Implementation of OFDM modem for the Physical Layer of IEEE 802.11a Standard Based on Xilinx Virtex-II FPGA

Farzad Manavi

A Thesis

in

The Department

of

Electrical Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Electrical Engineering) at
Concordia University
Montreal, Quebec, Canada

February 2004

Your file Votre référence
ISBN: 0-612-91079-2
Our file Notre référence
ISBN: 0-612-91079-2

ABSTRACT

Implementation of OFDM modem for the Physical Layer of IEEE 802.11a Standard Based on Xilinx Virtex-II FPGA

Farzad Manavi

In this thesis, a prototype design for the Physical Layer of IEEE 802.11a standard, which is based on Orthogonal Frequency Division Multiplexing (OFDM) technique, is presented. Implementation aspects of an OFDM modem on Xilinx Virtex II field programmable gate array (FPGA) are addressed. The system includes a synchronization circuitry used for packet detection and time synchronization.

The design flow starts with floating-point modeling with parameters specified by IEEE 802.11a for the physical layer of indoor Wireless Local Area Network (WLAN) modems. After algorithmic exploration and performance simulations, the fixed-point refinement is verified to compromise between sufficient arithmetic precision and high-speed hardware necessary for real-time communication systems. At this step of design, different synchronization schemes are examined and as the result of comparison; a modified algorithm based on the delayed correlation of received preamble symbols is selected for hardware implementation.

Finally, the architecture with lowest power consumption and required speed specified by the standard is achieved. This design is efficiently synthesized on 0.15μm /0.12μm CMOS 8-layer metal process Virtex II FPGA. The resulting hardware implementation is simulated at the system clock speed of 72MHz and analyzed from timing point of view to verify adequate performance.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my supervisor Dr. Yousef R. Shayan and truly appreciate his constructive comments and helpful discussions through the development of this research. I am grateful for his support and his help whenever I was in need.

I would like to thank Mr. Tadeusz Obuchowics for his promptly answering my questions and his technical support for CAD tools used in the VLSI lab of Concordia University. I also appreciate the other faculty member and friends who helped and encouraged me throughout my study in the Electrical and Computer Engineering department.

With deep sense of gratitude, I want to thank my parents, my sister and brother. They are very far from me but I wish to share this moment with them. They rendered me enormous support and inspiration during my study. The encouragement and motivation to start my study was given to me by my father.

Lastly, I owe my loving thanks to Sara for the inspiration and moral support she provided through my life in Montreal.

This Thesis is dedicated

to My Parents

TABLE OF CONTENTS

List of Figures.....xi

List of Tables.....xiii

List of Acronyms.....xiv

1 Introduction.....1

1.1 History and Related Works.....1

1.2 Objective of Thesis.....2

1.3 Overview of Thesis.....3

2 Physical Layer Architecture for IEEE 802.11a Standard.....5

2.1 802.11 Wireless Local Area Network.....5

2.1.1 802.11a WLAN Network Architecture.....6

2.1.2 802.11a Physical Layer Components.....9

2.2 OFDM Modulation Technique.....11

2.2.1 Basic Analysis for OFDM systems.....13

2.2.2 FFT and IFFT Analysis.....17

2.2.3 Small-scale Fading in Radio Channels.....19

2.3 OFDM System Evaluation for the PHY of 802.11a.....21

2.3.1 Parameter Calculation for OFDM System Used in 802.11a.....23

2.3.2 IEEE 802.11a Standard Specifications.....25

3 OFDM Modem Design and Simulation.....27

3.1 Implementation design flow.....27

3.2 Floating-Point modeling and Simulation.....30

3.3 Fixed-Point modeling and Simulation.....36

4 FPGA Implementation of OFDM Modem.....40

4.1 OFDM transmitter circuitry.....41

4.1.1 QAM Mapping circuitry.....42

4.1.2 IFFT circuitry.....44

4.1.3 Cyclic Extension circuitry.....50

4.1.4 Transmitter Controller circuitry.....51

4.2 OFDM receiver circuitry.....53

4.2.1 Cyclic Extension Removal circuitry.....54

4.2.2 FFT circuitry.....55

4.2.3 QAM Demapping circuitry.....56

4.2.4 Receiver Controller circuitry.....57

4.3 Implementation and Results.....58

4.3.1 VHDL code generation.....58

4.3.2 VHDL code simulation and test.....60

4.3.3 Synthesis the logic of modem.....61

4.3.4 Mapping the OFDM modem into FPGA and Results.....62

5 Synchronization circuitry for OFDM receiver.....64

5.1 Introduction to OFDM synchronizations.....64

5.2 OFDM synchronization schemes 66

5.3 OFDM synchronization circuitry.....68

6 Conclusion.....73

Appendix A. vfft64 Core pins and functionalities.....76

References.....79

List of Figures

Figure 2-1: IEEE 802 Standard Network Architecture.....6

Figure 2-2: 802.11a Data Link Layer and Physical Layer.....8

Figure 2-3: IEEE 802.11a PLCP frame format.....9

Figure 2-4: OFDM training structure.....9

Figure 2-5: Orthogonality among OFDM subcarriers.....12

Figure 2-6: The Basic structure for OFDM modem.....15

Figure 2-7: Block diagram of an OFDM transceiver.....21

Figure 3.1: Implementation design flow.....28

Figure 3-2: The floating-point model of the OFDM System.....31

Figure 3-3: Subcarrier Frequency allocation.....32

Figure 3-4: Input and Output of IFFT for Transmitter Side.....32

Figure 3-5: 64-QAM constellation bit encoding.....35

Figure 3-6: Bit-Error Rate performance of the OFDM system.....36

Figure 3-7: The fixed-point model for the OFDM system.....37

Figure 4-1: The block diagram of circuitry design for the OFDM modem.....40

Figure 4-2: The block diagram of circuitry design for the OFDM transmitter.....41

Figure 4-3: The circuitry used for QAM mapping.....42

Figure 4-4: Dual-Memory-Space IFFT Core interface.....45

Figure 4-5: DMS core configuration with detailed connection diagram.....46

Figure 4-6: Input data load timing.....47

Figure 4-7: FFT start timing.....48

Figure 4-8: DMS configuration- result and I/O timing.....48

Figure 4-9: The proposed IFFT circuitry for transmitter.....49

Figure 4-10: the circuitry for adding the cyclic extension.....50

Figure 4-11: Transmitter controller circuitry.....52

Figure 4-12: The block diagram of circuitry design for the OFDM receiver.....53

Figure 4-13: the circuitry for the cyclic extension removal.....54

Figure 4-14: The proposed FFT circuitry for receiver.....55

Figure 4-15: The circuitry used for Demapping.....56

Figure 4-16: Receiver controller circuitry.....57

Figure 4-17: the OFDM transmitter circuitry.....59

Figure 4-18: the OFDM receiver circuitry.....60

Figure 5-1: OFDM short preamble symbols.....68

Figure 5-2: Block diagram of OFDM Synchronizer.....69

Figure 5-3: Fixed-point model for synchronization system.....71

Figure A-1: the pins of vfft64 Core.....76

List of Tables

Table 2-1: Contents of the SIGNAL field.....10

Table 4-1: Area Report for OFDM modem.....63

Table 5-1: Area report for the synchronizer circuitry.....72

Table A-1: Defines the module pin functionality.....77

List of Acronyms

ADC Analog to Digital Converter

AGC Automatic Gain Control

BER Bit Error Rate

BW Bandwidth

CSMA/CA CSMA with Collision Avoidance

DAC Digital to Analog Converter

DCF Distributed Coordination Function

DLL Data Link Layer

DMS Dual-Memory-Space

DSSS Direct Sequence Spread-Spectrum

FDM Frequency Division Multiplexing

FFT Fast Fourier Transform

FHSS Frequency Hoping Spread-Spectrum

FPGA Field-Programmable gate Array

FSK Frequency Shift Keying

HDS Hardware Design System

HR/DSSS High-Rate Direct Sequence Spread-Spectrum

ICI Inter-Carrier Interference

IFFT Inverse Fast Fourier Transform

IP Intellectual Property

ISI Inter-Symbol Interference

LAN Local Area Network

LLC Logical Link Control

MAC Medium Access Control

MC Maximum-Correlation

MCM Multi-Carrier Modulation

ML Maximum Likelihood

MMSE Maximum Mean-Square-Error

MNC Maximum-Normalized Correlation

OFDM Orthogonal Frequency Division Multiplexing

OSI Open System Interconnection

PC Point Coordinator

PCF Point Coordination Function

PHY Physical Layer

PLCP Physical Layer Convergence Procedure

PMD Physical Medium Dependent

PSDU PHY Service Data Unit

RF Radio Frequency

RTL Register Transfer Level

RX Receiver

SMS Single-Memory-Space

SPW Signal Processing Worksystem

TMS Triple-Memory-Space

TX Transmitter

VHDL Very high-speed integrated circuit Hardware Description Language

1 Introduction

Orthogonal Frequency Division Multiplexing (OFDM) is a promising technique for achieving high data rate and combating multipath fading in wireless communications [35]. OFDM is one of the multi-carrier modulation (MCM) techniques. In OFDM, the transmitting data stream is divided into several parallel bit streams. Each of these data steams are modulated on individual carries or subcarriers. OFDM is distinguished from the other methods of MCM by its orthogonality property. As an introduction to the thesis, this chapter gives a literature review on OFDM and the organization of thesis.

1.1 History and Related Works

The parallel data transmission method is analyzed for the first time in a paper published in 1967 [31]. In this method, an available bandwidth is divided into several subchannels. These subchannels are independently modulated with different carrier frequencies. It was proved that the use of a large number of narrow channels combats delay and related amplitude distortion in a transmission medium effectively. Based on this concept and some more development OFDM was introduced through a US patent issued in 1970 [32].

In OFDM, the subchannels are orthogonal in frequency domain. The individual carriers have overlap with adjacent carriers but no interference happens among the subcarriers due to orthogonality. This means efficiency in using available bandwidth. The OFDM technique was utilized in some military systems in the 1960s [6].

The discrete Fourier Transform (DFT) was used to simplify the modulation and demodulation of parallel transmission data in 1971 [33]. By using this method, the banks of subcarrier oscillators and coherent demodulators required in the frequency-division multiplexing systems were eliminated.

Most of studies to use OFDM for high-speed modem and high-density recording were done in 1980s. For example, a paper published in 1981 [34] proposed an OFDM system based on the orthogonal multiplexing using DFT and QAM modulation.

The most significant applications of OFDM in 1990s were in Digital Audio Broadcasting (DAB), Digital Video Broadcasting (DVB) and Digital Subscribe Line (DSL). With increasing demand for wireless communication, there has been more attention to use OFDM technique for providing high-speed wireless service.

HiperLAN/2 and IEEE 802.11a are two wireless LAN applications of OFDM, which were introduced in 1999. The former is used in Europe but IEEE 802.11a standard is applied in North America. IEEE 802.11a standard specifies a physical layer based on OFDM modulation technique. Information signal is split into 52 separate subcarriers to provide transmission of data at a rate of 6, 9, 12, 18, 24, 36, 48 and 54 Mbps. IEEE 802.11a physical layer operates in 5 GHz radio frequency band.

1.2 Objective of thesis

IEEE 802.11a standard introduces the specification and parameters necessary for implementation of the physical layer. However, the design and selection of algorithms have been a challenging task since 1999. The literatures in this respect consider different aspects and complexity of implementation. Some of them present algorithms for the

receiver ([30], [36] and [37]). The most differences among proposed schemes are in their synchronization algorithms. There are literatures which discuss only the synchronization schemes (from [20] to [28]).

The objective of this thesis is to design and implement hardware for the physical layer of IEEE 802.11a. A system design is proposed for the OFDM modulator and demodulator. The system includes a synchronizer system which is designed for the receiver of physical layer. Based on a design flow demonstrated in this thesis, the floating-point, fixed-point and RTL models of the proposed OFDM and synchronizer systems are presented and simulated. The results of these three simulations are compared to each other in order to prove the correctness of functionality. The finalized circuitries for the systems are synthesized and mapped into Xilinx Field-Programmable Gate Array (FPGA).

During implementation of the OFDM modem, the overview of different schemes and algorithms are provided. The OFDM system is designed based on the performance required by the standard. Therefore, the algorithms provided for OFDM modems in other applications are modified to meet the specifications introduced by the IEEE 802.11a standard. In this thesis, the proposed circuitries for the OFDM transmitter, receiver and synchronizer are explained in details.

1.3 Overview of thesis

The next chapters of this thesis are organized as follows: Chapter 2 provides an introduction to the physical layer of IEEE 802.11a standard. First the network architecture of the standard is explained. Then OFDM as the modulation technique used

for the physical layer is analyzed. Finally, the specification and parameters of IEEE 802.11a and are presented at the end of the chapter.

Chapter 3 is dedicated to the system design and modeling of an OFDM modem that is used in the physical layer of IEEE 802.11a. The design flow, which is followed in this thesis, is demonstrated. The floating-point and fixed-point model for the OFDM modem are proposed and simulated. It is shown how fast and effective is the utilized design method to prototype the modem.

Chapter 4 is about the implementation aspects of the OFDM modem. First, the circuitry is designed for the system discussed in chapter 3. It is shown how the RTL model of the modem circuitry can be prepared with the available libraries and resources. The RTL model is also mapped into Xilinx Virtex II FPGA. The result and gate count for the designed OFDM modem are summarized in this chapter.

Synchronization in the OFDM modem is explained in chapter 5. The synchronization of OFDM receiver is separately discussed in this chapter because of complexity and importance of the issue. Several methods for the frame detection and time synchronization are explained. A synchronization system that provides the proper performance is proposed. Floating-point and fixed-point models for the proposed system are also presented in this chapter. Besides, the final RTL model of the synchronizer circuitry is mapped into Xilinx Virtex II FPGA.

Chapter 6 summarizes the works presented in the thesis and concludes the results. Appendix A includes the data sheets necessary for hardware implementation.

2 Physical Layer Architecture for IEEE 802.11a Standard Based on OFDM Technique

This chapter is an introduction to the physical layer of IEEE 802.11a standard and its architecture. First, the protocol and most frequently used acronyms are explained. Then the concept of the OFDM system as a modulation technique used in this standard is studied. The complexity and mathematical model is described and finally the specifications of IEEE 802.11a standard for the physical layer are demonstrated.

2.1 IEEE 802.11 Wireless Local Area Network

Ethernet technology has been used for wired Local Area Networks (LAN) for many years. Due to the increasing demand for mobile computer networks, in 1997, the standard committee of IEEE society drew up a wireless LAN (WLAN) standard called IEEE 802.11. This standard is an adaptation between widely used Ethernet and the wireless world. Because of the different nature of wireless medium, 802.11 standard needs to have additional management features and special physical layers.

Figure 2-1 shows the relationship between the members of IEEE 802 family and their places in OSI model. The upper networking layers are common for all 802 families. Therefore, the major differences between Ethernet and 802.11 are laid in Data Link Layer (DLL) and Physical Layer (PHY).

As specified in 802.11 standard, Data Link Layer consists of Logical Link Control (LLC) and Medium Access Control (MAC) sublayers. LLC hides the differences among 802 family members and Ethernet. It makes them indistinguishable as far as the network

layer is concerned. MAC determines how to access the medium and send data by doing the required setup for PHY. Meanwhile, PHY is dedicated to handle the details of data transmission and reception between two or more stations. Among the components depicted in Figure 2-1, only 802.11a will be briefly described in the next two subsections, as being used in this project.

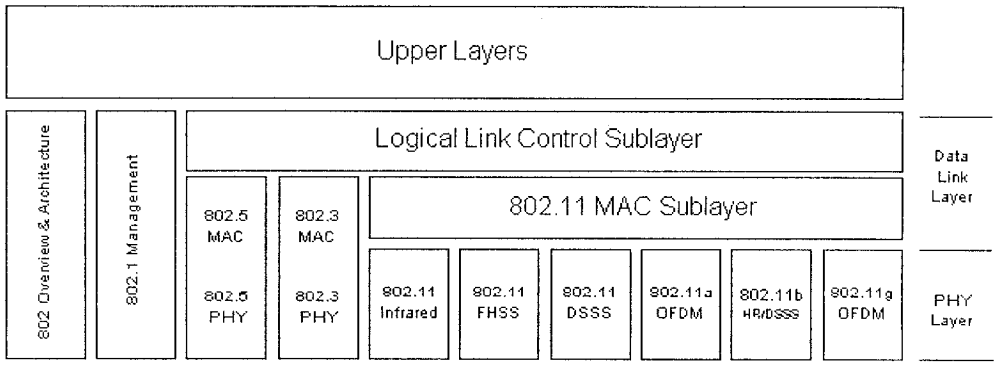

Figure 2-1: The Network Architecture for IEEE 802 Standard

2.1.1 IEEE 802.11a WLAN Network Architecture

As shown in Figure 2-1, the MAC sublayer for all 802.11 families is common and the differences start to be evident only in PHY. In 802.11, MAC has two modes of operation: Distributed Coordination Function (DCF) and Point Coordination Function (PCF).

DCF is the fundamental access method of 802.11 MAC. In this method, MAC does not use any central control. Stations compete for airtime. This occurs by utilizing a protocol called CSMA/CA (CSMA with Collision Avoidance). On the other hand in PCF, Point Coordinator (PC) asks the stations whether they have any frame to send or not. Then it gives privilege to those which have frames to send. Since the order of

transmission data is completely controlled by a base station in PCF mode, no collision ever occurs.

As we can see from Figure 2-1, there are five techniques used in 802.11 to send a MAC frame between two network points. Three of them are infrared, Frequency Hopping Spread-Spectrum (FHSS) and Direct–Sequence Spread Spectrum (DSSS). Because of the demand for higher bit rate data transmission, the standard committee of IEEE society added two other physical layers to 802.11 families in 1999. They are IEEE 802.11b and 802.11a. The former specifies PHY based on high-rate Direct-Sequence Spread-Spectrum (HR/DSSS) technique and the latter is based on Orthogonal Frequency Division Multiplexing (OFDM).

The major difference between 802.11a and 802.11b is in their modulation technique. IEEE 802.11b utilizes HR-DSSS with 11 million chips per second to achieve 11 MBps data rate in 2.4 GHz radio frequency band. Data rates in 802.11b are 1, 2, 5.5 and 11 MBps. However, the maximum data rate in 802.11a is 54 MBps and this technique uses a radio frequency band equal to 5 GHz.

In November 2001, the standard committee of IEEE society drew a new standard called 802.11g, which uses OFDM modulation technique but it operates in 2.4 GHz as its radio frequency band. IEEE 802.11g can transmit data with a rate up to 54 MBps, same as 802.11a. As it will be explained in the next chapter, OFDM has good spectrum efficiency and good immunity to multipath fading. It is also compatible with European HiperLAN/2 standard.

Before moving into the details of 802.11a PHY, it is necessary to define two more acronyms PLCP and PMD. Due to the nature of wireless environment, WLAN requires a

relatively complex physical layer. Therefore, IEEE 802.11 standard splits the PHY into two generic components: the Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD). Figure 2-2 shows these two components.

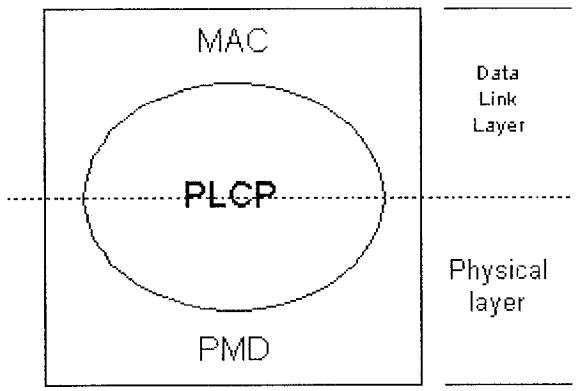

Figure 2-2: 802.11a Data Link Layer and Physical Layer

PLCP maps MAC frames by adding a number of fields to MAC frames. It defines a method to make PHY Service Data Unit (PSDU) into a framing format. The format is suitable for transmitting data between two or more stations using the associated PMD system. On the other hand, this is the PMD responsibility to transmit PLCP frames with radio waves through the air. In other words, PLCP sublayer makes it possible that 802.11 MAC operates with minimum dependency on the PMD sublayer. 802.11a PMD sublayer uses 5GHz band with OFDM modulation technique as a mean to transmit and receive data (including management information of the upper layer) between two or more stations.

By dividing the main layers into sublayers as mentioned before, the standard makes the architecture of 802.11 MAC independent of PHY. One of the advantages of 802.11 can be highlighted as flexibility and adaptability of this standard while all of its

complexity is hidden in its implementation. In the next section, some specifications and parameters which are necessary for 802.11a PHY implantation are demonstrated. References [1], [2] and [3] give the complete details of 802.11 network layers and sublayers.

