

RICE UNIVERSITY


Symbol Timing Synchronization for OFDM-based
WLAN Systems

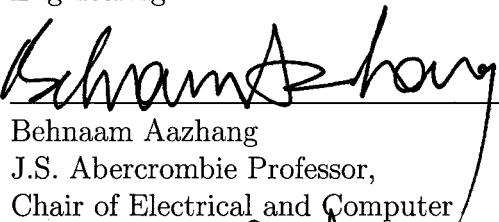
by

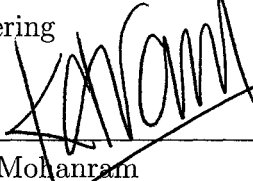
Manik Gadhiok

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
Master of Science

APPROVED, THESIS COMMITTEE:


Joseph R. Cavallaro, Chair
Professor of Electrical and Computer
Engineering


Behnaam Aazhang
J.S. Abercrombie Professor,
Chair of Electrical and Computer
Engineering


Kartik Mohanram
Assistant Professor of Electrical and
Computer Engineering

Houston, Texas

July, 2007

UMI Number: 1455238

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1455238

Copyright 2008 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

Symbol Timing Synchronization for OFDM-based WLAN Systems

by

Manik Gadhiok

In this work, we address the problem of symbol timing synchronization for Orthogonal Frequency Division Multiplexing (OFDM) WLAN systems. OFDM systems are known to be sensitive to synchronization errors and improving the accuracy of timing offset estimation can help improve the overall system performance.

We propose a method that reduces computational complexity while achieving performance comparable to the autocorrelation method commonly employed at the wireless receiver. The proposed method is based on the average magnitude difference function (AMDF). We present performance results for the proposed method for AWGN and Rayleigh-fading channels in the context of IEEE 802.11a short preamble sequence. We also propose a preamble sequence based on Golomb sequence, a sequence with low auto-correlation properties, and compare its timing estimation performance with that of the IEEE 802.11a short-sequence. Simulation results show significant performance improvements for AWGN as well as Rayleigh-fading channels.

We also present an experimental Field-Programmable Gate Array (FPGA) implementation of the symbol timing estimation block using an Intermediate-frequency based hardware transceiver and LabVIEW using fixed-point arithmetic.

Acknowledgments

I would like to take this opportunity to thank all the people at Rice University who have been instrumental in my graduate school learning. I would to thank my advisor, Dr. Joseph R. Cavallaro, for his support and interest in this work. I am grateful for his suggestions and guidance throughout the course of this project. This thesis would not have been possible without his support. I extend my appreciation to the committee members, Dr. Behnaam Aazhang and Dr. Kartik Mohanram for their suggestions and help. The folks at National Instruments have provided the funding, hardware platform and software for this project and I would also like to thank them. Also, thanks to Alexandre DeBaynst, who got me started on this problem and helped in the earlier stages of this project.

I would like to express my thanks to all my friends and colleagues in ECE department for making my graduate studies at Rice University, a wonderful and memorable experience. Finally, the biggest thanks and credit goes to my family who have been instrumental in shaping my life.

Contents

Abstract	ii
Acknowledgments	iii
List of Illustrations	vi
List of Tables	ix
1 Introduction	1
1.1 Thesis Contributions	2
1.2 Thesis Organization	2
2 Symbol Timing Synchronization in OFDM Systems	4
2.1 Overview of OFDM WLAN Systems	4
2.1.1 Orthogonal Frequency Division Multiplexing (OFDM)	5
2.1.2 OFDM for Wireless LANs	7
2.1.3 Wireless System Parameters	8
2.2 System Model	8
2.3 Symbol Timing Synchronization	10
3 Symbol Timing Estimation: Algorithm design and performance analysis	17
3.1 Related Work	18
3.2 Proposed symbol timing estimation method	19
3.2.1 Complexity Analysis	21
3.2.2 Simulation Results	23
3.3 Preamble sequence	26

3.3.1	Golomb Sequence	28
3.3.2	Performance Results	33
3.4	Conclusions	35
4	System Hardware and Design Environment	38
4.1	NI-5640R IF Transceiver Module	39
4.1.1	Hardware Components	40
4.1.2	ADC, DAC, and Host Interface	42
4.2	LabVIEW Programming Environment	44
4.2.1	LabVIEW Host	45
4.2.2	LabVIEW FPGA	46
4.2.3	NI-5640R Driver Software	46
5	FPGA Implementation of Symbol Timing Estimation	48
5.1	Fixed-Point Wordlength Analysis	48
5.2	System Partitioning	49
5.3	Host VI and Analog Interface	51
5.4	FPGA System Implementation	55
5.5	Synthesis Results	61
6	Conclusions and Future Work	64
	Bibliography	66

Illustrations

2.1	Illustration of orthogonal sub-carriers in OFDM signal	6
2.2	Block diagram of OFDM transmitter	10
2.3	Block diagram of OFDM receiver	10
2.4	Receiver signal processing	11
2.5	IEEE 802.11a OFDM frame	12
2.6	Received signal generated based on short symbol repetition	14
2.7	Normalized autocorrelation metric with timing offset of 30 samples .	15
3.1	Average magnitude difference function with a timing offset of 30 samples	20
3.2	Normalized correlation metric with a timing offset of 30 samples . . .	21
3.3	Probablity of correct estimation: Comparison for AMDF and modified average magnitude difference function (without square-root operation)	25
3.4	Probablity of correct estimation: Comparison between AMDF and Schmidl-Cox algorithm with different window size for computing signal energy	26
3.5	Mean estimation error estimation: Comparison between AMDF and Schmidl-Cox with different window size for computing signal energy .	27
3.6	Probability of correct detection: Comparison between Autocorrelation and AMDF methods	28
3.7	MSE comparison between Autocorrelation and AMDF methods . . .	29

3.8	Sample autocorrelation function of IEEE 802.11a STS (left) and Golomb sequence based training signal (right)	30
3.9	Frequency-domain sample autocorrelation function of IEEE 802.11a STS (left) and Golomb sequence based training signal (right)	31
3.10	Time-domain representation of IEEE 802.11a STS and Golomb sequence based training signal	32
3.11	Frequency-domain representation of IEEE 802.11a STS and Golomb sequence based training signal	33
3.12	Probability of correct detection for AWGN channel	34
3.13	Estimator MSE for AWGN channel	35
3.14	Probability of correct detection for multi-path channel	36
3.15	Probability of correct detection for multi-path channel: performance gains for higher SNRs	37
4.1	Block diagram of NI-5640R (adapted from [1])	41
4.2	NI-5640R configuring clocks (adapted from [1])	43
4.3	NI-5640R configuration loop	46
5.1	Verification of fixed point results	50
5.2	LabVIEW system overview	51
5.3	LabVIEW FPGA DAC loop	53
5.4	ADC loop	54
5.5	FPGA block diagram layout	55
5.6	Storing incoming data samples	56
5.7	LabVIEW FPGA symbol timing implementation	57
5.8	Initial norm computation	58
5.9	Updating the norm	58
5.10	Initial correlation computation	59

5.11 Correlation metric update	59
5.12 Estimation and DMA transfer	60
5.13 Timing offset estimation	60
5.14 Front panel view of FPGA VI	62

Tables

2.1	OFDM system parameters	9
3.1	Complexity analysis	23
5.1	FPGA synthesis results	61

Chapter 1

Introduction

This work addresses symbol timing synchronization for OFDM Wireless Local Area Networks (WLANs). We present a low-complexity method for estimating the timing offset that has performance comparable with the traditional auto-correlation based methods. Also, a preamble sequence based on Golomb sequence is proposed that exhibits low auto-correlation properties and results in better estimation performance at the receiver. Finally, a Field Programmable Gate Array (FPGA) based hardware prototype for this task is presented.

Orthogonal Frequency Division Multiplexing (OFDM) systems have gained popularity over the last few years for broadband wireless communication. Wireless Local Area Networks (WLAN) such as IEEE 802.11a [2] and HiperLanA [3] [4] have been designed and deployed successfully, along with some more recent standards such as IEEE 802.11n, IEEE 802.16 etc. for WLAN and Wireless Metropolitan Area Networks (WMAN) systems. By employing OFDM, these wireless communication systems strive to achieve good spectral efficiency, high data-rates, robust communication, and relatively lower computational complexity.

With the communication and signal processing algorithms research working to establish wireless communication links that can support data rates closer to the channel capacity, more signal processing capabilities are required at the wireless receiver. From a hardware designer's standpoint, architecture design and system prototyping are a key to support these algorithm advances, while meeting the size, battery life

and real-time budgets.

Orthogonal Frequency Division Multiplexing has proved to be an enabling technology for high data rate wireless communication networks [5]. Wireless LAN standards such as IEEE 802.11a/g/n as well as broadband wireless access systems such as IEEE 802.16 use OFDM modulation. Because of the bursty nature of data transmission and fast acquisition times needed, preamble-based methods are used to acquire symbol timing and carrier frequency synchronization. The sensitivity of OFDM receiver performance to symbol timing estimation and carrier frequency offset (CFO) estimation errors is well known [6] [7].

1.1 Thesis Contributions

The main contributions of this work are as follows. We address symbol timing synchronization issues for OFDM Wireless Local Area Networks (WLAN). We present a method for preamble-based symbol timing synchronization that exhibits lower-complexity and performs comparable to existing methods. Further, we present simulation results showing that a performance improvement can be achieved in WLAN systems by replacing the short symbols with low-autocorrelation sequences, for example, Golomb sequence. An FPGA-based hardware implementation of the symbol timing synchronization block is done using an intermediate frequency (IF) transceiver built around Xilinx Virtex-II Pro FPGA and LabVIEW software.

1.2 Thesis Organization

The thesis is organized as follows. The OFDM communication system being considered and overview of symbol timing estimation are described in Chapter 2. In Chapter 3, we present the proposed timing synchronization method along with simu-



lation results and complexity analysis. We also present an application of a sequence known to have good periodic correlation properties (Golomb sequence) to the problem of symbol timing estimation in the context of WLAN systems. The performance benefits of this preamble are discussed. In Chapter 4, we introduce the NI-5640R FPGA-based system hardware and the LabVIEW design environment. In Chapter 5, we present a hardware implementation of the symbol timing synchronization block. Finally, we end with conclusions and future work in Chapter 6.

Chapter 2

Symbol Timing Synchronization in OFDM Systems

In this chapter, we provide details of the wireless system considered in this work. We begin with the fundamentals of Orthogonal Frequency-Division Multiplexing (OFDM) technique in the context of Wireless Local Area Networks (WLAN) systems. Next, we consider the timing synchronization problem and the general receiver architecture for synchronization. Finally, we put the timing synchronization in context with an example of the training sequence and metric used for achieving coarse symbol timing estimation in an IEEE 802.11a receiver.

2.1 Overview of OFDM WLAN Systems

Wireless communication is increasingly becoming an integral part of our lives. By allowing for communication without wires, a wireless system may require lesser infrastructure, allow faster deployment, support more dynamic network usage, and roaming. Some of these wireless systems might be ad-hoc in nature, such as networks set-up to aid emergency workers; others might be more or less fixed with some roaming or mobility among users. Mobile phones allow users to communicate while moving at highway speeds and are supported by fixed infrastructure. High-definition digital TV viewers may receive signals from the satellite where the receiver position is assumed fixed relative to the satellite.

For a successful wireless communication system, the system design depends on

the nature of the communication system and usage. The signal processing algorithms and hardware necessary for a broadcast digital TV network, such as DVB-T, may be quite different from that necessary for an ethernet-based wireless LAN system. While a wireless receiver in a broadcast network may rightly assume that data is being transmitted continuously, a WLAN receiver should deal with the bursty or on/off nature of the data transmission.

For our work, we focus on wireless LANs. Wireless local area networks (WLANs), generally have fixed base-stations and allows users to connect to the internet using standard protocols, such as IEEE 802.11a. WLAN users generally don't require support for high mobility and can be assumed to be more or less stationary. Orthogonal frequency division multiplexing is one of the modulation schemes used for WLAN communication systems.

Wireless broadband networks , wireless personal area networks , and wireless local area networks are among the many applications where OFDM has been deployed or proposed as a standard. These include IEEE 802.16 Wireless metropolitan area networks (IEEE 802.16 WMAN), IEEE 802.11a/n, WiMAX and IEEE 802.15 WPAN.

2.1.1 Orthogonal Frequency Division Multiplexing (OFDM)

Orthogonal Frequency Division Multiplexing (OFDM) is a modulation scheme where the broadband wireless channel is subdivided into multiple parallel lower-data rate channels. The OFDM transmitter modulates information separately on each sub-channel by considering the signal bandwidth as comprised of multiple parallel channels. The sub-channels, also known as sub-carriers, are equally spaced, the sub-carrier spacing being an important parameter. Employing orthogonal sub-carriers allows an OFDM system to pack sub-carriers close to each other without causing any degra-

dation in performance. This increases the spectral efficiency, although the receiver performance is affected if the orthogonality between sub-carriers is not maintained due to effects of the wireless channel [6]. Figure 2.1 illustrates the concept of OFDM with 7 orthogonal sub-carriers. The dotted lines represent the center frequency for each of the sub-carriers. These are the ideal sample points; as shown in Figure 2.1, the contribution of all the other sub-carriers at these points is zero. If due to synchronization errors, the signal was sampled at some offset from these center frequencies, there would be a loss of orthogonality among the sub-carriers and consequent degradation in performance [6] [8].

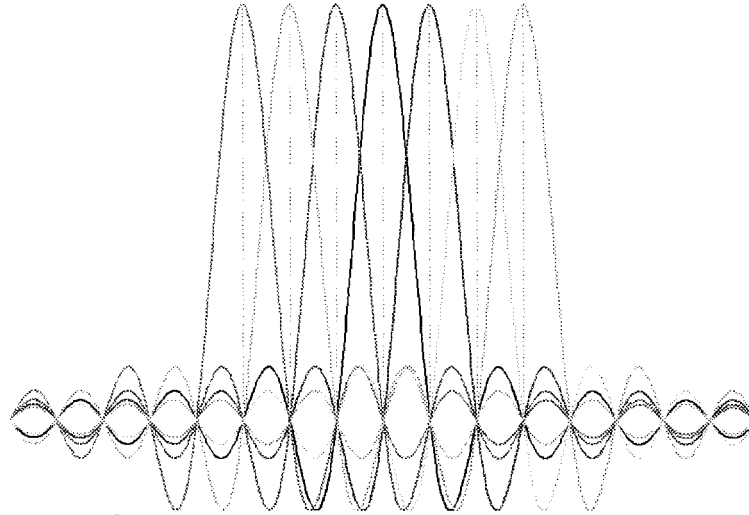


Figure 2.1 : Illustration of orthogonal sub-carriers in OFDM signal

OFDM communication systems are inherently more robust to impulse-noise and narrow-band interference. Also, by appending a cyclic prefix in the time-domain signal representation, the system is able to mitigate inter-symbol interference and improve sensitivity against multipath fading.

