiver sub-units are detailed to show one possible implementation method. Additionally, the architecture for the pruned FFT/IFFT core presented. This unit has the capability to performing pruned FFT/IFFTs at the expense of increased resource usage and complexity. An increase in complexity is expected, which can be seen in the software based pruned FFT algorithms.

Chapter 5

Algorithm Benchmarking

This chapter presents the benchmarking of the algorithm showing how well it performs in various channels.

5.1 MATLAB simulation

The MATLAB simulator of the NC-OFDM communication systems is broken into three portions: the transmitter, channel, and receiver. Each of these section is controlled by a number of parameters and are further broken into several individual files. This allows for a great range of flexibility for running simulations.

In NC-OFDM transmission, not all of the carriers are used since some carriers would interfere with the transmissions of primary users. For the simulations, used carrier blocks of size 11 to 11 + 15*floor(N/256) carriers were spaced a random distance between 10*floor(N/256) and 20*floor(N/256) carriers. The carrier arrangements did not change between trials. In the case when the channel is a multi-path channel, the carriers and channel will vary for a trial. This results in a random chance for the used carriers in the NC-OFDM transmission to have the chance to be in a relatively constant part of the channel, but also to exist valley in magnitude in the frequency domain. It is not ideal to transmit within these valleys, especially in relative deep valleys, but avoiding a valley entirely can drastically reduce the bandwidth. The decision to transmit on this null or to allocate more power to better channels is something that can be further investigated.

5.1.1 Performance Metrics

In order to show the performance of the NC-OFDM transceiver, *bit error rate* (BER) curves are shown for each channel possibility. Additionally, the initial carrier frequency offset estimate and tracking based on the received data symbols is also shown and precision of acquiring the start of the data symbol is presented.

5.1.2 Simulation Overview

The channels being presented for simulation are based off an AWGN channel with no primary users and a flat channel response. To this, the channel is changed to become a multi-path channel and the primary user is added to an AWGN channel with a flat channel to form two additional channels. The final channel is based on an AWGN channel with primary users and a multi-path channel. The size of the FFT/IFFT for simulations will be 1024, 2048, and 4096. The individual modulation scheme for the carriers will be 4-QAM, 16-QAM, and 64-QAM. The same modulation scheme and power levels will be used for all carriers, regardless of response of the channel.

A series of primary users are inserted in between each cluster of carriers used by the secondary user. Each primary user is OFDM in nature with a small random frequency shift to ensure that the zero crossings of the individual sinc pulses have the potential to interfere, in a worst case scenario, with the secondary user transmission. Each primary user sends out data symbol waits a random amount of sample between zero and fifty and then sends another symbol. Additional, a spectral mask is applied to each of primary users to ensure that the side-lobes of the transmissions are down to -60 dB where the secondary user is transmitting.

Table 5.1.2 shows the operating parameters used for the simulations

Table 5.1: Performance Using Hard Decision Detection

Parameter	Value
Simulated CFO	0
Cyclic Prefix Size	N/4
Sampling Frequency	64 MHz
Transmission Bandwidth	32 MHz
BER Error Bar Confidence Level	95%
Number of Data Symbols	5
Maximum Inter-Pilot Carrier Spacing	5 Sub-Carriers
Paths in multi-path Environment	10

A random frequency could have been chosen for the simulations, but would not have an appreciable effect on the results. The error in estimating the CFO is caused by the noise, channel, and primary users.

5.1.3 OFDM Operation with Primary Users

In order to establish the need for change in a OFDM transceiver, the performance of an OFDM transceiver is compared to another OFDM transceiver, but operating with no changes in a channel with multiple primary users. The operation of OFDM with Primary Users was done be deactivating a large section of carriers in the middle of the spectrum. No other changes were made to the receiver. Figure 5.1 shows the BER performance for a OFDM transceiver and Figure 5.2 shows the BER performance for the OFDM with primary users present for N=4096.

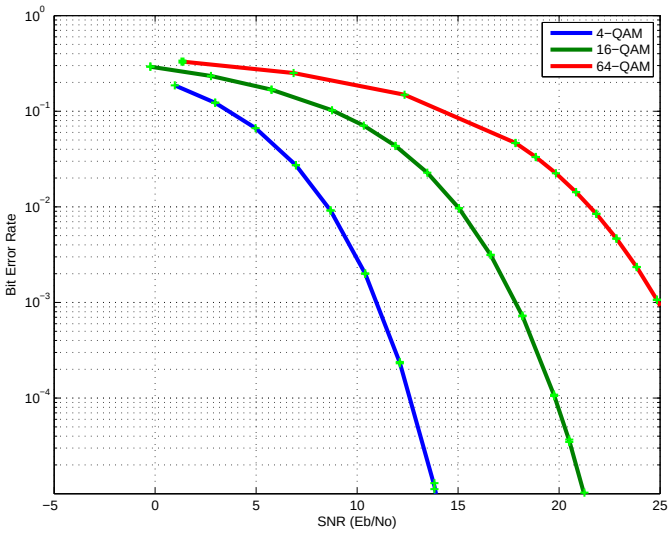


Figure 5.1: BER curves for N=4096

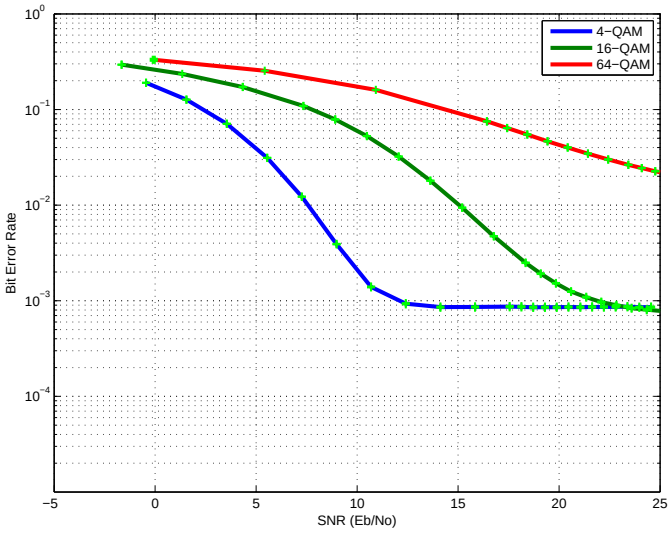


Figure 5.2: BER curves for N=4096 with Primary Users

As can be seen in the above graphs, the OFDM transceiver's BER graph has the characteristic waterfall appearance for a AWGN curve. In the case of the OFDM with primary users, the BER performance worsens for lower SNR values and has a waterfall curve appearance in the higher noise levels. The performance is also dependent on the what the primary

user is transmitting at a particular moment in time. A series of OFDM data symbols does not have the same effect as a repeated signal, like a OFDM preamble, does when it comes to triggering the detection of start of a secondary user transmission. These graphs should be compare with each other only. In the next several sections, the proposed NC-OFDM transceiver is tested in various channels.

5.1.4 Additive White Gaussian Noise Channel

The additive white noise channel provides a baseline to compare to for other channels. Figures 5.3, 5.4, and 5.5 show the results for BER simulation.

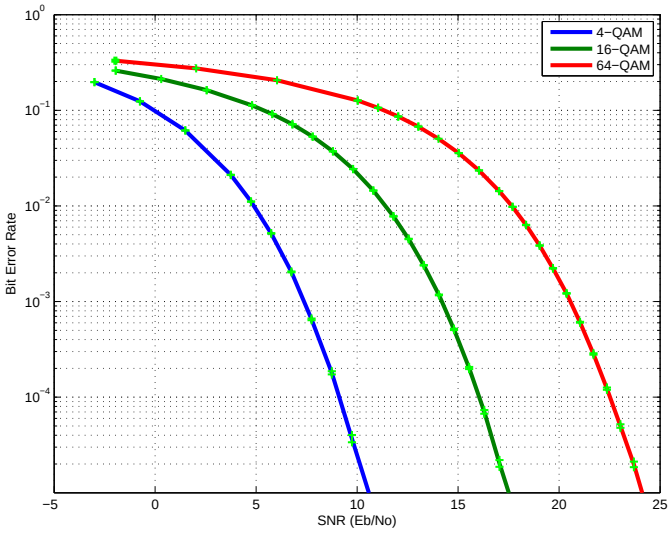


Figure 5.3: BER curves for N=1024

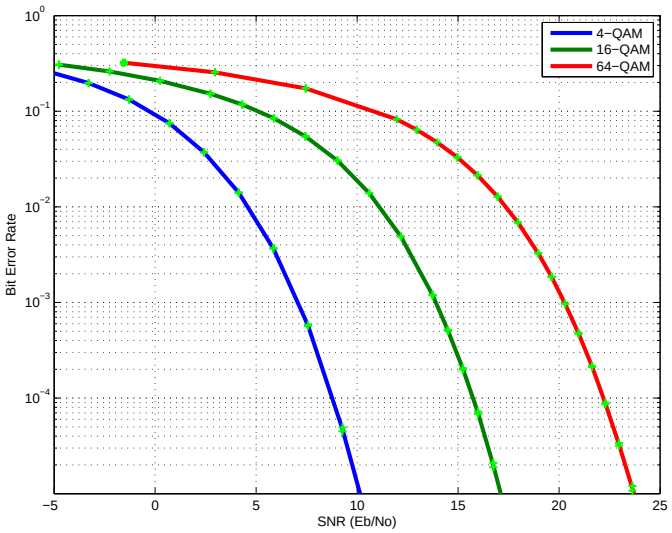


Figure 5.4: BER curves for N=2048

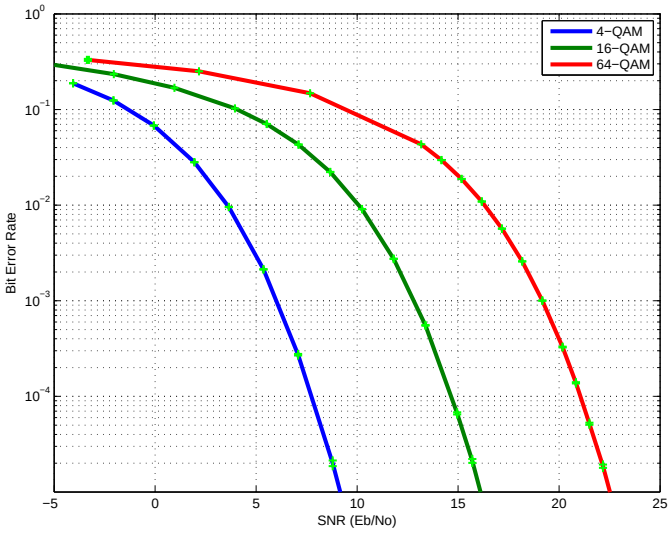
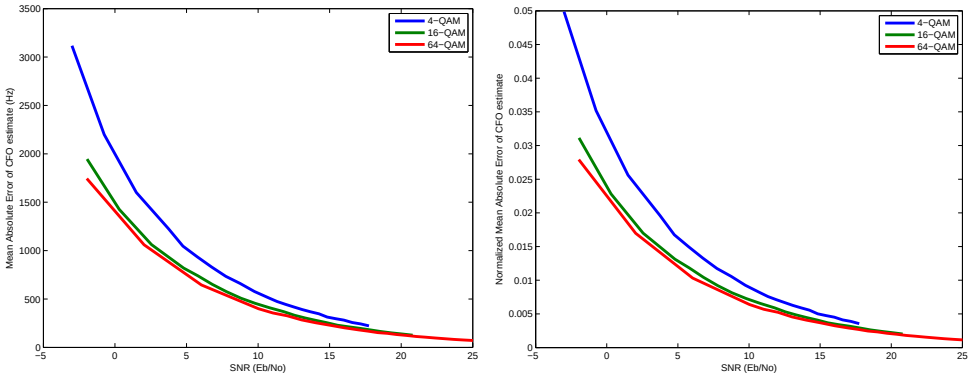


Figure 5.5: BER curves for N=4096

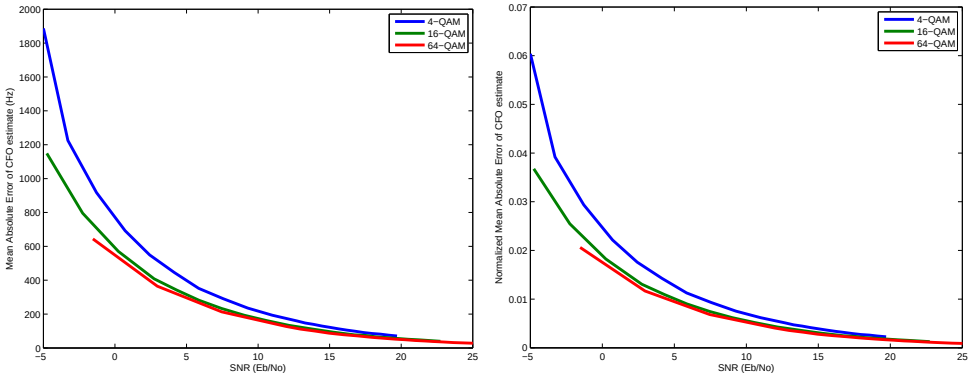
As expected, the bit error rate graphs for the AWGN channel are all similar in appearance and resemble the theoretical waterfall appearance. These form a basis to compare for the the channels with primary users and a multi-path channel. When comparing these graphs, it can be seen that the BER improves as the number of carriers increases. This is

caused mostly by an improving CFO estimate and CFO tracking. The improvement in the initial CFO estimate can be seen in Figures 5.6, 5.7, and 5.8 for the absolute mean error and Figures 5.9, 5.10, and5.11.