2.1.2 IEEE 802.11a Physical Layer Components

The 802.11a physical layer adds PLCP preamble, header and tailing bits to data as depicted in Figure 2-3.

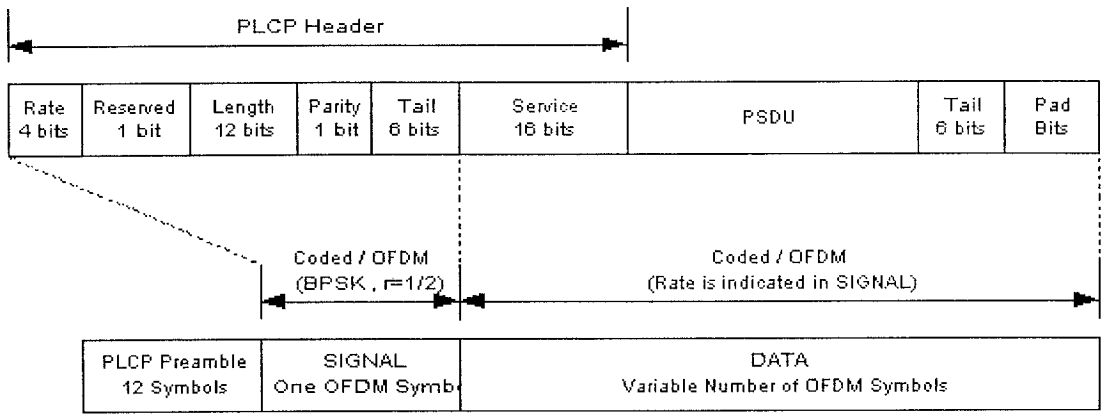

Figure 2-3: IEEE 802.11a PLCP frame format

The first part is preamble symbols, which lasts for 16 μS. It consists of a sequence of 10 short and 2 long training symbols. A guard time interval equal to 1.6 μS is also inserted between these two types of symbol as depicted in Figure 2-4.

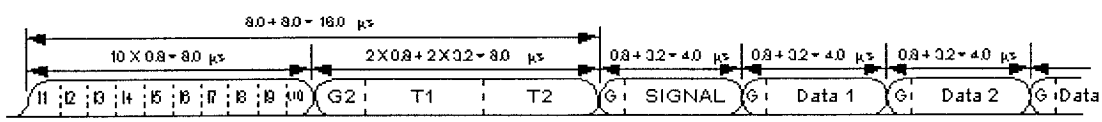

Figure 2-4: OFDM training structure

The guard time protects the long training symbols against multipath fading. The PLCP preamble is generally used for synchronization. The symbols t_1 to t_{10} in this figure are repeated symbols containing 16 samples each. They consist of 12 subcarriers. The short training symbols are used at the OFDM receiver for frame detection; coarse frequency offset estimation and large-scale time synchronization. They also can be used in diversity selection (such as antenna selection) and Automatic Gain Control (AGC) circuitries. After 1.6 μs guard time interval, two long training symbols are transmitted, which consist of 53 subcarriers (including a zero value at dc). Long symbols are used for fine-tuning, timing acquisition and improving channel estimation accuracy. The elements and other details of training symbols will be provided at the end of this chapter. The synchronization methods used in an OFDM receiver will be explained in Chapter 5.

As shown in Figure 2-3, PLCP header is transmitted at the beginning of signal and service field. Each part encodes the parameters needed during transmission. For example, the first four bits are used for specifying the data rate to the transmitter and receiver according to Table 2-1.

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

Table 2-1: Contents of the SIGNAL field

The other parts of PLCP headers are length, parity, tail and service bits. The final 16 bits unlike the other components are placed in the data field of PHY and used for initializing the scrambler of MAC frames. The details of MAC and PLCP are explained in reference [2].

2.2 OFDM Modulation Technique

As mentioned before, the modulation technique used in 802.11a PHY is OFDM, which is very closely related to another technique called Frequency Division Multiplexing (FDM). FDM means dividing the available bandwidth (BW) into slices called subcarriers. Each radio channel (or subcarrier) is dedicated to a network user to transfer its information. To protect a user from its spectral leakage to adjacent users, two guard bands are placed around the user's dedicated channel. This technique was widely used in the first generation of mobile telephones. Since guard bands are not used for transmitting information, they waste bandwidth and reduce the channel capacity. One solution to this problem is to overlap the channels without interfering them with each other. This can be achieved if signals have orthogonality property.

As shown in Figure 2-5, orthogonality means that the peak of each subcarrier exactly happens when the other signals have zero amplitude. It can also be seen that if data is multiplexed over a set of orthogonal subcarriers, more subcarriers will be transmitted through the bandwidth. This property increases the bandwidth efficiency of the OFDM technique.

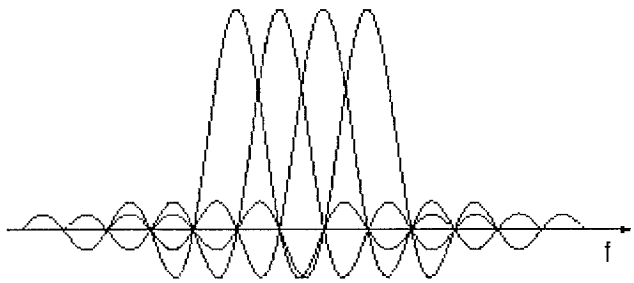

Figure 2-5: Orthogonality among OFDM subcarriers

Therefore, the basic principle of OFDM is to divide a high-rate data stream into a number of lower-rate streams that are transmitted over a number of multiplexed orthogonal subcarriers [6]. For example in an OFDM system with 7 subcarriers, the signal duration increases 7 times but instead of one carrier, there are 7 parallel orthogonal subcarriers in the system. No spectrum spreading happens and the bandwidth is identical to a single carrier system carrying the high-rate incoming data.

Because of symbol rate reduction in an OFDM system, transmitted signals are less affected by the channel time dispersion caused by multipath delay spread. This phenomenon is counted as the dominant impairment for the indoor wireless signal.

At the cost of a small reduction in bandwidth efficiency, a guard time is introduced to OFDM symbol at the transmitter. This technique avoids inter-symbol interference (ISI). If an OFDM symbol is cyclically extended into the guard time interval, it will protect the orthogonality among subcarriers and eliminates inter-carrier interference (ICI). As long as the delay spread is smaller than the guard time interval, there is no ISI or ICI within the OFDM symbols.

In the following subsection, first an OFDM system will be mathematically modeled. Then, it is shown that Inverse Fast Fourier Transform (IFFT) can be used in the

transmitter to generate an orthogonal signal in the frequency domain. On the other hand at the receiver side, the subcarriers are extracted from the received waveform by using Fast Fourier Transform (FFT) with a window outside the guard time interval.

2.2.1 Basic Analysis for OFDM Systems

An OFDM signal consists of a sum of digitally modulated (such as PSK or QAM) subcarriers transmitted in parallel. In general form we have:

$$s(t) = \sum_{-\infty}^{\infty} s_n(t) \qquad (2-1)$$

where $s_n(t)$ is the transmitted signal for the OFDM symbol number n. If this symbol starts at $t = t_s$, then one OFDM symbol is:

$$s_n(t) = \begin{cases} \mathrm{Re}\left\{ \sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} d_{i+N_s/2} \exp(j2\pi(f_c - \frac{i+0.5}{T})(t-t_s)) \right\} & , \ t_s \le t < t_s + T \\ 0 & otherwise \end{cases} \qquad (2-2)$$

where d_i is the complex QAM symbol. Parameters N_s and f_c are the number of subcarriers and the carrier frequency respectively. Parameter T is the OFDM symbol duration. The Real part of the complex variable is represented by $\mathrm{Re}\{\cdot\}$. For simplicity, the baseband notation is used in this analysis as shown in Equation (2-3):

$$s_n(t)=\begin{cases}\sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} d_{i+N_s/2}\exp(j2\pi\frac{i}{T}(t-t_s)) & , \ t_s\le t<t_s+T \\ \\ 0 & otherwise\end{cases} \tag{2-3}$$

The real and imaginary parts of this equation have to be multiplied by a cosine and sine of the desired carrier frequency to produce the final OFDM signal. Because of the orthogonality property, each subcarrier has an integer number of cycles in one OFDM symbol with period T. On the other hand, Equation (2-3) is the mathematical definition of inverse discrete Fourier transform for a QAM or BPSK symbol. At the receiver side, the transmitted k^{th} subcarrier can be extracted by down converting it with a frequency of k/T and then integrating the signal over T seconds. So the QAM value for a particular subcarrier comes from:

$$\int_{t_s}^{t_s+T}\exp(-j2\pi\frac{k}{T}(t-t_s))s_n(t)\,dt \tag{2-4}$$

$$=\int_{t_s}^{t_s+T}\exp(-j2\pi\frac{k}{T}(t-t_s))\sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} d_{i+N_s/2}\exp(j2\pi\frac{i}{T}(t-t_s))\,dt$$

$$=\sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} d_{i+N_s/2}\int_{t_s}^{t_s+T}\exp(j2\pi\frac{i-k}{T}(t-t_s))\,dt$$

$$=d_{k+N_s/2}\,T$$

that gives the desired output $d_{k+N_s/2}$ multiplied by a constant factor T. Since the frequency difference $\frac{i-k}{T}$ is an integer number of cycles within the integration interval

T, the integration result is always zero except for $i = k$. Equation (2-4) is the mathematical definition of the Fourier transform of $s_n(t)$. Figure 2-6 shows the basic structure of an OFDM system including modulator and demodulator.

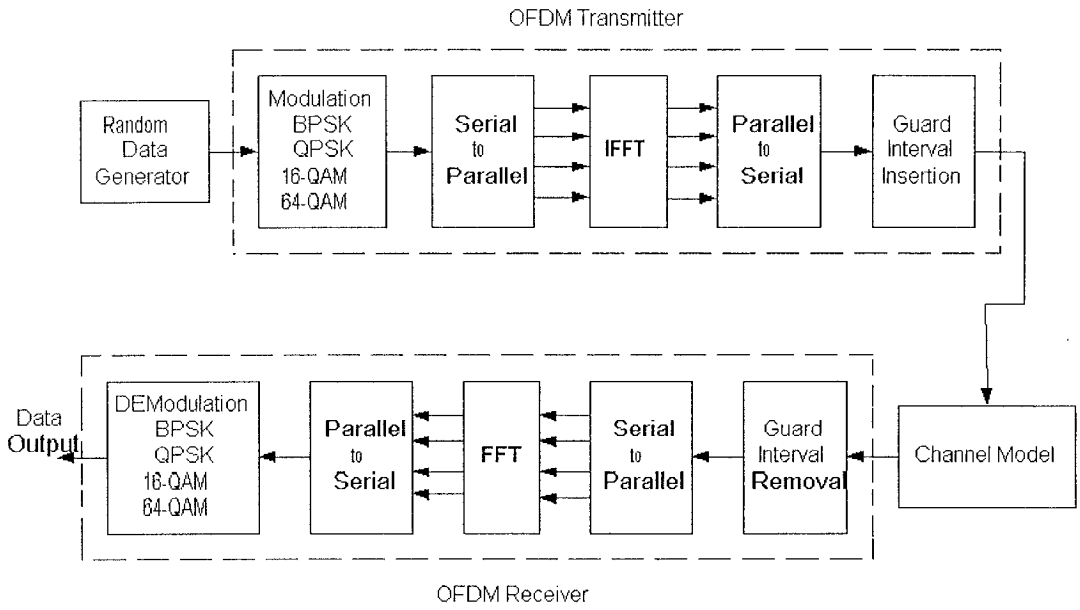

Figure 2-6: The Basic structure for OFDM modem

There are two more blocks to be added to this model to make it closer to real OFDM system. These are inclusion of guard time interval for ISI and ICI protection and windowing for pulse shaping to limit the transmission spectrum. So Equation (2-3) is rewritten as:

$$s_n(t) = \begin{cases} W_r(t-t_s)\sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} d_{i+N_s/2}\exp(j2\pi\frac{i}{T}(t-t_s-T_g)) & ,t_s \le t < t_s+T_g+T(1+\beta) \\ 0 \qquad\qquad otherwise \end{cases} \tag{2-5}$$

In Equation (2-5) $W_r(t-t_s)$ is the pulse-shaping filter and T_g is the guard time interval. The guard time duration needs to be more than multipath delay spread time of the channel. The parameter β is roll-off factor for the window. In order to maintain orthogonality of subchannels and preventing OFDM symbols from ICI, a cyclic prefix as a guard time interval is added to each OFDM symbol. The performance will improve with a price paid for a loss in both transmission power and bandwidth efficiency. Including a cyclic prefix, Equation (2-5) is modified as:

$$s_n(t)=\begin{cases} W_T(t-t_s)\sum_{i=\frac{N_s}{2}-1-L}^{\frac{N_s}{2}-1} d_{i+Ns/2}\ \exp(j2\pi\frac{i}{T}(t-t_s)) & , t_s \le t < t_s+T_g \\ W_T(t-t_s)\sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} d_{i+Ns/2}\ \exp(j2\pi\frac{i}{T}(t-t_s-T_g)) & , t_s+T_g \le t < t_s+T_g+T(1+\beta) \\ 0 \qquad otherwise & \end{cases} \tag{2-6}$$

L is the number of samples copied from the end of the symbol in the guard time. Therefore, the total OFDM symbol duration is $T+T_g$. But if $s(t)$ passes through a multipath channel whose impulse response is $h(t)$, the received signal will be as

$$r(t)=h(t)*s(t) \tag{2-7}$$

If additive noise is also considered in this model and we assume that the transmitted signal is sampled at the receiver with sampling period T_s. This equation can be written as:

$$r(kT_s)=\sum_{m=0}^{M-1} h_m(kT_s)\ s(kT_s-\tau_m)+n(kT_s) \tag{2-8}$$

In this equation, M is the number of multipaths in the channel model. Parameter $h_m(t)$ denotes the equivalent low pass impulse response for m^{th} multipath component. τ_m is delay time for m^{th} multipath component and T_s is the sampling period and equals to T/N_s.

According to 802.11a standard a complete block diagram needs to have some more blocks for error control, synchronization and channel estimation. Section 2.3 discuses the complete block diagram for 802.11a. Before that, FFT/IFFT algorithms and also fading in multipath channels have to be discussed.

2.2.2 FFT and IFFT Analysis

FFT/IFFT as the main blocks for an OFDM system are considered in this subsection. Assume a sequence with a finite number of samples { g_k }, k = 0, 1,..., N -1. Fourier transform is defined as:

$$G(f)=\sum_{k=0}^{N-1} g_k.\exp(-j2\pi fKT_c) \qquad (2-9)$$

where T_c is sampling period and $f=\frac{m}{NT_c}$ for m = 0,1,..., N -1. Equation (2-9) can be written as:

$$G(\frac{m}{NT_c})=\sum_{k=0}^{N-1} g_k.\exp(-j2\pi\frac{m}{N}K) \qquad (2-10)$$

For convenience in notation, the complex quantity W_N is defined as:

$$W_N = \exp(-j\frac{2\pi}{N}) \qquad (2-11)$$

So, Equation (2-10) is written as:

$$G_m = \sum_{k=0}^{N-1} g_k . W_N^{km} \quad , \quad m=0,1,....,N-1 \qquad (2-12)$$

And the inverse equation is:

$$g_k = \frac{1}{N}\sum_{m=0}^{N-1} G_m . W_N^{-km} \quad , \quad k=0,1,....,N-1 \qquad (2-13)$$

As it can be seen from equation (2-9), the direct computation of discrete Fourier transform requires $N(N-1)$ complex additions and N^2 complex multiplications. However, an algorithm called Fast Fourier Transform (FFT) allows discrete Fourier transform of a sequence of samples to be computed faster and more efficiently. As a condition for applying this algorithm, the number of samples in the sequence should be a power of 2. There are also some techniques used in FFT to increase the speed and decrease the complexity of computations. For example by using the radix-2 algorithm, an N-point FFT requires only $(\frac{N}{2}\log_2^N - N)$ complex multiplications and $N\log_2^N$ complex additions. Using a radix-4 algorithm can reduce the number of multiplications in FFT even further. The radix-4 butterfly can be used to efficiently build an FFT with a larger size [11], [12]. Another complexity advantage of FFT is the fact that FFT does not really require full multiplication, but rather phase rotation that can be efficiently implemented by the COrdinate Rotation DIgital Computer (CORDIC) algorithm [29].

Before defining the specification of 802.11a PHY, the fading channel parameters are briefly introduced in the next subsection. These parameters characterize a channel and they should be considered in the specification of a modem that deals with a wireless communication environment.

2.2.3 Small-scale Fading in Radio Channels

In a wireless channel, the transmitted signal passes through different paths on the way to the receiver due to existence of obstacles. Each path has delay and attenuation factor that is time variant. The amplitude variations in the received signal are due to the time variant multipath characteristics of the channel. Assume $C(\tau , t)$ is the impulse response of a channel when τ is the arriving time for the delayed version of a transmitted signal. If the impulse response of a channel is modeled as a zero-mean complex-valued Gaussian process, the envelop $| C(\tau , t) |$ at any instant t is Rayleigh distributed. For nonzero-mean case, the envelope is Rice distributed. The average received multipath power is an exponentially decaying function of excess decay for Rice and Rayleigh distributions [10].

Excess delay is the relative delay of the i^{th} multipath component as compared to the first arriving component. Since almost all delays are only a few percent smaller than the maximum measured delay spread, this parameter gives a good sense for the channel behavior in time domain.

The characteristics of a time varying channel can be described by the coherence time and Doppler spread. The coherence time is the duration over which the channel characteristics do not change significantly. Doppler spread in the frequency domain is a

measure of the spectral broadening caused by the time rate of changing in a mobile radio channel. It is defined as a range of frequencies over which the received Doppler spectrum is essentially non-zero. The spectrum of a sinusoid signal (with a constant envelope) transmitted through a channel determines Doppler spectrum. Slow changing channel means large coherence time or small Doppler spread.

Based on multipath time delay spread, we have two types of small scale fading: flat fading and frequency selective fading. And based on Doppler spread, we have: fast fading and slow fading. Fading affections due to Doppler spread can be summarized as:

Bit Time of transmitted signal (T_b) >> Coherence Time $(\Delta t)_c \Rightarrow$ Fast fading

Bit Time of transmitted signal (T_b) << Coherence Time $(\Delta t)_c \Rightarrow$ Slow fading

The Doppler spread and coherence time are inversely proportional to one another. A popular rule of thumb for modern digital communication is to define the coherence time as [10]:

$$\text{Coherence time: } (\Delta t)_c = \frac{0.423}{f_m} \tag{2-14}$$

and f_m is Doppler Shift and defined as:

$$\text{Doppler Shift: } f_m = (v / \lambda).\ \text{Cos}\ \alpha_m \tag{2-15}$$

where v is the constant velocity of moving receiver. λ is wavelength and equals to $\frac{the\ speed\ of light}{f_c}$ and α_m denotes the angle between transmitter and direction of mobile system.

Time delay spread was mentioned before when the guard time was defined in OFDM systems. We saw that OFDM technique improves the robustness against frequency selective fading by increasing the time duration of symbols. In the next section, the parameters necessary for designing an OFDM system is calculated.

2.3 OFDM System Evaluation for the PHY of 802.11a

Consider the OFDM modem as shown in Figure 2-7. This modem includes all blocks necessary for a practical wireless communication to have adequate performance.