4

From an architecture standpoint, an OFDM receiver is more amenable for hardware implementation. As communication systems push for higher data-rates using techniques such as multi-antenna systems [9], the signal processing required at the receiver becomes increasingly complex. The robustness of OFDM systems to channel distortion results in reducing the complexity of channel equalization needed at the receiver [10] [9]. Also, because OFDM uses Fourier transform, an efficient implementation using the Fast Fourier transform (FFT) algorithm is feasible [9].

The added flexibility in OFDM, whereby a sub-set of the sub-carriers can be turned off, is noteworthy. This facilitates having a common wireless system that can work across multiple geographies having different frequency licensing restrictions. The multi-band OFDM initiative can turn-off spectrum for taking advantage of this feature, as well as for protecting certain critical frequency bands [11]. Additionally, there has been a lot of interest recently in the area of cognitive radios and dynamic spectrum sharing. The flexibility of the OFDM system makes it a strong candidate for these systems [12].

2.1.2 OFDM for Wireless LANs

We consider a packet-based OFDM transmission system. The n th sample of the m th OFDM symbol can be represented as the inverse Discrete Fourier Transform (IDFT) of the complex data vector $d_{m,0}..d_{m,N-1}$

$$x_m(n) = \sum_{k=0}^{N-1} d_{m,k} \exp(j2\pi kn/N) \quad (2.1)$$

where $0 \leq n \leq N-1$. Here N represents the number of sub-carriers or equivalently the length of the DFT. A cyclic prefix of length N_g is added to give the m th transmitted OFDM symbol $x_m = [x_{m,N-N_g}, \dots, x_{m,N-1}, x_{m,0}, \dots, x_{m,N-1}]$. A preamble sequence is

inserted at the beginning of each frame of data before transmission. The transmitted signal passes through the quasi-static wireless multipath-channel, assumed constant over one OFDM symbol, and additive white Gaussian noise (AWGN) is added. In the absence of any synchronization errors, the receiver can perform a DFT in order to recover the original data vector. In practice, the DFT and IDFT operations are implemented using the Fast Fourier Transform (FFT) algorithm.

Because of the bursty nature of data transmission and fast acquisition times needed, preamble-based methods are used to acquire symbol timing and carrier frequency synchronization. The sensitivity of OFDM receiver performance to symbol timing estimation and carrier frequency offset (CFO) estimation errors is well known.

2.1.3 Wireless System Parameters

The IEEE standard body has proposed many standards based on OFDM that are used for WLAN and wireless metropolitan area networks (WMAN). These include IEEE 802.11a/g/n and IEEE 802.16. We have chosen the IEEE 802.11a system, though the recent IEEE 802.11n standard also has similar system parameters. All of these packet-based OFDM communication systems require a preamble-sequence to be prefixed to the transmitted signal to facilitate fast signal acquisition and recovery of received signal. The system parameters are shown in Table 2.1.

2.2 System Model

Here we describe the wireless communication system under consideration. Figure 2.2 shows the block diagram of the transmitter. The data or information bits are mapped using conventional modulation such as Quadrature Phase Shift Keying (QPSK), Quadrature Amplitude Modulation (QAM). These symbols are then modulated using

Table 2.1 : OFDM system parameters

Parameter Name	Parameter Value
OFDM Symbol Duration	3.2 microseconds
Guard Interval	0.8 microseconds
FFT Size	64
Number of data sub-carriers	52
Sub-carrier spacing	312.5 kHz
Sampling Rate	$1/T$
Modulation Scheme	QPSK

the Inverse fourier transform (IFFT) operation. The IFFT is a block operation and modulates N parallel data streams, where N is the size of the transform. A guard interval is appended to the transform output to form a complete OFDM symbol. Next, the frame is generated by appending a preamble sequence to one or more OFDM data symbols. The preamble sequence helps the receiver in packet detection, synchronization and channel estimation tasks. We discuss the preamble sequence structure and its use at the receiver in Chapter 3. Finally, the baseband signal undergoes oversampling, pulse-shaping, and digital-to-analog conversion (D/A) and transmission occurs.

At the receiver, the received signal first undergoes analog-to-digital conversion (A/D) and downsampling. There is typically an automatic gain control (AGC) loop that helps with setting the signal amplitude. Figure 2.3 shows the receiver block diagram.

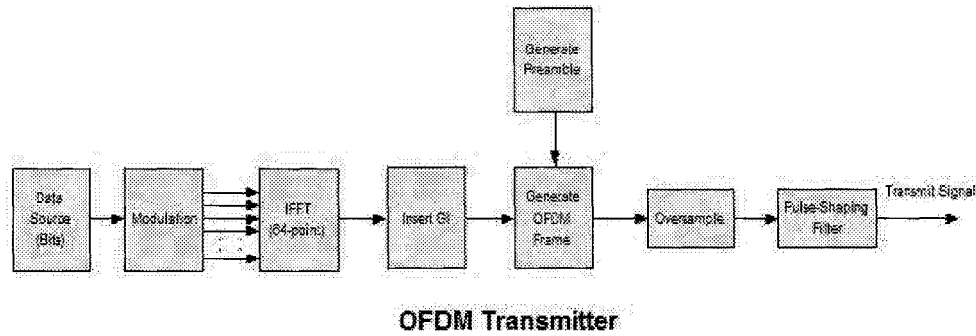


Figure 2.2 : Block diagram of OFDM transmitter

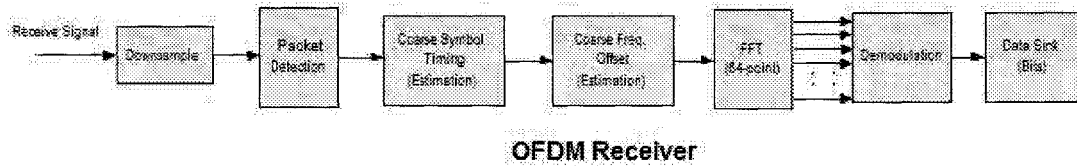


Figure 2.3 : Block diagram of OFDM receiver

2.3 Symbol Timing Synchronization

The performance of a wireless receiver is measured by its ability to successfully recover or decipher transmitted information. In the context of wireless communications, the baseband signal processing algorithms generally assume perfect synchronization. In practical systems, however, synchronization at the receiver is a critical aspect that impacts the performance of the entire system. In fact, synchronization is one of the main challenges in OFDM systems. Before actual processing and decoding of the received data can occur, synchronization needs to be achieved. The receiver signal processing blocks downstream depend on this and consequently are susceptible to errors in synchronization.

For packet-based burst-mode communication systems (for example WLAN), the receiver has to first detect the presence of a packet and determine the beginning of the data. For an OFDM system, the receiver signal processing algorithms are implemented in the frequency-domain. Therefore, estimating the beginning of the data or equivalently the start of the FFT window has a direct impact on the receiver performance. This task is called symbol timing synchronization. OFDM systems are also sensitive to frequency offsets - generally caused due to carrier frequency mismatch or doppler spread. By utilizing the structure in the transmitted signal, most OFDM receiver architectures apply coordinate rotation digital computer (CORDIC) to the results of symbol timing synchronization to estimate this frequency offset.

Figure 2.4 shows the functions performed by a typical OFDM WLAN receiver. Once the received signal has undergone analog-to-digital conversion and downconversion, the baseband IQ (in-phase and quadrature-phase) samples are available for processing. The first task is to detect the presence of a packet and estimate the start of the Fast Fourier transform (FFT) window.

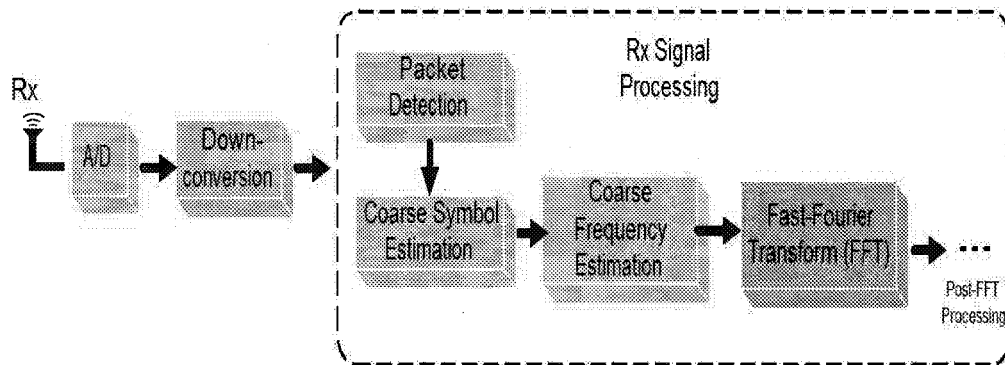


Figure 2.4 : Receiver signal processing

For bursty packet-based communication systems, preamble-based methods are

used for achieving timing and frequency synchronization. In contrast to the blind or non-data aided methods, the preamble-based synchronization gives accurate and reliable estimates much faster and within the first one or two received symbols. Figure 2.5 shows the structure of the OFDM frame for the IEEE 802.11a standard. At the beginning of a new transmission, the transmitter sends a known training sequence (preamble) followed by the data bits. The receiver uses knowledge of this preamble sequence for synchronization. The receiver employs the short training sequence to achieve packet detection, automatic gain control (AGC), symbol timing synchronization and frequency offset estimation. The long sequence is then used to further refine the timing and frequency estimates as well as for estimating the wireless channel.

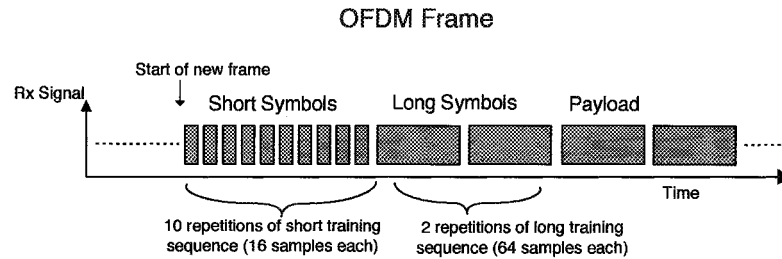


Figure 2.5 : IEEE 802.11a OFDM frame

Relating back to Figure 2.4, the coarse symbol estimation block uses correlation-based methods (typically autocorrelation) to detect where the packet begins and dictates the samples to which the Fast Fourier transform is applied. Based on the output of the coarse symbol estimation, a CORDIC block can then be used to find the coarse frequency offset in the received signal. Typically, the coarse estimation block is expected to find the start of the FFT window within a certain range and the fine estimation blocks then strive to improve this estimate further.

In many practical implementations, the sequence of events can be described as



follows. First, a packet detector will detect the presence of a packet. One way to do this is to use the received signal strength. Then, the coarse symbol timing estimation performs autocorrelations to find the beginning of a short symbol. Once the correlation result crosses a specified threshold, the corresponding sample index is given as the coarse timing estimate. The coarse estimate for the frequency offset in the received signal is then computed and the frequency correction is applied to the incoming signal. At the same time, the fine timing estimation computation is triggered. Generally, this also involves sliding window autocorrelation over the received signal, this time for searching for the location (sample index) for the long symbols. This fine symbol timing estimate is then used in conjunction with the CORDIC block to refine the frequency offset estimate. Finally, after applying the frequency offset correction, the Fast-Fourier Transform (FFT) operation is performed and channel estimation and other signal processing can be done.

We now take a closer look at the autocorrelation metric used for coarse symbol timing synchronization. The estimation involves computing a sliding window autocorrelation over the received signal. Mathematically, the normalized autocorrelation function, $M(d)$, is written as [13]:

$$M(d) = |P(d)|^2 / R(d)^2 \quad (2.2)$$

where

$$P(d) = \sum_{m=0}^{L-1} r^*(d+m)r(d+m+L) \quad (2.3)$$

and

$$R(d) = \sum_{m=0}^{L-1} |r(d + m + L)|^2 \quad (2.4)$$

In the equations above, $r(n)$ are the received signal samples, and $R(d)$ is the energy of the received signal currently being used as input to the correlation block. Also, L denotes the length of the sequence: $L = 16$ for the short sequences. Figure 2.6 shows the short symbol repetition under zero-noise conditions. Here, the length-16 short sequence is repeated 10 times, the sequence beginning at index=31 and ending at index=190. In order to illustrate the shape of the metric function, a timing offset of 30 samples is introduced and data packets are appended at the end of the short symbol repetition.