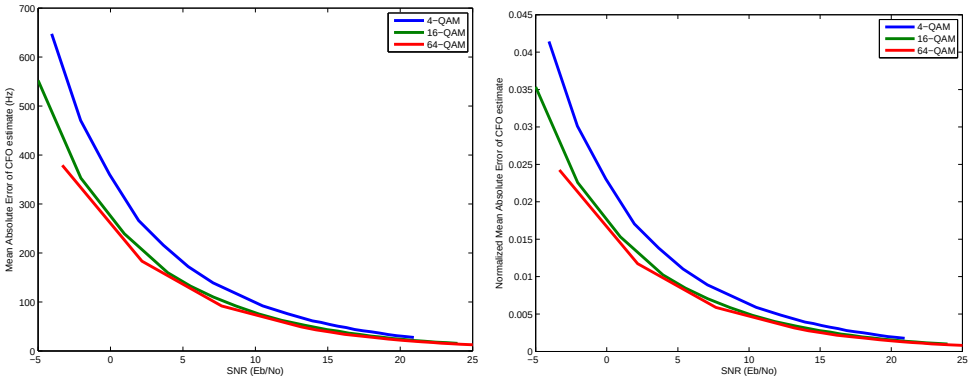
(a) Absolute (b) Normalized

Figure 5.6: CFO initial estimate mean for N=1024



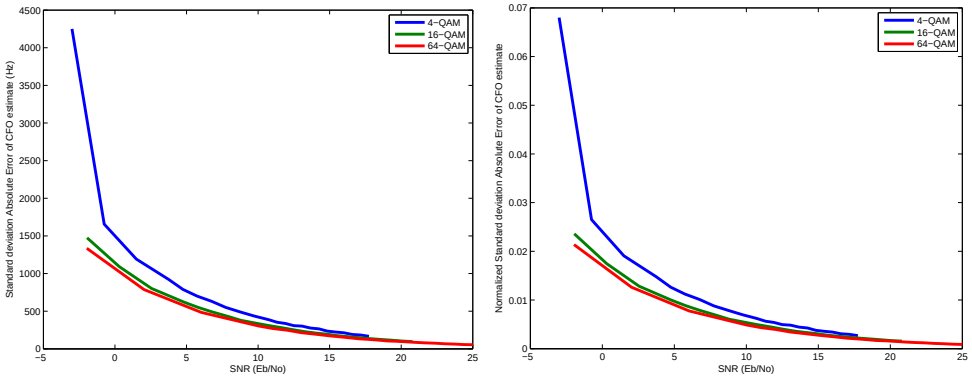
(a) Absolute (b) Normalized

Figure 5.7: CFO initial estimate mean for N=2048



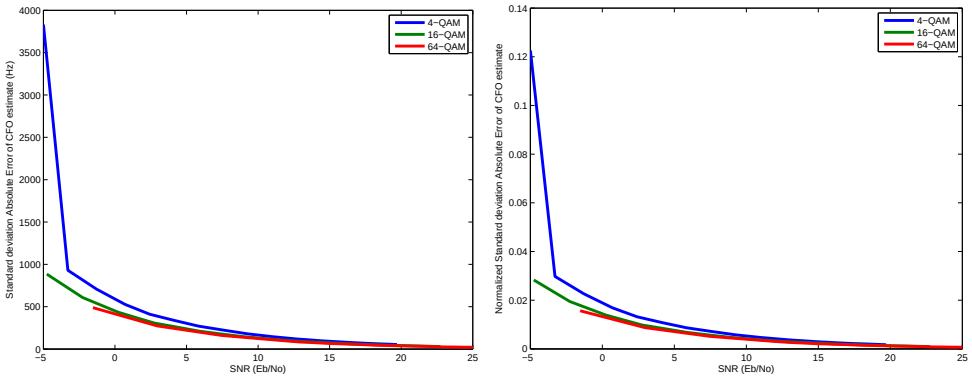
(a) Absolute (b) Normalized

Figure 5.8: CFO initial estimate mean for N=4096



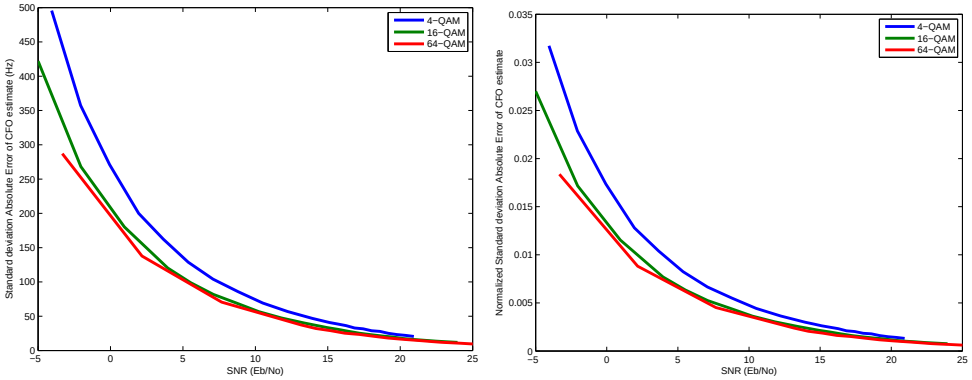
(a) Absolute (b) Normalized

Figure 5.9: CFO initial estimate standard deviation for N=1024



(a) Absolute (b) Normalized

Figure 5.10: CFO initial estimate standard deviation for N=2048

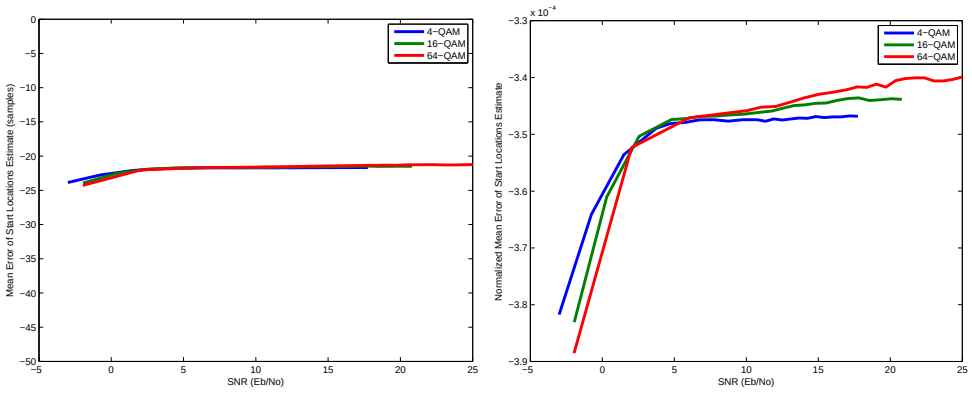


(a) Absolute (b) Normalized

Figure 5.11: CFO initial estimate standard deviation for N=4096

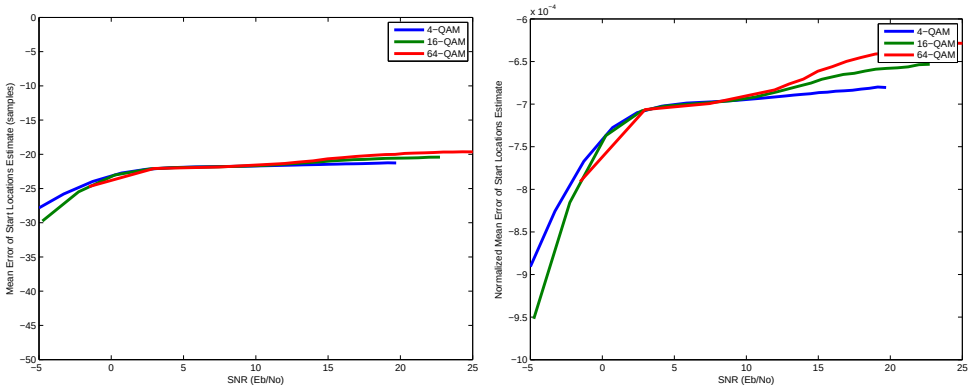
When comparing these graphs, the absolute mean error improves as the number of carriers increases. This suggests that the as the number of carriers increases, the CFO estimate improves. This is the case in term so of frequency, but when the CFO estimate is normalized to the inter carrier spacing, it changes slightly. When normalized, the CFO does improve, but not as much suggested when looking at the non-normalized graphs.

The accuracy of detecting the individual data symbols is another important benchmarking point for synchronization. Figures 5.12, 5.13, and 5.14 show absolute mean error and Figures 5.15, 5.16, and 5.17.



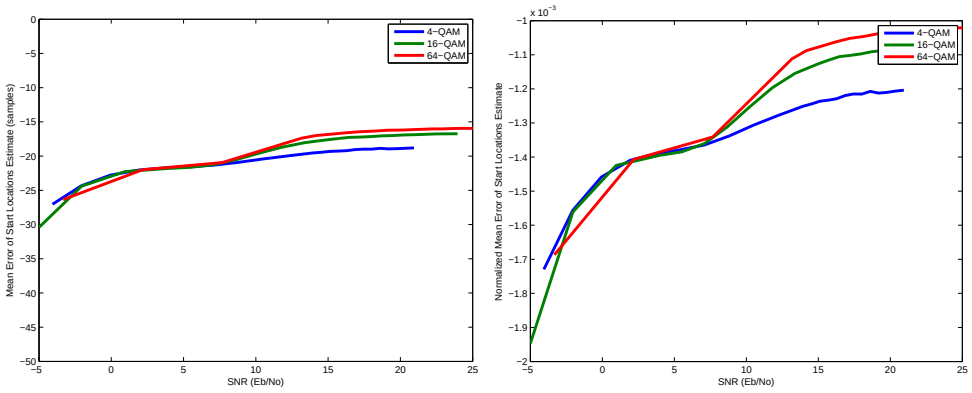
(a) Absolute (b) Normalized

Figure 5.12: Data acquisition mean absolute error for N=1024



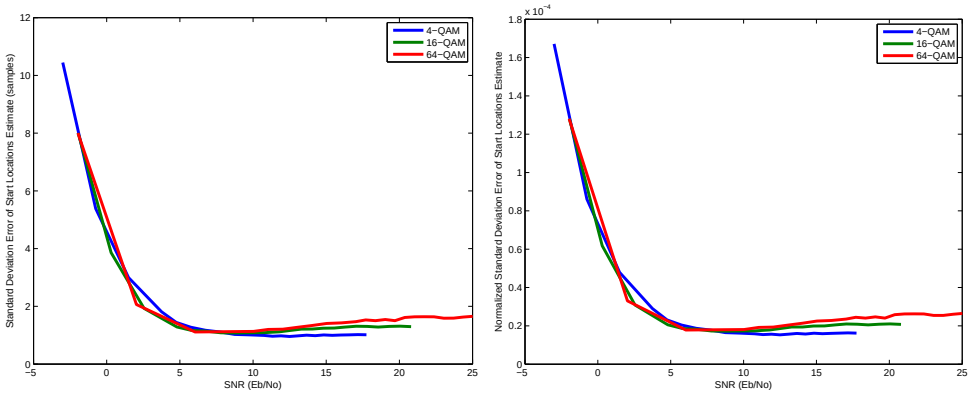
(a) Absolute (b) Normalized

Figure 5.13: Data acquisition mean absolute error for N=2048



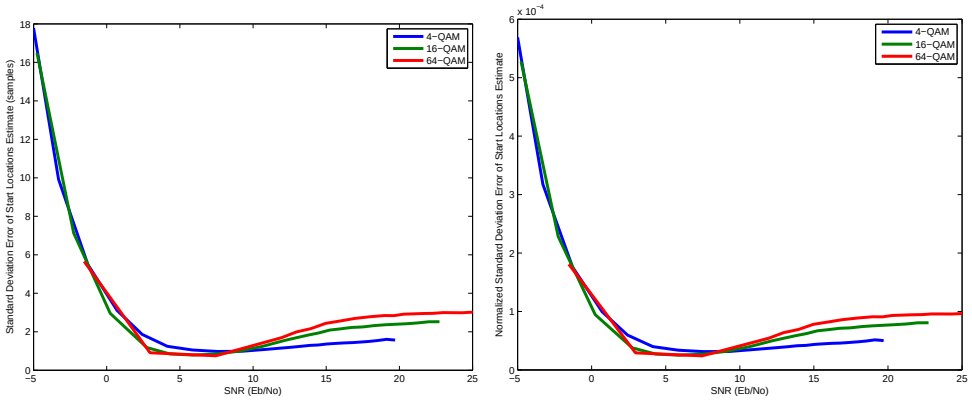
(a) Absolute (b) Normalized

Figure 5.14: Data acquisition mean absolute error for N=4096



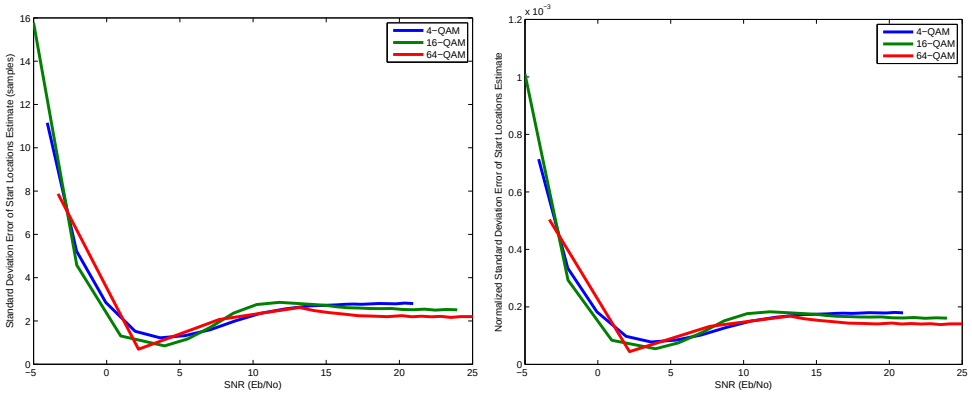
(a) Absolute (b) Normalized

Figure 5.15: Data acquisition standard deviation absolute error for N=1024



(a) Absolute (b) Normalized

Figure 5.16: Data acquisition standard deviation absolute error for N=2048



(a) Absolute (b) Normalized

Figure 5.17: Data acquisition standard deviation absolute error for N=4096

The absolute errors does not vary appreciably between different number of carriers. There is an improvement as the noise level decreases, which is expected. The normalized error does show an improvement. This is mostly likely caused from how the timing metric is derived from the filtered received signal which is compared to the non-filtered, but delayed by approximately half the length of the primary user filter.