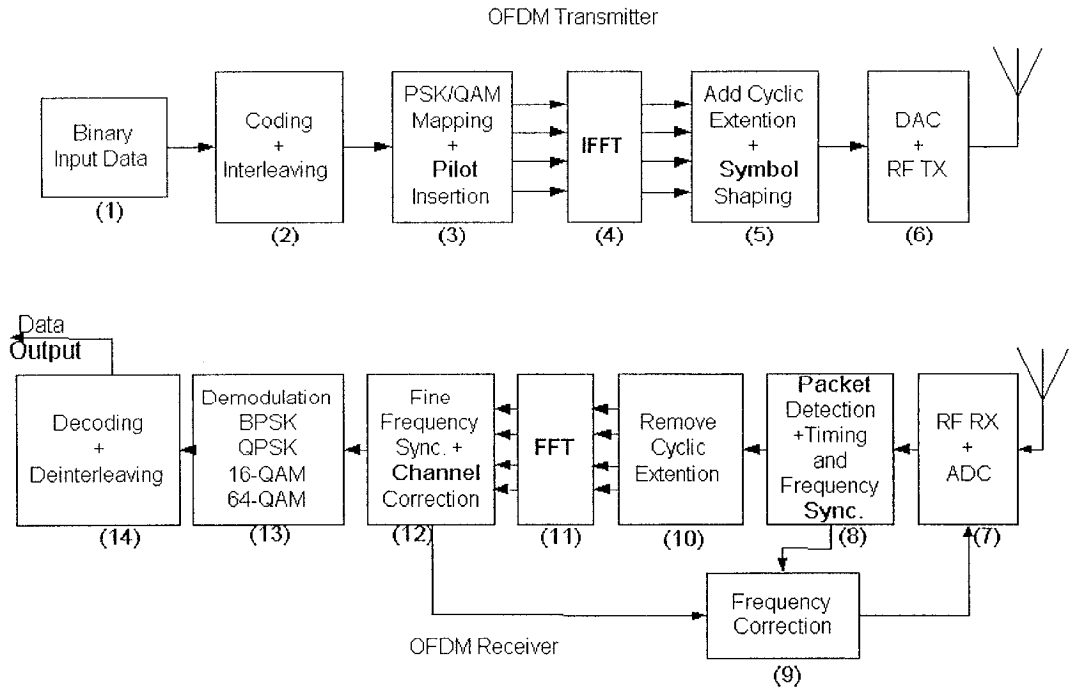

Figure 2-7: Block diagram of an OFDM transceiver

Blocks (3), (4), (5), (10), (11) and (12) are the same as the ones in Figure 2-6 and have the same functionality as described before. For simplicity, some blocks are shown with multiple functionalities but in reality they can include several separate sub blocks.

In block (2), there is a convolutional encoder as specified in 802.11a standard to improve the performance of the system. For a wireless design working in RF band, there are some blocks such as Analog to Digital Converter (block (7)), Digital to Analog Converter (block (6)), Radio Frequency transmitter (RF TX) or Radio Frequency Receiver (RF RX). However, in a baseband design there is no need to consider these blocks. Blocks (8), (9) and (12) are for the packet detection and synchronization purpose. Pilot symbols also are needed for channel estimation and fine-time synchronization process and they are added in block (3). These symbols are removed in block (12) and used for the channel estimation for fine frequency synchronization. Block (14) is decoder, which uses Viterbi algorithm.

To design an OFDM system as depicted in Figure 2-7, we need to calculate the following parameters:

- Number of subcarriers
- Guard time
- Symbol duration
- Subcarrier spacing
- Modulation type per subcarrier
- Type of forward error correction coding

The parameters that are necessary for calculating the above parameters are:

- Available bandwidth
- Required bit rate
- Tolerable delay spread
- Doppler values

In the next subsection, the parameters for an illustrative OFDM system are calculated. The example is very close to the OFDM modem used for 802.11a PHY. The final section gives the summarized specifications. The differences between calculated parameters in the next section and the standard specification can be compared.

2.3.1 Parameter Calculation for OFDM System Used in 802.11a

Now, as an example, we evaluate the parameters for an OFDM system with the following assumptions:

- Bandwidth < 20 MHz
- Delay Spread 200 ns
- Bit rate 54 Mbps

This assumption is based on the maximum possible bit rate in 802.11a standard according to table (2-1). The delay spread about 200 ns is suitable for indoor WLAN systems [10]. A guard time about four times the delay spread is a safe value to prevent ISI in the receiver. Therefore, an 800 ns guard time satisfies the safe condition for data transmission without ISI.

OFDM symbol duration should be at least five times the guard time [6]. So, the total symbol duration including guard time is 4.0 μs. This value is also equal to the OFDM symbol duration assigned by IEEE 802.11a standard. The subcarrier spacing for 3200ns OFDM symbol without guard time is 312.5 kHz.

Since the maximum bit rate is 54 Mbps and symbol duration is 4.0 μs, each symbol has to carry 216 bits. If we use 64-QAM for modulation and convolution coding (with rate ¾) for the coding part, there will be $6 \times 3/4 = 4.5$ bits per symbol per

subcarrier. So the number of subcarriers for an OFDM symbol is 216 / 4.5 = 48 subcarriers.

Subtracting the space occupied by 48 subcarriers from the assigned bandwidth (20 MHz), the extra available bandwidth will be:

$$20 \text{ MHz} - (48 \times 312.5\text{kHz}) = 5.0 \text{ MHz}$$

This part of bandwidth can be used for three proposes:

- Leaving some zero subcarriers to provide oversampling necessary to avoid aliasing
- Windowing
- Pilots for channel estimation

The number of subcarriers should be at least four times the constraint length of the convolution coding part. Practically, the constraint length should be less than 10 [6]. If the complexity of implementation is concerned, the constraint length of 7 is proper for the design.

For FFT/IFFT block, this OFDM system has 64 subcarriers and therefore for the transmitter part, we can use 64-point IFFT containing stages with four radix-4 butterflies. A 64-point FFT using the radix-4 algorithm requires only 96 complex multiplications or phase rotations and 384 additions. The extra zeros should be added in the middle of the data vector rather than appending them at the end. The calculated parameters are summarized as:

- Guard time: 800ns
- Symbol duration: 4.0μs
- Subcarrier spacing: 312.5 kHz

- Number of subcarriers: 64
- Modulation: 64-QAM
- Convolution Coding with rate ¾ and constraint length 7
- IFFT/FFT: 64-point using radix-4 algorithm

These parameters are calculated according to the assumptions done at the beginning of this subsection. In reality, these assumptions are true for the OFDM system used in IEEE 802.11a PHY. However, the bit rate 54 MHz is the maximum bit rate permitted by the standard. For a lower bit rate, only the parameters related to the type of modulation (such as BPSK, 16-QAM and 64-QAM) and Convolution coding rate need to be changed while other parameters remain the same. The next subsection summarizes the specifications assigned by IEEE 802.11a standard for the physical layer.

2.3.2 IEEE 802.11a Standard Specifications

Major parameters of the OFDM PHY [2] are as follows.

- Information data rate: 6, 9, 12, 18, 24, 36, 48 and 54 Mbps (6,12 and 24 Mbps are mandatory)
- Modulation: BPSK OFDM, QPSK OFDM, 16-QAM OFDM,64-QAM OFDM
- Error correcting code: K = 7 (64 states) convolutional code
- Length: 1 - 4095
- Coding rate: 1/2, 2/3, 3/4
- Number of subcarriers, total: 52
- Number of data subcarriers: 48
- Number of pilot subcarriers: 4

- Subcarrier frequency spacing: 0.3125 MHz (= 20 MHz/64)
- IFFT/FFT period: 3.2 μs
- PLCP preamble duration: 16 μs ($T_{short} + T_{long}$)
- OFDM symbol duration: 4.0 μs
- Guard time interval: 0.8 μs
- Occupied bandwidth: 16.6 MHz

Most of the parameters specified by the standard are equal to the assumed parameters given for the example discussed in subsection 2.3.1. However, the total number of subcarriers in 802.11a PHY is 52. This value comes from 48 data subcarriers plus four Pilot subcarriers. There are also 12 zero subcarriers, which make 64 subcarriers for IFFT block. Arranging of subcarriers will be discussed in the next chapter, where an OFDM system is designed as shown in Figure 2-6.

3 OFDM Modem Design and Simulation

In this chapter, the design flow of an OFDM modem used for the physical layer of IEEE 802.11a is demonstrated. The system at each step of design is modeled and the simulation results are provided. The block diagram of an OFDM system depicted in Figure 2-6 is the base for this chapter. The implementation method, finalized circuitry and system simulation results are described.

3.1 Implementation design flow

The design flow chosen for this project is shown in Figure 3.1. The system design begins with floating-point modeling. The floating-point model is prepared based on the mathematical model. This model lets us to explore and compare the performance of different algorithm and schemes. System modification and optimization should be done at this step of design. For this OFDM modem design, the environment used for floating-point modeling is Signal Processing Worksystem (SPW) from Cadence. All libraries necessary for modeling and simulating an OFDM system exist in SPW.

Bit Error Rate (BER) verses bit energy to noise spectral density (E_b / N_0) curve is an important measurement to evaluate the performance of a communication system. Therefore, the BER simulation result of the floating-point model can be compared with references or proven equations to evaluate the performance of the design.

As depicted in Figure 3-1, the second step in design flow is fixed-point modeling and simulation. It starts with converting the block diagrams of floating-point model into fixed-point model.

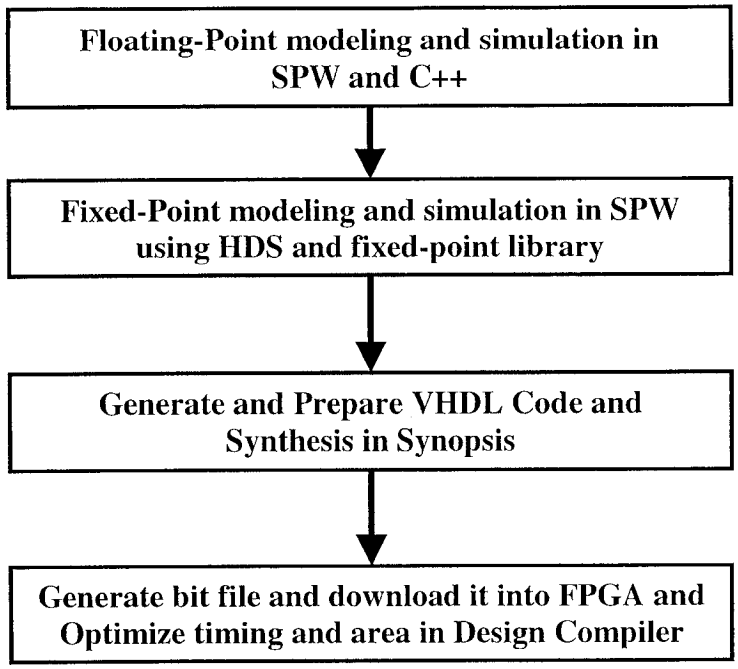

Figure 3-1: Implementation design flow

The number of bits assigned to each block determines the complexity of the system. The greater is this value; the more complex is the system. On the other hand, decreasing number of assigned bits increases the calculation error in each block. This leads to lose some performance in the system. Therefore, there is always a compromise between the complexity and performance of the system. By comparing the BER curve of the fixed-point model and the one obtained from floating-point simulation, the performance degradation due to having finite number of bits in the fixed-point model is determined.

When the numbers of assigned bits are completely identified for the design, it is time to generate a Register Transfer Level (RTL) model for the system in order to synthesize the circuitry. The environment used for this purpose is Hardware Design System (HDS), which is a set of libraries from SPW. The blocks inside the floating-point

model are replaced with the libraries from HDS. At this step we need to identify the combinational and sequential logics that are important to synthesis the circuitry. For example some blocks like control systems or some parameters like clock speed are designed in HDS modeling. The functionality of HDS model can be checked with fixed-point model. The same as previous step, the input of fixed-point model are given to HDS model and the result can be compared with each other.

One of the major advantages of HDS modeling is to generate Very high-speed integrated circuit Hardware Description Language (VHDL) codes. VHDL codes can be generated easily by running only a command in HDS. Some blocks such as FFT/IFFT do not exist in HDS. For simulation purpose, the same floating-point blocks can be used in HDS. However for implementation, they should be imported from other libraries (such as Xilinx Coregen). Some VHDL codes are required to be prepared manually.

After VHDL code generation, they are synthesized in Synopsis Design Compiler targeting FPGA as the hardware for implementation. The area and/or time constraints should be given to the compiler. After generating a synthesized file, it is possible to analyze the circuitry in "Design & Analyzer" tool. "Design & Analyzer" provides the environment that we can synthesize and analyze the VHDL. For the blocks, which do not meet timing constrains, we need to change the design like using pipeline architecture or to increase the compiler's effort for synthesizing VHDL codes.

The last step in Figure 3-1 is to generate a bit file. The synthesized circuitry is mapped into FPGA. This task is called "Place and Rout" and final report from this step gives the exact time delays and gate counts. Another important result of "Place and Rout" is to generate a file called "Bit File". The file can be downloaded into FPGA to realize

the circuitry. This step is mostly done automatically using Xilinx Design Manager. As another test to check the system, the gate level of circuitry is simulated and the results are compared with simulation results achieved from previous step.

The next two sections are dedicated for the floating-point and fixed-point modeling and simulation. It will be shown how the modeling and simulations are done in SPW.

3.2 Floating-Point Modeling and Simulation

Figure 3-2 depicts the floating-point model for the OFDM modem used in IEEE 802.11a PHY. The model includes the transmitter, the receiver and the channel. In this model, "Random Data" block generates binary '0' and '1'input data, which are uniformly distributed. The number of sample per input bit is one. The next block, called "Number to Symbol", maps binary data into an integer number. For 64-QAM, each six bits is mapped into a number between '0' to '63' level. "Number to Symbol" block is a lookup table, which gives I and Q values to each integer input. The values are assigned according to constellation defined for 64-QAM. Then, there is a serial to parallel block that is occurred in "Complex Scalar to Vector" block. Each 48 QAM symbols coming to this block make a vector at its output. The following block is "Insert Pilot" which adds four pilots to the vector constructed by the previous block. The detail of how to insert these pilots into the vector is demonstrated in Figure 3-3. This figure shows the subcarrier frequency allocation as specified by IEEE 802.11a Standard [2]. The 48 QAM symbols and four pilots *are* shown by d_0 to d_{47} and P_{-21}, P_{-7}, P_7, P_{-21} respectively.

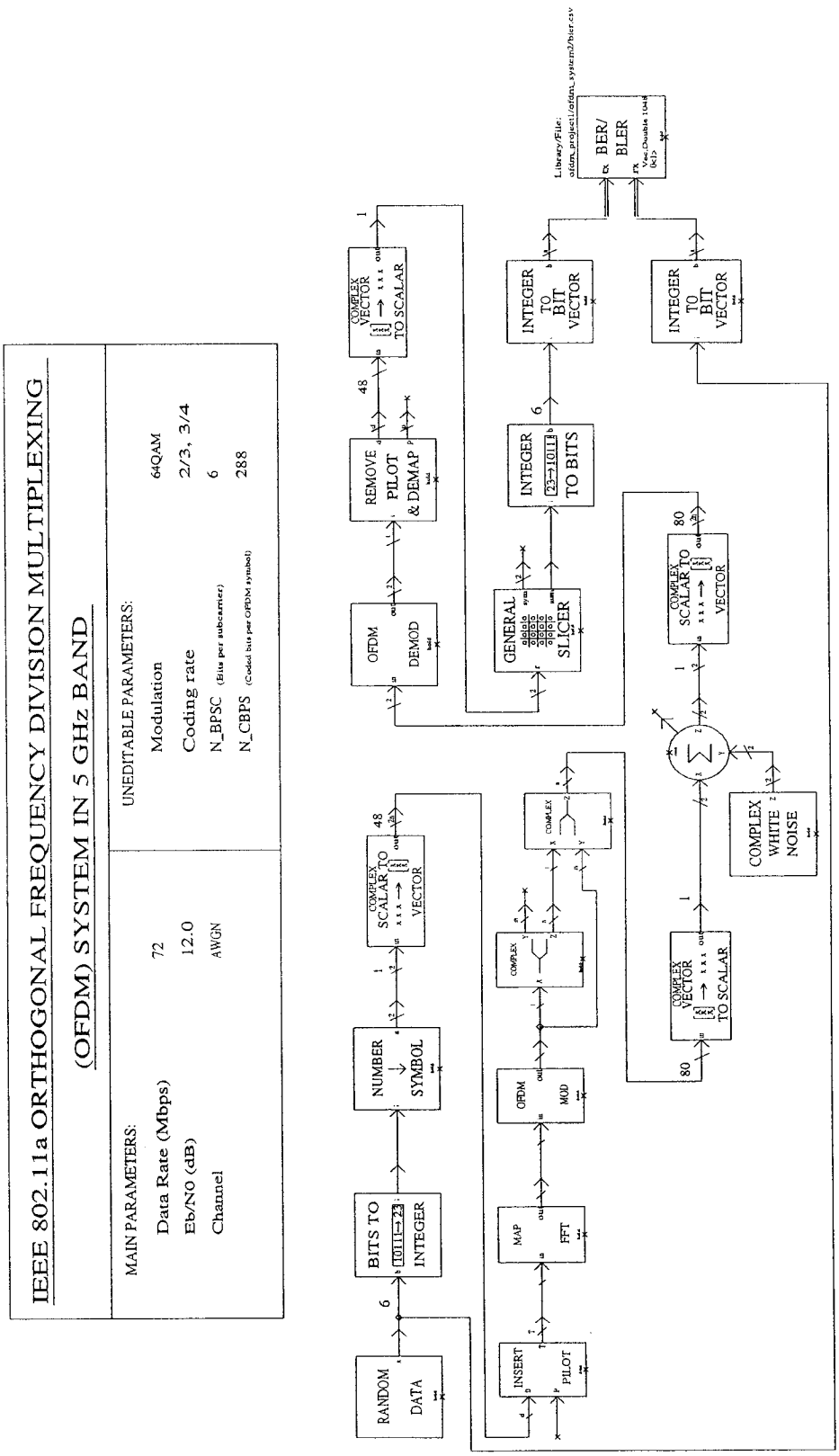

Figure 3-2: The floating-point model of the OFDM System

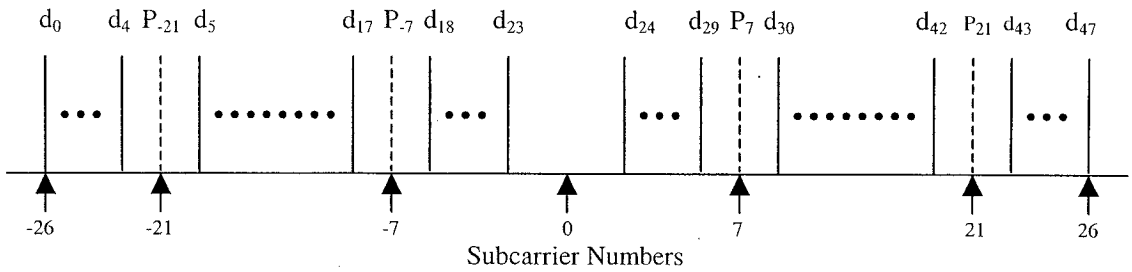

Figure 3-3: Subcarrier Frequency allocation

The next block after "Insert Pilot" in Figure 3-2 is called "Map FFT". It is responsible to prevent aliasing that might happen in the digital-to-analog converter at the output of transmitter. This task has been done by inserting 12 zero symbols into the vector coming from "Insert Pilot" block. These zeros should be added in the middle of the vector to ensure that nonzero values are mapped onto subcarriers around 0 Hz [6]. These subcarriers and zero symbols are mapped as shown in Figure 3-4. The term "Null" refers to the location of zero symbols.

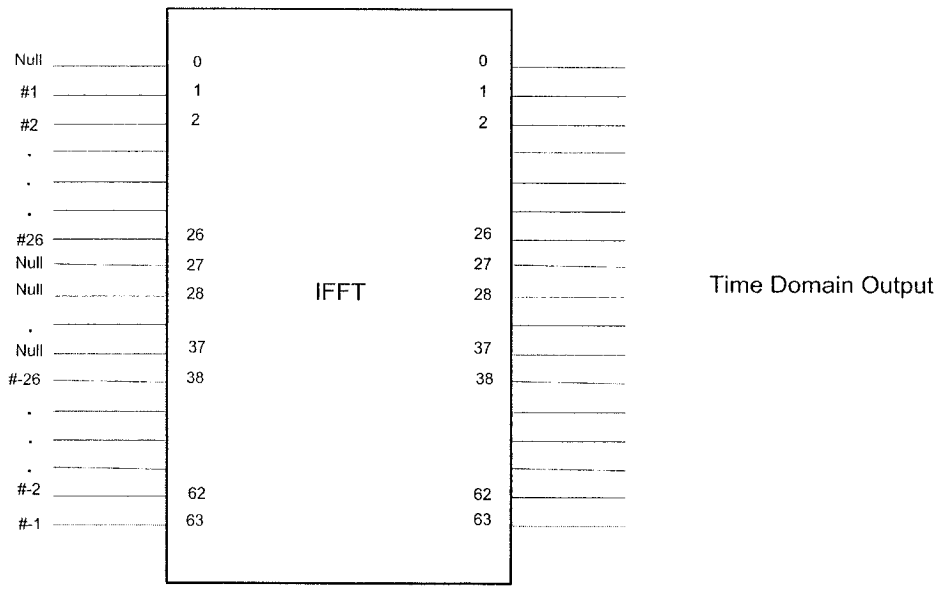

Figure 3-4: Input and Output of IFFT for Transmitter Side

The next block in Figure 3-2 is "OFDM Mod", which computes IFFT of the input vector. The input to this block is now a vector with the length of 64. The corresponding IFFT vector is also of the same length. The two following blocks after "OFDM Mod" are used to insert guard time into OFDM symbol. The guard time is created by inserting the last 16 symbols of the IFFT vector to the beginning of the same IFFT vector. Therefore we have an OFDM vector consisted of 80 symbols. This OFDM symbol is then delivered to "Complex Vector to Scalar" block which is used as a parallel to serial converter. All the above mentioned blocks construct the OFDM transmitter.