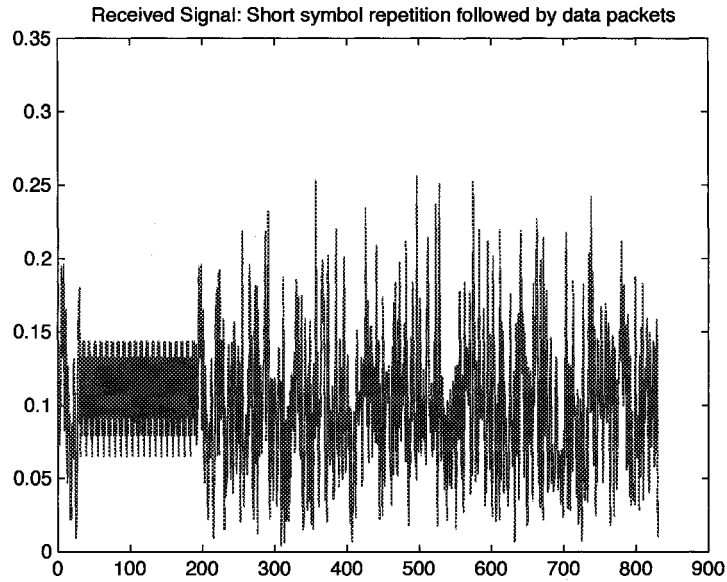


Figure 2.6 : Received signal generated based on short symbol repetition

The metric $M(d)$ for this signal is shown in Figure 2.7. The metric reaches its maximum value of 1 at index=31, and then stays at this value until index=159. At

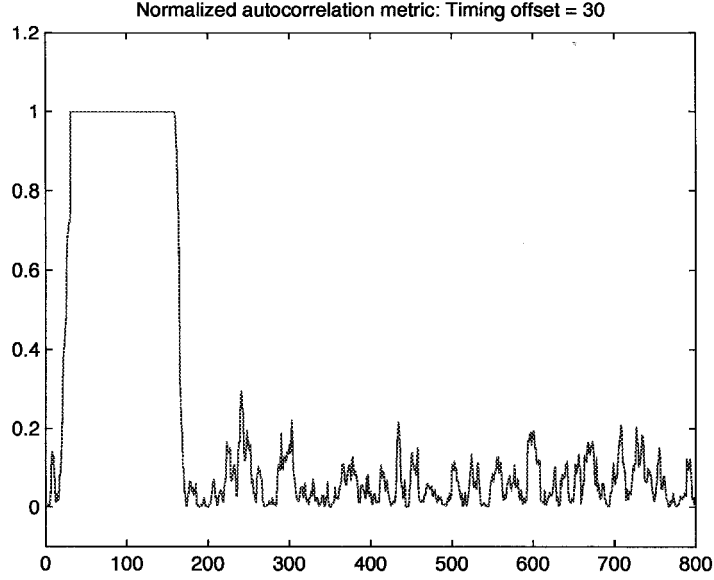


Figure 2.7 : Normalized autocorrelation metric with timing offset of 30 samples

these index values, the signals in the two different correlation windows (equation 3.4) match exactly. Beyond this index, the match starts decreasing and consequently the correlation decreases until it becomes very small. Therefore, many practical systems give the first crossing-point when the signal is greater than a set threshold as the symbol timing estimate. The plateau-like shape of the metric adds robustness to channel effects such as multi-path. However, it also results in a high estimator variance. The estimator variance can be reduced by modifying the training sequence or metric definition such that there are clear peaks at the beginning of the training symbols. In [14] [15] for example, the authors propose a modified-preamble sequence to reduce the estimator variance. We consider these modifications in more detail in Chapter 3.

We will now proceed to address the problem of preamble-based symbol timing

estimation in more detail. While our focus is on coarse estimation using the short sequence, the similarity between the coarse and fine estimation algorithms implies that the approach may be applicable for fine-estimation as well. A performance improvement gained by improving the coarse symbol timing estimation enables better overall system performance. The frequency offset estimation, for instance, relies on the timing estimate for its accuracy and therefore will directly benefit from improved symbol timing estimates.

Chapter 3

Symbol Timing Estimation: Algorithm design and performance analysis

In this chapter, we focus on the symbol timing synchronization algorithms in more detail. The goal of symbol timing synchronization is to find the beginning of the Fast Fourier Transform (FFT) window for wireless OFDM receivers. For OFDM systems, signal processing such as channel estimation, decoding and other tasks are typically done in the frequency domain, i.e. post-FFT. Therefore, the FFT results and consequently the samples chosen for the FFT input affect the performance of the signal processing algorithms. Generally, these algorithms are designed and evaluated assuming perfect synchronization at the wireless receiver. In practical systems, though, achieving timing and frequency synchronization at the wireless receiver is a challenging problem [6] [8] [10].

We propose a preamble-based low-complexity timing synchronization algorithm and present performance results in the context of coarse symbol timing synchronization. We also present an application of a well-known sequence to symbol timing synchronization. The simulation results show that the new sequence can result in a significant performance improvement over the IEEE 802.11a short symbol, especially for multi-path channels.

3.1 Related Work

In recent years, the problem of symbol timing synchronization for OFDM systems has received a lot of attention. Among the many approaches taken to solve this problem, the preamble-based timing synchronization methods are most applicable to burst-mode communication systems such as WLANs, primarily due to the long acquisition time needed by other non preamble-aided methods in generating an accurate estimate.

A timing estimation method proposed by Schmidl and Cox [13] estimates symbol timing offset using received signal correlation based on the preamble signal. Minn, Bhargava et. al. [14] and Park et.al. [15] have proposed modified preamble schemes that try to reduce the variance of this estimator. The majority of the methods proposed in the literature are based on autocorrelation of the received signal, including [16] [7] [13]. The correlation is computed either to detect the cyclic prefix, or the periodicity of the preamble sequence. By exploiting the apriori knowledge of the preamble sequence, the methods try to maximize the similarity between two sliding windows [13] [14] [15]. In [17], the authors present a different approach for cyclic-prefix based timing estimation where they minimize the anti-correlation or dissimilarity between the sliding windows. Also known as average magnitude difference function (AMDF), this method has been studied for pitch detection of voice signals [18] [19]. However, cyclic-prefix based timing estimation methods require averaging over multiple OFDM symbols in order to generate an accurate estimate. In contrast, preamble-based estimation does not require averaging over multiple OFDM symbols, and consequently meet the stringent acquisition time requirements for burst-mode wireless OFDM transmission systems.

There has been other work in the literature that try to improve the accuracy of symbol timing estimates. However, due to the increased computation complexity of



these methods, for example, [20] [21], they may not have been pursued for hardware implementation. In the context of IEEE 802.11a systems, methods that use cross-correlation over the long preamble sequence in conjunction with autocorrelation over the short symbols have also been proposed. However, in addition to the increased computational complexity, the longer latency of these methods is a drawback. To the best of our knowledge, the Schmidl-Cox metric for symbol timing synchronization [13] (with slight variations in the way normalization is done or maximum is chosen) is the most widely used method in hardware today.

We propose a simple method that lowers the computational complexity while striving to maintain similar performance. Indeed, performance results show that the performance matches that of the modified Schmidl-Cox method [14] for AWGN channels. For a review of the results presented in this chapter, we refer the reader to [22].

3.2 Proposed symbol timing estimation method

In contrast to the autocorrelation-based method presented in Chapter 2, we consider a new metric for preamble-based synchronization. Given a received signal, $r(d)$, the receiver detects the repetition pattern by computing the following metric:

$$M(d) = \sum_{m=0}^{L-1} |r(d+m) - r(d+m+L)| \quad (3.1)$$

This has been referred to as the average magnitude difference function [17]. The motivation for this metric can be understood as follows. $M(d)$ is a measure of the difference between the received signal in the two windows. In a perfect/noiseless scenario, the above metric will be zero when the received signal samples in the two windows match up exactly. This is the case when the received signal is comprised of

a repeated pattern with periodicity L . On the other hand, when the received signal is comprised of any other component, the difference will be non-zero and will indicate the level of disagreement. Just as autocorrelation is a measure of similarity between two signal windows, the metric above, average magnitude difference function is a measure of dissimilarity.

Figure 3.1 shows the shape for the metric for the received signal shown earlier in Figure 2.6. For comparison, the normalized correlation metric for the same signal is shown in Figure 3.2.

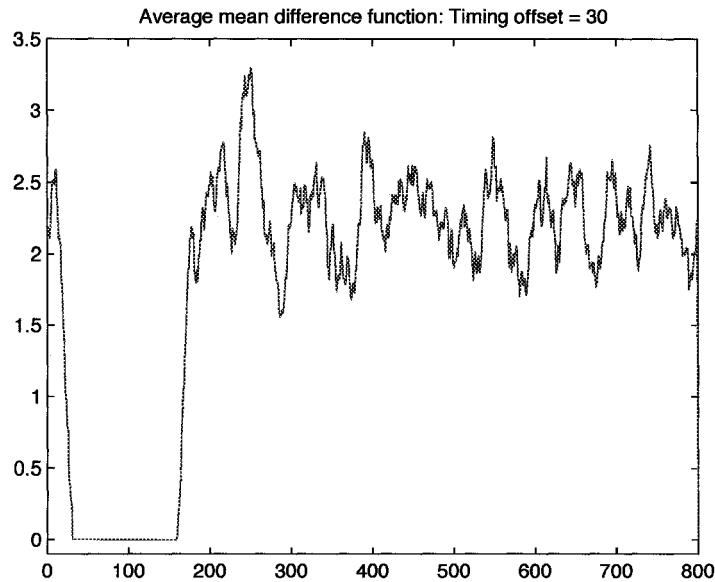


Figure 3.1 : Average magnitude difference function with a timing offset of 30 samples

We also note that the use of the difference structure in equation 3.1 is not restricted to autocorrelation. For example, schemes that use cross-correlation methods could also benefit from the lower complexity of this computation.

We introduce a variation of the above metric as follows:

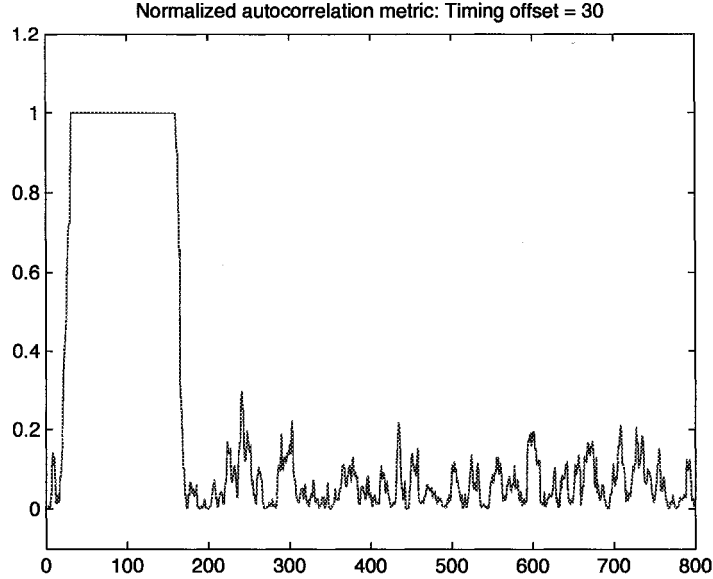


Figure 3.2 : Normalized correlation metric with a timing offset of 30 samples

$$M(d) = \sum_{m=0}^{L-1} |r(d+m) - r(d+m+L)|^2 \quad (3.2)$$

This eliminates the square root operation and is motivated by simpler hardware implementation. We will evaluate the effect of this on the performance of the symbol timing synchronization method.

3.2.1 Complexity Analysis

The main advantage of this method is its significantly reduced computational complexity. We consider again the normalized autocorrelation metric defined by Schmid-Cox [13]; the metric that most implementations of timing synchronization blocks are based on [6] [8] [23]:

$$M_1(d) = |P(d)^2|/R(d)^2 \quad (3.3)$$

where

$$P(d) = \sum_{m=0}^{L-1} r^*(d+m)r(d+m+L) \quad (3.4)$$

and

$$R(d) = \sum_{m=0}^{L-1} |r(d+m+L)|^2 \quad (3.5)$$

It has been shown that the performance of this method can be improved tremendously by replacing $R(d)$ above with $R_2(d)$ given by [14]:

$$R_2(d) = \sum_{m=0}^{2L-1} |r(d+m)|^2 \quad (3.6)$$

We then define the new metric as:

$$M_2(d) = |P(d)^2|/R_2(d)^2 \quad (3.7)$$

For reference, we also look at the autocorrelation metric with no normalization, i.e.

$$M_3(d) = |P(d)^2| \quad (3.8)$$

To summarize, the difference between the three metrics is simply that M_1 uses L samples, M_2 uses $2L$ samples, and M_3 uses zero samples to compute the signal energy during the normalization step. Note that they all still need $2L$ samples for computing $P(d)$, the autocorrelation function.

Table 3.1 shows the computational complexity for the three different metrics.

The method proposed in [13] is shown in the top row of Table 3.1. Many implementations, for example [24], use the method shown in the second row (M_2). Our simulation results confirm findings of [14] that the performance is much better when using the second metric M_2 . From Table 3.1, we note that the AMDF method has much lower computational complexity than M_2 . We will see later that the performance of the AMDF method is similar to that of the modified Schmidl-Cox method proposed in [14].

With many future generation wireless systems leaning towards more advanced signal processing and error-correction methods such as LDPC [25] [26] [27], the limitations posed by the size constraints of the hardware become increasingly evident[28]. When using FPGA platforms for prototyping a complete system, the constraints can be the size of the FPGA and the number of multipliers as well. An advantage of AMDF is the significant reduction in the number of multiplies required.

3.2.2 Simulation Results

In this section, we present a performance analysis for the proposed method. All simulations were done in MATLAB with parameters similar to IEEE 802.11a. The

Table 3.1 : Complexity analysis

Method	Samples Needed	Real Additions	Real Multiplications
Schmidl-Cox (M_1)	$2*L$	$4*L$	$6*(L-1)$
M_2	$2*L$	$6*L$	$8*L$
M_3	$2*L$	$4*L$	$4*L$
AMDF	$2*L$	$4*L$	$2*L$

parameters of the data packets, such as modulation scheme chosen (for eg: BPSK, QPSK, 4-QAM), number of pilot tones, etc. does not impact the preamble-based timing estimation block because it operates on the preamble signal. However, we note that some of these parameters, for example modulation scheme, can affect the frequency estimation and correction blocks (errors caused by a given offset) as these affect the frequency-domain signal decoding.

It is a well-known fact that the channel profile has a direct impact on the performance of the receiver. The symbol timing synchronization scheme needs to be robust to channel effects and provide the best performance for that channel environment. A multipath channel presents a greater challenge in detecting the correct timing point. Degradation in performance may be expected due to inter-symbol interference (ISI).

Figure 3.3 compares the performance of AMDF with the modified AMDF method (Equation 3.2) in terms of the probability that the symbol timing estimate matches the actual offset exactly, under AWGN channel conditions. The key difference is the elimination of the square-root operation for the second method. The results are averaged over 500 Monte-Carlo simulations. The performance of both schemes is almost identical. From here on, we use AMDF to refer to the modified AMDF equation 3.2. Another thing to note in Figure 3.3 is that the performance is very poor for SNR 0-8dB or so and then improves steadily as we might expect.