5.1.5 Multi-path Channel

The multi-path channel is another channel that is analyzed and can be used as a baseline to compare with for a channel with primary users and multi-path. The impulse response of the channel has ten taps. The first tap has a value of '1'. The next nine taps are complex values, to ensure a non-symmetric frequency response, formed by the multiplication of the previous tap's value by a random complex value whose magnitude is between and .14142 and .42426 with a random phase rotation. In a multi-path channel, the BER curve has the appearance of the AWGN channel at high noise levels, but then will diverge from the AWGN BER curve as the noise level decreases. This effect is caused by making an inaccurate approximation of the channel. Figures 5.18, 5.19, and 5.20 show the BER curves for this channel.

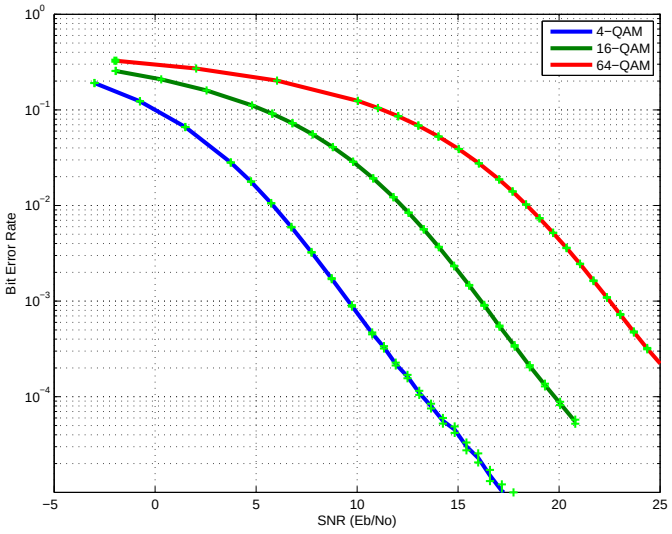


Figure 5.18: BER curves for N=1024

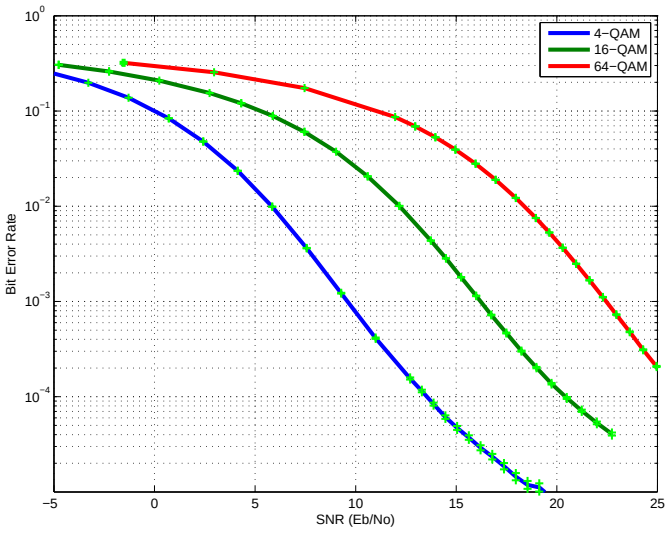


Figure 5.19: BER curves for N=2048

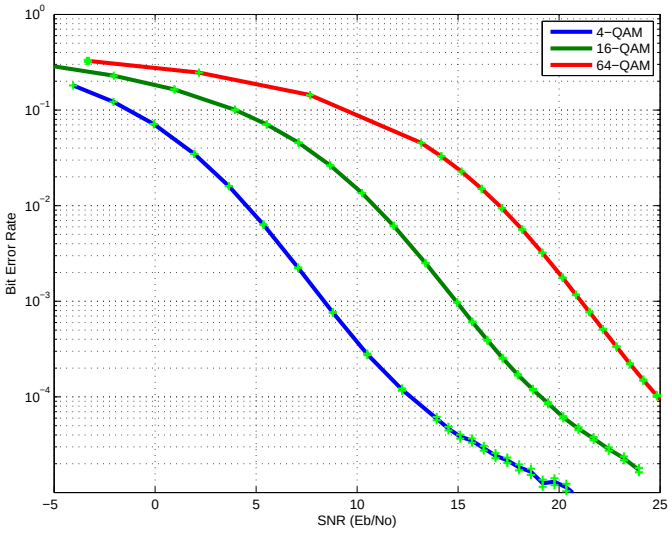
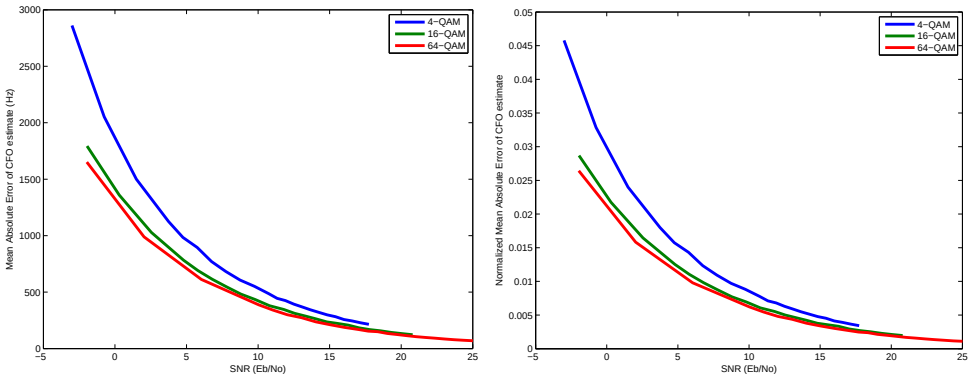


Figure 5.20: BER curves for N=4096

As can be seen in the above graphs, the BER curves have the expected shape. When comparing the differing number of carriers and the BER curves from the AWGN channel, two observations can be made. The BER does improve slightly in high noise conditions and tends to worsen in lower noise conditions.

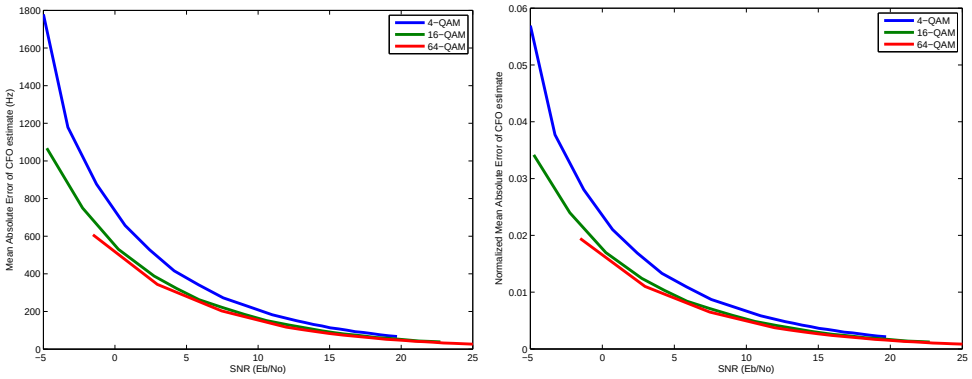
Figures 5.21, 5.22, and 5.23 show the mean absolute error for the initial CFO estimate.

Figures 5.24, 5.25, and 5.26 show the standard deviation of the absolute error for the initial CFO estimate.



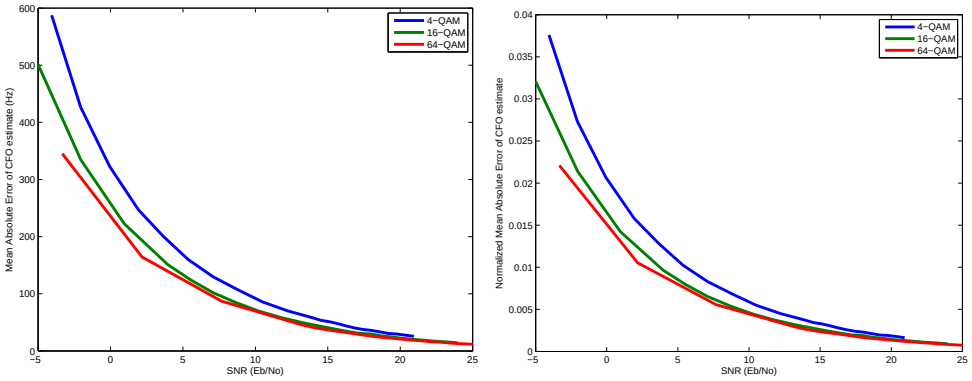
(a) Absolute (b) Normalized

Figure 5.21: CFO initial estimate mean for N=1024



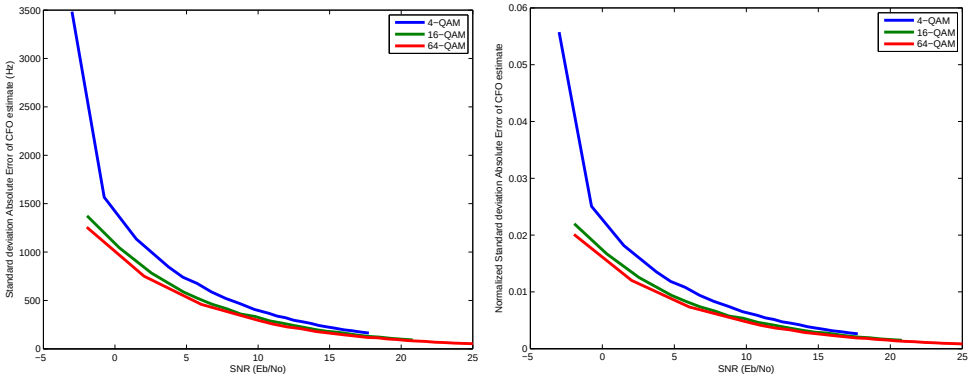
(a) Absolute (b) Normalized

Figure 5.22: CFO initial estimate mean for N=2048



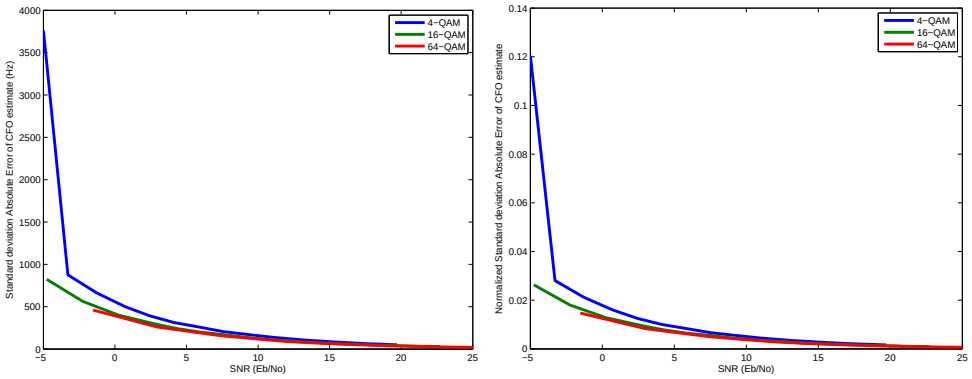
(a) Absolute (b) Normalized

Figure 5.23: CFO initial estimate mean for N=4096



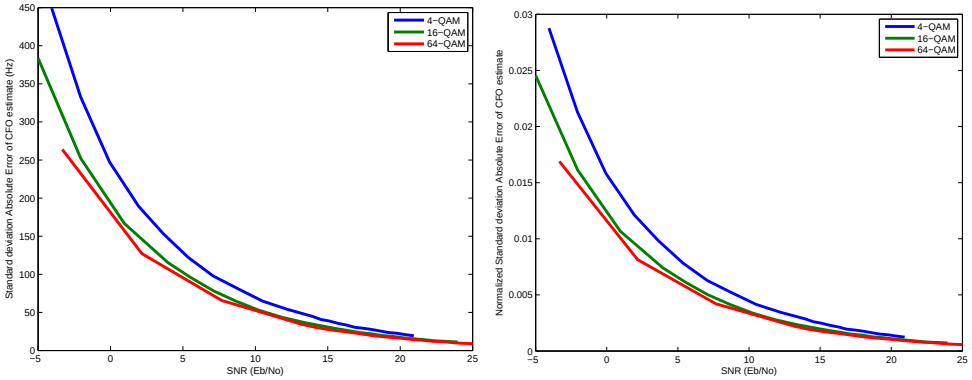
(a) Absolute (b) Normalized

Figure 5.24: CFO initial estimate standard deviation for N=1024



(a) (b) Normalized

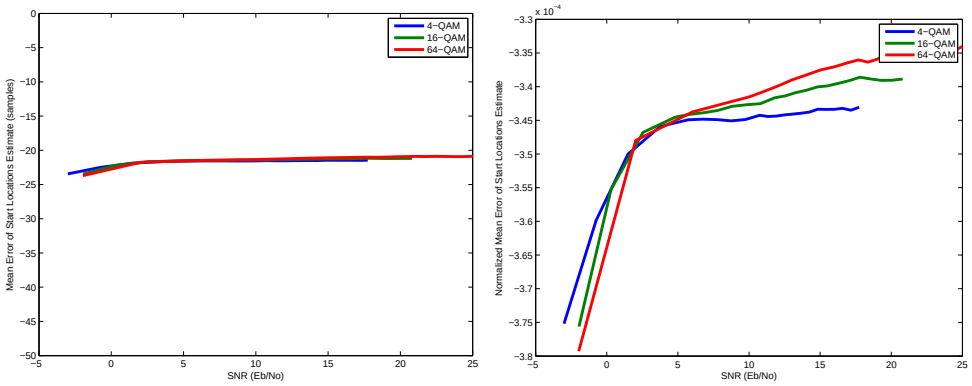
Figure 5.25: CFO initial estimate standard deviation for N=2048



(a) Absolute (b) Normalized

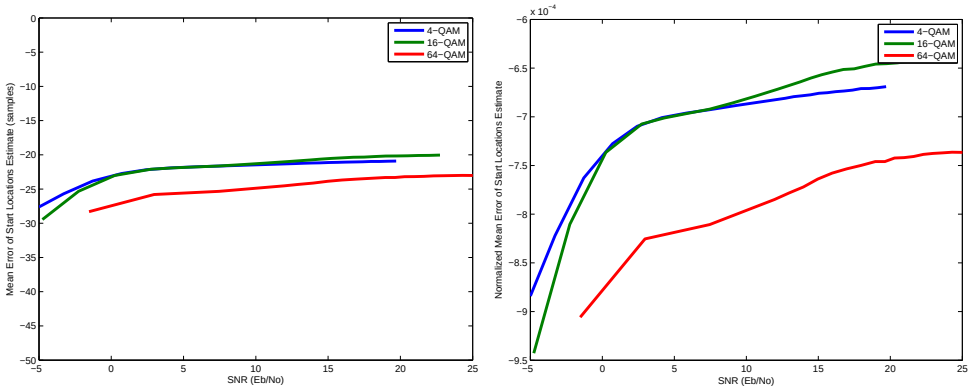
Figure 5.26: CFO initial estimate standard deviation for N=4096

The accuracy of detecting the individual data symbols is another important benchmarking point for synchronization. Figures 5.27, 5.28, and 5.29 show absolute mean error and Figures 5.30, 5.31, and5.32.