To model the channel in Figure 3-2, we used a block called "Complex White Noise". A complex adder adds the generated white noise by this block to the OFDM symbols at the output of the OFDM transmitter. On the receiver side, the first block is "Complex Scalar to Vector" that is a serial to parallel converter. Again we have a vector of length 80 at the output of this block. Then we have "OFDM Demod" block, in which the first 16 symbols (added as the guard time in the transmitter side) are removed and then the 64-point FFT of the remaining symbols is calculated. The output of FFT block is now consisted of the 48 original QAM symbols, four pilots and zero symbols. However we need to extract the 48 QAM symbols. This task is done by "Remove Pilot & Demap" block, which has two vectors at the output. One vector of the length 48 (QAM symbols) and the other a vector of the length four (pilots). We have placed "Complex Vector to Scalar" block to convert 48 QAM vector to serial symbols. The following block is "General Slicer" which remaps QAM symbols from I and Q format to integer numbers. "Integer to Bits" block assigns six bits for each integer number. The remaining last three blocks are used for Bit Error Rate (BER) calculation. "Error Controller" compares the

received bits at the output of the receiver with the input bits generated by "Random Data" block. In the case of any differences between these two bits, the error will be counted until all generated bits are received. Then, "Error Controller" calculates the bit error rate.

To draw BER curve, we need to generate Gaussian distributed noise signals and add them to OFDM symbols. If the sampling frequency is $R_s = \frac{1}{N}$, the noise variance would be:

$$\sigma_n = \frac{N_0 R_s}{2} = \frac{R_s}{2.SNR_s} = \frac{1}{2.N.SNR_s} = \frac{1}{2.N.SNR_b.\log_2^M} \qquad (3-1)$$

where SNR_b is Signal to Noise Ratio for transmitted bits and M is number of bits in each QAM symbol. For 64-QAM, M is equal to 64. In this equation, it is assumed that signal carrier symbols are normalized to one. Equation (3-1) needs to be multiplied by 42 if the constellation assigned for 64-QAM is like Figure 3-5. The average power for this constellation is 42. The BER simulation result for the floating-point model is shown in Figure 3-6. The number of input bits is 2.88 million bits and 64-QAM is used for symbol mapping. For QAM modulation scheme in a white Gaussian noise channel, the BER is given by the following equation [14]:

$$P_b \approx \frac{2(1-L^{-1})}{\log_2^L} Q\left(\sqrt{\left(\frac{3\log_2^L}{L^2-1}\right)\frac{2E_b}{N_0}}\right) \qquad (3-2)$$

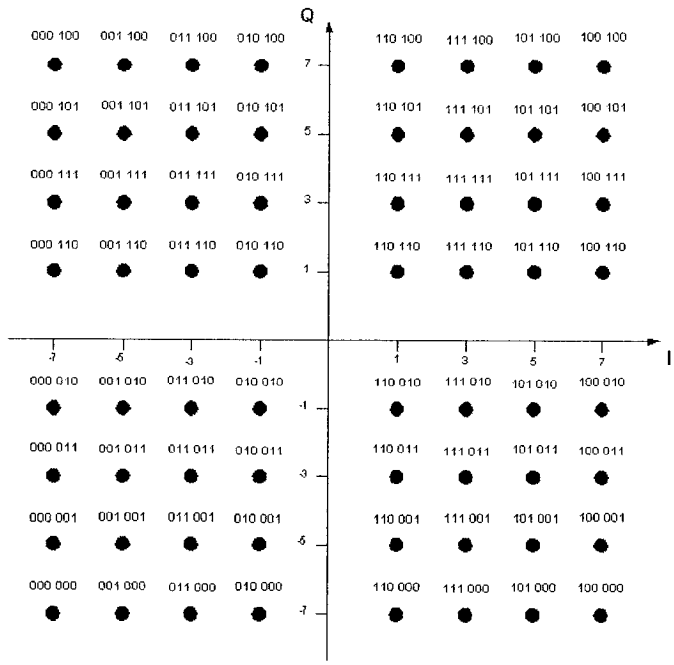

Figure 3-5: 64-QAM constellation bit encoding

where L is the number of amplitude levels in one dimension of constellation, which is equal to eight for 64-QAM. Figure 3-6 compares the 64-QAM BER curve for theory Equation (3-2)), floating-point, 16-bit fixed-point and 12-bit fixed-point simulations. As shown in the figure, the 16-bit fixed-point BER curve is exactly the same as theoretical and floating-point curves. However, the 12-bit fixed-point curve has degradation in performance. Figure 3-6 shows that at BER equal to 10^{-5}, the 16-bit model with 1 dB less SNR has the same performance as 12-bit fixed-point model. Therefore, in the implementation of the OFDM modem, 16-bit fixed-point is considered.

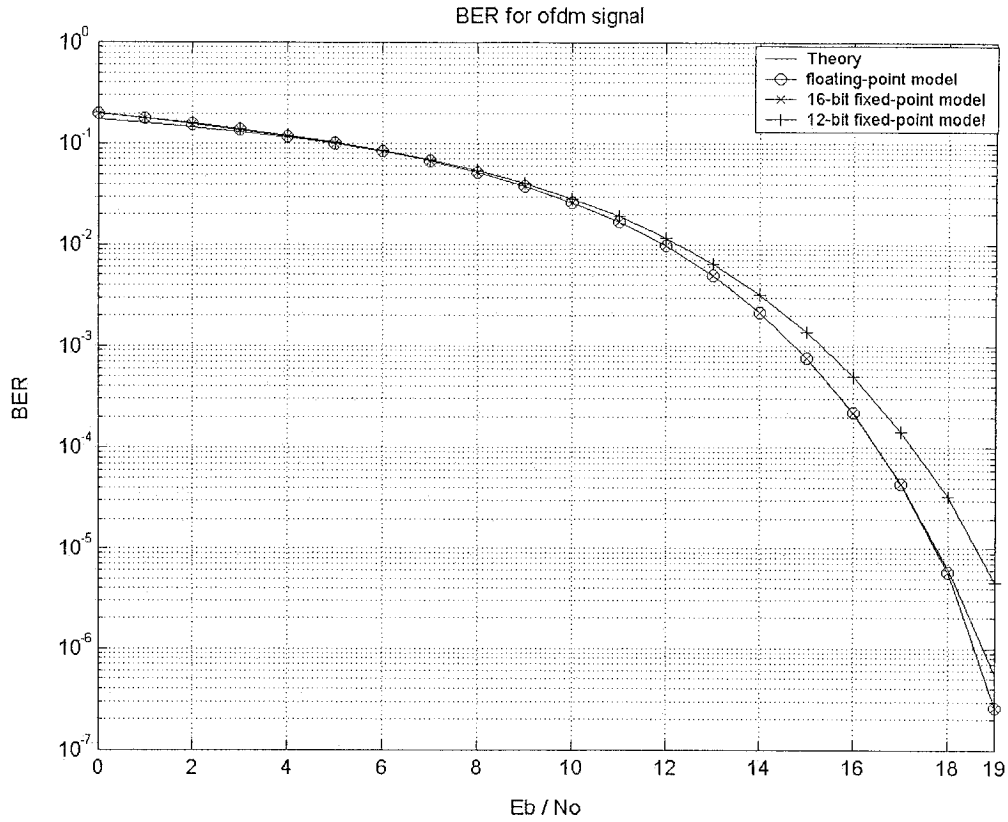

Figure 3-6: Bit-Error Rate performance of the OFDM system

3.3 Fixed-Point Modeling and Simulation

Fixed-Point modeling starts from executing "Convert Floating to Fixed-Point" command in SPW. A format needs to be chosen to assign the number of bits necessary for calculations in each block. Figure 3-7 shows the block diagram of the OFDM modem in Fixed-Point modeling. For this design 16 bits are used for all signal processing computations including IFFT/FFT. Fifteen bits of them are integer and the most significant bit is assumed as 2's complement sign bit. The notation used by SPW to show this format is <16, 15, t> in which "t"stands for 2's complement. Some blocks use different formats depending on the precision necessary for their computations.

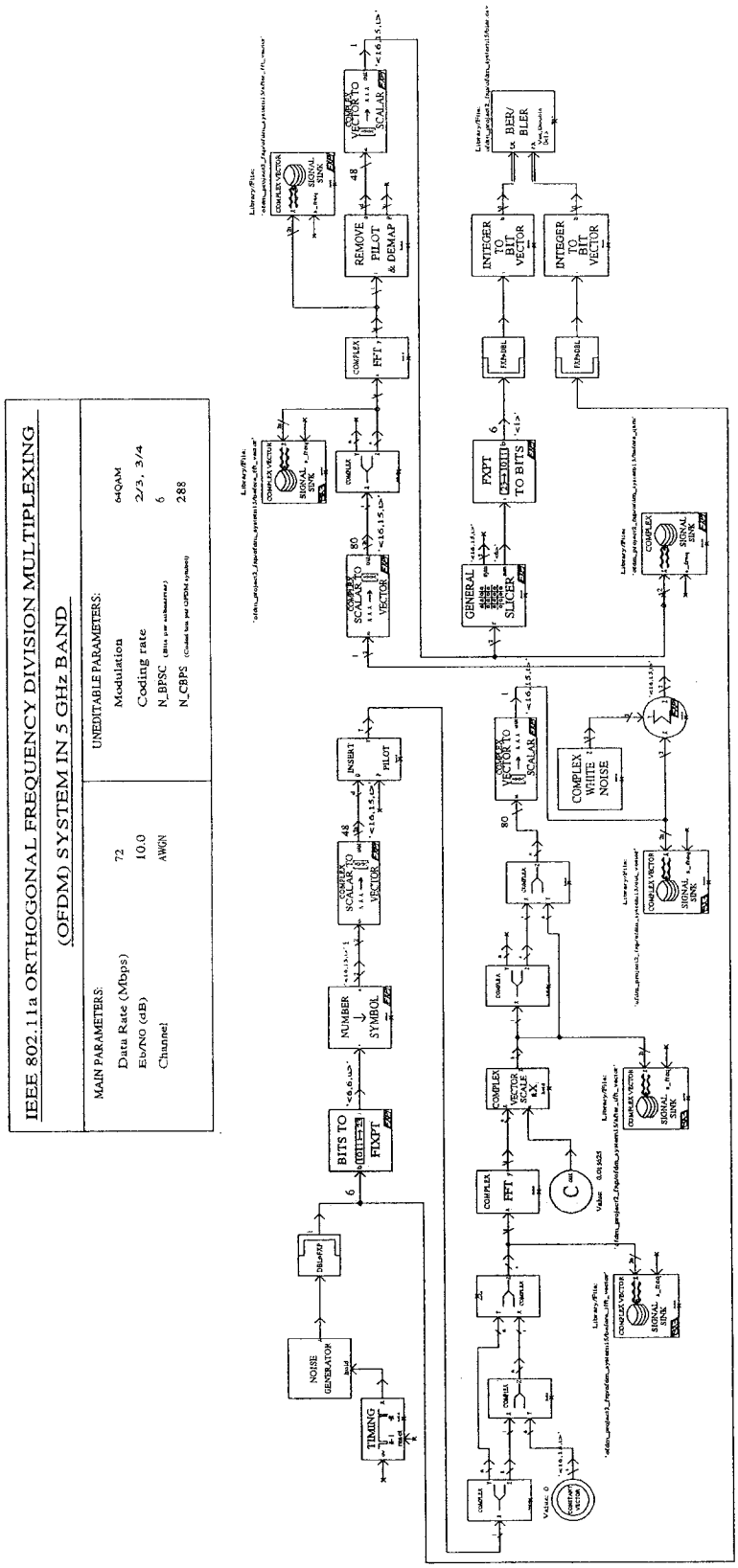

Figure 3-7: The fixed-point model for the OFDM system

When we convert the floating-point model to fixed-point, there are some blocks like "Random Data" and IFFT/FFT that do not have any equivalent in the fixed-point or HDS library. Therefore, the same floating-point block with an additional "DBL-FXP" or "FXP-DBL" block can be used for the simulation. "DBL-FXP" converts double precision value (floating-point) to an assigned fixed-point value. "FXP-DBL" converts a fixed-point input into double precision value.

In Figure 3-7, "Noise Generator" and "Timing" blocks generate the random input in floating-point format. These inputs are converted to <1, 1, u> format by "DBL-FXP" block. The parameter "u" in <1, 1, u> shows the unsigned format of fixed-point values. For 64-QAM mapping, "Bits to FIXPT" block receives six bits and convert it to an integer value between '0' and '63'. Therefore, the fixed-point format necessary to show this range of integer numbers is <6, 6, u>. The same format <16, 15, t> is assigned for the rest of the blocks in the transmitter side. As it will be described in the next chapter, VHDL codes for IFFT/FFT blocks are imported from Xilinx Coregen library. According to data sheet for these blocks, the format for FFT/IFFT computation is <16, 15, t>. Therefore, this format is chosen for the rest of the blocks.

Both imported FFT and IFFT blocks divide the final result by 64 [15]. To correctly model the functionality of imported blocks, "Complex Vector Scale aX" block is used to model the scale factor of 64. This block multiplies the output of "Complex IFFT" by 0.015625. Most of the blocks in Figure 3-7 have the same functionality as the ones in the floating-point model but they do the computations in fixed-point format.

"Complex White Noise" is a floating-point block, same as the one in Figure 3-2 but its output is converted into <16, 15, t> format with "DBL-FXP" block. The receiver

side starts after the adder. From that point to "FXPT TO BITS" block, all blocks have <16, 15, t> format. "General Slicer" remaps I and Q inputs into the related integer numbers. The constellation configuration plays an important role in BER reduction. As mentioned before, Figure3-5 shows the 64-QAM constellation as specified by IEEE 802.11a standard. This constellation configuration in SPW is therefore saved in a file which is accessible to both "General Slicer" in the receiver and "Number Symbol" block in the transmitter side.

The proposed fixed-point model is then simulated to verify that it works perfectly in accordance with the results obtained from the floating-point model. As both fixed-point and floating-point models are served with the same inputs, the resulted outputs are also the same. The BER verses E_b/N_0 curve is given in Figure 3-6. The curve is exactly similar to the one for floating-point model and this proves that the model was correctly converted from floating-point to fixed-point format. Note that there exist several sinks in figure3-7 to monitor the signals in various points of the system. These sinks are mainly used for debugging purpose.

By finalizing the fixed-point model, the system design for the OFDM modem is finished and the system is ready to be implemented. As shown in Figure 3-1, the next step is to prepare VHDL codes and to synthesis the circuitry. The next chapter is dedicated to circuitry design and VHDL code generation and final simulation.

4 FPGA Implementation of OFDM Modem

The last steps in the design flow are to prepare VHDL codes and to synthesize the circuitry which was shown in Figure 3-1. The details of these two steps are discussed in this chapter. The modeling and simulation of the OFDM system used for the physical layer of IEEE 802.11a have been explained in the previous chapter. We saw that the 16-bit fixed-point model (as depicted in Figure 3-7) had the same performance as the floating-point model. That fixed-point model is used as the base for the proposed OFDM modem circuitry design. A block diagram for modem implementation is demonstrated in Figure 4-1. This circuitry is constructed by three main blocks: "Transmitter", "Receiver" and "Frame detection and Synchronization". The transmitter and receiver circuitries are discussed in this chapter and the circuitry for synchronizer is the subject of the next chapter. To design the modem controller, we need to have all the blocks necessary for the standard modem and all the signals coming from the upper network layers.

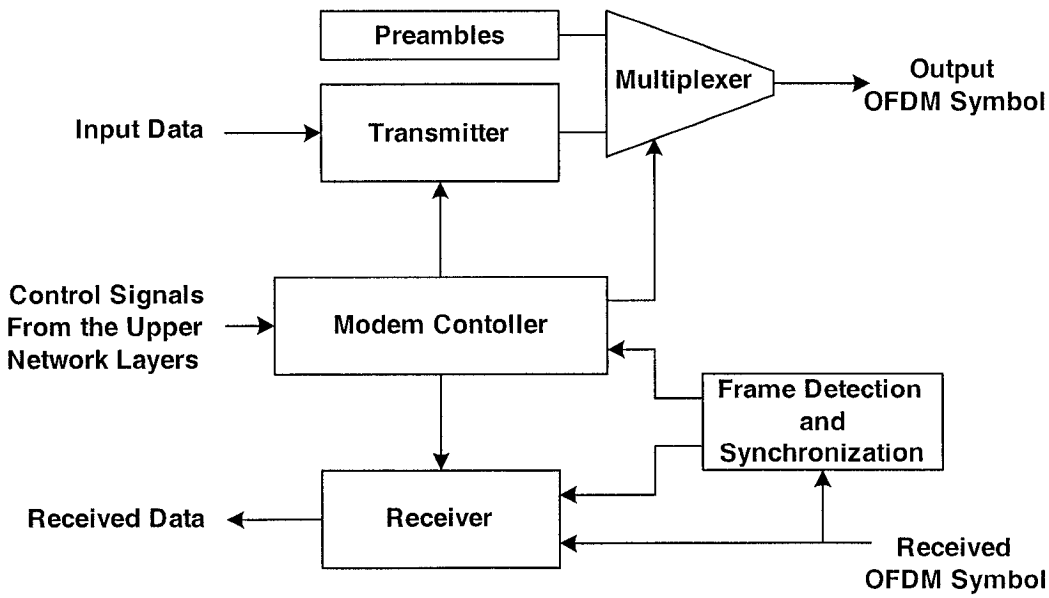

Figure 4-1: The OFDM modem block diagram for circuitry design

The short and long preambles were explained in Chapter 2. They are used at the beginning of data transmission and are predefined symbols to the receiver. We store these symbols in a ROM, which is shown as the "Preamble" block. The modem controller selects whether to send OFDM symbols or preamble symbols through the "Multiplexer" block in Figure 4-1.

In this chapter first a circuitry is proposed for the OFDM modem implementation. This circuitry includes the OFDM transmitter and receiver. Then the circuitries for the transmitter and receiver are synthesized. After analysis of these circuitries, the final logic of them is mapped into FPGA.

4.1 OFDM transmitter circuitry

The "Transmitter" block shown in Figure 4-1 can be split into four major building sub-blocks as demonstrated in Figure 4-2. Similar to what we had for the fixed-point model, the first task is QAM mapping. "Mapping" is followed by two other blocks which are "IFFT" and "Add Cyclic Extension". Their functionalities are discussed earlier in fixed-point model. "Transmitter Controller" schedules the operation of three other blocks. In following subsections, detail design of each block is presented.

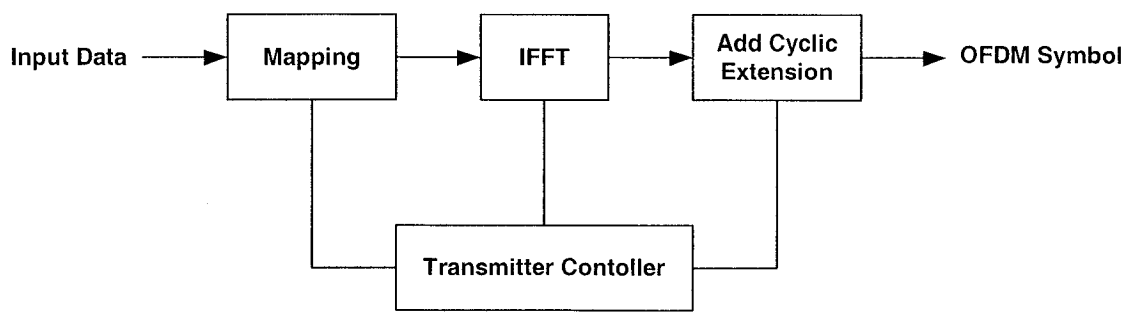

Figure 4-2: The OFDM transmitter block diagram for circuitry design

4.1.1 QAM Mapping circuitry

The circuitry used for QAM mapping in the OFDM transmitter is depicted in Figure 4-3. In this figure, the first block is "Serial to Parallel Shift Register". It has the same functionality as "Bits to fixp" block in the fixed-point model (Figure 3-7). It consists of a shift right register and a latch, which are both clocked with the same rate as the input data. Six bits from input bit stream are serially shifted into a register. Then they are latched in the "Latch" block for six clock cycles. The two inputs "enable" and "clear" for "Serial to Parallel Shift Register" are coming from the transmitter controller. The "Latch" block is also scheduled through the input "Hold" which comes from the transmitter controller. "out_parallel" is a six-bit output and gives the integer number corresponding to the input data that have been shifted into the shift register.