Figure 3.4 compares the four metrics M_1 , M_2 , M_3 and AMDF in terms of their probability of correct detection under AWGN channels. This is the likelihood that the timing estimate matches the correct timing offset exactly. ACFN16 (M_1) and ACFN32 (M_2) represent that 16 and 32 samples respectively have been used for computing signal energy. We observe that M_2 and AMDF perform comparably, while the performance of M_1 and M_3 is much worse. As we saw in Table 3.1, M_3 requires



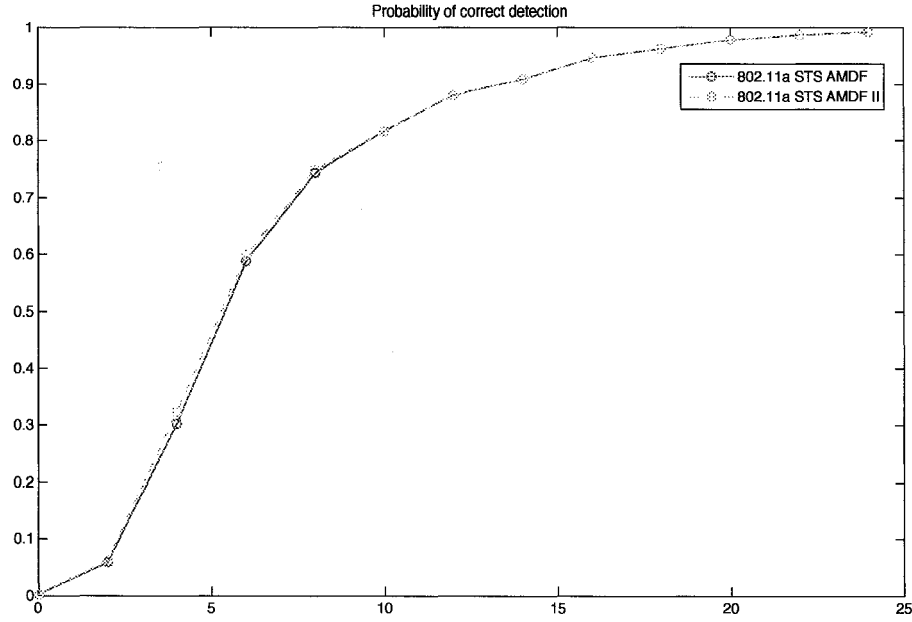


Figure 3.3 : Probability of correct estimation: Comparison for AMDF and modified average magnitude difference function (without square-root operation)

much less computations as compared to M_2 .

The mean estimation error for the same simulation is shown in Figure 3.5. Again, we can confirm that the performance of M_2 and AMDF is superior to the other methods.

We now proceed to evaluate the performance of AMDF relative to M_3 . Recall, M_3 is based on the method proposed by [13]. From here on, we use the term autocorrelation metric to refer. Figure 3.6 shows the probability of correct detection under AWGN channel conditions.

The corresponding MSE is shown in Figure 3.7. Again, we observe the performance of AMDF matches that of the M_2 closely.

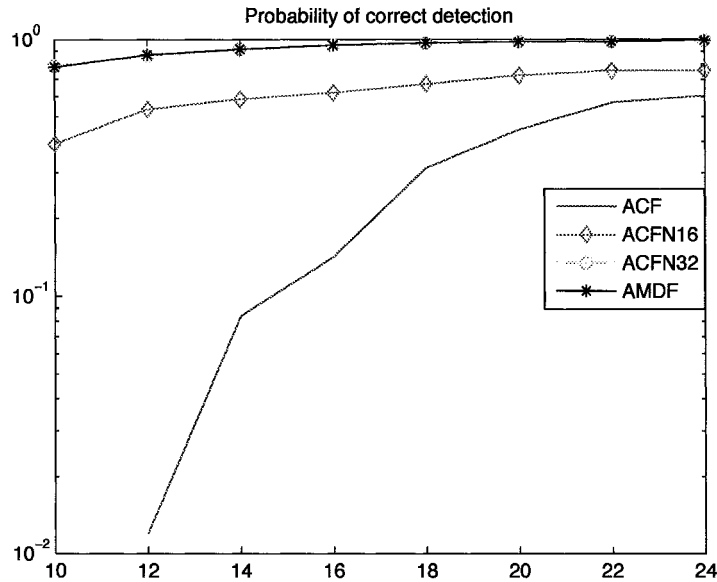


Figure 3.4 : Probability of correct estimation: Comparison between AMDF and Schmidl-Cox algorithm with different window size for computing signal energy

3.3 Preamble sequence

Burst-mode OFDM systems utilize the preamble signal to achieve rapid signal acquisition. This is also known as the training signal. The structure of the preamble chosen is a key design aspect of the wireless communication system. The performance results discussed so far assume the short training symbols of IEEE 802.11a. In this section, we introduce a preamble sequence that maintains the same energy as the IEEE 802.11a short training symbol but results in much better performance for multi-path channels.

The acquisition time needs to be short for WLAN systems. The receiver, when listening on a packet, needs to detect the presence of the packet and start processing it within a short amount of time. This time is dictated by the medium-access-control (MAC) protocol, which might require the receiver to acknowledge reception within

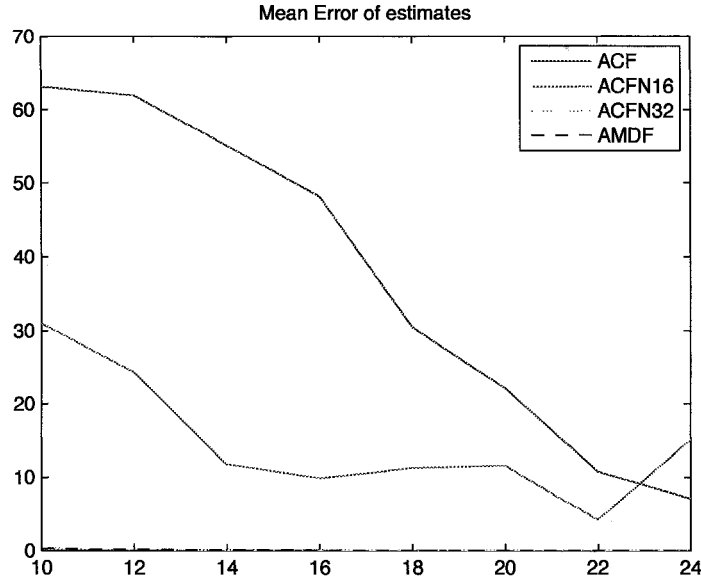


Figure 3.5 : Mean estimation error estimation: Comparison between AMDF and Schmidl-Cox with different window size for computing signal energy

a window of time to avoid retransmission. This time will also be dependent on the complexity of the rest of the receiver signal processing.

The acquisition time has to be kept in mind when designing the symbol timing synchronization architecture. For systems like IEEE 802.11a, there is a repetition of the short symbol to generate the complete preamble. At the receiver, there is a trade-off between the acquisition time and the accuracy of timing estimation. Specifically, a larger correlation window will translate into a better estimate at the cost of increased time required for estimation.

For hardware implementation, the number and latency of operations that need to be performed have a tremendous impact on the system. In addition to the number of gates and power consumption, the number and latency of operations affects the acquisition time and the ability of the receiver to successfully detect a packet.

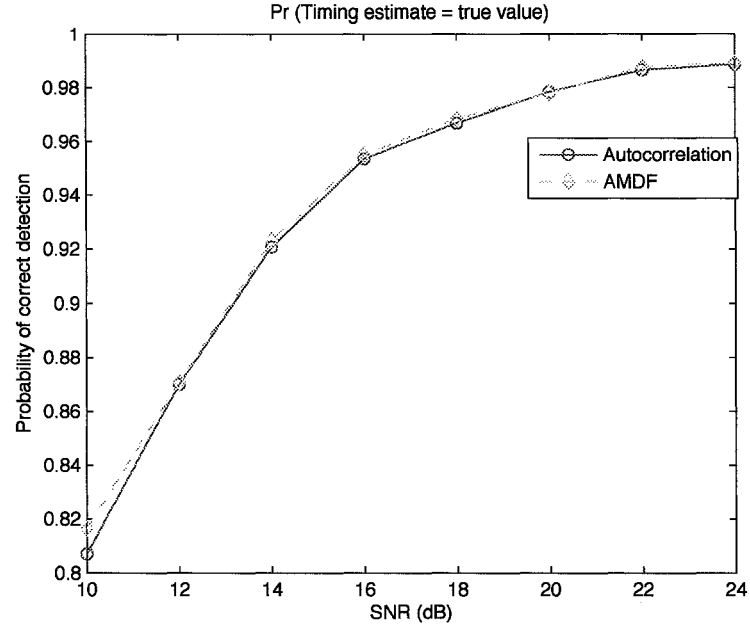


Figure 3.6 : Probability of correct detection: Comparison between Autocorrelation and AMDF methods

A receiver architecture based on computations over the received signal has the advantage that the number of operations required can be reduced by re-using results from previous computation. However, a cross-correlation based scheme, where the received signal is correlated (or subtracted, in the case of AMDF) with a template sequence does not leave room for any reuse of previous results.

3.3.1 Golomb Sequence

We now consider a sequence with low periodic autocorrelation generated using the following equations [29]. Let L denote the length of the sequence and define

$$\alpha = \exp(2\pi i L) \quad (3.9)$$

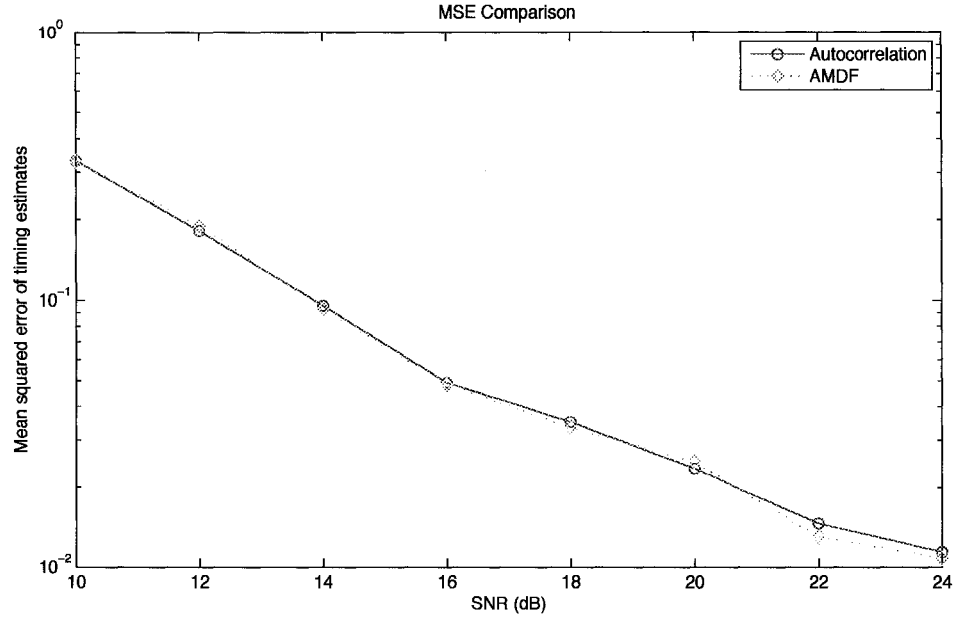


Figure 3.7 : MSE comparison between Autocorrelation and AMDF methods

Then, the Golomb sequence consists of $(\alpha_k)_{k=1}^L$ complex numbers that are chosen using the following equation:

$$\alpha_k = \alpha^{(k-1)k/2} \quad (3.10)$$

This sequence is known as Golomb sequence. This is a polyphase sequence exhibiting a constant envelope in the time-domain. We have chosen the length to be 16 samples, the same as the short training sequence length [2].

Figure 3.8 shows the sample autocorrelation of the IEEE 802.11a short training symbol (STS) and length-16 Golomb sequence for different lags.

For the ideal sequence, the sample autocorrelation should be unity for a lag of zero, and zero for all other lags. We note that the mean value of the sample autocorrelation

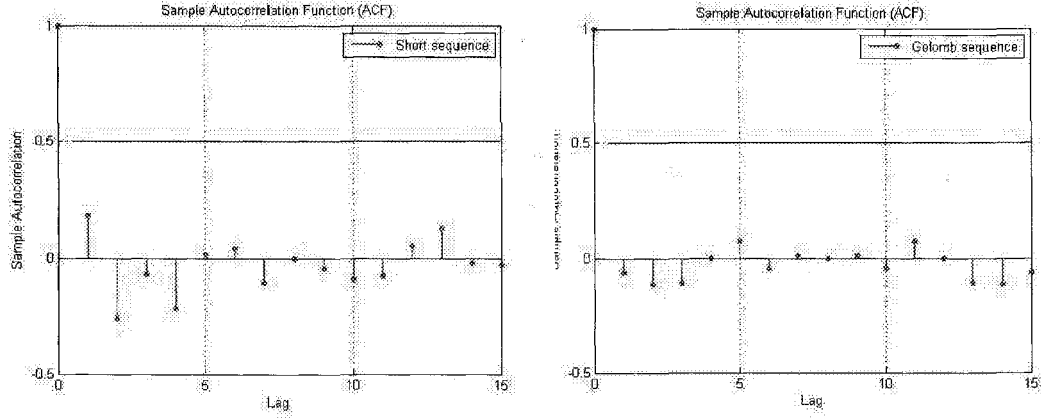


Figure 3.8 : Sample autocorrelation function of IEEE 802.11a STS (left) and Golomb sequence based training signal (right)

function for all non-zeros lags is much smaller for the Golomb sequence as compared with the IEEE 802.11a STS. A commonly used metric for comparing auto-correlation properties of sequences is the inverse of the sum of sidelobe energy, mathematically denoted as:

$$X = 1 / \sum_{lag=0}^{L-1} ACF(m) \quad (3.11)$$

Here ACF represents the sample autocorrelation function. A larger value of X is desirable because it denotes a small mean-value for the sample autocorrelation function at non-zero lags. The value of X for IEEE 802.11a STS and Golomb sequence is 0.7457 and 1.1840 respectively. Hence, Golomb sequence does indeed exhibit better autocorrelation properties and it may be expected to be more robust to noise relative to the short training sequence. For completeness, Figure 3.9 shows the frequency-domain sample autocorrelation function.