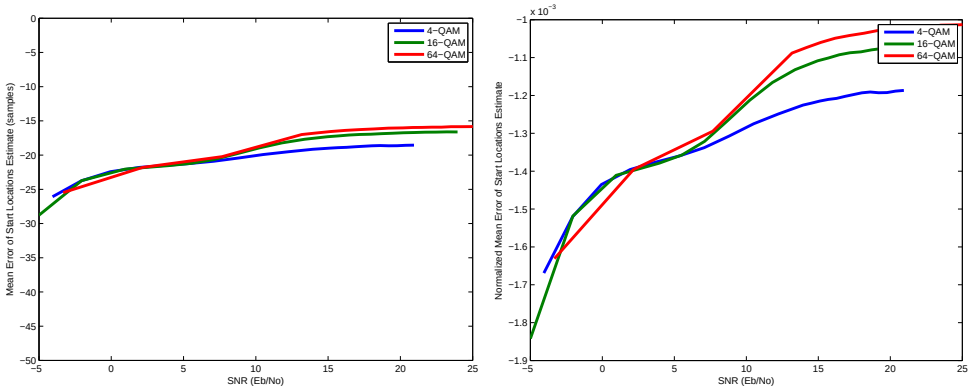
(a) Absolute (b) Normalized

Figure 5.27: Data acquisition mean absolute error for N=1024



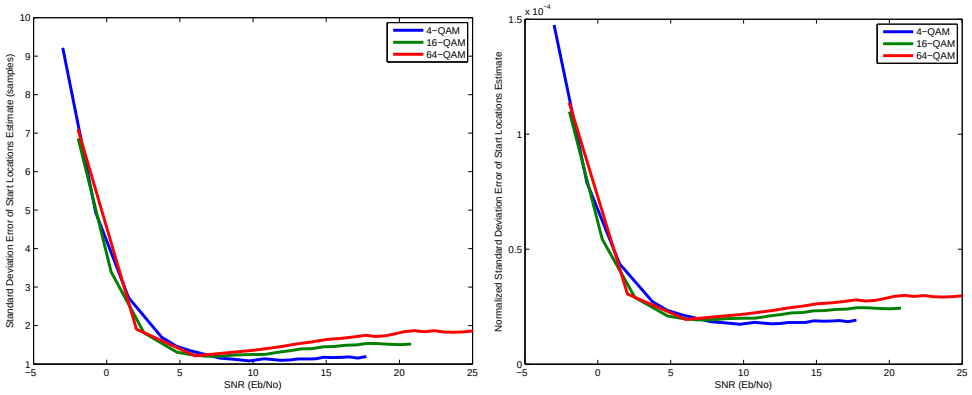
(a) Absolute (b) Normalized

Figure 5.28: Data acquisition mean absolute error for N=2048



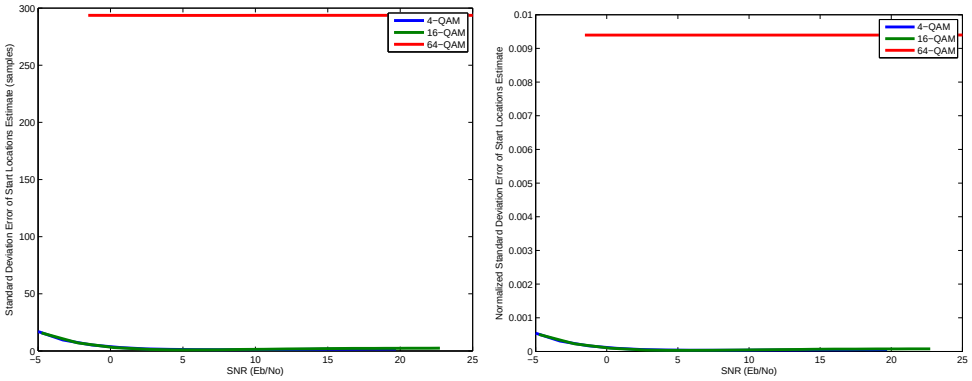
(a) Absolute (b) Normalized

Figure 5.29: Data acquisition mean absolute error for N=4096



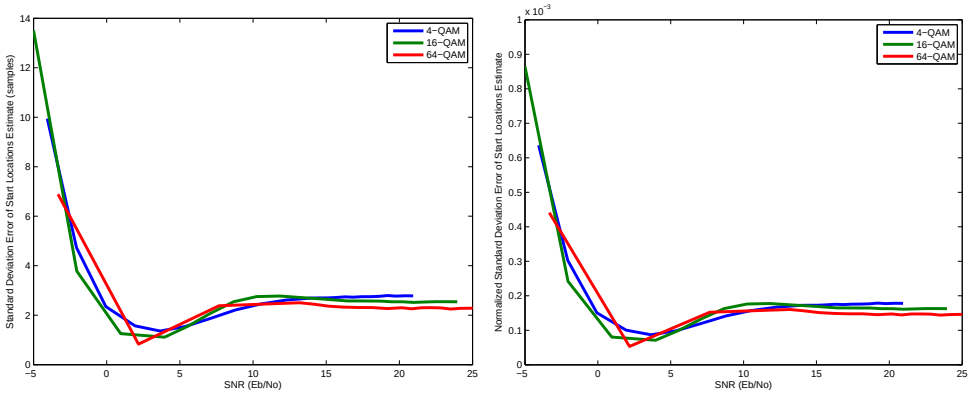
(a) Absolute (b) Normalized

Figure 5.30: Data acquisition standard deviation absolute error for N=1024



(a) Absolute (b) Normalized

Figure 5.31: Data acquisition standard deviation absolute error for N=2048



(a) Absolute (b) Normalized

Figure 5.32: Data acquisition standard deviation absolute error for N=4096

The accuracy of detecting the start of data symbols has a similar appearance to the AWGN channel graphs. All of the graphs have a similar offset from zero. This further proves that the detection process from the filtered received data is causing the delay. Since the offset is fairly constant across differing noise levels, differing number of carriers, and differing QAM modulation schemes, the delay constant can be adjusted as such or it could be corrected by individually rotating the carriers in the frequency domain after the FFT is taken.

5.1.6 Primary User Present Channel

The AWGN with primary users is one of the most important channels being analyzed since NC-OFDM's purpose is to use non-contiguous chunks of spectrum. Additionally, the spectrum that is not being used the NC-OFDM transceiver pair may have other transmissions occurring. When testing this particular channel, the primary user was chosen to be multiple OFDM of the same carrier size to maximize potential interference. The primary users were also filtered to such that they do not overlap in the spectrum with the NC-OFDM pair. This assumes that the spectrum determined to be of use for the NC-OFDM is chosen to have no overlap with the primary users, including spectral bleeding from the primary users. If this is requirement is relaxed, the BER curves would deteriorate since the spectral bleeding from the primary users would appear as addition noise to the NC-OFDM transmission pair. Figures 5.33, 5.34, and 5.35 show the BER curves for this channel.

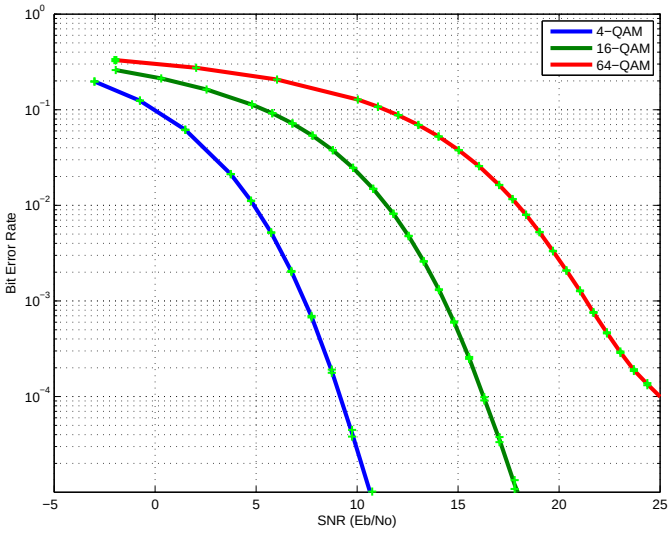


Figure 5.33: BER curves for N=1024

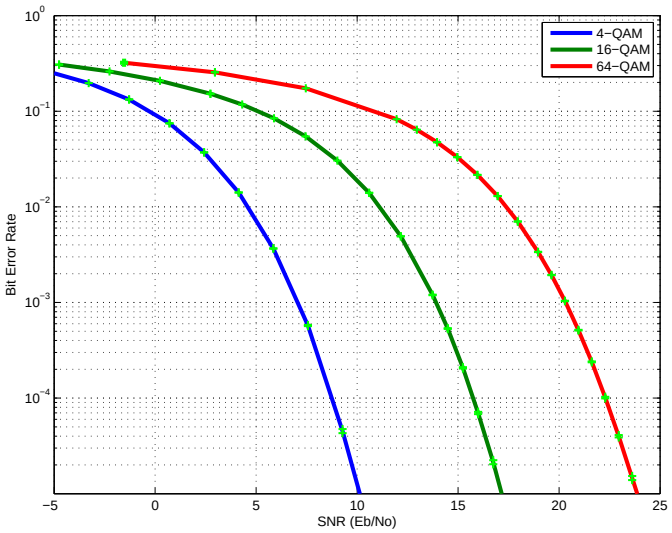


Figure 5.34: BER curves for N=2048

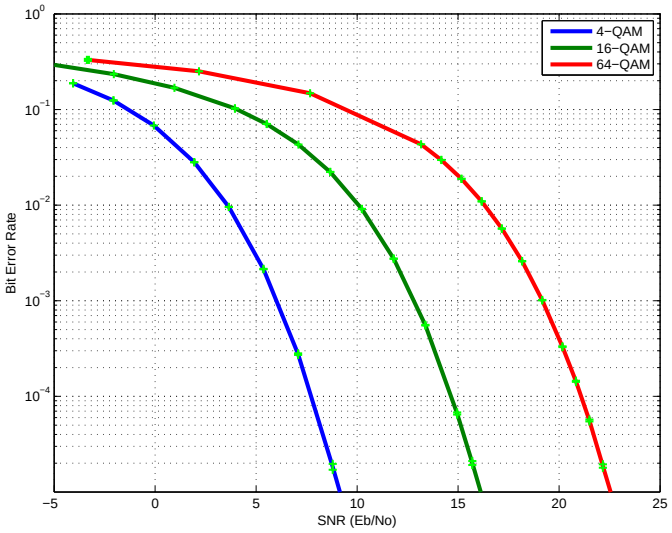
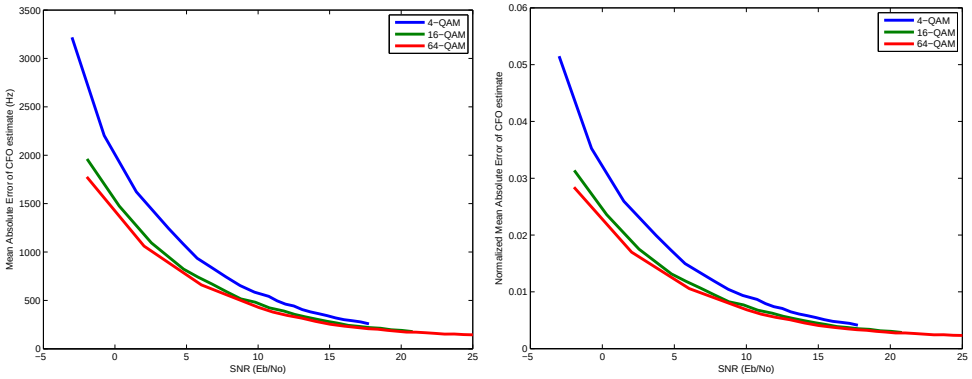


Figure 5.35: BER curves for N=4096

As expected the BER curves improve as the number of carriers increases. Additionally, when N=4096 and N=2048, the change is very small. When N=1024, the change is most pronounced especially when 64-QAM is used. Since the the changes are minimal

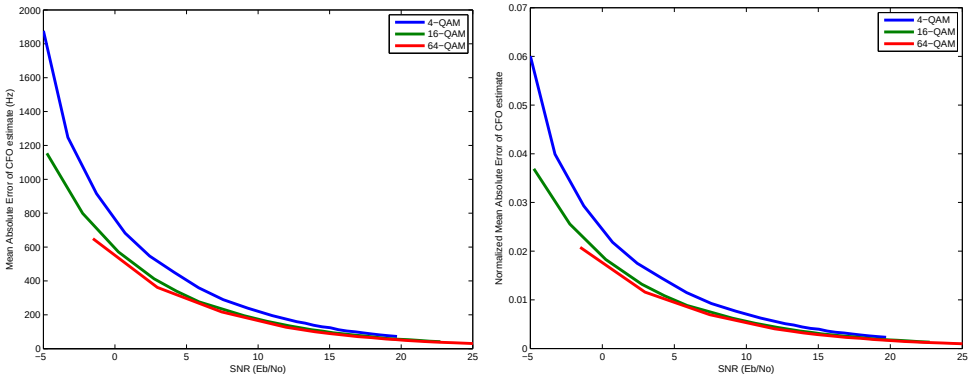
Figures 5.36, 5.37, and 5.38 show the mean absolute error for the initial CFO estimate.

Figures 5.39, 5.40, and 5.41 show the standard deviation of the absolute error for the initial CFO estimate.