The next block "QAM Mapper" is a combinational logic. It maps the input integer data into a constellation point as shown in Figure 3-5 for 64-QAM. However, there is a difference between the constellation used in the circuitry and the one shown in Figure 3-5

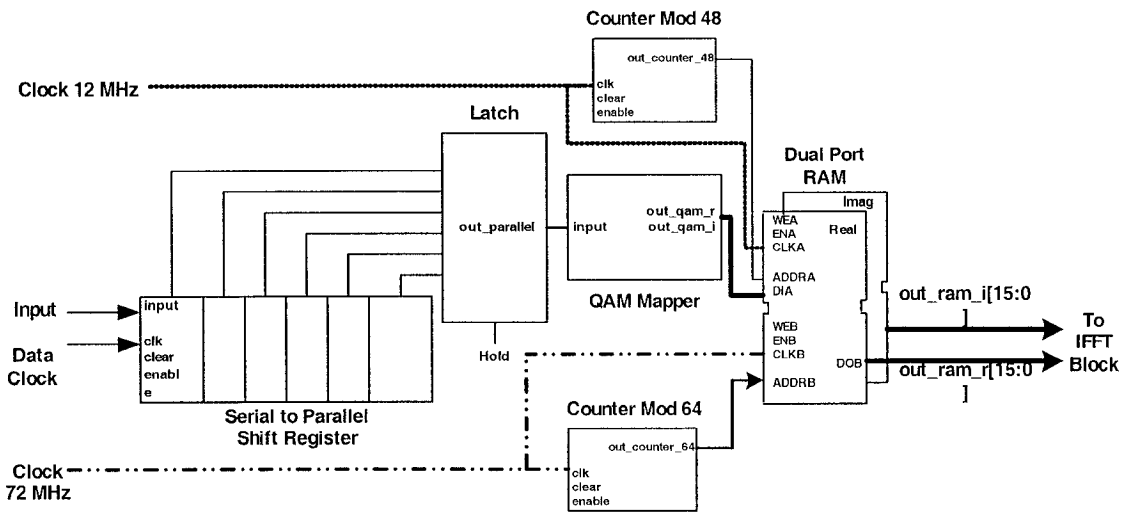

Figure 4-3: The circuitry used for QAM mapping

due to the mapping scale. It means that instead of mapping the input data into I and Q in the range of –7 to 7, they are mapped in the range of –28672 to 28672 (4096 × 7). This makes the output numbers large enough to provide us with more precise result of the computations done in the following IFFT block. The "QAM Mapper" block gives the real and imaginary components of QAM as two separated outputs. Each of them are formatted in 16-bit 2's complement. These outputs are shown by "out_qam_r" and "out_qam_i" in Figure 4-3. The circuitry is fed by the input continuously, therefore "out_qam_r" and "out_qam_i" are generated as continuous streams. However they are processed with higher clock speed in groups of 48 symbols. This explains why we need to store them in a "Dual Port RAM". In reality "out_qam_r" and "out_qam_i" symbols are stored in two separated dual port RAMs called "Dual Port RAM imag" and "Dual Port RAM real" respectively. In Figure 4-3, for the sake of simplicity, both outputs of "QAM Mapper" are drawn as a single line. To store QAM symbols, a counter (shown by "Counter Mod 48") gives the address to Port A of "Dual Port RAM". The counter is clocked (in pin "clk") with a speed equal to 12 MHz. It counts from 38 up to 63 and then from 1 to 26 in accordance with the specification shown in Figure 3-4. In each group of 48-symbol, there are 288 bits. For input data with bit rate equal to 72 MHz, the bit time duration is 13.888 ns. Therefore the mapping process should be done within a 4-μs time interval. The real and imaginary components of mapped symbols are grouped in a vector of 48 words. Each word is formatted in 16-bit 2's complement.

Four pilot symbols also can be stored in "Dual Port RAM" while IFFT block is not reading the stored words. These symbols are addressed and written in the dual port RAM through the pins located in Port B. The words stored inside the dual RAM are read

from the output of Port B and addressed by "Counter Mod 64". Therefore, Port A of "Dual Port RAM1" is only used for writing the input data into the RAM. But Port B is used for writing Pilots (inserting Pilots) and reading the stored words from the RAM. All other pins (e.g. "enable" and "clear") which are shown in Figure 4-3 but not discussed here are connected to the "Transmitter Controller" block.

4.1.2 IFFT circuitry

To meet IEEE 802.11a standard, the IFFT block requires highly pipelined architecture. A core from Coregen library is chosen for this purpose, which is called vfft64 [15]. This core computes a 64-point complex forward or inverse FFT. The input data and output samples are vectors of 64 complex values represented as 16-bit 2's complement numbers. Sixteen bits are used for each real and imaginary component. The vfft64 transform engine employs Cooley-Tukey radix-4 decimation-in-frequency IFFT [16]. This algorithm requires the calculation of columns or ranks of radix-4 butterflies. The radix-4 butterflies are referred to as dragonflies. For 64-point, there are three dragonfly ranks, and each rank includes 16 dragonflies. Appendix A defines the pins and the module functionality for vfft64 core. Three configurations are supported for vfft64:

- Single-Memory-Space (SMS)
- Dual-Memory-Space (DMS)
- Triple-Memory-Space (TMS)

The differences among these configurations give system designers maximum flexibility of the I/O interface and memory architecture selection. No data storage has been included in the core itself. Therefore, other modules like dual port RAMs need to be imported.

SMS configuration provides the simplest memory and I/O interface to the IFFT core. In this mode, while input and output operation is in progress, the IFFT core is idle. This means that SMS has explicit I/O operations, which take some extra clocks and increases latency. This inefficiency can be overcome with DMS or TMS configuration, which use additional memory banks. Figure 4-4 shows DMS configuration.

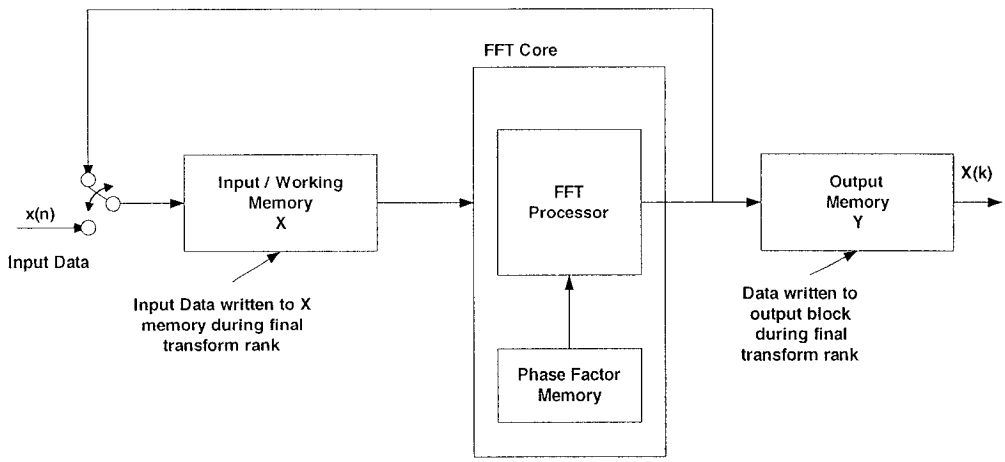

Figure 4-4: Dual-Memory-Space IFFT Core interface

This mode allows input, computation and output operations to be overlapped. More detailed figure is given in Figure 4-5. The labels of connections are written beside each pin. For example, the outputs of "RAM X" are "DOB[15:0]" which are connected to the input "DI_R[15:0]" and "DI_I[15:0]" of "FFT CORE". Their connections are labeled as DI[15:0] and DR[15:0].

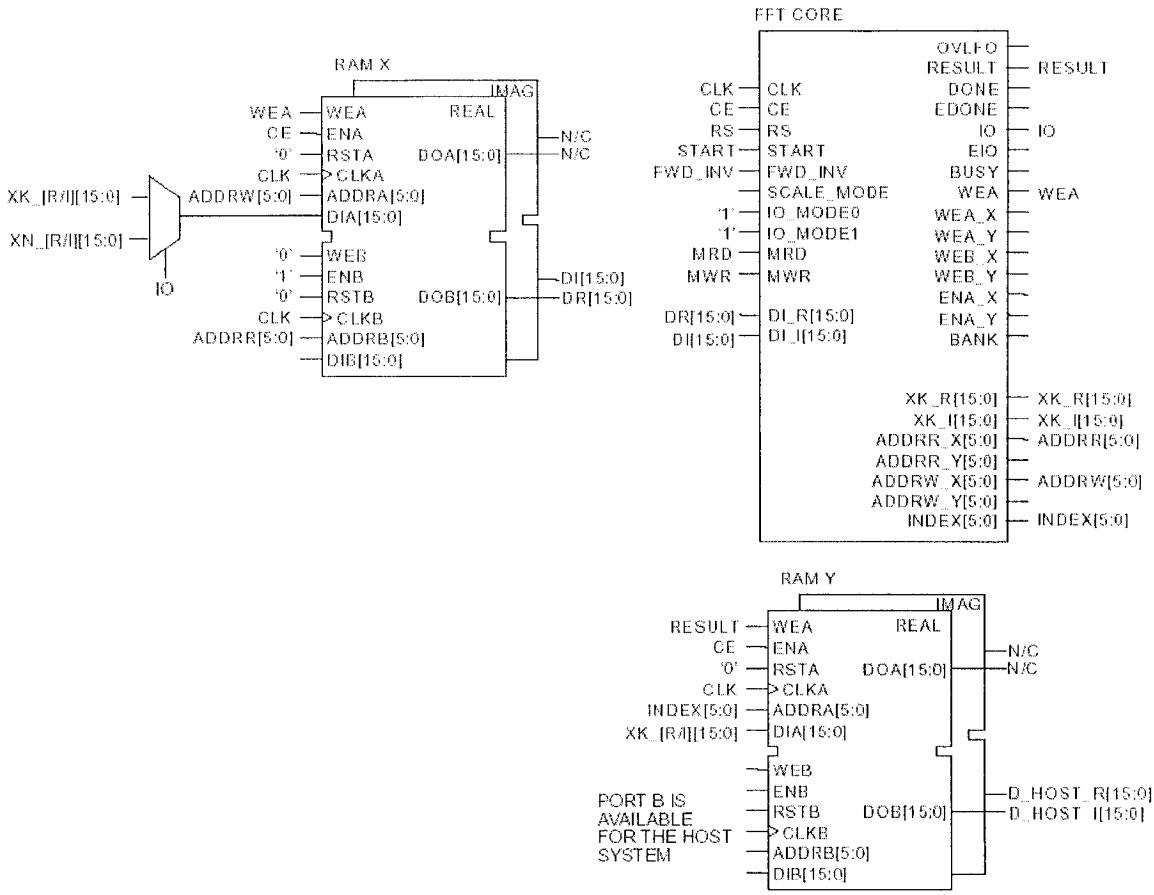

Figure 4-5: DMS core configuration with detailed connection diagram

There are three phases for the IFFT computation process:

- Data Load Phase
- Compute phase
- Result Upload phase.

A data loading operation is to write data into memory on the rising edge of the clock. This memory is shown by "Memory X" in Figure 4-4 and "RAM X" in Figure 4-5. To store the data in memories serially, this phase needs 64 clock pulses as shown in Figure 4-6.

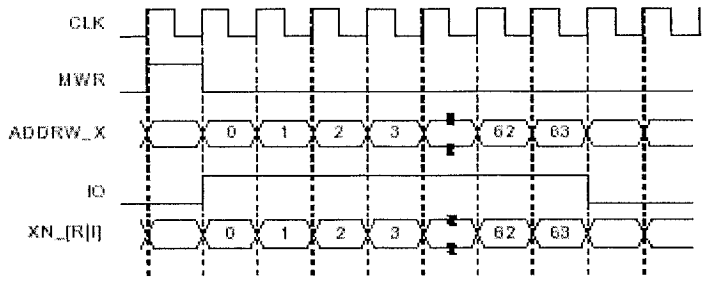

Figure 4-6: Input data load timing

"MWR" signal starts the data loading operation. Immediately after this signal, the IFFT core put the addresses (necessary to store data) in the address bus "ADDRW_X". The data should be ready on the input of multiplexer shown as XN_[R|I]. The other multiplexer input XK_[R|I] comes from the IFFT core and is used by the core during IFFT computations.

After completion of Data Load phase, the IFFT engine starts computation phase with applying "START" signal. Once the IFFT computation is completed, the resulted data is stored again in "Memory X" in Figure 4-4 and "RAM X" in Figure 4-5. The computed data is in digit-reversed order due to decimation-in-frequency format of used IFFT algorithm. Figure 4-7 depicts the timing for computation phase. In this diagram, the controller schedules the "START" signal but the others (except CLK) are internal signals coming from IFFT core.

In Upload phase, data is read back from memories and unscrambled so that it is presented in "Output Memory Y" (shown in Figure 4-4) in natural order. Concurrent with this operation, a new vector of input data can be written into "Memory X". Therefore,

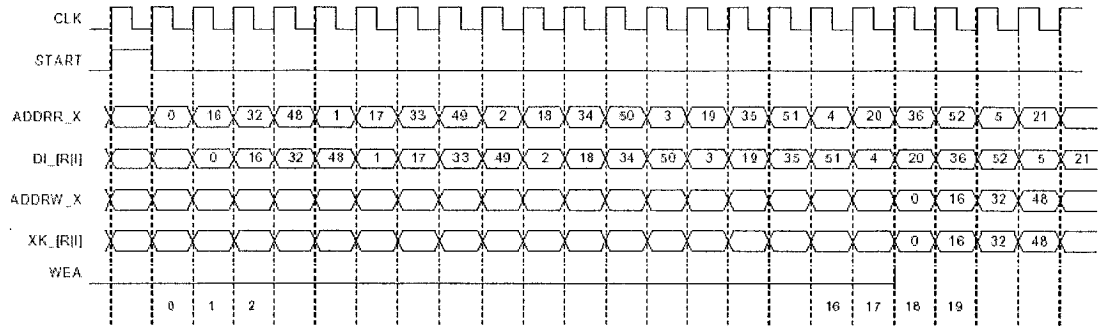

Figure 4-7: FFT start timing

data load, computation and output phase operations are overlapped in DMS configuration. The completed IFFT output vector is placed in the output buffer. Figure 4-8 shows the related uploading timing in DMS configuration.

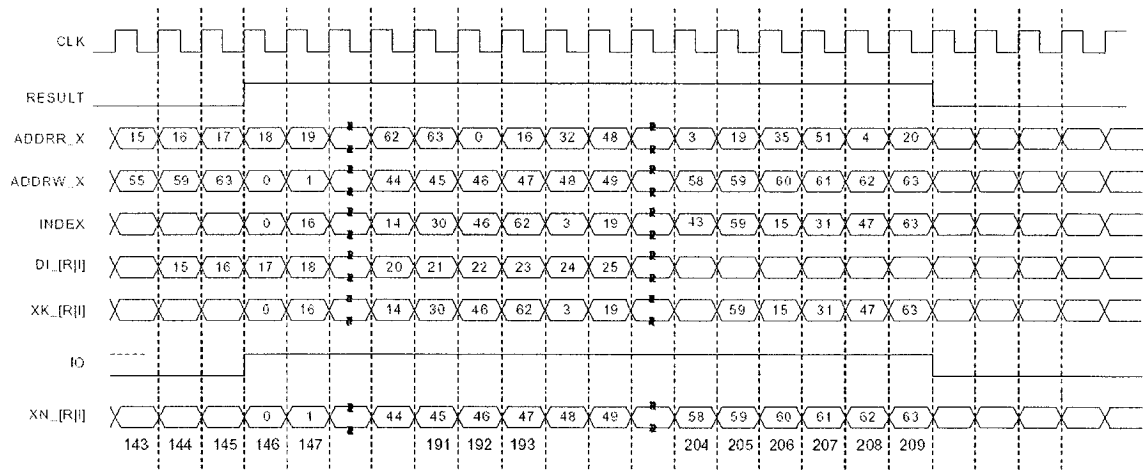

Figure 4-8: DMS configuration- result and I/O timing

In DMS configuration, the first series of 64 data require 209 clock cycles to complete IFFT computations but the next IFFT result vectors will be available every 192 clock cycles. The additional 17 clock cycles required by the first IFFT are associated with the pipeline depth of the IFFT arithmetic unit [15]. The clock speed for the IFFT core is 72 MHz. Therefore, the core can compute the IFFT result within 3.2 μs as specified by

the standard. Thus, we can use DMS configuration in the design for IFFT circuitry in the transmitter. Figure 4-9 depicts the proposed IFFT circuitry using DMS configuration.

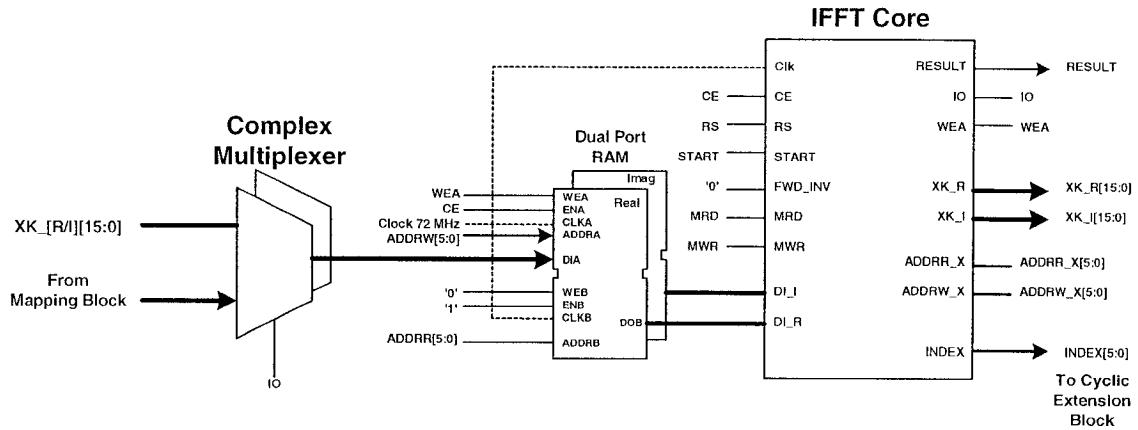

Figure 4-9: The proposed IFFT circuitry for transmitter

In this figure, one of the inputs to multiplexer comes from the "Dual Port RAM" output (shown in Figure 4-3 as "out_ram_r" and "out_ram_i"). In Data Load phase these inputs are stored in "Dual Port RAM" within 64 clock cycles. The clock speed ("clk" input) for both ports of the dual port RAM is 72 MHz. This memory has the same functionality as "Memory X" in Figure 4-4 and "RAM X" in Figure 4-5. After completing Data Load Phase, the IFFT core computes IFFT data and stores them inside the same dual port RAM blocks. This time the input to "Dual Port RAM" comes from "XK_[R|I]" output of "IFFT Core". The data has to pass through the multiplexer which is controlled by "IO" output of the core. The phase factors used in the IFFT computation are generated within the core. For the Compute phase, the core reads stored data from Port B of the dual port RAM.

There are some outputs which are used only for the Upload phase. They are "INDEX", "RESULT" and "XK_[I/R]" as shown in Figure 4-9. These outputs are

connected to a dual port RAM in the next block of the OFDM transmitter circuitry as it is discussed in the next subsection.

4.1.3 Cyclic Extension circuitry

Figure 4-10 shows the proposed circuitry for adding the cyclic extension symbols. "Dual Port RAM 2" and "Counter Mod 80" insert cyclically extended guard time into the OFDM symbol. The input pin "DIA" for "Dual Port RAM 1" is connected to XK_[R/I] coming from "IFFT Core" (see Figure 4-9). The addresses for storing computed IFFT symbols are from "INDEX". The input pin "WEA" which enables Port A to write the input data into "Dual Port RAM 1" is controlled by "RESULT" as shown in Figure 4-9. Therefore, Port A of the first dual port RAM shown in Figure 4-10 is controlled only by the signals coming form the IFFT core.