In this case for the value of the metric X, the STS and Golomb sequence is 0.7801 and 1.5300 respectively. Again, Golomb sequence exhibits better autocorrelation



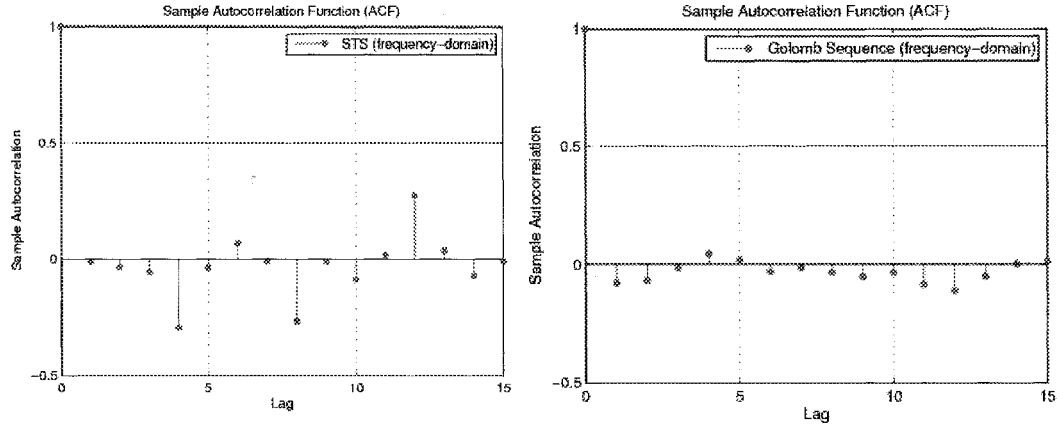


Figure 3.9 : Frequency-domain sample autocorrelation function of IEEE 802.11a STS (left) and Golomb sequence based training signal (right)

properties in terms of the above metric. We note that since the timing synchronization typically operates before the frequency offset estimation, it is highly unlikely that the correlation will be done in the frequency-domain (if we take an FFT before correcting for the timing offset, the resulting data will suffer from ICI due to loss of orthogonality of sub-carriers).

We now study the performance of the two sequences under various channel environments. For fair comparison, we have normalized the energy of the Golomb sequence to be the same as the short training sequence.

Figure 3.10 shows the time-domain domain representation of the two training signals. Observe that the short preamble based on Golomb sequence has a constant amplitude, thus alleviating the problem of high peak-to-average power ratio.

Figure 3.11 shows the frequency-domain representation of the two training signals. Observe that with the exception of three frequency indices - 29, 33, 37 - in Golomb sequence, the frequency indices where the IEEE 802.11a short training signal (STS) and the Golomb sequence are zero match. Thus, both signals have the same guard

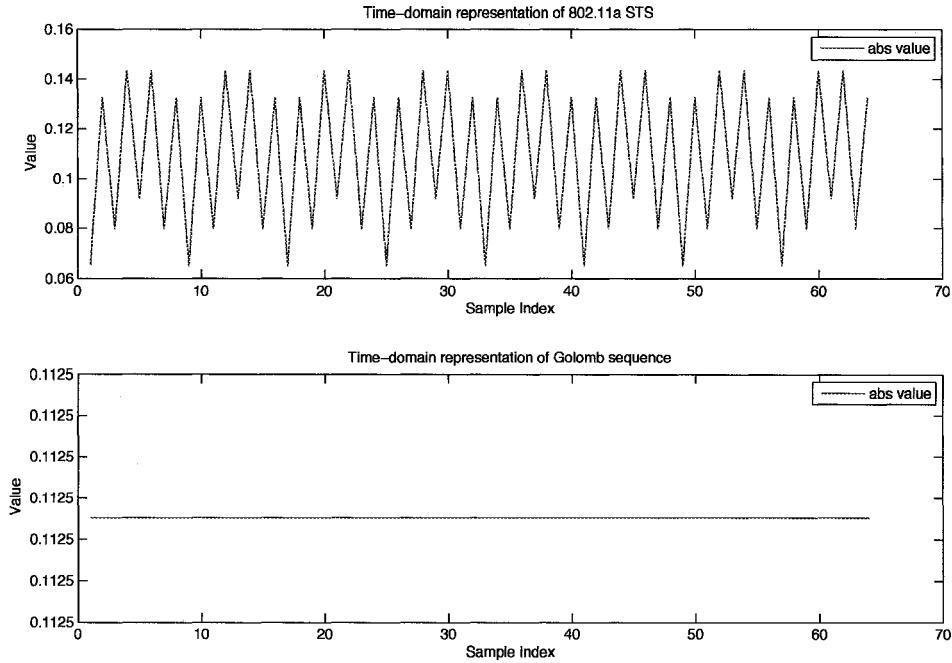


Figure 3.10 : Time-domain representation of IEEE 802.11a STS and Golomb sequence based training signal

bands at either end of the frequency spectrum.

We believe that a further exploration of preamble-design space is very desirable, especially when multiple-antenna systems come. This is similar to the exploration and design done in [30], where the authors devise an optimization criteria and method for designing a preamble sequence taking into account factors such as AGC, guard sub-carriers, multi-path propagation for OFDM-based power line communication systems. In our work, we show the performance benefits of using a new length-16 sequence that has the same energy as the short training sequence and shares many of its characteristics. At the same time, the new preamble signal (in baseband) has a peak to average power ratio of unity.

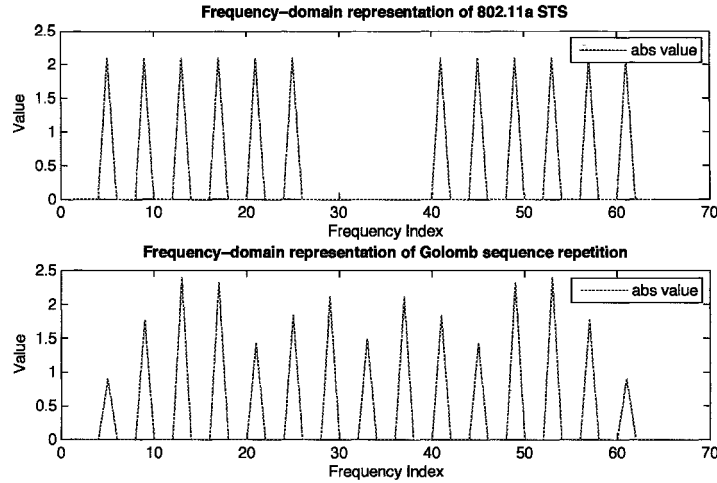


Figure 3.11 : Frequency-domain representation of IEEE 802.11a STS and Golomb sequence based training signal

3.3.2 Performance Results

Here, we show present simulation results comparing the performance of symbol timing synchronization using short training sequence and Golomb sequence. The results are based on Monte-Carlo simulations with various channel environments.

Figure 3.12 shows the probability of correct detection for the AWGN channel. We see that the Golomb sequence based training signal performs better than the 802.11a short training signal with the gap being more prominent for lower SNRs. The corresponding MSE of the timing estimator is shown Figure 3.13.

Finally, we consider the performance of symbol timing estimator under multi-path channel. The multipath channel model used is based on IEEE 802.11n and represents a typical residential environment (LOS conditions) with a rms delay spread of 15ns and a 10dB Ricean K-factor at the first delay. The probability of correct detection for multi-path channel is shown in Figure 3.14. Observe the general degradation in performance compared to AWGN. At the same time, the benefits of the better

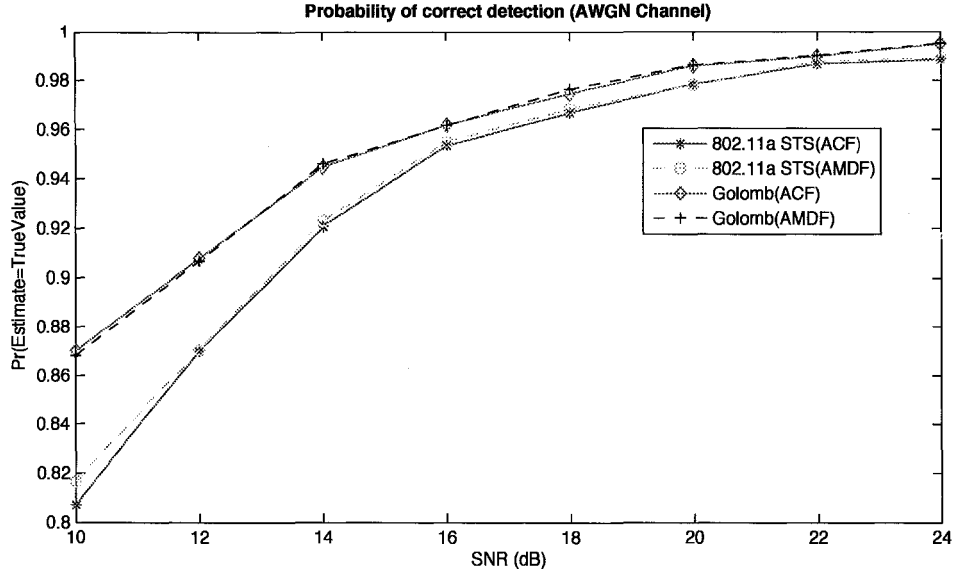


Figure 3.12 : Probability of correct detection for AWGN channel

correlation properties of Golomb sequence are more visible. The preamble based on Golomb sequence gives significantly better performance than the short sequence.

We show the same figure with a closer focus on higher SNRs in Figure 3.15. The probability of correct detection (estimated value = correct value) starts near 0.6 rather than near 0.8 as in the case of AWGN channels. Also, the trend from lower SNR to higher SNR is similar for each metric, but the gap between Golomb sequence based training signal and 802.11a STS is very significant even at high SNRs. Finally, we note that the AMDF method performance follows the general trend of the higher-complexity autocorrelation method.

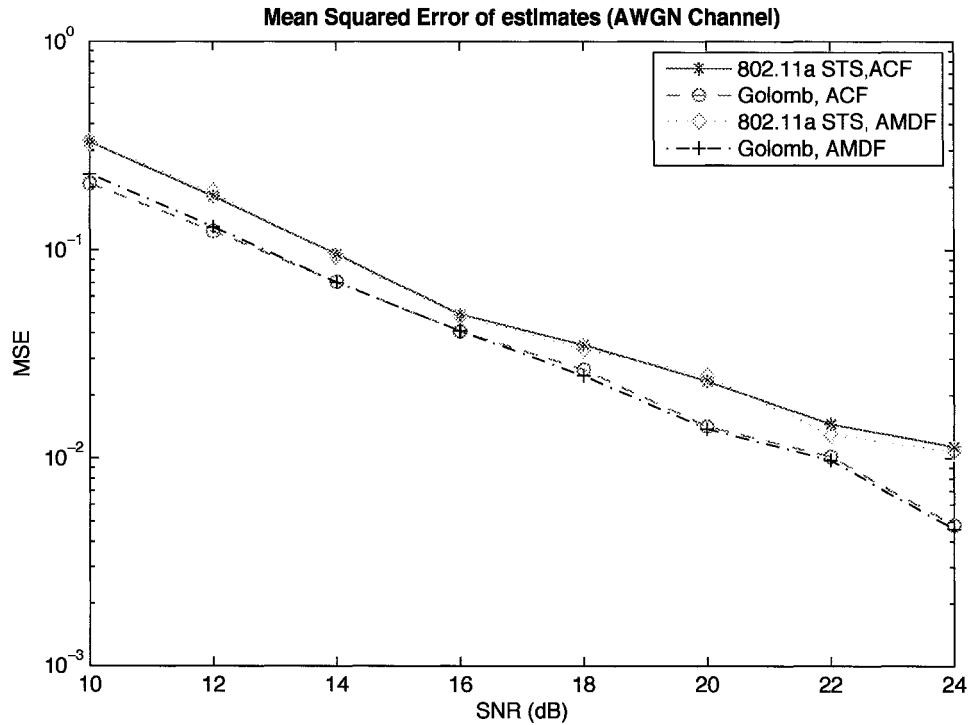


Figure 3.13 : Estimator MSE for AWGN channel

3.4 Conclusions

We have proposed a preamble-based low-complexity timing synchronization algorithm whose performance matches closely the widely-used autocorrelation-based metric based on [13]. We also present an application of a well-known sequence to symbol timing synchronization. The simulation results show that the new sequence, used to build the short training symbol, can result in a significant performance improvement over the IEEE 802.11a short symbol, especially for multi-path channels.

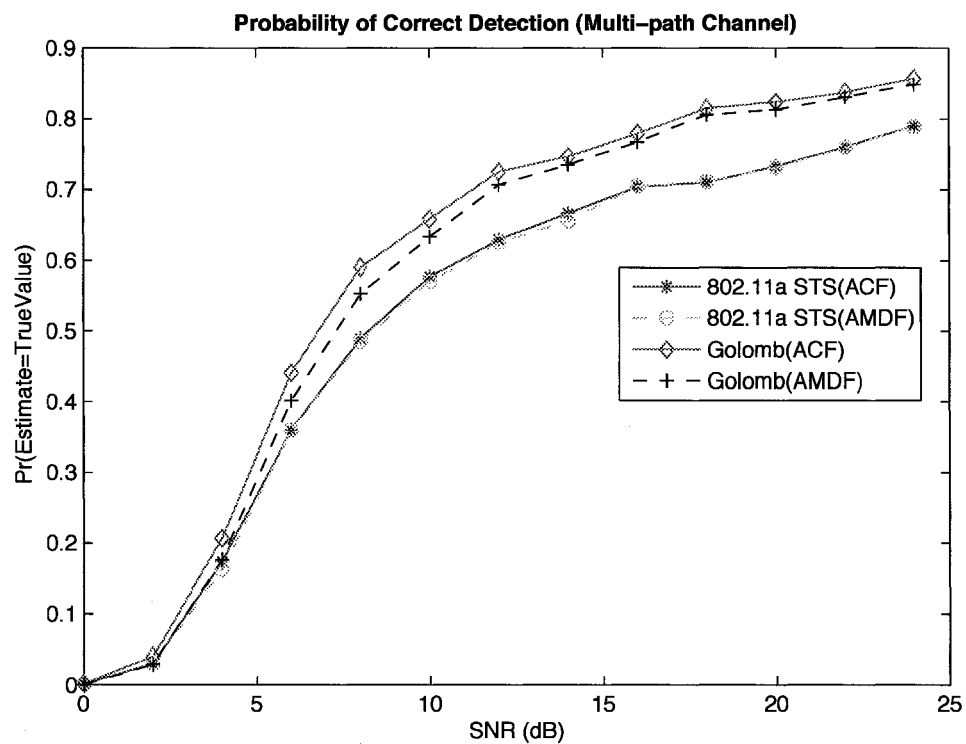


Figure 3.14 : Probability of correct detection for multi-path channel

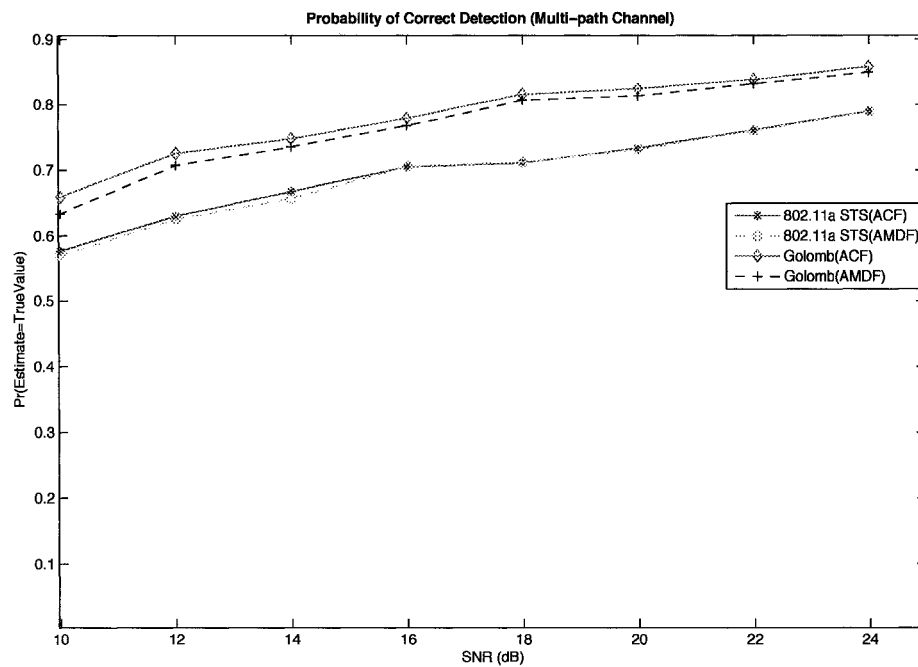


Figure 3.15 : Probability of correct detection for multi-path channel: performance gains for higher SNRs

Chapter 4

System Hardware and Design Environment

In this chapter we discuss the NI-5640R FPGA-based system hardware and the National Instruments LabVIEW design environment that was used to prototype the symbol timing synchronization block in hardware.