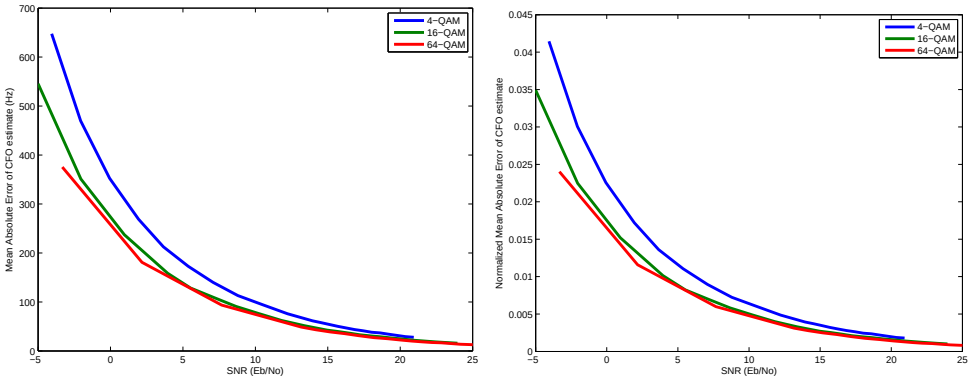
(a) Absolute (b) Normalized

Figure 5.36: CFO initial estimate mean for N=1024



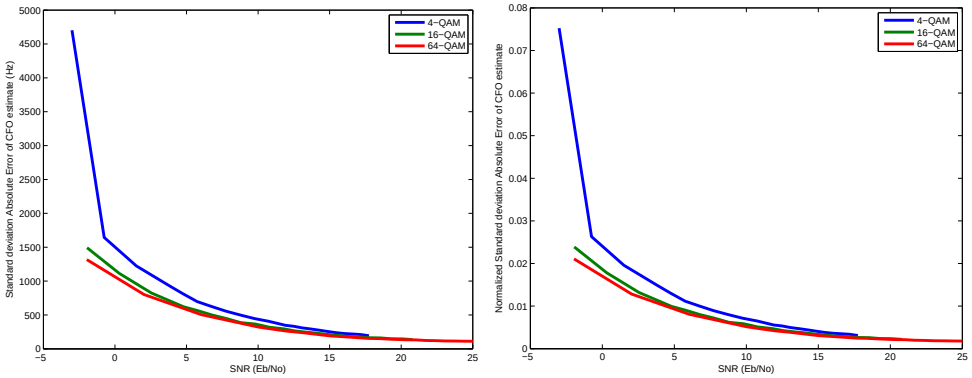
(a) Absolute (b) Normalized

Figure 5.37: CFO initial estimate mean for N=2048



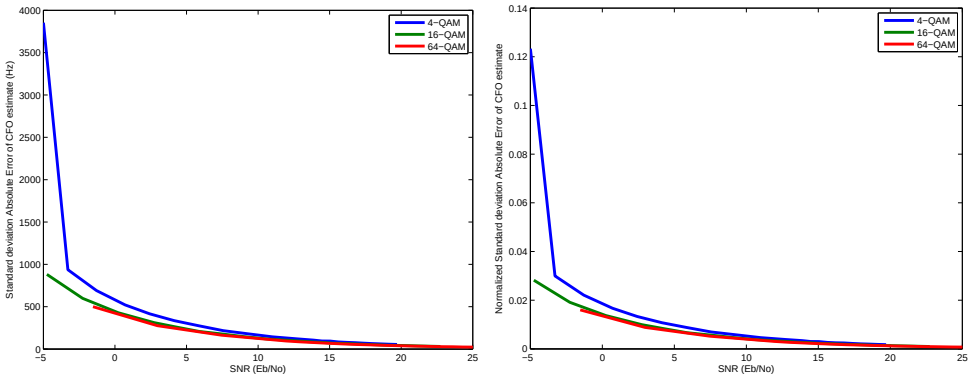
(a) Absolute (b) Normalized

Figure 5.38: CFO initial estimate mean for N=4096



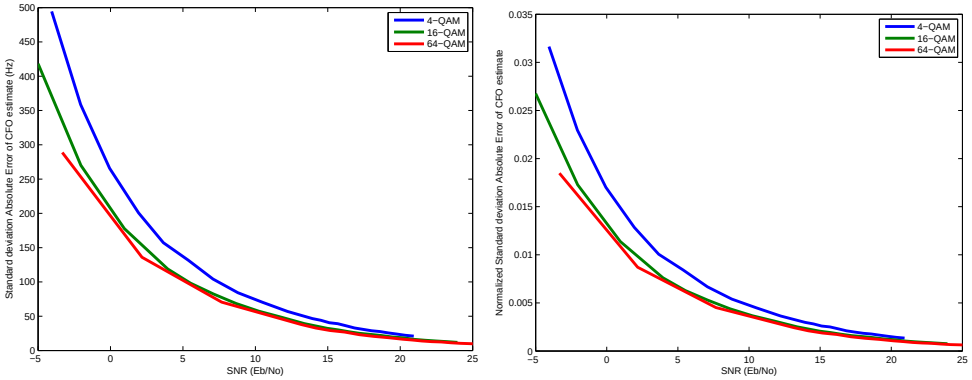
(a) Absolute (b) Normalized

Figure 5.39: CFO initial estimate standard deviation for N=1024



(a) Absolute (b) Normalized

Figure 5.40: CFO initial estimate standard deviation for N=2048

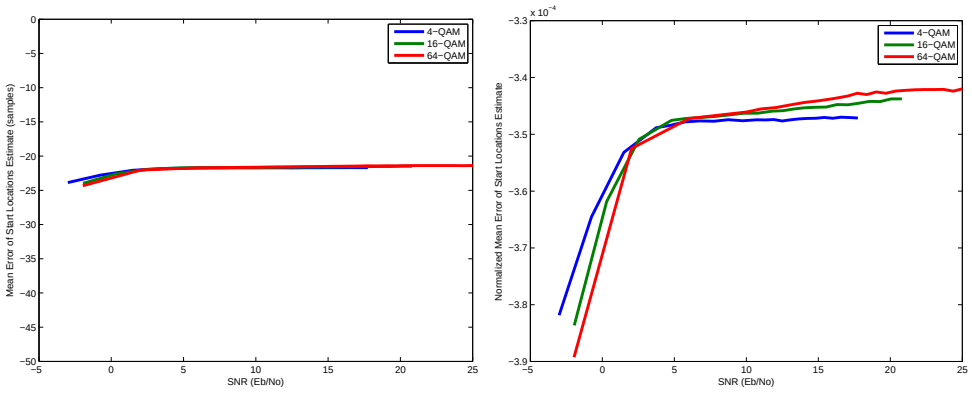


(a) Absolute (b) Normalized

Figure 5.41: CFO initial estimate standard deviation for N=4096

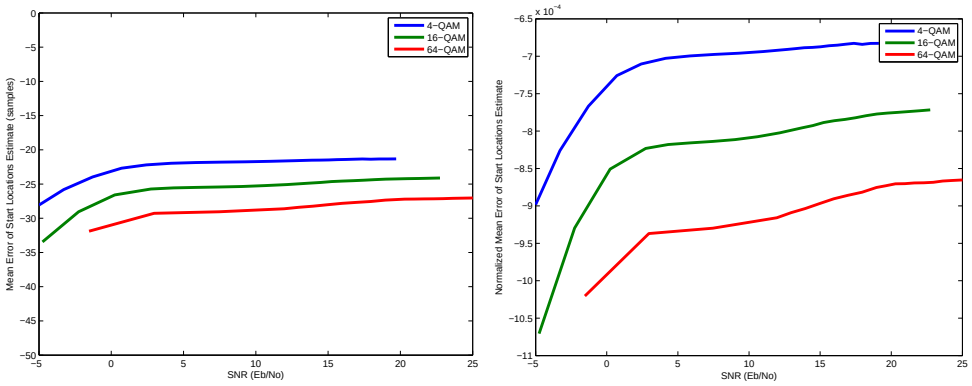
When comparing the CFO initial estimates, the behavior is similar to that of the AWGN channel. Additionally, when comparing the performance when the number of carriers is the same to the AWGN channel, there is little change.

The accuracy of detecting the individual data symbols is another important benchmarking point for synchronization. Figures Figures 5.42, 5.43, and 5.44 show absolute mean error and Figures 5.45, 5.46, and 5.47.



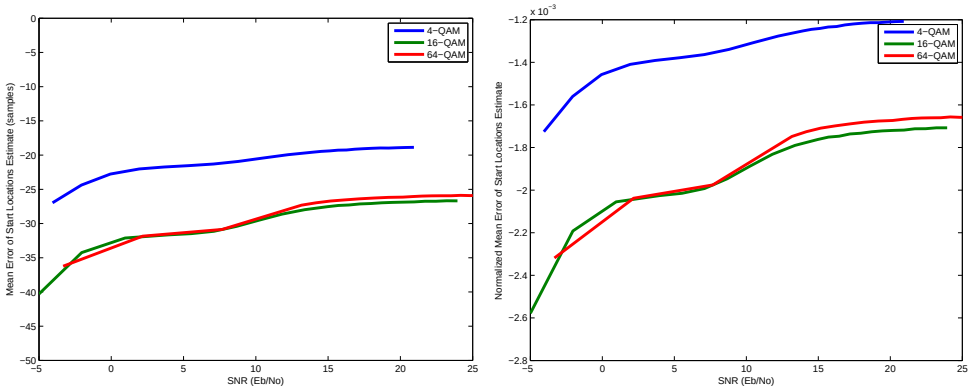
(a) Absolute (b) Normalized

Figure 5.42: Data acquisition mean absolute error for N=1024



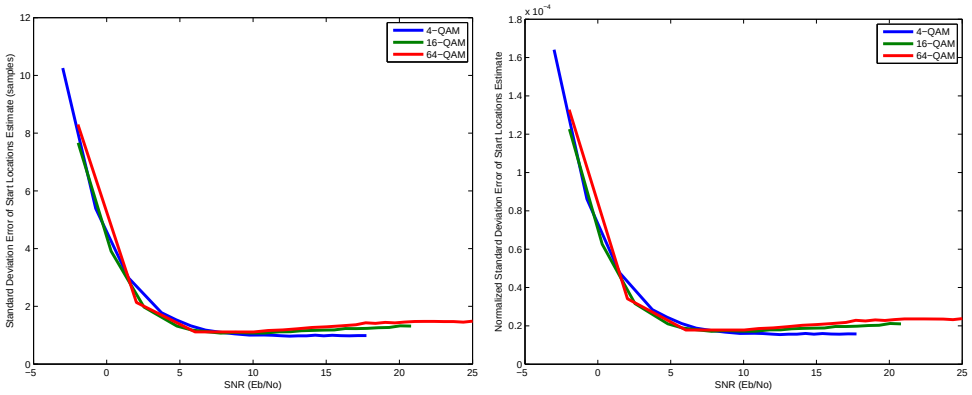
(a) Absolute (b) Normalized

Figure 5.43: Data acquisition mean absolute error for N=2048



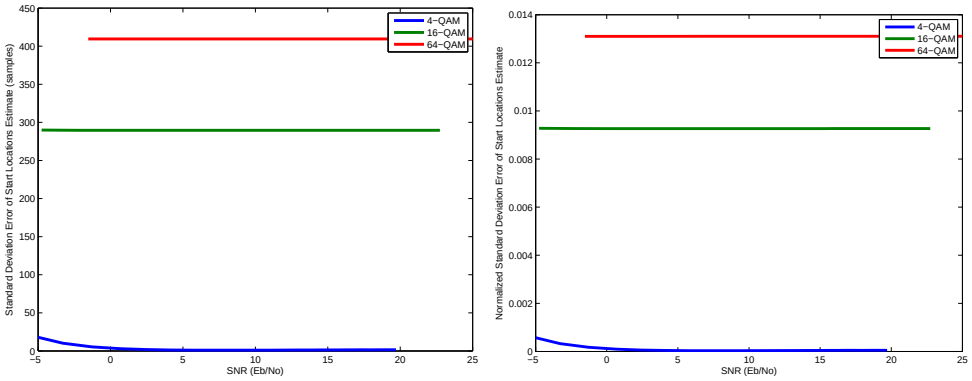
(a) Absolute (b) Normalized

Figure 5.44: Data acquisition mean absolute error for N=4096



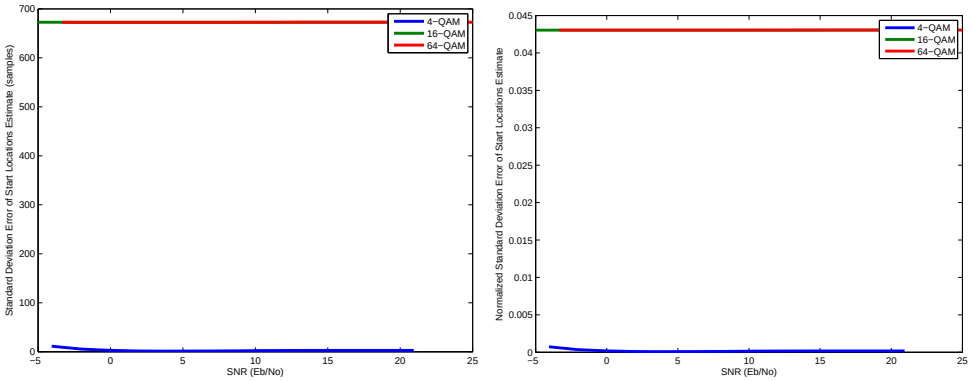
(a) Absolute (b) Normalized

Figure 5.45: Data acquisition standard deviation absolute error for N=1024



(a) Absolute (b) Normalized

Figure 5.46: Data acquisition standard deviation absolute error for N=2048



(a) Absolute (b) Normalized

Figure 5.47: Data acquisition standard deviation absolute error for N=4096

Based on theses graphs and the graphs from the AWGN channel, the NC-OFDM transceiver is function successfully.

5.1.7 Primary User in Multi-path Environment

The primary user and multi-path channel is last channel being tested. Figures 5.48, 5.49, and 5.50 show the BER curves for this channel.