"Counter Mod 64_1" gives addresses to the Port B of "Dual Port RAM 1" as depicted in Figure 4-10. The stored data is transferred to "Dual Port RAM 2" within 64

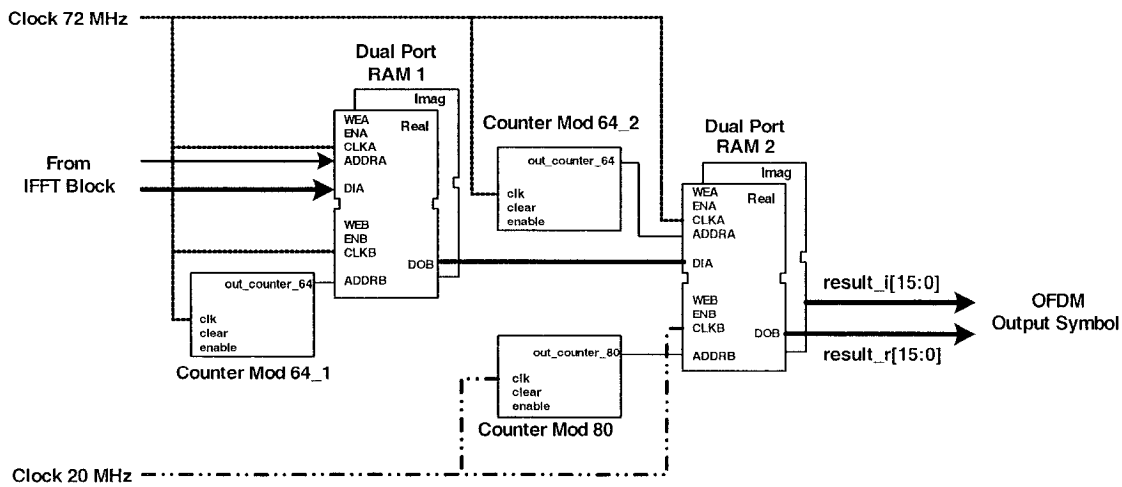

Figure 4-10: the circuitry for adding the cyclic extension

clock cycle with clock speed equal to 72 MHz. "Counter Mod 64_2" gives the addresses to Port A of "Dual Port RAM 2". Both counters count from 0 to 64 with clock speed of 72 MHz. By using "Dual Port RAM 1" and the arrangement shown in figure 4-10, we make the timing for adding cyclic extension more independent of the timing necessary for IFFT computation.

"Counter Mod 80" gives addresses to the Port B of "Dual Port RAM 2". The counter counts from 48 to 63 and then counts from 0 to 63. Since the clock for this part is 20 MHz, sixteen symbols are inserted into one OFDM symbol that creates a guard time equal to 800 ns. The output symbols are read from port B of "Dual Port RAM 2" with a clock rate equal 20 MHz. Therefore, one OFDM symbol takes 4 μs, which is in accordance with the standard. The counters and dual port RAMs are controlled by the transmitter control circuitry, which is discussed in the next subsection.

4.1.4 Transmitter Controller circuitry

The transmitter controller circuitry is illustrated in Figure 4-11. It is responsible to control all blocks existing in the OFDM transmitter. The controller is a counter which counts from 0 to 287. It schedules all blocks of the transmitter circuitry with each clock pulse. It includes also some combinational logics to make the proper control signals. The clock speed for the controller is 72 MHz. and applied to "clk" input pin. As shown in Figure 4-11, there are some control signal inputs to the counter which is coming from the OFDM modem controller circuitry. A brief description for each output is written besides the output pins.

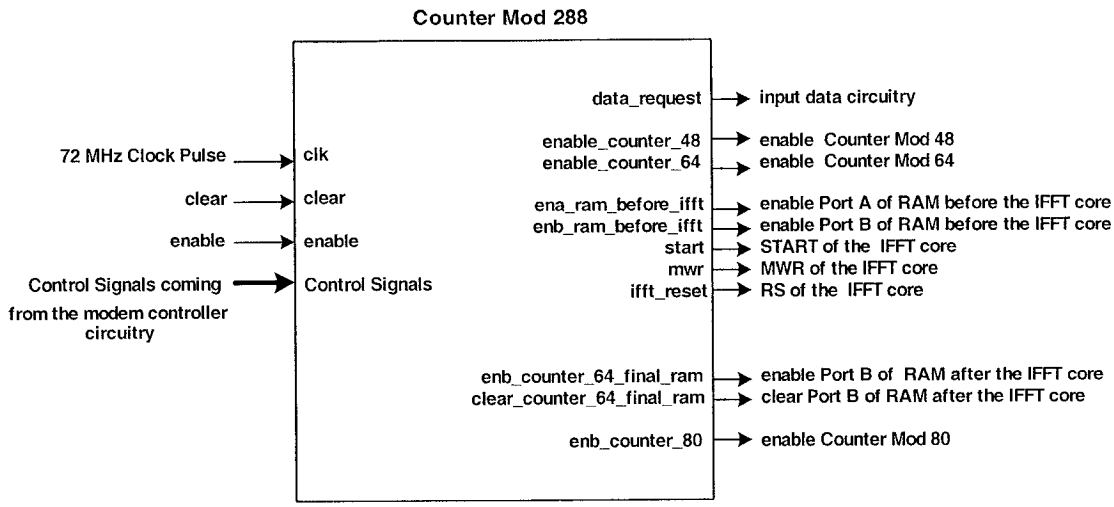

Figure 4-11: Transmitter controller circuitry

"data_request" signal is sent to the output of transmitter to show the circuitry is ready to receive the input data. Whenever the shifted input data is mapped to QAM symbol, "enable_counter_48" is set to '1'. At this time the symbols need to be stored in "Dual Port RAM' shown in Figure 4-3. The transmitter controller set "enable_counter_64" to '1' when the words stored in "Dual Port RAM" (in Figure 4-3) need to be read by the IFFT core. The stored word is available at the output of RAM one clock cycle after applying its address. Thus "Counter Mod 64" must start counting one clock cycle before the core starts its Data Load phase. "ena_ram_before_ifft" and "enb_ram_before_ifft" are enable signals for both ports of "Dual Port RAM". The enable signal for Port A is set to '1' when the mapped data are stored into the RAM. The other one is '1' only when "FFT Core" wants to address the RAM in order to read the stored words.

Four outputs of the transmitter controller are going to the IFFT core. They are discussed in subsection 4.1.2. In Figure 4-11, "enb_counter_64_final_ram" is an output

to enable Port B of "Dual Port RAM 1" shown in Figure 4-10. It is connected to "ENB" pin of this RAM. It is also connected to "enable" pins of "Counter Mod 64_[1/2]. "enb_counter_64_final_ram" controls also "ENA" pin of "Dual Port RAM 2". Finally "enb_counter_80" is set to '1' when transmitter starts to send OFDM symbols. This control signal is also connected to "ENB" of "Dual Port RAM 2". It remains '1' for the whole transmission time.

4.2 OFDM receiver circuitry

Figure 4-12 shows the block diagram of circuitry design for the OFDM receiver. This diagram depicts the sub-blocks that construct the "Receiver" block in Figure 4-1. Same as the OFDM transmitter, the receiver circuitry is split into four major blocks. The first task of the receiver is Cyclic Extension removal. Next blocks are used for the demodulation of received OFDM symbols. They have the same functionality as discussed for the fixed-point model proposed in Figure 3-7. The details of each block are demonstrated through the following subsections.

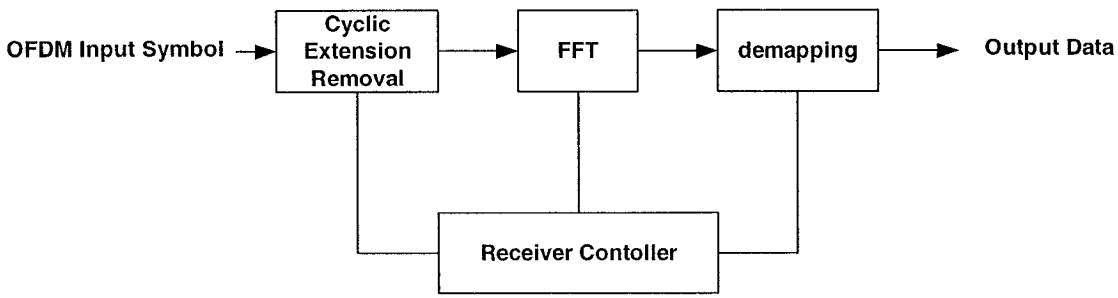

Figure 4-12: The block diagram of circuitry design for the OFDM receiver

4.2.1 Cyclic Extension Removal circuitry

Figure 4-13 shows the proposed circuitry for removing the cyclic extension symbols. The received OFDM symbols are stored in Port A of the dual port RAM continuously. The addresses for the RAM are given by "Counter Mod 64_1". This counter is similar to "Counter Mod 64_1" shown in Figure 4-10. The inputs "DIA" for the dual port RAM are "input_[r/i] that are the received OFDM symbols. The other inputs for Port A and "Counter Mode 64_1" are coming from the receiver controller circuitry, which it will be discussed later in this chapter. The clock speed for Port A is 20 MHz; therefore it takes 3.2μs to store one OFDM symbol.

Port B of "Dual Port RAM' in Figure 4-1 is used for reading the stored OFDM symbols. "Counter Mod 64_2" gives addresses to the Port B. The stored symbols are read from port B of "Dual Port RAM" with a clock rate equal to 72 MHz. The same clock is also given to "Counter Mod 64_2". This counter counts from 0 to 63 whenever "ENB" is set to '1'. The outputs of Port B are sent to the FFT block. All the other pins for Port B and "Counter Mod 64_2" are controlled by the receiver controller circuitry.

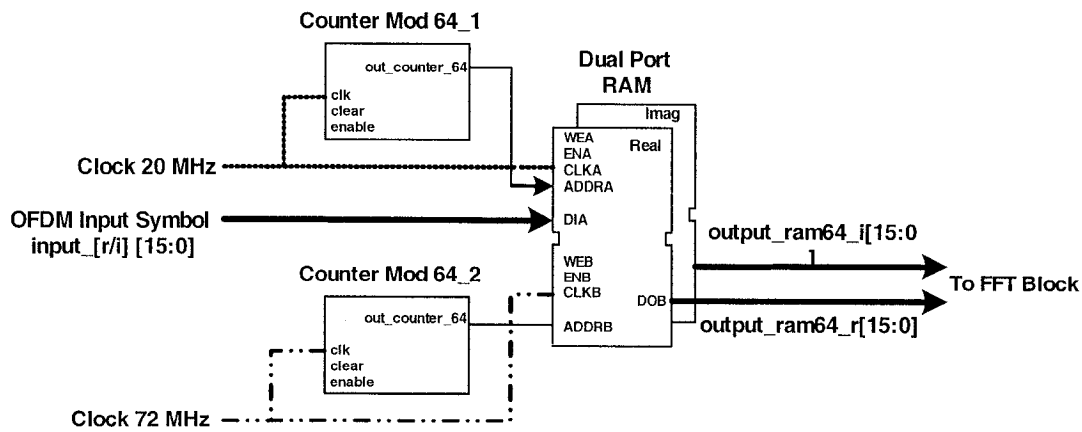

Figure 4-13: the circuitry for the cyclic extension removal

4.2.2 FFT circuitry

The FFT core in receiver is the same as the IFFT core presented for the transmitter. Therefore the same discussion as presented in subsection 4.1.2 is valid for the FFT core. All the configuration setup for the imported Coregen FFT code is done in the same fashion as IFFT except pin "FWD_INV" of vfft64 core. This pin should be '0' for IFFT and '1' for FFT computation. Figure 4-14 depicts the proposed FFT circuitry using DMS configuration. For "Complex Multiplexer", "Dual Port RAM 1" and "FFT Core" the interconnections are the same as the ones shown in Figure 4-9. Therefore, the discussion provided in subsection 4.1.2 remains valid for these blocks. One of the inputs to multiplexer comes from the dual port RAM shown in Figure 4-13.

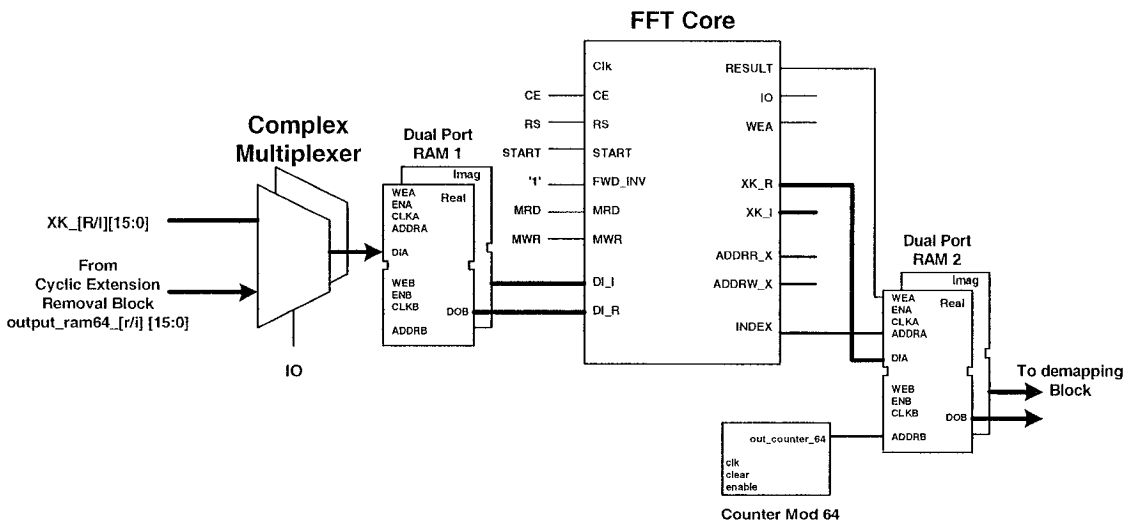

Figure 4-14: The proposed FFT circuitry for receiver

There are two additional blocks in Figure 4-14 compares to Figure4-9. One of them is "Dual Port RAM 2" which stores the result from "FFT Core". It has the same role as "RAM Y" in Figure 4-5 for DMS configuration. Therefore Port A of this RAM is connected to FFT core similar to the one that was used in the cyclic extension circuitry

(see Figure 4-10) in the transmitter circuitry. Port B of "Dual Port RAM 2" is used to read the results stored in the RAM. The addresses for reading Port B are given by the output of "Counter Mod 64", which counts from 0 to 63 when "ENB" of this RAM is set to '1'. The receiver controller circuitry sets pin "ENB" of "Dual Port RAM 2" to '1' for 64 clock cycles. The controller controls also the counter and "FFT Core".

4.2.3 QAM Demapping circuitry

Figure 4-15 depicts the circuitry used for QAM demapping in the OFDM receiver. In this figure, "Dual Port RAM" stores 48 QAM symbols coming from the FFT block. The addresses to store data in Port A are given by "Counter Mod 64", which counts from 0 to 64 when "ENA" is set to '1'. The clock for Port A is 72 MHz. The stored words are read from Port B, which is addressed by "Counter Mod 48" continuously. This counter counts from 38 up to 63 and then from 1 to 26 in accordance with the specification shown in Figure 3-4. All of the data reading process should be done within 4 μs time interval. The clock speed for reading data from Port B is 12 MHz.

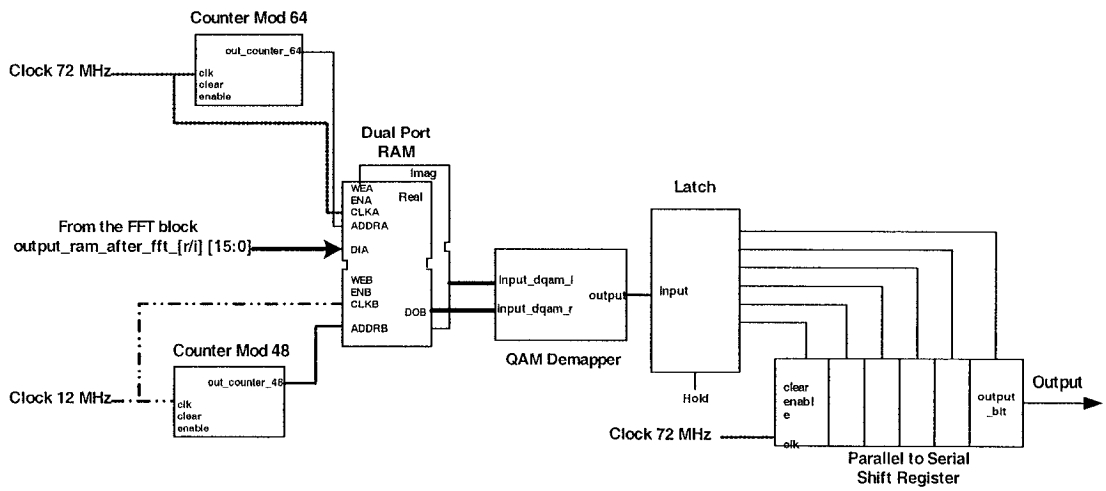

Figure 4-15: The circuitry used for Demapping

Then "QAM Demapper" demaps the complex I and Q input words into a six-bit integer number according to the decision region of 64-QAM constellation. "Latch" block keeps the input value constant for the next parallel to serial block. The parallel to serial block shifts the hold values into its output within six clock pulses. Therefore, the speed for the parallel to serial register needs to be 72 MHz. All of the controller inputs for each block in Figure 4-15 (e.g. "enable" and "clear") are connected to "Receiver Controller" which is discussed in the next subsections.

4.2.4 Receiver Controller circuitry

The receiver controller circuitry is illustrated in Figure 4-16. It is responsible to control all blocks existing in the OFDM receiver. The controller is a counter which counts from 0 to 287. It schedules the timing of all blocks in the receiver circuitry with each clock pulse. The clock speed for the controller is 72 MHz. It includes also some combinational logics to make the proper control signals. As shown in Figure 4-16, there are some control signal inputs to the counter which are coming from the OFDM modem controller circuitry.

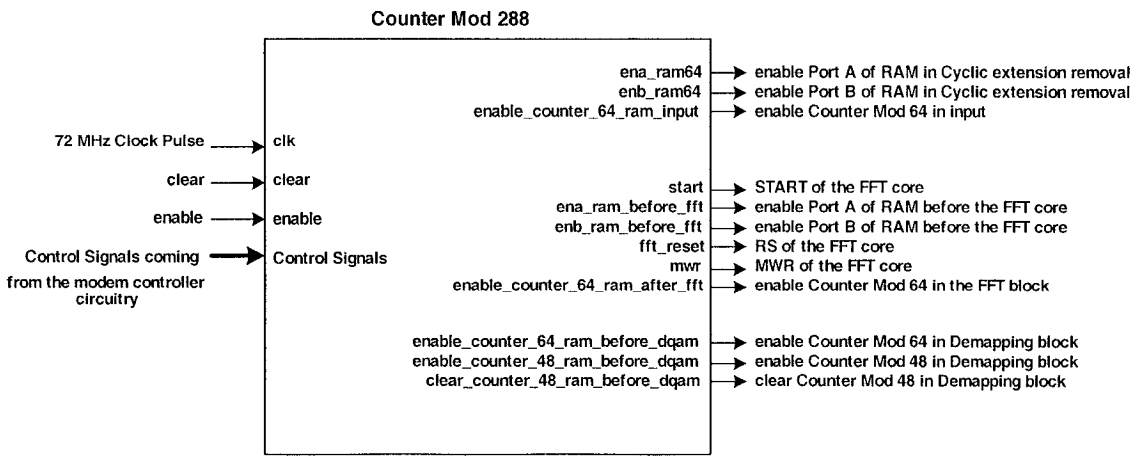

Figure 4-16: Receiver controller circuitry

"ena_ram64" enables Port A of the dual port RAM in Demmaping block to store the received OFDM symbols. For reading the stored data from Port B, the controller sets "enb_ram64" and "enable_counter_64_ram_input" to '1' for 64 clock cycles with speed of 72MHz.

There are some outputs used for controlling the FFT block. They do the same task as described for the control signals in the transmitter controller circuitry. For the Demapping block, "enable_counter_64_ram_before_dqam" controls both "ENA" pin of the dual port RAM and "enable" pin of "Counter Mod 64". "enable_counter_48_ram_before_dqam" controls both "ENB" pin of the dual port RAM and "enable" pin of "Counter Mod 48". "clear_counter_48_ram_before_dqam" is connected to "clear" pin of "Counter Mod 48". This control signal is '1' for only one clock cycle.

4.3 Implementation and Results

The next step in the design is to describe the circuitry in VHDL code. Then we can synthesize the codes and analyze them to verify that the circuitry meets the required design specification. The synthesized codes are mapped into FPGA and the hardware for the circuitry is realized.