Wireless system designers use hardware prototyping extensively to achieve a successful system design. Using reconfigurable devices such as Field Programmable Gate arrays (FPGAs), wireless communications blocks are implemented in hardware and connected to an analog interface to provide a complete working prototype. The prototype helps understand the hardware trade-offs involved and exhibits the feasibility of a hardware implementation and the correctness of the communication block. The NI-5640R hardware and LabVIEW software allow us to achieve this purpose. An understanding of this hardware and associated software will assist the reader in following the details of the hardware implementation described in the next chapter.

The NI-5640R hardware platform used in this project provides computational resources, supporting components and the ability to interface with analog intermediate-frequency(IF) signals. The LabVIEW software together with other modules allows us to target the hardware.

4.1 NI-5640R IF Transceiver Module

The NI-5640R hardware is a PCI card from National Instruments Inc. that plugs into the PC and provides connectors for interfacing with analog signals. Briefly, the card has a Xilinx Virtex-II Pro FPGA, ADCs, DACs, voltage-controlled crystal oscillator, and a timebase (clock configuration chip). The board takes power from the PCI bus and has voltage regulators to provide power to the FPGA and other on-board components. The documentation provided by the manufacturer can be referred to for more details.

For our project, we were interested in a hardware architecture built around a field programmable gate array (FPGA). Field programmable gate arrays are now being widely used for rapid prototyping as well as for implementation of parallel systems. The FPGA is an inherently parallel computation engine that is designed as an array of logic blocks, along with memory blocks as well as interconnects for moving data. The FPGA package can provide sufficient Input/Output pins to allow rapid moving of data between the computation blocks inside the FPGA and the rest of the system. This parallelism in computation and I/O resources are similar to the parallelism available in an application specific integrated circuit (ASIC). However, the programmable nature of the FPGA allows systems to be designed and tested at a much lower cost, by avoiding costs associated with fabrication and testing of silicon. Also, the platform architecture and the hardware/software design tools make it possible to test different algorithms without making hardware connectivity changes or other modifications in the overall system.

For prototyping real-world wireless communication systems successfully, the ability to get real-world signals into the system is an important consideration. The analog interface with the integrated analog-to-digital converters (ADC), and digital-

to-analog (DAC) converters in the NI-5640R card provide this capability. Wireless communication systems, available commercially, generally operate in the radio frequency (RF) range. The signal processing algorithms are implemented in baseband, meaning raw data samples without the presence of any carrier signal. The data generated for transmission must be modulated onto a RF carrier signal before transmission can occur. The NI-5640R card has digital upconversion and digital downconversion capabilities to achieve modulation and demodulation with an intermediate frequency (IF) range carrier signal. The IF range is not suitable for direct transmission; an intermediate frequency (IF) to radio frequency (RF) upconversion must be done by external hardware before transmission. As mentioned previously, RF signal output is what a typical commercially available communication system employs for transmission. However, for prototyping purposes, an IF transceiver can be sufficient for studying the implementation in a practical environment.

For our wireless system, we interface with IF signals. Since there isn't an IF-to-RF solution available for the NI-5640R card, over-the-air operation is not feasible. Therefore, we use a loopback cable connecting the analog input and output of the NI-5640R card. The implementation details are discussed in Chapter 5.

4.1.1 Hardware Components

Figure 4.1 shows the block diagram of the NI-5640R FPGA based IF transceiver. The Xilinx Virtex-II Pro FPGA is the main computing engine. The interface for analog I/O is provided by four SMA (SubMiniature version A) connectors. The analog-to-digital conversion (ADC) and digital downconversion (DDC) are done using the AD6654 integrated ADC/DDC receiver chip. Following this process, the analog IF input will be available to the FPGA user as baseband I/Q samples. The digital up-

conversion (DUC) and digital-to-analog conversion (DAC) are done with the AD9857 chip. There are two ADC/DDC chips (AD6654) and DDC/DAC chips (AD9857) respectively thereby providing the ability for two parallel input and output channels. Both the ADC and DAC devices provide an interface with 16-bit precision. While the ADC provides the I and Q data streams in parallel, the DAC interleaves the I and Q input data streams. From an operational standpoint, the DAC clock frequency should be twice as fast as the ADC clock frequency to meet the achieve data-rate. There are low-pass filters on-board the NI-5640R for the analog data-streams.

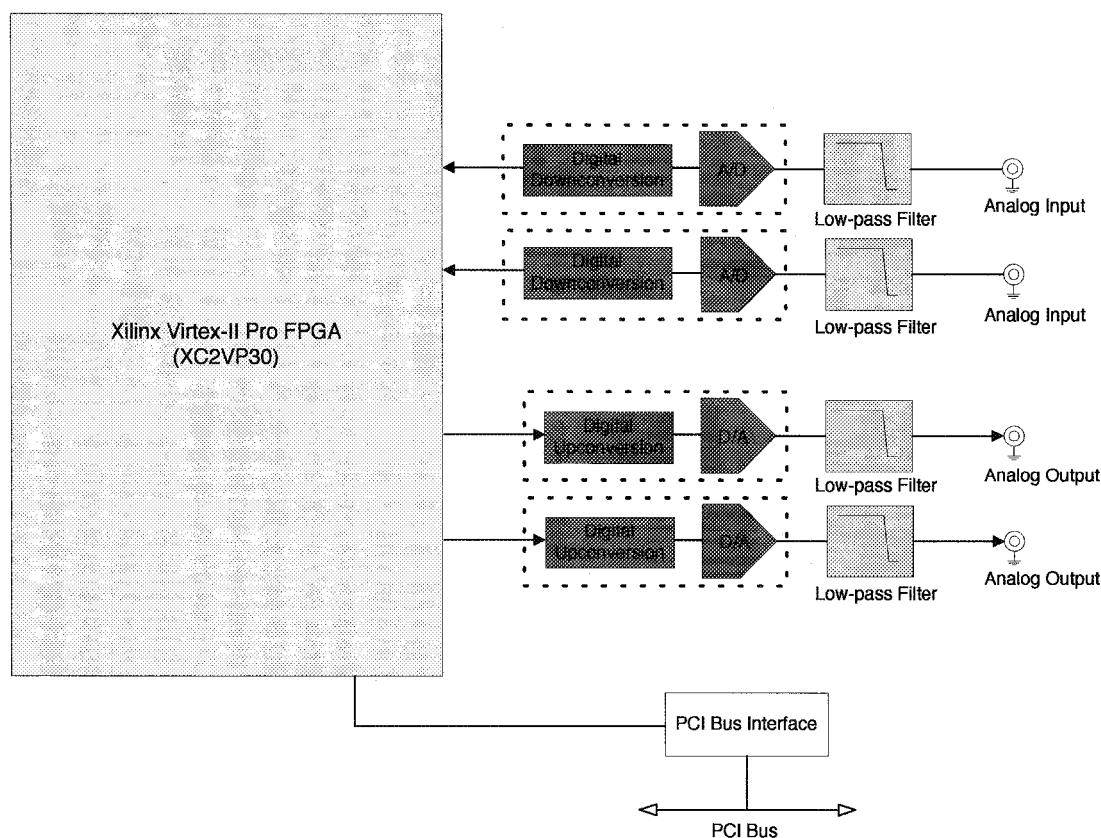


Figure 4.1 : Block diagram of NI-5640R (adapted from [1])

4.1.2 ADC, DAC, and Host Interface

The ADC operation is described below. The incoming IF signal, after digitization, undergoes downconversion which performs frequency translation, band-selective filtering, and sample rate conversion. The signal which is centered around carrier frequency (F_c) is brought down to baseband. For our system, we are interested in a signal bandwidth of 20MHz. Therefore a sampling rate higher than 20MHz is desirable to allow for decimation filter transition band effects. The DDC decimation filter co-efficients should be chosen to avoid aliasing effects and to provide proper scaling to avoid numerical overflows. The sampling frequency (F_s) should be chosen such that there is no signal content at DC, $F_s/2$ and multiples of this frequency. We select a sampling frequency of 100MSamples/sec.

Also, the frequency-range supported by the ADC is from 250kHz to 80MHz. The lower frequency limit is due to AC-coupling, while the higher limit is where the transition region of the onboard analog low-pass filter begins.

The DAC takes in interleaved I and Q data and performs sample-rate conversion, interpolation, and analog conversion. It provides a 16-bit interface at the input. Based on the desired IQ rate, the interpolation factor is calculated and the conversion from baseband to IF occurs.

NI-5640R connects to the Host computer using the PCI bus. There is a PCI-bridge chip (NI-STC2) that provides capabilities for direct-memory access (DMA) transfers between the program running on the Host machine and the FPGA target. Presently, NI-5640R driver software supports DMA only from the Host to the FPGA, but not in the reverse direction. The 32-bit PCI bus can provide a maximum theoretical bandwidth of approximately 127 MBytes/sec. Considering complex baseband samples with 16-bit representation for I and Q data, this limits the maximum theoretical



sample rate to approximately 31.75 MSamples/sec.

Figure 4.2 shows the configuration steps for the ADC, DAC clocks as well as the clocks available to the FPGA. We use the on-board 200MHz voltage controlled crystal oscillator (VCXO) for generating the various clocks. The configuration clock is used for all communication between the HOST and FPGA done by LabVIEW, such as front-panel connectors, read/write operations and DMA transfers.

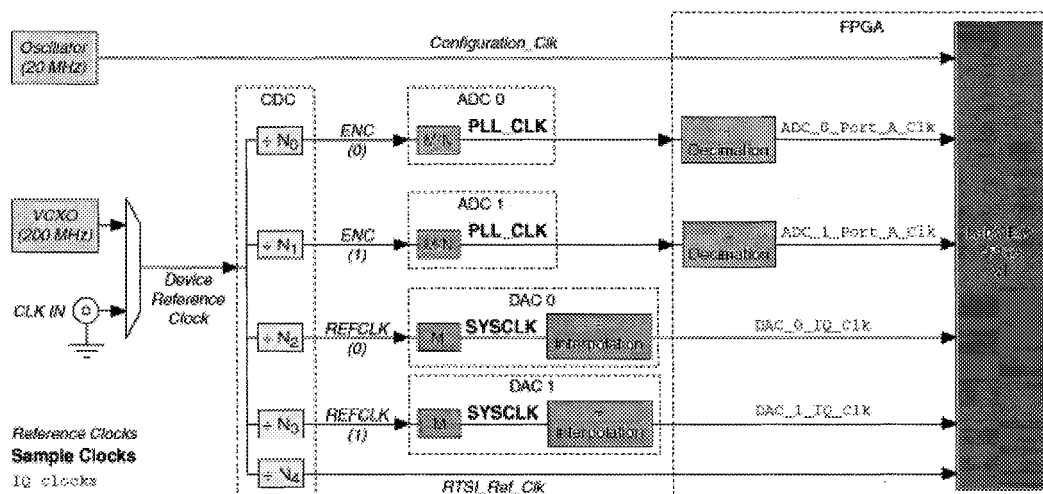


Figure 4.2 : NI-5640R configuring clocks (adapted from [1])

The reference clocks, interpolation/decimation rates and clock dividers need to be set in order to configure the IQ sample rate. The reference clock frequency is set using the graphical user-interface provided by the LabVIEW project. The clock dividers, and interpolation/decimation rates are configured by the user with the help of HOST VIs. The details are discussed in section 4.2

We use the onboard 200MHz free-running Voltage-controlled crystal oscillator (VCXO) as the primary clock source from which other clocks are derived. While the

NI-5640R hardware provides the ability to use an external clock signal as a reference clock source, we do not use this feature for our application. This can be helpful if the system is extended to use multiple FPGA boards.

4.2 LabVIEW Programming Environment

LabVIEW is a graphical programming environment. The program is represented using a block diagram based approach. The program, known as a Virtual Instrument (or VI), is translated into code appropriate for the target platform. A program in LabVIEW is called a Virtual Instrument (VI).

We use LabVIEW 8.0 software and FPGA module to implement the symbol timing synchronization algorithm on the FPGA target. The configuration of the other hardware resources on the NI-5640R card are done using the NI-5640R driver software. The prototyping environment provided by LabVIEW allows us to integrate code running on the HOST computer with the FPGA code and thereby enabling hardware co-simulation.

The code is managed using a LabVIEW project. The LabVIEW code (or VIs) are segregated according to the target. In our project, we have a HOST VI running on the HOST target, and an FPGA VI with supporting sub-VIs that run on the NI-5640R FPGA target. The details of these are discussed below. Also included in the project, is the system clock information. Our system implementation includes an ADC clock, DAC clock, RTSI clock, and a configuration clock. The clock frequency for these are configured using the Project interface.