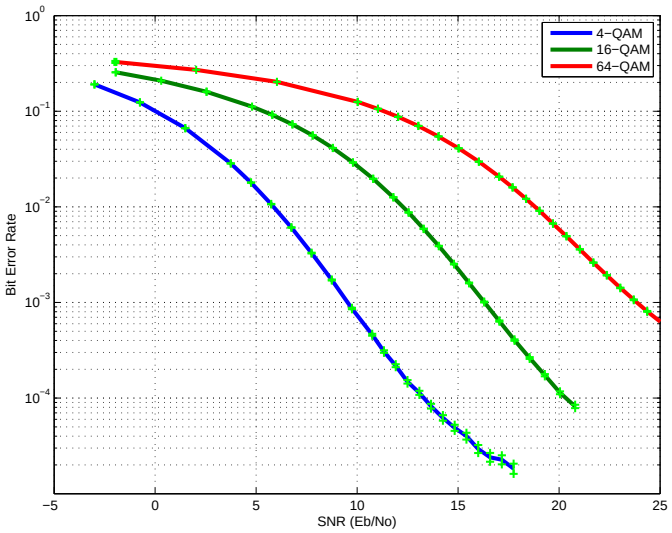


Figure 5.48: BER curves for N=1024

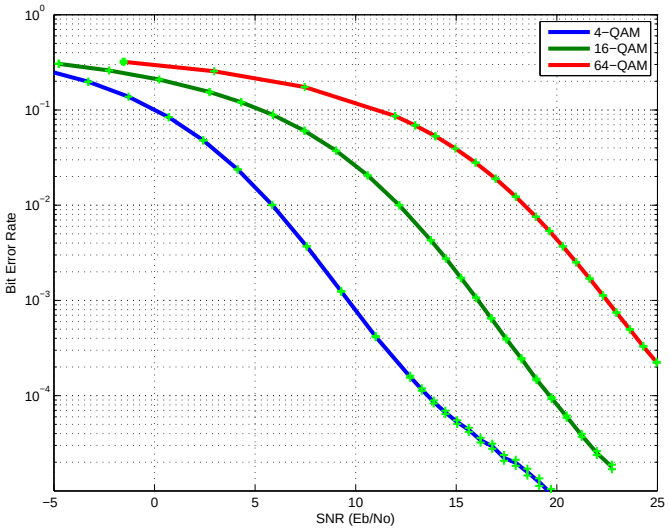


Figure 5.49: BER curves for N=2048

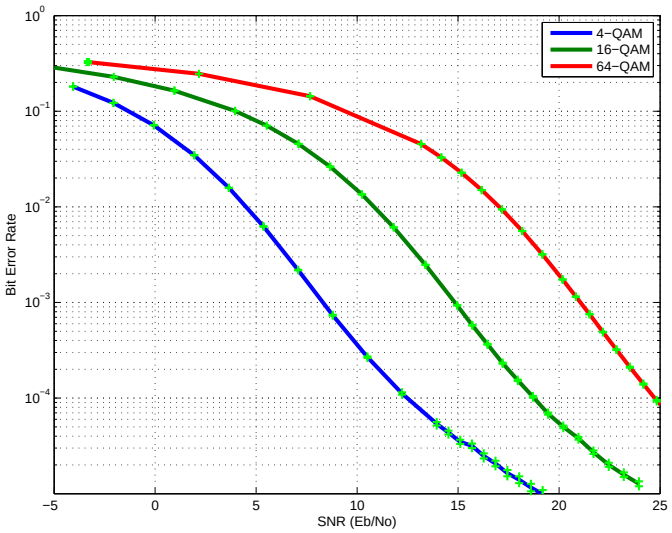
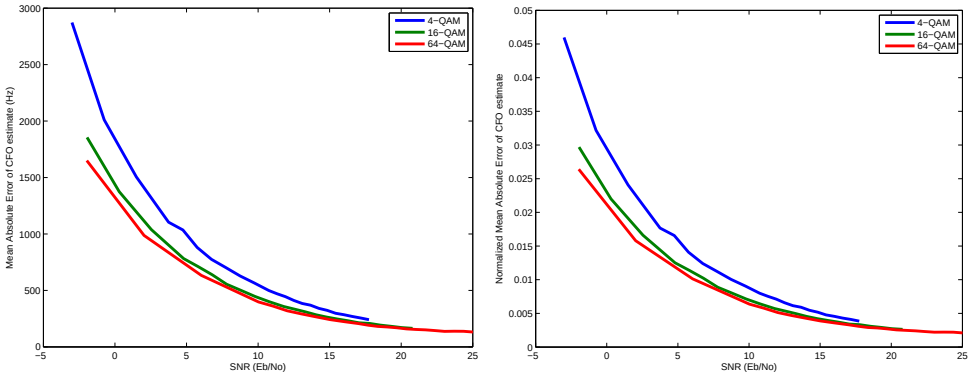


Figure 5.50: BER curves for N=4096

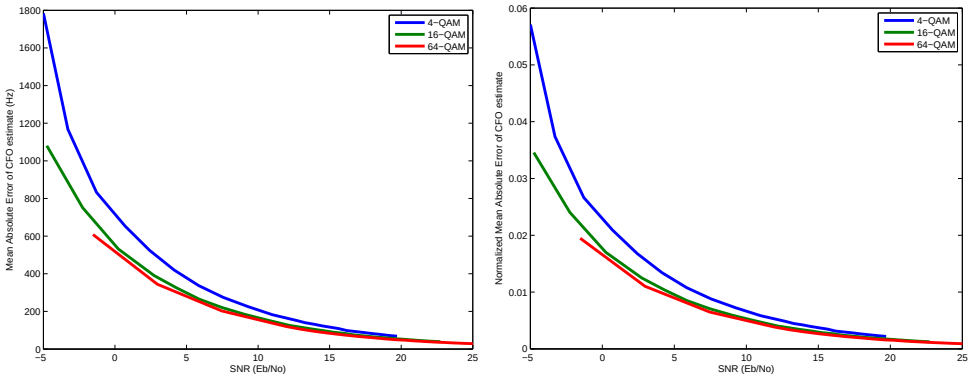
The BER curves show what is expected. The multi-path effects dominates in the lower noise levels, BER improves as the number of carriers increases, and minimal changes when compared to multi-path channel.

Figures 5.51, 5.52, and 5.53 show the mean absolute error for the initial CFO estimate. Figures 5.54, 5.55, and 5.56 show the standard deviation of the absolute error for the initial CFO estimate.



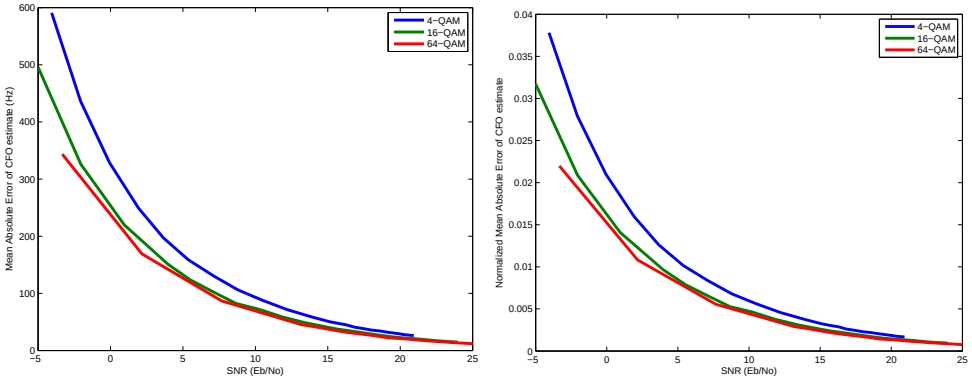
(a) Absolute (b) Normalized

Figure 5.51: CFO initial estimate mean for N=1024



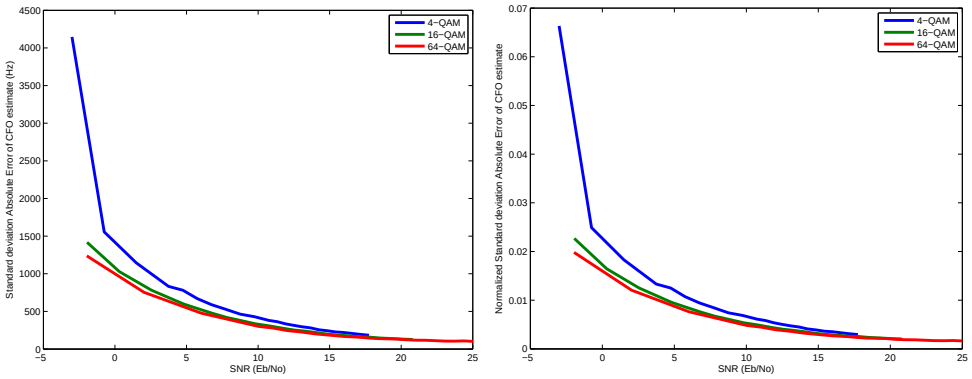
(a) Absolute (b) Normalized

Figure 5.52: CFO initial estimate mean for N=2048



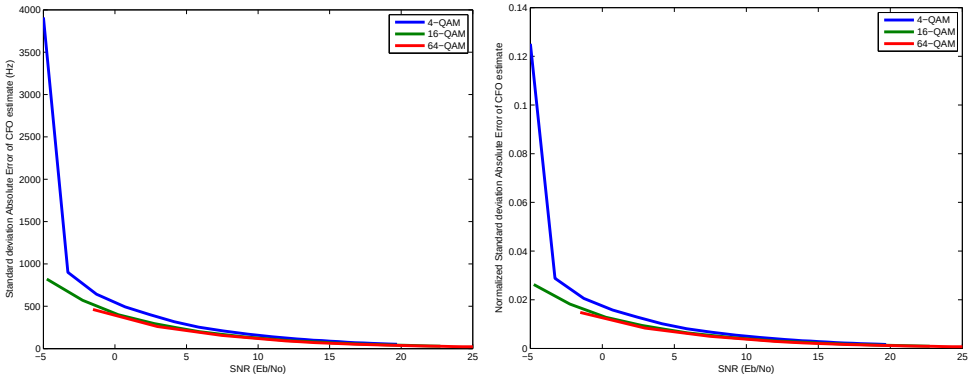
(a) Absolute (b) Normalized

Figure 5.53: CFO initial estimate mean for N=4096



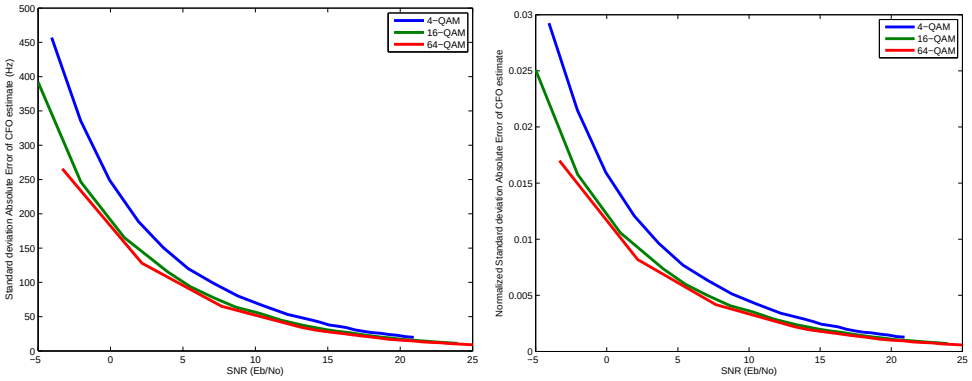
(a) Absolute (b) Normalized

Figure 5.54: CFO initial estimate standard deviation for N=1024



(a) Absolute (b) Normalized

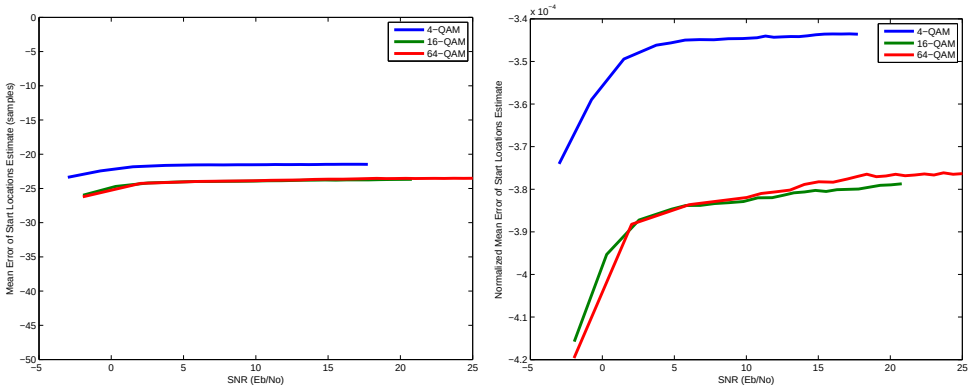
Figure 5.55: CFO initial estimate standard deviation for N=2048



(a) Absolute (b) Normalized

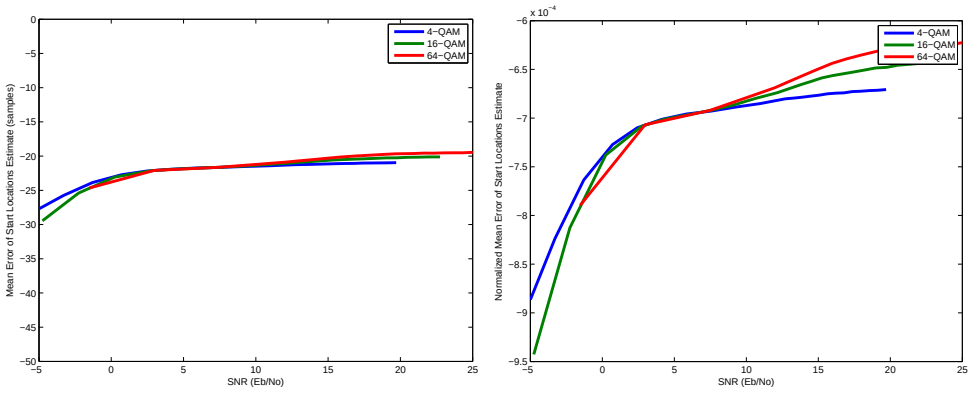
Figure 5.56: CFO initial estimate standard deviation for N=4096

The accuracy of detecting the individual data symbols is another important benchmarking point for synchronization. Figures 5.57, 5.58, and 5.59 show absolute mean error and Figures 5.60, 5.61, and 5.62.