4.3.1 VHDL code generation

The OFDM transmitter and receiver circuitries are summarized in Figures 4-17 and 4-18, respectively. The circuitry has three major blocks which their details have been discussed in this chapter. To simplify the diagram, the controllers are not illustrated. The

interconnections shown in the figure are used for the data and addresses. The connectors which transfer data are shown with bold lines. Figure 4-18 depicts the OFDM receiver circuitry. In these two figures the dual port RAM blocks, IFFT core and multiplexers are imported from Coregen library. Therefore, we have VHDL codes for those blocks. The supplier of these cores; which is Xilinx in this design, does not provide RTL level codes.

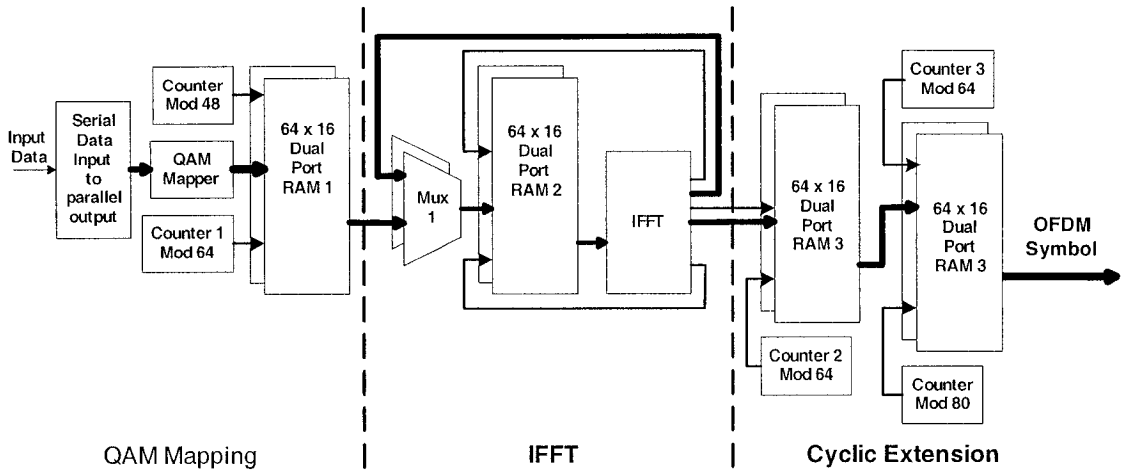

Figure 4-17: the OFDM transmitter circuitry

However, it gives the configuration necessary for a VHDL design. In "Design and Analyzer" tool, these cores are displayed as boxes but their timing for simulations are known.

VHDL codes for the other blocks in transmitter and receiver including the counters and controllers are prepared manually. All the blocks have a sequential VHDL codes except "QAM Mapper" in Figure 4-17 and "QAM Demapper" in Figure 4-18 which have combinational logic. The design is hierarchal and each block can be simulated separately as it will be explained in the next subsection

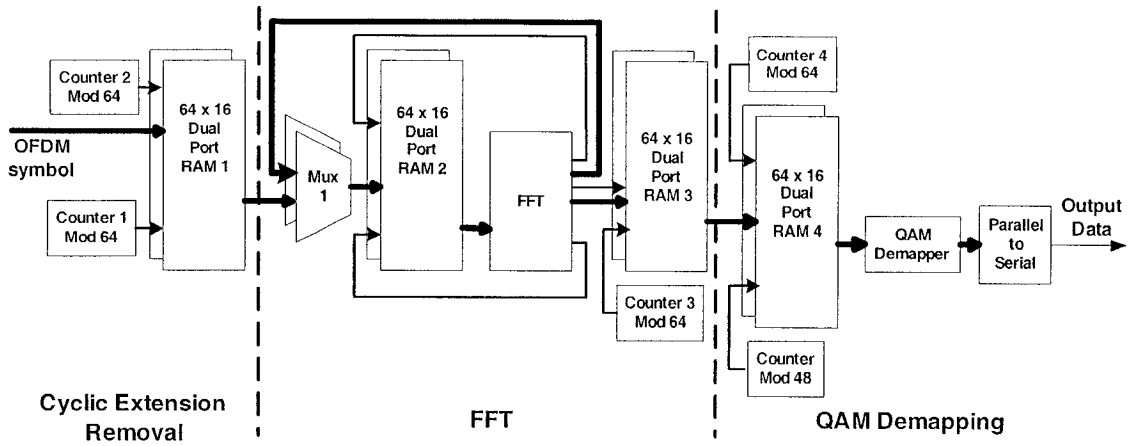

Figure 4-18: the OFDM receiver circuitry

4.3.2 VHDL code simulation and test

To simulate the VHDL codes of the circuitries, we need to have some test vectors. The input data in VHDL simulator is exactly the same as the input file used in fixed-point model. In "Signal Calculator" of SPW, we can save the ASCII file of any signal monitored in the model. The input of fixed-point model is saved in a readable ASCII file. The signals of some test points are also saved in ASCII files. These signals are monitored by "Complex signal Sink" blocks shown in Figure 3-7. To transmit one OFDM symbol, we need to generate 288 (48 × 6) bits, which change with clock cycle every 13.888 ns (1/72 MHz). Therefore, the test vector for VHDL code should have a length of at least 288 bits. The results of simulation for each block of circuitry must be compared with the values obtained from the fixed point model.

The tool used for code simulation is "VHDL Debugger" from Synopsys. To control the simulation, a command file is prepared. This file stores the commands that simulator needs to read and execute. Through the command lines, the input values obtained from the ASCII files of the fixed-point model are entered. The simulator

executes the command contained in the file and shows the result in the "Waveform Viewer" window. The timing and value ('0' or '1') of waveforms are measurable in this window. The displayed values in this window are compared with the values obtained from the fixed-point model ASCII files. This test is done for different test vectors to make sure the modem has a correct functionality.

According to the simulation results, the total time required for one OFDM symbol computation in transmitter is 8.1 μs or 583 clock cycles. The circuitry is highly pipelined and the input data are fed into the circuitry continuously.

IEEE 802.11a standard asks for sending one short and one long preamble at the beginning of data transmission. They need totally 16 μs or 1152 clock cycles for transmission. Therefore, the circuitry can start computations 7.9 μs after the starting point of the first symbol of the short preamble.

For the receiver side, the circuitry needs 7.9 μs or 569 clock cycles to extract the transmitted bits. We can also overlap this required time interval with the timing necessary for synchronization process, as it will be discussed in the next chapter.

4.3.3 Logic Synthesis

The targeted hardware to implement the circuitry is FPGA. Virtex II family from Xilinx is chosen for this purpose. The family delivers complete solutions for telecommunication, wireless, networking and DSP applications. All cores used in this design are available for Virtex II family.

The device used for this design is XC2V6000FF1152-4. There exist 6 million gates on the device including 33,792 slices and 144 RAM blocks (each one is 18 Kbits).

The internal clock speed can go up to 420 MHz, which meets completely the maximum speed requirements of the OFDM circuitry.

The FPGA mapping phase starts with performing logic synthesis. Synopsys is used to synthesis the logic. First, we prepare an ASCII text file which is called shell script. The script includes all the commands and constraints necessary for "Design Compiler", which is another tool of Synopsis. By running the commands in the script, Synopsys creates two directories. One is called SEDIF which contains the Xilinx netlist file. This file is a netlist of basic logic gates. The other one is the Synthesized directory which holds the result of synthesis procedure. We elaborate and analyze the synthesized circuitry in "Design and Analyzer". This tool needs to read the database format file with extension "db". This file is saved in Synthesized directory. The timing and critical path reports of synthesized logic are available in "Design and Analyzer". These are the results of synthesis after FPGA cells have been replaced with gates. To obtain the area report of the modem VHDL codes, the synthesized logic is mapped into FPGA.

4.3.4 Mapping the OFDM modem into FPGA and Results

Xilinx tools are used to map SEDIF file into Virtex II XC2V6000. The tools use this file as input and convert it into a netlist of Xilinx Logic Cells. This process is called "technology mapping". The mapping optimizes the logic depending on the timing and area constraints. This phase generates the Logic Cells. The next phase, which is called routing, locates these Logic Cells within the targeted FPGA. After these two steps, the tools create a file which is used to program the FPGA. This file (with a ".bit" extension) is called Configuration file. "Xilinx Design Manager" tool is used for mapping and

routing of the synthesized logic. After mapping and routing phase, the area report for this circuitry is summarized in table 4-1.

	Slices	Flip-Flpos	RAM Blocks	Total LUT	Total Gates
QAM Mapping	49	45	4	73	263035
IFFT/FFT	992	1650	2	1716	164169
Cyclic Ext.	96	34	4	142	263583
QAM Demap.	46	30	4	70	262897
Remove Prefix	11	13	2	19	131290
Total Receiver	1150	1713	10	1937	690533
Total Transmitter	1115	1694	10	1873	690048
Total Modem	2265	3407	20	3810	1380581

Table 4-1: Area Report for OFDM modem

The timing analysis (reported from "Xilinx Timing Analyzer" tool) shows no timing conflict in the modem circuitry. The reported maximum net delay is equal to 10.023 ns and therefore, the synthesized circuitry meets the timing constraint. The floor plan is plotted by the "Floorplanner" tool of Xilinx. The cells left unused on this chip can be mapped for the synchronizer circuitry and future development in this OFDM modem.

5 Synchronization circuitry for OFDM receiver

Synchronization is one of the major tasks of any receiver. There are two synchronization tasks. One is timing synchronization and the other one is frequency synchronization. Some algorithms and methods of synchronization will be discussed in this chapter. A synchronizer circuitry based on the delayed correlation of input preambles is proposed for the OFDM receiver. The circuitry is modeled in SPW and implemented. The final result of implementation on FPGA will be demonstrated at the end of the chapter.

5.1 Introduction to OFDM synchronizations

Frequency synchronization estimates and corrects carrier frequency offsets of the received signal. Any frequency offset immediately results in ICI [6]. Since a practical oscillator does not produce a carrier at exactly one frequency, there is a random phase jitter at the received baseband signal. Therefore, the frequency (as a time derivative of the phase) is not always constant. As a result, there is no constant space between OFDM subcarriers any more. This causes ICI in an OFDM receiver. For a single carrier receiver, phase noise and frequency offsets only degrade SNR performance of the receiver. However, in OFDM modems, they introduce interference among carriers as well. This extra sensitivity to phase noise and frequency offsets is considered as a disadvantage of OFDM relative to single carrier receivers. However this degradation can be kept at minimum with utilizing a frequency synchronizer, which adjusts the local oscillator of the receiver.

On the other hand, timing synchronization has to find the symbol boundaries to prevent ISI and ICI. As discussed in Chapter 2, OFDM is robust with respect to ISI while the timing offset of an OFDM symbol (due to multipath effects of the channel) is less than the guard time interval. However, there is a relationship between symbol timing and the demodulated subcarrier phases [20]. In other words, when the timing of a symbol changes, the phases of subcarriers will also change. In an OFDM receiver, channel estimation can be performed to estimate these phase rotations among all subcarriers.

There are two techniques used to realize OFDM synchronization. The first one is based on the cyclic extensions [30], which is suitable for tracking or blind synchronization in circuit-switched connections. The second one is based on training symbols, which is used for high-rate packet transmission where the synchronization time needs to be as short as possible. These training symbols are known to the receiver. The receiver must continuously scan for incoming data and rapidly detect the training symbols and do its synchronization tasks.

IEEE 802.11a standard has introduced two short and long preambles for synchronization purpose as discussed in chapter 2 (see Figure 2-4). The short preamble symbols are used for frame detection; coarse frequency offset estimation and large-scale time synchronization. On the other hand, the long preamble training symbols are utilized for frequency fine-tuning and improving channel estimation accuracy. A pilot-assisted phase tracker enhances synchronization by removing the residual phase errors. Nevertheless, the first task of synchronization circuitry in an OFDM receiver is to detect the training symbols. In some literature, this task is called "frame detection". Several methods have been proposed for frame detection and synchronizations based on sending

training symbols. Some of these schemes and their applications are considered in the next section.

5.2 OFDM synchronization schemes

The simplest synchronization method is Maximum-Correlation (MC) scheme [21]. In this method, the complex correlation of the received input symbols r_n in a window of N_s is computed according to the following equation:

$$S_n = \sum_{m=0}^{Ns-1} r^{*}_{n+m} \quad r_{n+m+Ns} \qquad (5-1)$$

In this equation n is the sample number. Based on the maximum value of $|S_n|$, the starting point of the frame is detected. This method is not suitable for multipath fading signals because the scheme does not consider the average power inside the synchronization window. The average power can be considered in the other method called Maximum Mean-Square-Error (MMSE) scheme as proposed in [22]. The average power of the input frame in time window N_s can be calculated by using Equation (5-2):

$$P_n = \sum_{m=0}^{Ns-1} |r_{n+m}|^2 \qquad (5-2)$$

The starting point of the frame can be estimated at the maximum point of the following expression:

$$|S_n| - \frac{1}{2}\left(P_{n+N_s} + P_n\right) \qquad (5-3)$$

As the Signal to noise ratio of the received signal decreases, the error in this estimation starts to fail its detection task. Another method based on Maximum-Likelihood (ML) can be used to overcome this problem. In ML scheme a coefficient called ρ is multiplied to the average power mentioned in Equation (5-3). This method is proposed in [23] and implemented in [24] for time and frequency estimations.

The ML and MMSE schemes have shown to be efficient in continuous OFDM applications [22], [25]. For burst OFDM applications like IEEE 802.11a, other methods can be derived, which are mostly based on Maximum-Normalized Correlation (MNC) proposed in [26]. In this scheme, if we have L complex samples in one-half of the first training symbol (excluding the cyclic prefix), then the estimation window in Equations (5-1) and (5-2) will be L. The starting point of a frame can be estimated from the following equation:

$$M_n = \frac{|S_n|^2}{(P_{n+L})^2} \qquad (5-4)$$

The above criterion is considered as timing metric. As it can be seen from Equation (5-4), the window slides along in time. Meanwhile the receiver searches for the first training symbol. The parameter P_{n+L} is the received energy for the second half-symbol. The timing metric reaches a flat value within a length equal to the length of the guard time interval minus the length of the channel impulse response. At this value, there is no ISI to distort the received signal. For the AWGN channel, the timing metric reaches its maximum value inside a window with a length of guard time. Inside this window, the start point of the frame can be estimated without a loss in the received SNR.

Based on the above concept, a scheme is proposed in [27]. In this method, a threshold is assumed for the normalization of the complex correlation. For the observation time window T_o, the normalized correlation should be greater than this threshold. However, the frame detection is achieved if the normalized correlation value remains lower than the threshold in the verification window T_v. The threshold, T_v and T_o can be set according to three parameters: the targeted SNR value, the input word-length and the preamble sequence. Based on this scheme, a synchronization system is proposed in the next section. To make the design more adequate for FPGA implementation, several modifications will be also introduced.

5.3 OFDM synchronization circuitry

One of the properties of the short preamble symbol in IEEE 802.11a standard is its periodicity. Therefore, we can exploit this property by doing a correlation of incoming signal with delayed version of the same signal. This is called "Delayed Correlation".

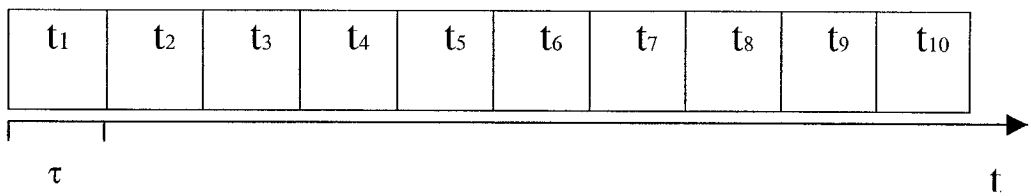

Figure 5-1: OFDM short preamble symbols

Figure 2-4 is redrawn as Figure 5-1 depicting only the short preamble signal. We can compute the correlation between a symbol and delayed version (equal to 4τ) of that symbol as defined in Equation (5-1). The summation is computed on time duration τ,

which is equal to 16 symbols. A modification can be applied here to rewrite the summation of Equation (5-1) into a recursive structure [28].

$$S_n = \sum_{i=n}^{n+N-1} x(i) \quad \Rightarrow \quad S_{n+1} = S_n + x(n+N) - x(n) \tag{5-5}$$

In some literatures, this technique is called "Sliding Summation". However, to apply this equation, we need to have a complex multiplier, two adders (one for addition and one for subtraction) and two delay blocks as shown in Figure 5-2.

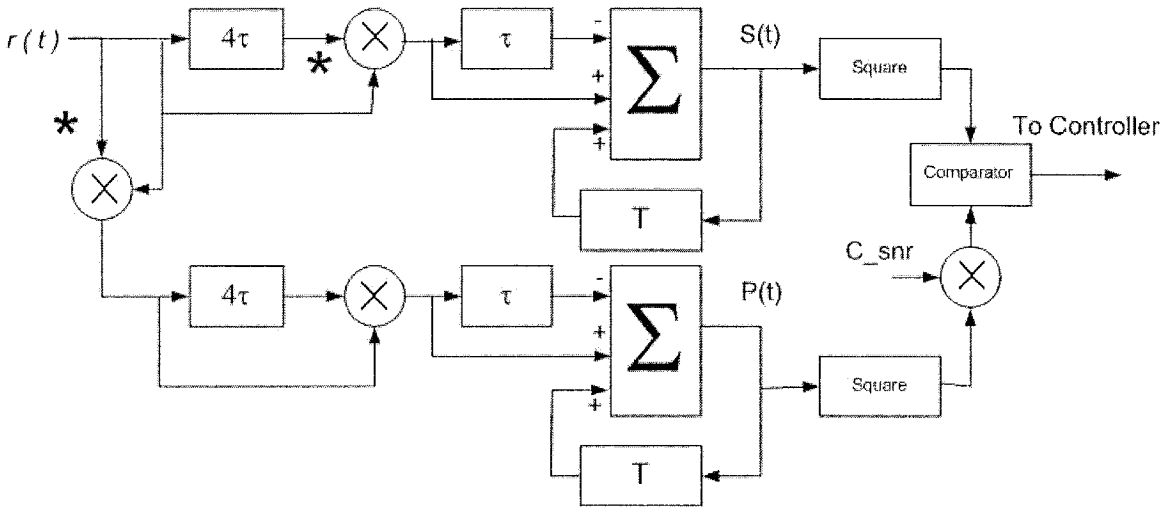

Figure 5-2: Block diagram of OFDM Synchronizer

The average power estimation defined in Equation (5-2) can be done with the same techniques (recursive summation structure). In Figure 5-2, the input signal *r(t)* is correlated with 4τ delayed version of itself. The parameter T is the sampling time and each delay τ is equal to 16 T. The * sign stands for complex conjugate of a signal. C_snr is a coefficient less than 1, which determines the threshold in the following equation:

$$|S_n|^2 \geq threshold \cdot P_n^2 \qquad (5-6)$$

We can start designing a synchronization scheme according to the block diagram in Figure 5-2. The design flow is very similar to the one explained in Chapter 4. A floating-point model is first prepared in SPW. The model is simulated in AWGN channel and the result shows that the system is able to detect correctly the preambles.

Based on the block diagram shown in Figure 5-2, a fixed-point model is presented as shown in Figure 5-3. The blocks in this figure follow the functionality explained for Figure 5-2. The counter in this figure observes the validity of Equation (5-6) for five short preamble symbols. Therefore, the square of the correlated symbols should be more than the square of power for about 5 symbols. The real and imaginary parts of the correlation result are available at the output of the design. They can be used for phase estimations and frequency synchronization. This task can be done by converting their Cartesian representation to polar coordinates with an algorithm called CORDIC [29].

To implement and synthesize the circuitry, the RTL level of VHDL codes is generated automatically in HDS. In figure 5-3 the multiplier cores from Coregen libraries are used. The VHDL code for the synchronizer is simulated and the circuitry is synthesized in Synopsys. The results from mapping the circuitry into FPGA are given in Table 5-1.