4.2.1 LabVIEW Host

LabVIEW software, originally targeted for measurement and automation has grown in support and user-base to include simulation of signal processing, math, as well as digital communication systems. The LabVIEW Host VIs have access to toolkits that have pre-designed blocks for implementing the functionality for many common blocks used in these areas. This is good from a stand-point of supporting future extensions to the system in simulation.

The LabVIEW Host VI runs on the PC and the user interacts and controls the program execution using this VI. The front-panel shows the controls that can be used to configure the code on the fly and indicators that depict the results and debugging information. The VI includes a reference to the FPGA VI that needs to be downloaded onto the FPGA target and executed.

This VI contains some configuration code for the NI-5640R hardware and therefore is necessary in order for the hardware to function properly. The FPGA VI cannot be run interactively, rather, the Host VI should open a reference to it. The LabVIEW HOST VI configures the hardware resources on the NI-5640R card based on user input. This includes the analog-to-digital converters (ADC), digital-to-analog converters (DAC), and their respective IQ clock rates (sampling rates). The HOST VI does this by writing the user parameters to hardware registers via the interface provided by the NI-5640R driver software.

A key element of the HOST VI is the state-machine for transferring data back-and-forth between the HOST and the FPGA. We discuss the details in Section 5.2.

4.2.2 LabVIEW FPGA

The FPGA VI contains all the user code that will run on the FPGA as well as code for acquiring and sending data using the analog interface. For the NI-5640R target, the FPGA VI cannot be run in interactive mode, i.e. cannot be run stand-alone without the HOST VI. The driver software and configuration VIs that are used on the HOST target for configuring the various hardware parameters such clock dividers, interpolation/decimation rates, ADC and DAC profiles, call on FPGA code for these tasks. The details of our FPGA implementation are discussed in Section 5.2.

4.2.3 NI-5640R Driver Software

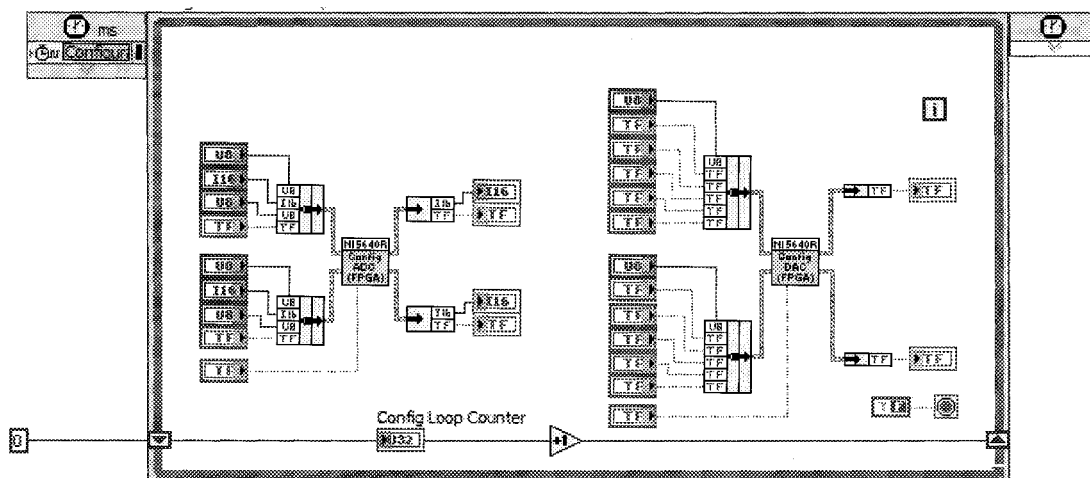


Figure 4.3 : NI-5640R configuration loop

Figure 4.3 shows the configuration code provided by National Instruments with the NI-5640R card. The figure shows a single-cycle timed loop (SCTL) that runs every clock tick of the configuration clock. For the NI-5640R hardware, the configuration clock runs at a fixed frequency of 20MHz. The DAC and ADC have many configurable

parameters that are set using registers on the respective device. When the user makes changes in the hardware system the HOST code relies on this loop to implement them. For example, if the user changes the DAC carrier frequency, the changes are propagated from the HOST VI to the actual hardware registers on the DAC via the configuration loop.

Chapter 5

FPGA Implementation of Symbol Timing Estimation

In this chapter, we present the hardware implementation of the symbol timing synchronization block on the NI-5640R transceiver built around the Xilinx Virtex-II Pro FPGA chip.

5.1 Fixed-Point Wordlength Analysis

While designing hardware for wireless communication systems, there are area, power, and performance budgets that need to be met. This is especially true for portable consumer devices such as laptops where the size and battery life are important considerations. Supporting floating-point operations in hardware requires a lot more area and power relative to fixed-point operations. Consequently, there is a need to translate the wireless algorithms from using floating-point numbers to numbers using fixed-point representation. In fixed-point representation, the decimal place in the bit-representation of the numbers is known at every stage of the computation, thus saving resources required for decoding and shifting numbers when dealing with floating-point representation. During the conversion from floating-point to fixed-point, the goal is to minimize the hardware resources and power needed while striving to maintain the desired level of performance.

LabVIEW FPGA can represent numbers in fixed-width precision. Unlike Application specific integrated circuits (ASICs) where the wordlength can be made arbitrary,

the wordlength for LabVIEW FPGA target needs to be chosen as either 8,16, or 32 bits. The precision required varies based on the application, but for many systems that need high-precision, 24-bits typically seems to be a good trade-off point between performance and hardware resources. For our LabVIEW FPGA implementation, we use 16-bit precision to represent the fixed-point numbers. We found that 8-bits are not sufficient to cover the required range of values, while 32-bits leaves a lot of the precision unused resulting in a lot of unnecessary hardware overhead. Also, as discussed in Chapter 4, the interface to the A/D and D/A in the NI-5640R transceiver provides 16-bit precision, thus making 16-bit wordlength a suitable choice.

As we've seen, the primary motivation for converting from floating-point to fixed-point operations is the significantly higher hardware cost associated with floating-point arithmetic. The trade-off is that the fixed-point operations typically allow for a smaller dynamic range, therefore introducing performance loss.

Figure 5.1 shows the results of the timing metric computation, comparing the fixed-point and floating-point implementations. In this particular case, we compare the results from correlation metric computation performed in MATLAB floating-point arithmetic with the same metric computed in fixed-point LabVIEW code. The simulations are for additive white gaussian noise (AWGN) channel using the short sequence repetition and normalization correlation computation [13].

5.2 System Partitioning

In this section, we give a system-level description of our transceiver. As discussed in Chapter 2, the source generates information bits which are mapped to symbols belonging to the QPSK constellation. This QPSK data is then modulated using the Inverse Fourier Transform (IFFT) operation. Next, the cyclic-prefix is appended to

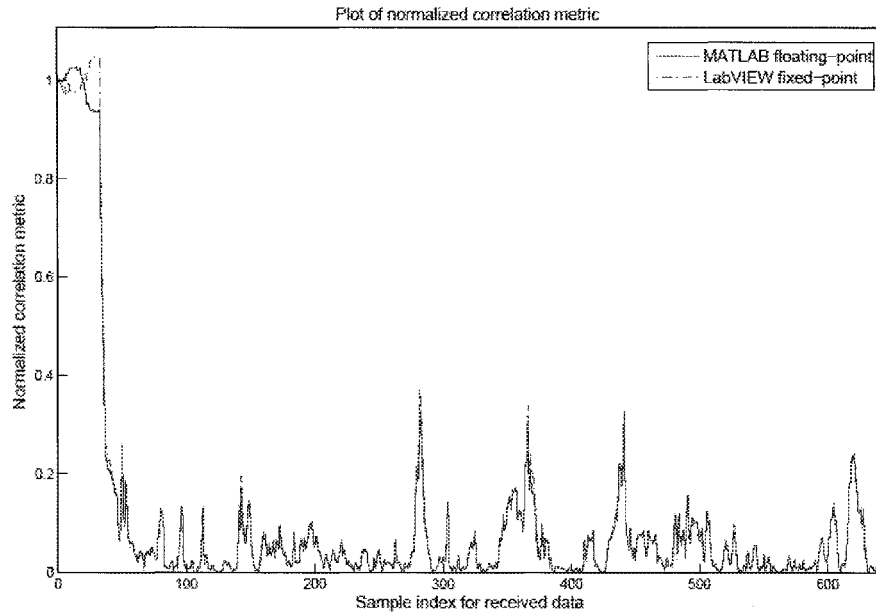


Figure 5.1 : Verification of fixed point results

blocks of IFFT output to form the corresponding OFDM symbols to generate the data frame. Finally, the preamble sequence is appended at the beginning to form the OFDM transmitted frame. In our implementation, the generation of transmitted data and the wireless channel effects are applied in simulation. The received data is then read into the LabVIEW HOST VI from a file. The HOST performs I/Q interleaving to match the data format expected by the DAC interface and sends the data to the FPGA for transmission. These samples are then transmitted via the DAC and using a loopback cable fed back into the ADC for acquisition. In a real wireless transceiver system, instead of the loopback cable, there will be a suitable upconverter and RF antenna at the transmit side followed by an RF antenna and suitable downconverter at the receiver side. In the absence of these components though, the loopback cable provides a suitable alternative to help understand the effects of the analog interface.

The system-level view just described above is depicted in Figure 5.2.

As an aside, we note that the symbol timing algorithm operates on the preamble sequence and therefore isn't directly affected by the constellation selected for the data. However, the BER degradation caused due to a fixed symbol timing error can be expected to be greater for higher-order constellations.

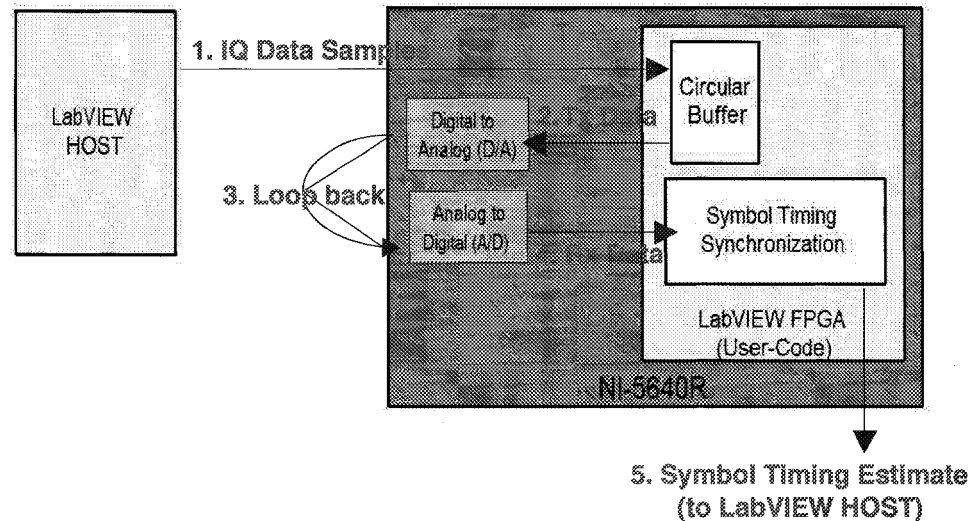


Figure 5.2 : LabVIEW system overview

5.3 Host VI and Analog Interface

The LabVIEW HOST VI serves as a medium to target the FPGA VI. As discussed previously, the HOST VI downloads the appropriate bitfile to the FPGA and controls the code running on the FPGA. Post-configuration, we load the transmitted data into the HOST VI and send it to the FPGA for transmission. In our implementation, the FPGA VI loops over this data set for transmission of data until the user loads a new data set via the HOST VI. The HOST VI also enables the user to control values for

parameters (eg: threshold) and observe the results from the timing metric computation. The HOST/FPGA interface is supported by the PCI bus and Direct Memory Access (DMA) mechanisms. For our implementation, we did not have DMA support for data transfers from the HOST to the FPGA. Hence, a state machine is used to achieve this. The controls and indicators provided on the FPGA VI (and hence accessible to the HOST VI) can be written to or read from using the configuration clock running at 20MHz. In our implementation, we tried to minimize the number of arrays that are used and especially arrays that have a front-panel control or indicator as these take significant FPGA resources. Finally, the DMA transfer speeds are dictated by the PCI bus in the existing hardware. Therefore, even if DMA transfers from the HOST to FPGA were available, bi-directional data transfers at 20MSamples/sec are not feasible. For our application, the data movement is primarily from the FPGA to the HOST VI - i.e. the results of the metric computation being transferred from the FPGA.

Figure 5.3 shows the FPGA code for transmission of data via the DAC. This loop also provides the state-machine for transferring the initial data set from the HOST to the FPGA VI. Over the course of this project, we designed various methods for the Host to FPGA communication. Presently, the NI-5640R hardware and software does not have support for interrupt service requests (IRQs) or direct memory access (DMA) transfers. After iterating through many versions of the state machine for transferring data from the HOST to the FPGA, we settled on this one that is provided by NI due to limitations we encountered with the data structures supported for a single-cycled timed loop and for crossing clock boundaries. Although the structure shown in Figure 5.3 is not very intuitive, it provides the desirable results.

Here, the loop shown is a single-cycled timed loop. Every clock tick of the DAC IQ

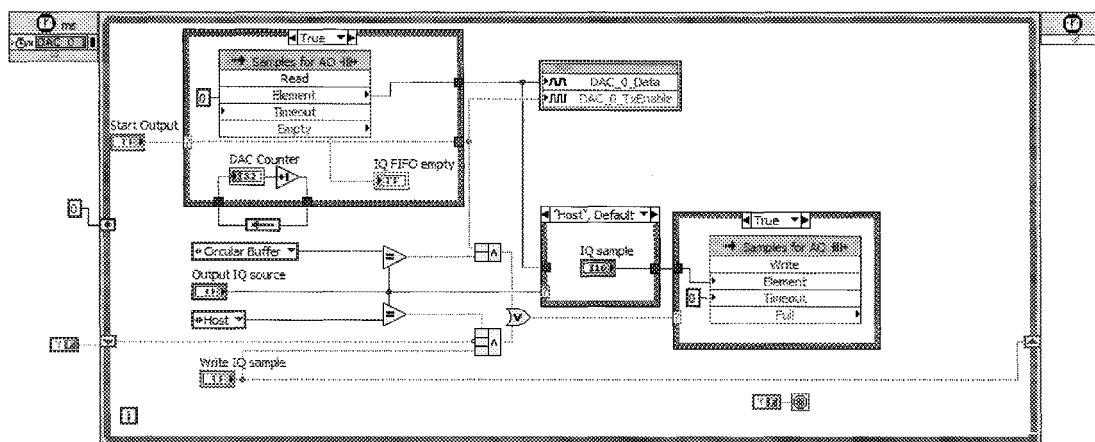


Figure 5.3 : LabVIEW FPGA DAC loop

Clock, a sample is received by the DAC and transmitted when the TxEnable signal is asserted (=logic 1 value). The HOST VI code interleaves the I and Q samples and these are stored in the 'Samples for AO' FIFO buffer in the FPGA VI. Once all the baseband I/Q samples have been sent, the HOST asserts the 'Start Output' signal and the DAC operation is triggered. Following this, the DAC loops over the samples in the FIFO buffer and transmits them unless there is an interruption and a new sample set provided by the HOST VI. The arrow symbols on either end of the loop symbolize one shift-register. The shift-register is initialized to a boolean value of false outside the loop. Inside the timed loop the 'Write IQ sample' variable is wired to the input of the shift register. The shift-register output together with the current value of 'Write IQ sample' detect a rising edge on this signal.