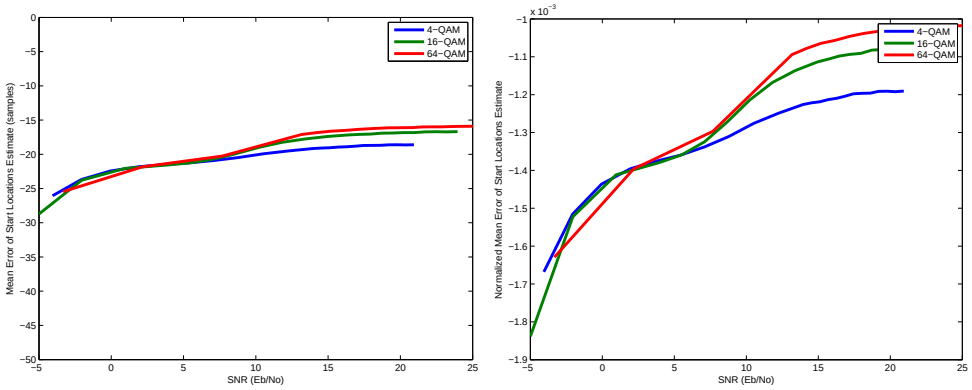
(a) Absolute (b) Normalized

Figure 5.57: Data acquisition mean absolute error for N=1024



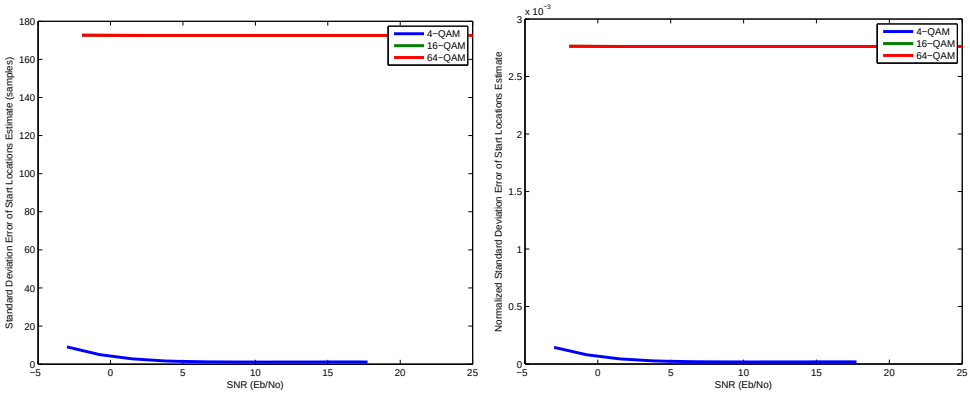
(a) Absolute (b) Normalized

Figure 5.58: Data acquisition mean absolute error for N=2048



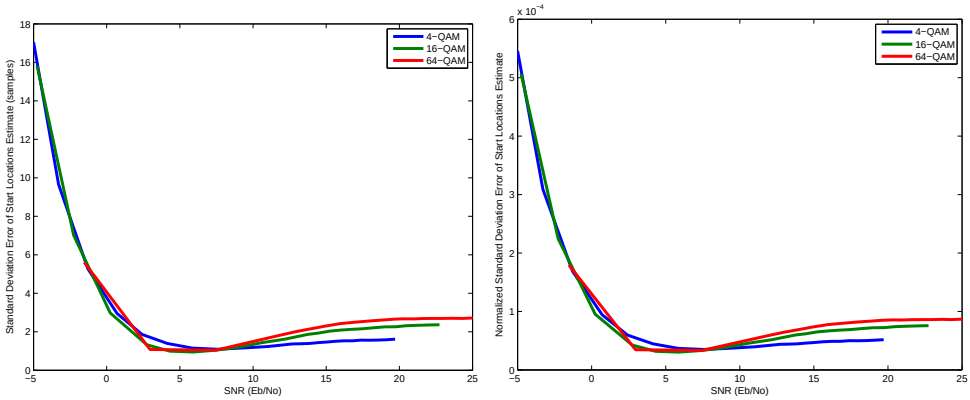
(a) Absolute (b) Normalized

Figure 5.59: Data acquisition mean absolute error for N=4096



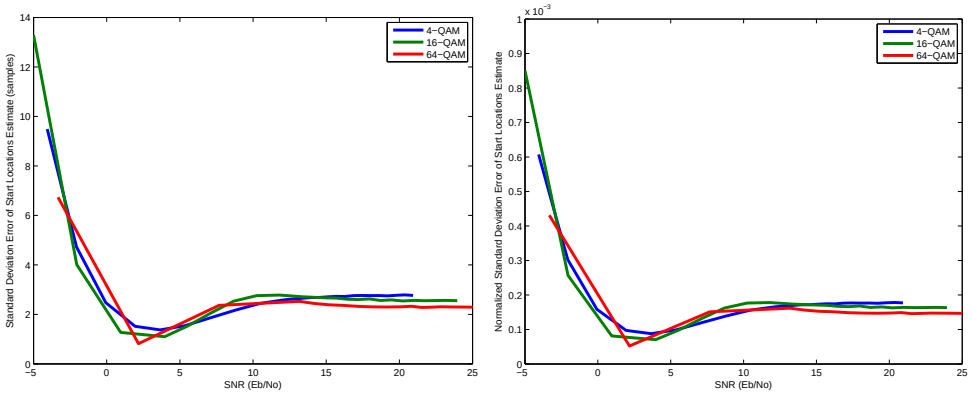
(a) Absolute (b) Normalized

Figure 5.60: Data acquisition standard deviation absolute error for N=1024



(a) Absolute (b) Normalized

Figure 5.61: Data acquisition standard deviation absolute error for N=2048



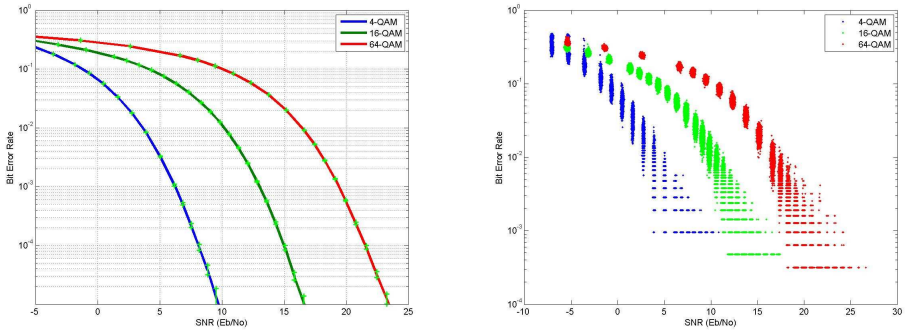
(a) Absolute (b) Normalized

Figure 5.62: Data acquisition standard deviation absolute error for N=4096

This final channel also shows that the NC-OFDM transceiver is able to transmit successfully in a multi-path channel with primary users. It is expected that this channel would the best model of a real world environment where signals are reflected off objects and primary users are present. In the case where the paths are not significant, the primary user channel would is a better model. In either case, the performance is similar to that of the multi-path channel and the AWGN channel.

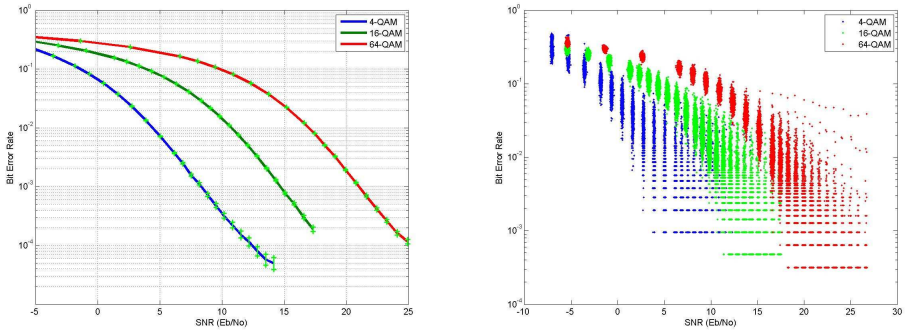
5.1.8 Preamble Detection

In the course of running benchmarks for N=512, it was seen that in some channel perform as expected and others had a very high bit error rate for low noise conditions. The only possible cause would be for a large incorrect estimation of the CFO. It was then determined that the root cause was a incorrect detection of the short preamble. Figures 5.63, 5.64, 5.66, and 5.65 show the BER curves and scatter plots for the case when N=512.



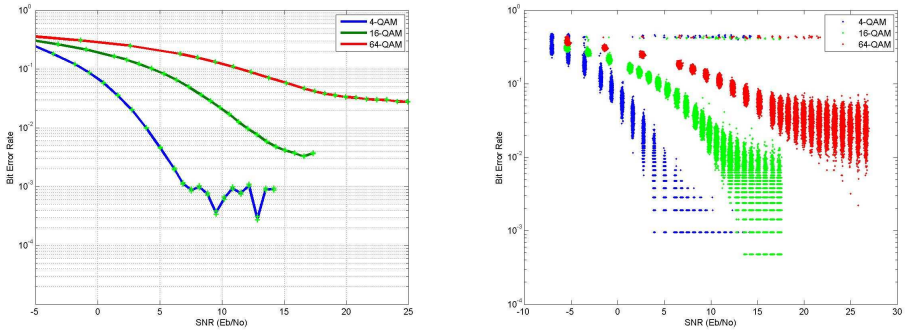
(a) Average Plot (b) Scatter Plot

Figure 5.63: BER curves for N=512 in AWGN channel



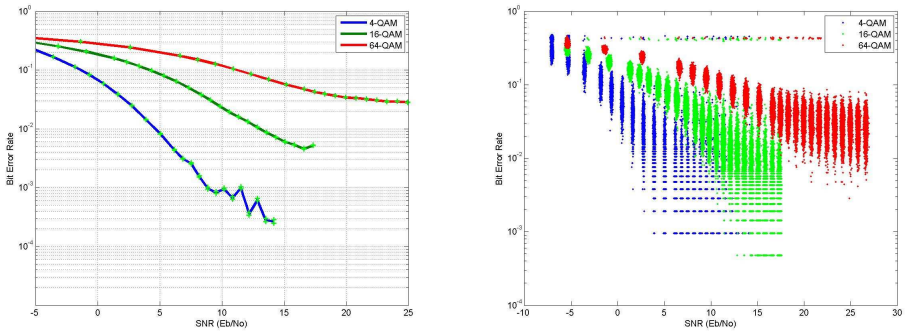
(a) Average Plot (b) Scatter Plot

Figure 5.64: BER curves for N=512 in Multi-path Channel



(a) Average Plot (b) Scatter Plot

Figure 5.65: BER curves for N=512 in Primary User Channel



(a) Average Plot (b) Scatter Plot

Figure 5.66: BER curves for N=512 in Primary User Channel with Multi-path

The averaged BER curve and a scatter plot of the corresponding BER curve are both presented. The scatter plot is shown to better illustrate the failures in detecting the short preamble. Each point represents the individual BER for a particular trial, QAM modulation, and SNR level for a single trial. The scatter plot shows there is an underlining BER that has an expected appearance. The failures in detecting the short preamble can best be seen in the outliers near 10^{-1} BER. As can be seen the above BER curves, the only the AWGN and Multi-path channels behave as expected. It should also be noted that there many trials in which MATLAB crashed. A crash signifies the short preamble was missed entirely. Most crashes occurred in the Primary User channel and Multi-path with Primary User channel.

There were few MATLAB crashes with other channels. The failures are most apparent in the last two channels. This was seen on occasion in the case of N=1024, but a more advanced detection system was implement so it was no longer seen in that case. In addition to the the detection algorithm, the normalization term was adjusted to see if what change it had on the false detection of the short preamble. Table 5.1.8 and Table 5.1.8 show a summary of the results for various detection methods.

Table 5.2: Detection Failures for 1000 packets Case 1

	Low Noise		High Noise	
Normalization Method	Delay 0	Delay 1	Delay 0	Delay 1
Samp1	3	3	9	8
Samp2	0	0	8	6
Samp1 + Samp2	0	0	8	6

Table 5.3: Detection Failures for 1000 packets Case 2

	Low Noise		High Noise	
Normalization Method	Delay 0	Delay 1	Delay 0	Delay 1
Samp1	30	8	23	28
Samp2	0	29	9	26
Samp1 + Samp2	0	21	12	26

Samp1 and Samp2 are the two sliding windows of the autocorrelation for the received data. In Case 1, they are N/8 in size and N/4 in Case 2. For the columns *Delay 0*, the Samp2 sliding window follows directly behind Samp1 sliding window. For the columns *Delay 1*, the Samp2 sliding window follows behind the Samp1 sliding window, but is delayed by the sliding window size. As can be seen from the two tables, the Case 1's Sampl+Samp2 with a delay of 1 and Case 1's Samp2 with a delay of 1 perform the best. This suggests the that further investigation must be done for N=512 if it is to be used.

A matched filter and cross correlator was considered, but both were deemed not practical as both would require N/2 multipliers, assuming the clock rate of the multipliers is the same as the sampling frequency. This comes from N/8 samples times 4 real multipliers per a complex multiply. This is possible if only simulations were being run, but impractical to implement in most FPGAs.

5.2 FFT and IFFT Pruning Results

The pruned FFT/IFFT core provides an important function to the transceiver. The purpose of a pruned FFT/IFFT core is to save computations and thus time. Figure 5.67 shows the theoretical relative speed as the percentage of used sub-carriers decreases.

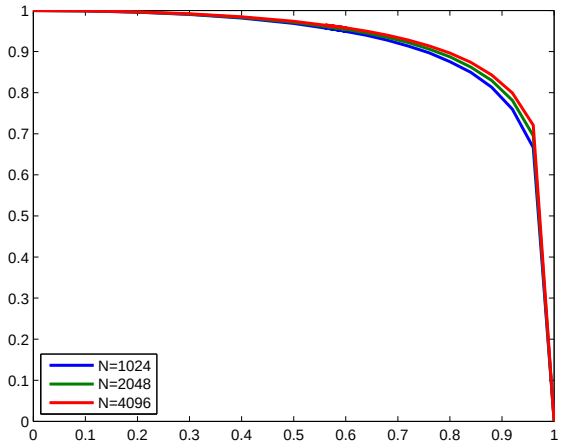


Figure 5.67: Graph showing relative number of cycles for a pruned FFT.