Due to the use of the multiplier code from Coregen library, "Xilinx Design Manager" has mapped the multiplier blocks into FPGA using the multiplier resources on Virtex II. The circuitry VHDL code is synthesized with medium effort for the Synopsys compiler. In this circumstance, the maximum net delay reported from Xilinx Design Manager is 9.015ns, which meets the time constraints. As it can be seen from Figure 5-3,

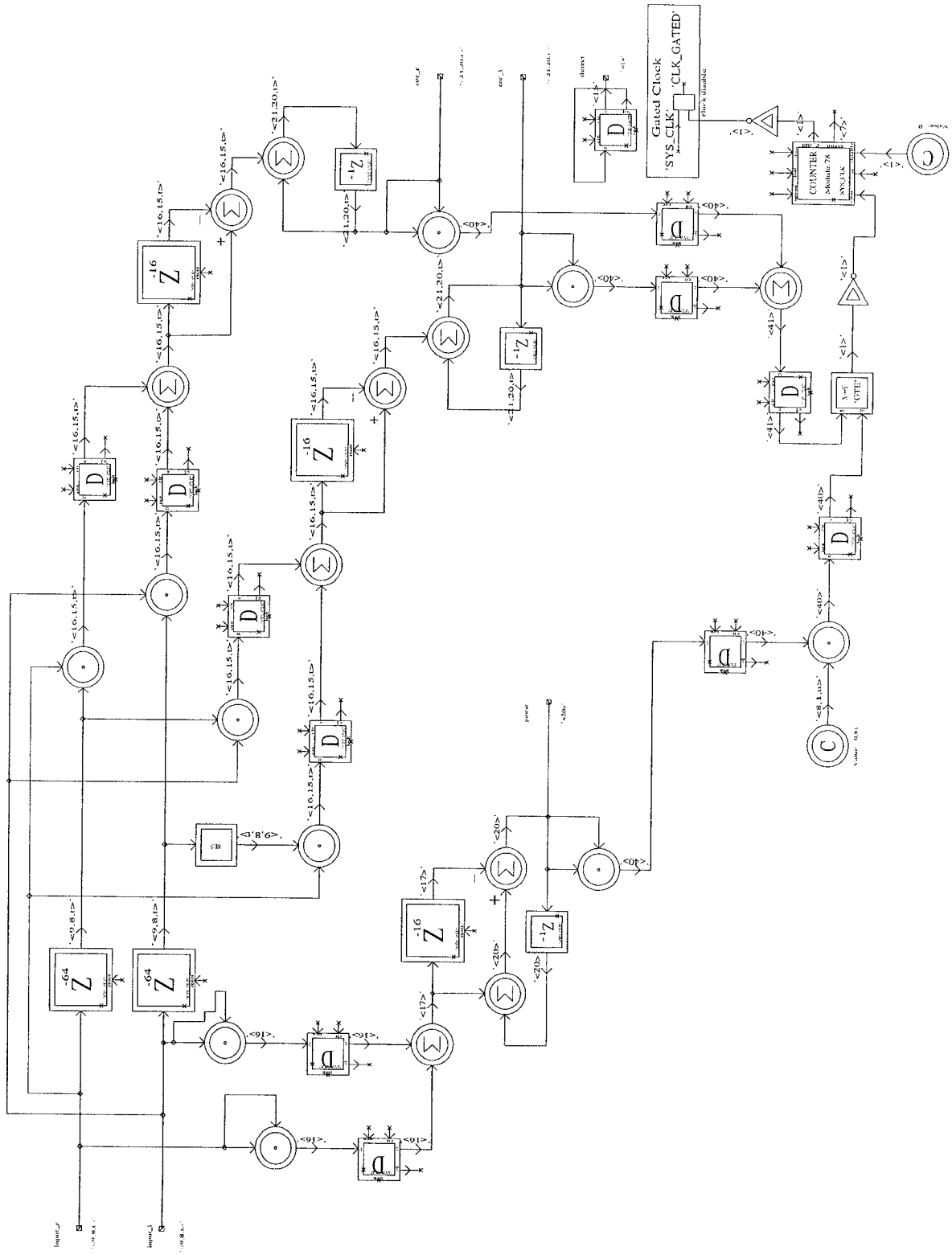

Figure 5-3: Fixed-point model for synchronization system

	Slices	Flip-Flops	Multipliers	Total Gates
Synchronizer Circuitry	1409	2303	18	97003

Table 5-1: Area report for the synchronizer circuitry

three pipeline stages are used in this design. Without these pipelines, the circuitry failed to meet the time constraints with the same medium effort which was assigned for the Synopsis compiler.

The total number of gates required for implementation of the OFDM modem is 1484000 gates. This number is calculated by adding the two areas reported in Table 5-1 and Table 4-1. the maximum net delay for the OFDM modem is 10 ns. The latency for the transmitter, receiver and synchronizer circuitries is 583, 569 and 3 clock cycles respectively.

6 Conclusion

In this thesis, a prototype design of a baseband OFDM modem was presented. The modem is based on physical layer of IEEE 802.11a standard. The modem can transmit and receive the data with bit rate up to 72 MBits/s. This is the highest speed specified by IEEE 802.11a standard for indoor wireless LAN.

The modem includes circuitries for transmitter, receiver and synchronizer. The synchronizer is able to do frame detection and symbol timing synchronization. To design the circuitry, the design flow was demonstrated in the thesis. This flow can be utilized to explore and prototype different communication and DSP algorithms. It also facilitates the evaluation of the hardware performance for various algorithms.

The design flow starts with floating–point modeling and simulation of the proposed OFDM modem in SPW. Bit Error Rate (BER) verses bit energy to noise spectral density (E_b / N_0) curve was used to evaluate the performance of the modem. The BER simulation result of the floating-point model was compared with proven equations.

Then, the floating-point model was converted to fixed-point. The number of bits representing each block was determined based on fixed-point simulation. By comparing the BER curves of the fixed-point and the one obtained from floating-point simulation, the performance degradation due to having finite number of bits in the fixed-point model was determined. To implement the circuitry, a hardware model of the system was prepared in HDS which is a library in SPW for hardware design. This library is able to automatically generate VHDL codes for the design.

For implementation, the transmitter circuitry was split into four major blocks: QAM Mapper, IFFT, Cyclic Extension and controller. The receiver also was split into

four blocks: Cyclic Extension Removal, FFT, QAM Demapper and the receiver controller. For FFT/IFFT and Dual Port RAM, it was shown how to use imported VHDL codes from the Intellectual Property of Xilinx. The VHDL code simulation results were compared with the fixed-point simulation to verify the circuitry.

The transmitter and receiver were implemented on Xilinx Virtex II FPGA. The VHDL codes were synthesized in Synopsys. Then "Xilinx Design Manager" mapped the circuitry into the FPGA XC2V6000 device. After mapping and routing phases, the reported area shows that the transmitter and receiver circuitries occupied about 1,380,580 gates on the device and the maximum delay in the circuitry was about 10ns. Since the maximum clock speed used in the system has been 72 MHz, the synthesized circuitry has met the timing constraints.

Synchronization as one of the major tasks of the OFDM receiver was demonstrated separately. Several algorithms were discussed and finally a synchronizer circuitry based on the delayed correlation of input preambles was proposed for the OFDM receiver. The algorithm measured the ratio between correlation and power of the received signal. To implement the receiver a technique called recursive summation was used. This synchronizer can be used for frame detection and symbol time synchronization. The design flow was shown for the synchronizer circuitry. It was modeled in SPW and implemented on FPGA. The reported area for this circuitry was 97,000 gates on XC2V6000 and the maximum delay in the circuitry was about 9 ns. Therefore, the final OFDM modem circuitry occupied 1,480,000 gates with maximum delay equal to 10ns.

The unused gates on the device can be mapped for the future work. For example frequency synchronizer, channel estimator, coder and interleaver are some of the blocks that can be added to the OFDM modem to implement a complete physical layer as specified by the standard.

Appendix A

vfft64 Core pins and functionalities

The vfft64 symbol and its pins are shown in Figure A-1. The functionality of each pin is given in TableA-1 [15].

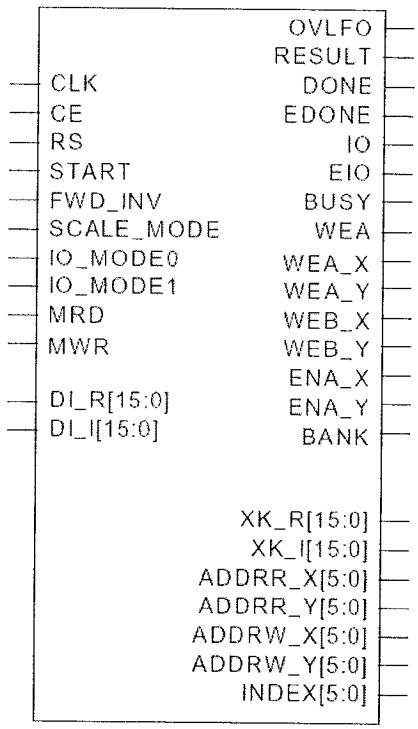

Figure A-1: the pins of vfft64 Core

The setup done on this core for the OFDM modem is:

- FWD_INV for the transmitter = '0' and for the receiver = '1'
- IO_MODE0 = '1'
- IO_MODE1 = '1'
- The others depend on the operation of the module

Table A-1: Defines the module pin functionality

Signal Name	Direction	Description
CLK	Input	Master clock (active rising edge)
RS	Input	Reset (active high)
START	Input	Start processing (active high). This signal must be synchronized with CLK.
CE	Input	Clock enable (active high)
FWD_INV	Input	Defines if a forward (FWD_INV=1) or inverse (FWD_INV=0) FFT is performed. This signal is sampled when START is asserted for all memory configuration options (SMS, DMS and TMS). When the DMS and TMS configurations are selected, where FFTs are computed continuously, FWD_INV is sampled when MODE_CE is asserted by the core. This permits alternating forward and inverse FFTs to be performed.
SCALE_MODE	Input	FFT scaling control. When SCALE_MODE=0 the FFT output vector is scaled by 1/64. When SCALE_MODE=1 the FFT output vector is scaled by 1/128. This signal is sampled when START is asserted for all memory configuration options (SMS, DMS and TMS). When the DMS and TMS configurations are selected, where FFTs are computed continuously, SCALE_MODE is sampled when MODE_CE is asserted by the core. This permits different scaling options to be used for successive FFTs without any interruption to processing.
MWR	Input	Input data write strobe (active high). This signal must be synchronized with CLK.
MRD	Input	Result vector read strobe (active high). This signal must be synchronized with CLK.
DI_R[15:0]	Input	Input data bus – real component. The real component of the input data vector is presented to the core on this port. Two's complement data format is assumed.
DI_I[15:0]	Input	Input data bus –imaginary component. The imaginary component of the input data vector is presented to the core on this port. Two's complement data format is assumed.
IO_MODE0	Input	Together with the IO_MODE1 pin, these signals define the type of memory interface to be used, single, dual or triple memory space configurations. The memory configurations also effectively define the method by which data is presented to the core and read back from the core. The precise functionality of IO_MODE0 and IO_MODE1 are defined in the I/O Interface and Memory Configurations section of this document.
IO_MODE1	Input	I/O interface and memory configuration select pin – refer to the signal description for IO-MODE0 for details.
OVFLO	Output	Active high Arithmetic overflow indicator. Even when employing a 2-bit scale factor for each FFT processing phase, certain input signals can cause arithmetic overflow. This pin indicates that an internal arithmetic overflow has been generated. When additional scaling is employed by setting SCALE_MODE=1, there is no possibility of overflow occurring and this signal will not be active. OVFLO is removed when the core is reset by asserting RS, when START is asserted, or at the beginning of the next output result vector as indicated by DONE.
RESULT	Output	FFT result strobe (active high). This signal indicates that a new (I)FFT result vector is available. It frames the XK_R and XK_I output buses.
DONE	Output	FFT complete strobe (active high). DONE will transition high for one clock cycle at the end of the RESULT strobe.
EDONE	Output	Early done strobe (active high) EDONE goes high for one clock cycle immediately prior to RESULT going active.
IO	Output	IO cycle strobe. (active high) This signal is only intended to be used with the dual-memory-space core configuration.
EIO	Output	Early I/O strobe. (active high) This signal is only intended to be used with the dual-memory-space core configuration.
BUSY	Output	Core activity indicator (active high). This signal will go high in response to the start signal and will remain high while the core is processing data.
MODE_CE	Output	The operation mode signals FWD_INV and SCALE_MODE are sampled when

Table A-1 (cont.): Defines the module pin functionality

		MODE_CE is active. This is an active high signal that is asserted by the core for one clock period several clock cycles prior to the start of a new computation in the dual-memory-space and triple-memory-space core configurations.
WEA	Output	Block RAM port A write enable strobe (active high). This write signal is used for the single-memory-space configuration.
WEA_X	Output	Block RAM port A write enable strobe (active high). This write signal is used for the dual- and triple-memory-space configurations.
WEA_Y	Output	Block RAM port A write enable strobe (active high). This write signal is used for the dual and triple memory space configurations.
WEB_X	Output	Block RAM port B write enable strobe (active high). This write signal is used for the dual- and triple-memory-space configurations.
WEB_Y	Output	Block RAM port B write enable strobe (active high). This write signal is used for the dual and triple memory space configurations.
ENA_X	Output	Block RAM port A enable signal (active high). This memory enable signal is used for the single, dual and triple-memory- space configuration options.
ENA_Y	Output	Block RAM port A enable signal (active high). This memory enable signal is used for the triple-memory-space configuration option.
BANK	Output	Memory bank select signal to be used with the triple- memory-space core configuration.
ADDRR_X[5:0]	Output	Block RAM read address bus used for the single, dual and triple memory configurations.
ADDRR_Y[5:0]	Output	Block RAM read address bus used for the dual and triple memory configurations.
INDEX[5:0]	Output	Output data re-ordering bus. This bus is used in the DMS and TMS memory configurations for re-ordering the transform result (XK_R and XK_I) as it is written to the result memory. The INDEX bus generates a radix-4 digit reversed sequence of addresses.
ADDRW_X[5:0]	Output	Block RAM write address bus used for the single, dual and triple memory configurations.
ADDRW_Y[5:0]	Output	Block RAM write address bus used for the dual and triple memory configurations.
XK_R[15:0]	Output	FFT result bus – real component. The real component of the FFT result vector is presented on this bus. The values are in two's complement format.
XK_I[15:0]	Output	FFT result bus –imaginary component. The imaginary component of the FFT result vector is presented on this bus. The values are in two's complement format.

References

1. IEEE Std 802.11-1999, Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) specifications, The Institute of Electrical and Electronics Engineers, Inc.
2. IEEE Std. 802.11a-1999, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: High-speed Physical Layer in the 5 GHz Band, The Institute of Electrical and Electronics Engineers, Inc.
3. A.S. Tanenbaum, *Computer Networks*, Upper Saddle River, NJ : Prentice Hall, 4th edition 2003
4. M. Loukides, *802.11 Wireless Network: The Definitive Guide,* Sebastopol, CA : O'Reilly, 1st edition 2002
5. A. Burr, *Modulation and Coding for wireless communications*, Harlow, Eng. : Prentice Hall, 1st edition 2001
6. R. van Nee and R. Prasad, *OFDM Wireless Multimedia Communications.* Artech House, Boston, 2000.
7. B.P. Crow, I. Widjaja, L.G. Kim, Sakai, "IEEE 802.11 Wireless Local Area Networks", *IEEE Communication Magazine,* Volume: 35, Issue: 9, Pages:116 – 126, September 1997
8. R. van Nee, G.A water, M. Morikura, H. Takanashi, M. Webster, K.W. Halford, "New High-Rate Wireless LAN Standards", *IEEE Communication Magazine,* Volume: 37, Issue: 12 , Pages:82 – 88, December 1999

9. S. Armour, A. Nix, D. Bull, "Complexity evaluation for the implementation of a pre-FFT equalizer in an OFDM receiver", *IEEE Transactions on Consumer Electronics*, Volume: 46, Issue: 3 , Pages:428 - 437, August 2000

10. T.S. Rappaport, *Wireless Communications: Principle and Practice,* Prentice Hall, Upper Saddle River, NJ07458, 2nd edition 2002

11. R.E. Blahut, *Fast Algorithms for Digital Signal Processing,* Reading, Mass.: Addison-Wesley Pub. Co, 1985

12. A.V. Oppenheim, R.W. Schafer, *Discrete-Time Signal Processing,* Prentice Hall, Englewood Cliffs, NJ 07632, 1989

13. J.G. Proakis, D.G. Manolakis , *Introduction to Digital Signal Processing,* Macmillan Publishing Company, 866 Third Avenue, New York, New York 10022, 1989

14. B. Sklar, *Digital Communications, Fundamentals and Applications*, Prentice Hall Inc., Upper Saddle River, New Jersey 07458, 2001

15. Xilinx Inc. online technical documentation for "Intellectual Property and IP cores" and "Hardware Data Sheet for Virtex II Platform", http://www.xilinx.com/literature/

16. J.W. Cooley and Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Comput., Volume: 10, Pages: 297-301, April 1965

17. S.D. Brown, R.J. Francis, J. Rose, Z.G. Vranesic, *Field-Programmable Gate Arrays,* Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061 USA

18. S. Sjoholm, VHDL for Designers, London ; New York : Prentice Hall 2003

19. K. Shahill, *VHDL for Programmable Logic,* Addison-Wesley Publishing Company Inc., 2725 Sand Hill Road, Menlo Park, CA94025, 1996

20. T.N. Zogakis, and J.M. Cioffi, "The effect of timing jitter on the performance of a discrete multitone system", *IEEE Transactions On Communications*, Volume: 44, No. 7, Pages: 799-808, July 1996

21. T. Keller and L. Hanzo, "Orthogonal Frequency Division Multiplexing Synchronization Techniques for Wireless Local Area Networks", *IEEE PIMRC*, Pages: 963-967, Taipei, Taiwan, 1996

22. P.R. Chevillat, D. Maiwald, and G. Ungerboek, "Rapid Training of a Voiceband Data-Modem Receiver Employing an Equalizer with Fractional-T Spaced Coefficients", *IEEE Transactions On Communication.*, Volume: 35, No. 9, Pages 869-876, 1987.

23. J.J. van de Beek, M. Sandell and P.O. Börjesson, "ML Estimation of time and Frequency Offset in OFDM Systems", *IEEE Transactions on Signal Processing,* Volume: 45, No.7, Pages: 1800-1805, July 1997

24. S. Johansson, P. Nilson and M. Torkelson, "Implementation of an OFDM Synchronization Algorithm", 42nd *Midwest Symposium on Circuits and Systems*, Volume: 1, Pages:228 - 231, 8-11 August 1999

25. M. Sandell, J.J. van de Beek and P.O. Börjesson, "Timing and Frequency Synchronization in OFDM Systems using the Cyclic Prefix", *Proc. International Symposium on Synchronization,* Pages: 16-19 , Essen, Germany, 1995

26. T.M. Schmidl and D.C. Cox, "Robust Frequency and Timing Synchronization for OFDM", *Transactions on Communication*, Volume: 45, No: 12, Pages: 1613-1621, December 1997.

27. L.D. Kabulepa, A.G. Ortiz and M. Glesner, "Design of an Efficient OFDM Burst Synchronization Scheme", *IEEE International Symposium on Circuits and Systems*, (ISCAS 2002), Volume: 3, Pages: III-449 - III-452, 26-29 May 2002

28. L. Schwoerer, "VLSI suitable Synchronization Algorithms and Architecture for IEEE 802.11a Physical Layer", *IEEE International Symposium on Circuits and Systems*, (ISCAS 2002), Volume: 5, Pages: V-721 - V-724, 26-29 May 2002

29. J.E. Volder, "the CORDIC Trigonometric Computing Technique", *IRE transaction Electronic Computers*, Volume: EC-8, Page: 330-334, 1959

30. H. Zou, B. McNair and B. Daneshrad, "An OFDM Receiver for High-Speed Mobile Data Communications", *IEEE Global Telecommunications Conference*, 2001. GLOBECOM '01, Volume: 5, Pages: 3090 – 3094, 25-29 November 2001

31. B. Saltzberg , "Performance of an Efficient Parallel Data Transmission System", *IEEE Transaction on Communication*, Volume: 15, Issue: 6, Pages: 805 –811, December 1967

32. "Orthogonal Frequency Division Multiplexing", US Patent No. 3,488,455 filled in November 14, 1966, Issued in January 6,1970

33. S. Weinstein, P. Ebert, "Data Transmission by Frequency-Division Multiplexing Using the Discrete Fourier transform", *IEEE Transaction on Communications,* Volume: 19, Issue: 5, Pages: 628-634, October 1971

34. B. Hirosaki, "An Orthogonally multiplexed QAM System Using the Discrete Fourier Transform", *IEEE Transaction on Communications,* Volume: 28, Issue: 7, Pages: 982-989, July 1981

35. J. Terry, J. Heiskala, *OFDM Wireless LANs: A Theoretical and Practical* Guide, Sams Publishing, 201 West 103rd St., Indianapolis, Indiana, 46290,USA, Second printing with corrections: September 2002

36. C.F. Hsu; Y.H. Huang; T.D. Chiueh, "Design of an OFDM receiver for high-speed wireless LAN", *The 2001 IEEE International Symposium on Circuits and Systems*, Volume: 4 , Pages:558 – 561, 6-9 May 2001

37. M. Engels, W. Eberle, B. Gyselinckx, "Design of a 100 Mbps wireless local area network", *International Symposium on Signals, Systems, and Electronics, 1998*, Pages: 253 – 256, 29 September-2 October 1998