From the above graph, it can be seen that a pruned FFT does not save many cycles for total FFT time until approximately 60 % of carriers are deactivated. From there, the reduction in the number of butterflies, and thus cycles, decreases rapidly. The distribution of the carriers for the simulation was random, thus were not grouped into contiguous blocks of carriers. The effect of random distribution versus contiguous blocks of carriers was not investigated. This was done since a pruned FFT/IFFT core was chosen specifically for a NC-OFDM transceiver, but can be used other applications and a specific distribution choice would only show results for that specific distribution that may not be used in other possible applications. Additionally in the case of contiguous block of carriers, the size of the contiguous blocks of carrier and their location can vary greatly while having the same number of carriers. This has the potential for widely differing number of cycles. When comparing this algorithm on the basis of number of computations saved to pruning algorithms that remove portions of butterflies, this algorithm would not save as many cycles. However,

this algorithm was designed to allow for easier implementation in hardware. In a simplistic view, all that needs to change for radix-2 FFT/IFFT core would be the incrementing of the addresses for the butterfly unit, as opposed to going with an approach that may abandon butterfly unit for general execution units or use specialized partial butterfly units.

The rapid roll off as the percentage of active carriers approaches zero shows that pruned butterflies exist mostly in the first or last several stages. This is caused from the intertwined nature of the stages of the FFT/IFFT. In an extreme case, of only one needed butterfly in the initialization vector, the number of needed butterflies doubles as it progresses through the stages.

5.3 Summary

As can be seen, the NC-OFDM transceiver is able to transmit at suitable SNR when the size of the FFT/IFFT is N=1024, N=2048, or N=4096 carriers. It is expected that it will work for larger number of carriers. The reduction in computation time for the pruned FFT is not appreciable for most percentages of unused carriers. At approximately 70 %, the reduction rapid decreases.

Chapter 6

Conclusion

Chapter 5 presented the results and explained how they relate to each other. In general, when N=1024, 2048, or 4096, NC-OFDM can successfully transmit data over the tested channels with acceptable BER performance. Since this thesis presents an initial NC-OFDM synchronization algorithm and transceiver architecture and form the base for a larger system, there are many areas that could be further researched. Some of these research areas are explained in the Future Work section.

6.1 Synchronization Algorithm

From the results presented in Chapter 5, it can be seen that the synchronization algorithm can produce acceptable BER curves for the tested channels. The BER curves have an expected shape of either a smooth fall off or a smooth fall leading to a plateau. Additionally, the results show that the size of N should be maximized. This is backed up by how the BER rates improve and the CFO estimate improves. When the number of carrier is increased, the whitespaces in the spectrum can be better filled and the side-lobes, inherit to OFDM based modulation schemes, is reduced. This is due to fact that the larger a rectangular pulse is in the time domain, the narrower the sinc pulse in the frequency domain. Although not tested, a rapidly changing Doppler shift would be most noticeable with higher carrier sizes.

The detection of the short preamble, thus the start of the NC-OFDM packet, is some-

thing that did not work consistently when N=512. This may not be a of much concern considering it can perform as expected for N=1024, 2048, and 4096 and it is better to use more carriers.

6.2 FFT/IFFT Pruning in Hardware

A pruned FFT/IFFT core was also presented. Although it can reduce the number of butterfly computation for an IFFT or FFT, there is a large increase in resource usage to enable FFT/IFFT pruning in hardware. Since the FFT and IFFT are tied to the receiver and the transmitter tightly, they should not be separated. A split hardware/software solution to the FFT and IFFT should not be done since it will add additional latency, especially if the data is passed from hardware, to software, and then back to hardware.

6.3 NC-OFDM Transceiver Architecture

The architecture and algorithm proposed did allow for an architecture that is realizable in hardware. A matched filter and cross correlator was avoided. Although the pruned FFT/IFFT core has an inherent large increase in resource usage, since the address generation is no longer done by a small unit, the unit was designed to reduce the needed resources by calculating only the necessary data at that moment in time while ensuring the pipeline does not stall.

6.4 Future Work

The work presented in this thesis can be expanded upon or integrated with other research. An important part of the complete NC-OFDM transmission system is the addition of PAPR reduction and side-lobe suppression, neither of these were explicitly investigated in this thesis.

The NC-OFDM transceiver should have additional work done. The cyclic prefix and guard interval are presently fixed in size. This should be parameterizable based on the impulse response of the channel. Additionally, a re-estimation of the CFO may not be

entirely necessary if the oscillators are of sufficient quality and the transmissions are close enough together. This allows for the removal of the long preamble and possibly the short preamble. The cyclic prefix of the data symbols can be used to adjust the CFO estimate should both the short and long preambles be removed. At present, the maximum number of carriers is 4096, which could be increased. Based on the results, this would improve the BER by a small amount and would help to reduce side-lobe levels. If a cyclic postfix is also deemed to be necessary, additional restructuring would need to be made.

The last section that code be expanded upon relate to the data symbols. In the simulations, the data was uncoded. Additionally, no consideration were made on power levels or QAM scheme used on the individual carriers based on the SNR or channel response. The pilot carrier arrangement was also fixed and did not change if the channel response was better or worse. A simple channel equalizer was used for the simulations. The channel equalizer did not use past history of estimates to form a better estimate of the channel, particularly in a noisy channel environment. Another option is to change the pilot carrier arrangement style. At present, the pilot carriers are at fixed locations. There are two possible options, entire data symbols can be comprised of pilot carriers or the pilot carrier arrangement can sparse and the pilot carrier location change over time.

Another issue with NC-OFDM synchronization is the *sampling frequency offset* (SFO). The sampling frequency is not identical between the DAC of the transmitter and the ADC of the receiver. The SFO can induce additional errors since it induces an offset in the frequency similar to a CFO.

The next logical step for the NC-OFDM transceiver is to integrate it into a full communication system using the USRP 1 or 2, MATLAB, and ideally, a FPGA development board. As it stands, the transceiver creates random data for the simulation runs. It is a relatively simple change to random data source to be the output from a channel coder. Additionally, the cancellation carriers can be set to their appropriate values based on the data. With completion of a finalized carrier modulator and demodulator, cancellation carrier creator, and an optional transmit filter, the NC-OFDM transceiver will be a complete physical layer. The USRPs are used as a RF-front end and will be connected to a computer through USB or Ethernet. Since an architecture of much of the physical layer of a NC-OFDM transceiver

was presented, system integration with a FPGA board would be part of this next step. Integrating the FPGA board with the USRPs and Matlab could be a formidable task.

Bibliography

[1] J. Acharya, H. Viswanathan, and S. Venkatesan, *Timing acquisition for non contiguous OFDM based dynamic spectrum access*, Proceedings of the IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN (Chicago, IL, USA), October 2008.

[2] Rogerio G. Alves, P. L. Osorio, and M. N. S. Swamy, *General FFT pruning algorithm*, Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Lansing, MI, USA), August 2000.

[3] Mark W. Chamberlain, *A software defined HF radio*, Proceedings of the IEEE Military Communications Conference (Rochester, NY, USA), October 2005.

[4] Yung-Fang Chen and Jean-Wei Chen, *A fast subcarrier, bit, and power allocation algorithm for multiuser OFDM based systems*, IEEE Transactions on Vehicular Technology, (2008).

[5] J. W. Cooley and J. W. Tukey, IEEE Transactions on Electronic Computers.

[6] Hiroshi Harada, *Software defined radio prototype toward cognitive radio communication systems*, Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN (Baltimore, MD, USA), November 2005.

[7] Simon Haykin, *Communication systems*, 4th ed., John Wilev & Sons Inc., 2001.

[8] F. Iacomacci, C. Morlet, F. Autelitano, G.C. Cardarilli, M. Re, E. Petrongari, G. Bogo, and M. Franceschelli, *A software defined radio architecture for a regenerative on board*

processor, Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (Rome, Italy), June 2008.

[9] Joesph Mitola III, *Software radios: Survey, critical evaluation and future directions*, IEEE Aerospace and Electronic Systems Magazine (1993).

[10] Preston A. Jackson, Cy P. Chan, Jonathan E. Scalera, Charles M. Rader, and M. Michael Vai, *A systolic FFT architecture for real time FPGA systems*, Proceedings of the High Performance Embedded Computing Conference (Lexington, MA, USA), 2004.

[11] Jongkyung Kim, Sangjin Lee, and Jongsoo Seo, *Synchronization and channel estimation in cyclic postfix based OFDM system*, Proceedings of the IEEE 63rd Vehicular Technology Conference, Spring (Melbourne, Australia), May 2006.

[12] F. Kristensen, P. Nilsson, and A. Olsson, *Flexible baseband transmitter for OFDM*, Proceedings of the IASTED International Conference on Circuits, Signals, and Systems (Cancun, Mexico), 2003.

[13] Chan-Tong Lam, D.D. Falconer, and F. Danilo-Lemoine, *Iterative frequency domain channel estimation for DFT-precoded OFDM systems using in-band pilots*, IEEE Journal on Selected Areas in Communications (2008).

[14] Yu Tai Ma, *A VLSI-oriented parallel FFT algorithm*, IEEE Transactions on Signal Processing (1996).

[15] A.B. MacKenzie, J.H. Reed, P. Athanas, C.W. Bostian, R.M. Buehrer, L.A. DaSilva, S.W. Ellingson, Y.T. Hou, M. Hsiaon, Jung-Min Park, C. Patterson, S. Raman, and C. da Silva, *Cognitive radio and networking research at virginia tech*, Proceedings of the IEEE (2009).

[16] John D. Markel, *FFT pruning*, IEEE Transactions on Audio and Electroacoustics (1971).

[17] G. J. Minden, J. B. Evans, L. Searl, D. DePardo, V. R. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill,

A. M. Wyglinski, and A. Agah, *KUAR: A flexible software-defined radio development platform*, Proceedings of the 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN (Dublin), April 2007.

[18] Joesph Mitola, *Cognitive radio for flexible mobile multimedia communications*, Proceedings of the IEEE International Workshop on Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 (San Diego, CA, USA), November 1999.

[19] P.H. Moose, *A technique for orthogonal frequency division multiplexing frequency offset correction*, IEEE Transactions on Communications (1994).

[20] Institute of Electrical and Electronics Engineers, *Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, 2003, IEEE Std 802.11g-2003.

[21] ———, *Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications high-speed physical layer in the 5 GHz band*, 2003, IEEE Std 802.11a-1999(R2003).

[22] Jeffrey D. Poston and William D. Horne, *Discontiguous OFDM considerations for dynamic spectrum access in idle TV channels*, Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN (Baltimore, MD, USA), November 2005.

[23] John G. Proakis, *Digital communications*, 4th ed., McGraw-Hill, 2000.

[24] R. Rajbanshi, A.M. Wyglinski, and G.J. Minden, *Peak-to-average power ratio analysis for NC-OFDM transmissions*, Proceedings of the IEEE 66th Vehicular Technology Conference, Fall (Baltimore, MD, USA), October 2007.

[25] Rakesh Rajbanshi, Qi Chen, Alexander M. Wyglinski, Joseph B. Evans, and Gary J. Minden, *Comparative study of frequency agile data transmission schemes for cognitive radio transceivers*, Proceedings of the first international workshop on Technology and policy for accessing spectrum (New York, NY, USA), January 2007.

[26] Rakesh Rajbanshi, Qi Chen, Alexander. M. Wyglinski, Gary J. Minden, and Joesph B. Evans, *Quantitative comparison of agile modulation techniques for cognitive*

radio transceivers, Proceedings of the 4th IEEE Consumer Communications and Networking Conference (Las Vegas, NV, USA), January 2007.

[27] Rakesh Rajbanshi, Alexander W. Wyglinski, and Gary J. Minden, *An efficient implementation of NC-OFDM transceivers for cognitive radios*, Proceedings of the 1st International Conference on Cognitive Radio Oriented Wireless Networks and Communications (Mykonos Island), June 2006.

[28] T.M. Schmidl and D.C. Cox, *Robust frequency and timing synchronization for OFDM*, IEEE Transactions on Communications, (1997).

[29] Yang Shenyuan, Yang Jie, and Wei Na, *MMSE simplification algorithm in time-domain for OFDM channel estimation*, Proceedings of the 7th World Congress on Intelligent Control and Automation (Chongqing), June 2008.

[30] S. Singh and S. Srinivasan, *Architecturally efficient FFT pruning algorithm*, Electronic Letters (2005).

[31] Steven W Smith, *Digital signal processing: A practical guide for engineers and scientists*, 1st ed., Newnes, 2003.

[32] H.V. Sorensen and C.S. Burrus, *Efficient computation of the DFT with only a subset of input or output points*, IEEE Transactions on Signal Processing (1993).

[33] Timo Weiss, Joerg Hillenbrand, Albert Krohn, and Friedrich K. Jondral, *Mutual interference in OFDM-based spectrum pooling systems*, Proceedings of the IEEE 59th Vehicular Technology Conference, Spring, May 2004.

[34] Timo A. Weiss and Friedrich K. Jondral, *Spectrum pooling: An innovative strategy for the enhancement of spectrum efficiency*, IEEE Communications Magazine (2004).

[35] Xilinx, *Virtex-5 FPGA XtremeDSP design considerations user guide*, [Online:] http://www.xilinx.com/support/documentation/user_guides/ug193.pdf, 2009.

[36] ———, *Xilinx virtex-5 family overview*, [Online:] http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2009.

[37] N. Yee, J. P. Linnartz, , and G. Fettweis, *Multicarrier CDMA in indoor wireless radio networks*, Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (Yokohama, Japan), September 1993.

[38] Zhou Yuan, S. Pagadarai, and A.M. Wyglinski, *Cancellation carrier technique using genetic algorithm for OFDM sidelobe suppression*, Proceedings of the IEEE Military Communications Conference (San Diego, CA, USA), November 2008.

[39] Guoping Zhang and F. Chen, *Parallel FFT with CORDIC for ultra wide band*, Proceedings of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (Singapore), September 2004.

[40] Mark Zwolinski, *Digital system design with VHDL*, 2nd ed., Prentice Hall, 2004.