The data acquisition via the ADC is shown in Figure 5.4. The single-cycle timed loop operates every clock cycle of the ADC IQ Clock. The 16-bit IQ samples are stored in a FIFO buffer for later processing. The 'Start Output' signal ensures valid samples will be stored. The ADC loop is simpler than the DAC loop which has the

additional functionality of transferring data from the Host to the FPGA. This loop is part of the acquisition code provided with the NI-5640R driver.

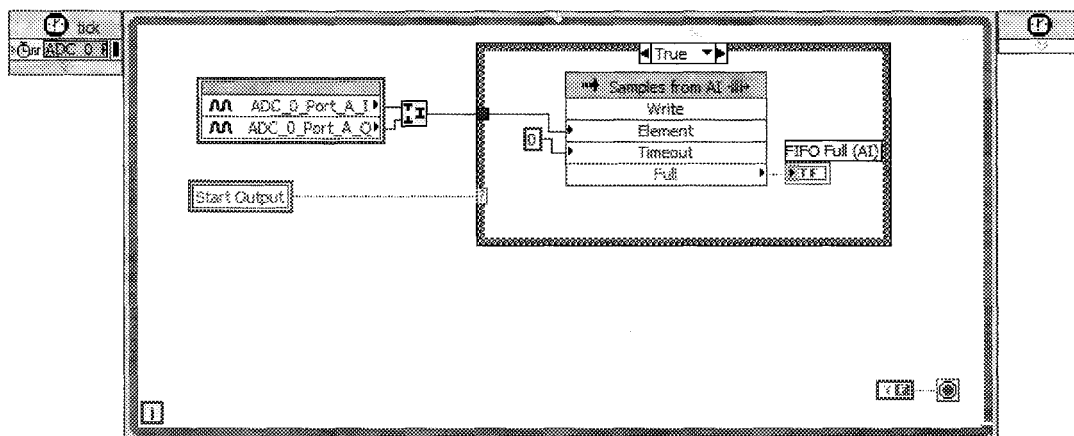


Figure 5.4 : ADC loop

As discussed in Chapter 4, the NI-4640R hardware is capable of providing multiple clock rates for the different on-board components. Further, the programmable interpolation and decimation filters result in the sampling rate being controllable by the user. We set the A/D and D/A sample rates to 25MSamples/sec in order to support the 20MHz bandwidth requirement for typical OFDM WLAN systems (IEEE 802.11a). In the case of the A/D and D/A, this signifies the sample rate required after data has been downconverted (A/D) and the rate at which baseband data can be supplied (D/A), respectively. Because there are sample rate conversions occurring within the A/D and D/A chips, the actual clock rate at which the A/D and D/A run is much higher. The A/D and D/A are clocked at 100MHz and 200MHz respectively. The reason for the 2x difference is that while the A/D provides IQ data in parallel, the D/A requires the IQ data to be interleaved. Hence, with this configuration the rate at which baseband IQ samples are processed is the same for both the devices.

5.4 FPGA System Implementation

In this section, we describe the hardware implementation of the symbol timing estimation algorithm. We start with a high-level view of the FPGA VI. Figure 5.5 shows a bird's eye of the block diagram for the top-level FPGA VI. On the left side is the configuration loop discussed in Chapter 4 and the two loops for acquisition and transmission of data. Note that the three loops are each running using their own separate clocks and are not connected (in the sense of wires going in or out) to any other blocks/loops. FIFO buffers provide the interface for moving data between the acquisition, transmission loops and the timing estimation. The configuration loop does not require data transfers; it uses the front-panel controls on the FPGA VI to configure the appropriate hardware registers.

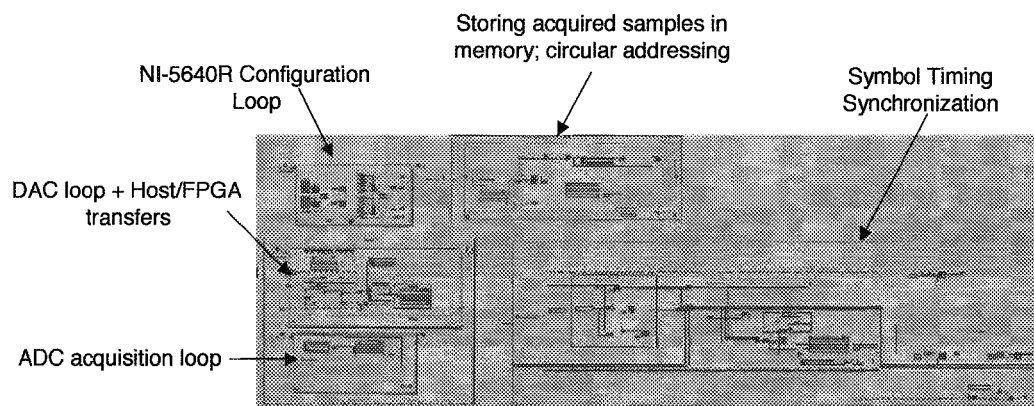


Figure 5.5 : FPGA block diagram layout

The loop on the top-right transfers the acquired samples from the FIFO buffer into memory. The FIFO buffer (without support for strided access) is not a suitable data structure for the access pattern required by the symbol timing estimation algorithm. Additionally, the FIFO is not very well suited as the samples read must be dropped,

whereas the wireless receiver will require access to the samples for further processing (performing the FFT operation). Thus, the FIFO buffers are used primarily to acquire data and transfer it between clock domains - in this case the ADC clock-domain to the top-level clock. The acquired samples are written to memory and the symbol timing estimation loop - a while loop that terminates when estimation is complete or the user stops the program - performs memory read operations to access the acquired data.

We now look at each of the components described above in a little more detail. Figure 5.6 shows the code for storing the samples acquired via the analog-to-digital converter.

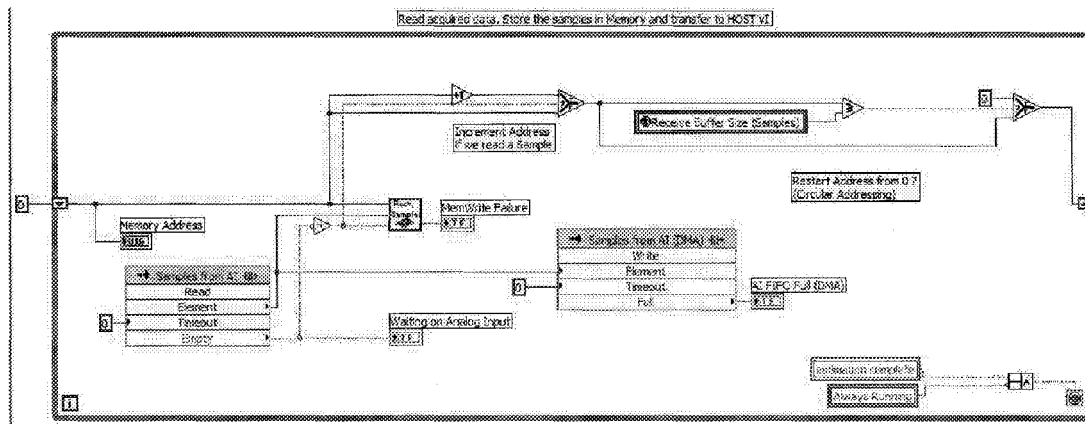


Figure 5.6 : Storing incoming data samples

Figure 5.7 shows the while loop for the metric computation and symbol timing estimation. As with any LabVIEW block diagram, the data flows from left to right. On the far left, we have the address lines and some simple address generation logic for strided-access. This is followed by the memory reads for accessing the data samples required for the current position of the two sliding windows. In the iterative mode,

the algorithm needs three samples to be read for every iteration. Next, on the right side are four sub-VIs for computing the timing estimation metric for the updated position of the sliding windows.

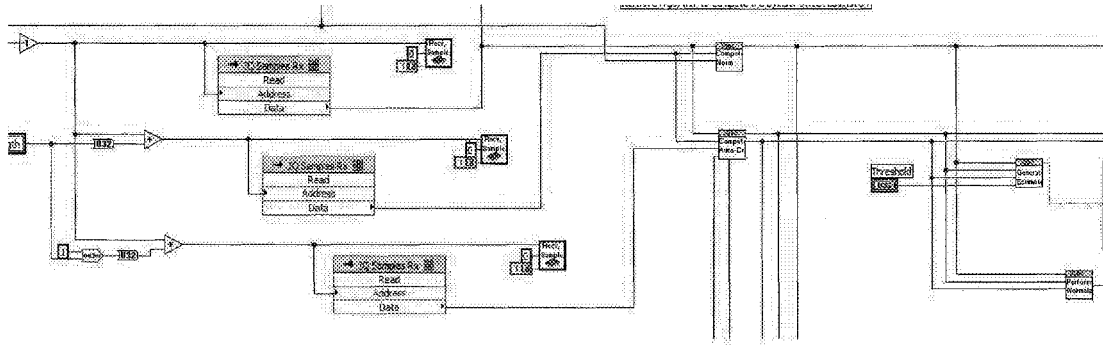


Figure 5.7 : LabVIEW FPGA symbol timing implementation

We discuss the implementation of Schmidl-Cox method (ACFN16) below. The AMDF implementation follows the same structure and is very similar to the ACFN16 implementation except that the AMDF-based method does not require the norm computation and the correlation computation VI is modified to implement Equation 3.2 instead of Equation 2.2. The code architecture, including the A/D, D/A loops, data storage, and program flow remain the same.

The initial norm computation is shown in Figure 5.8. This stage requires computing the norm for all the samples in the sliding window. Once we switch to the iterative mode, significantly fewer operations are required for computing the norm since most of the samples used for the previous norm computation are also present in the updated sliding window (the position has changed but that doesn't affect the

results from the previous iteration. The main computational block is the complex multiplier sub-VI.

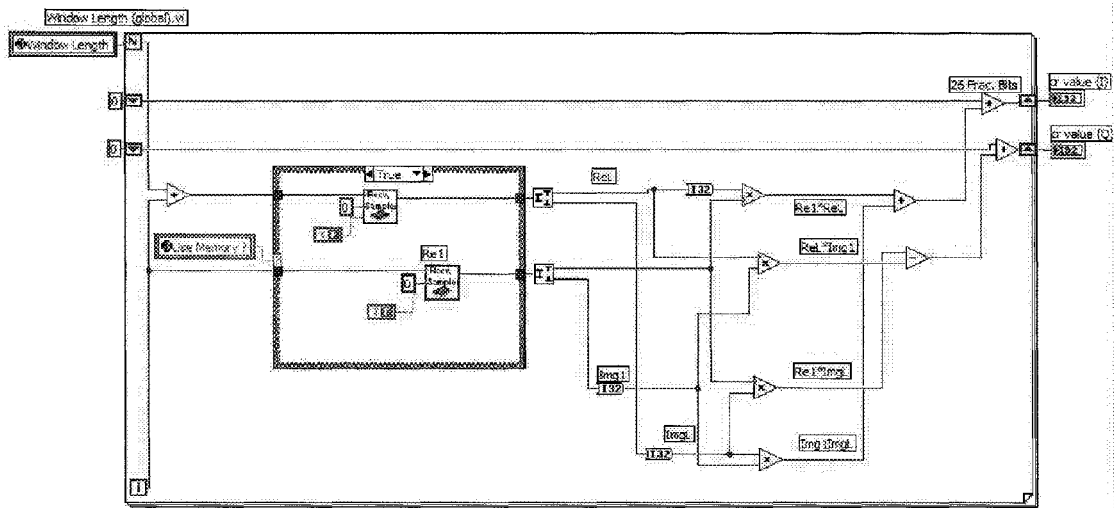


Figure 5.10 : Initial correlation computation

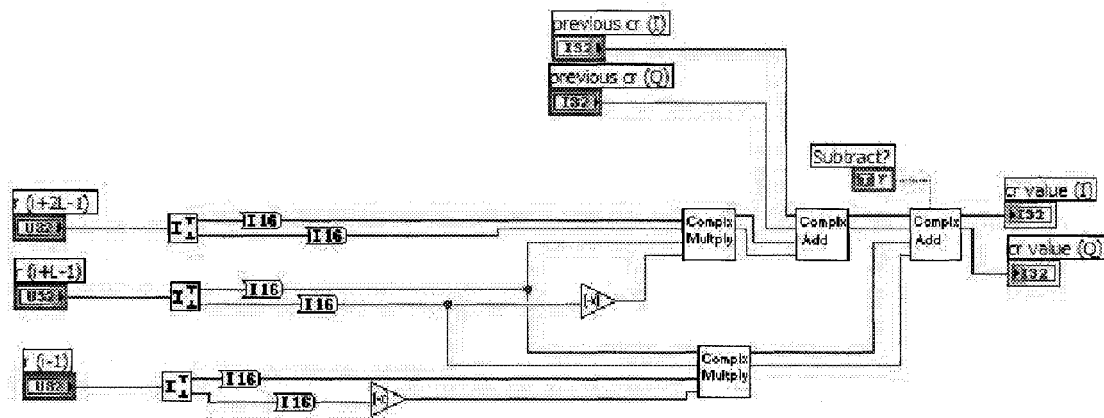


Figure 5.11 : Correlation metric update

In Figure 5.12, a snapshot of the estimation block is presented. The autocorrelation metric is normalized with the energy of the received signal and the threshold-

based estimation is applied while also transferring the normalized metric to the HOST via DMA transfer. Finally, Figure 5.13 shows the code for the threshold-based comparison.

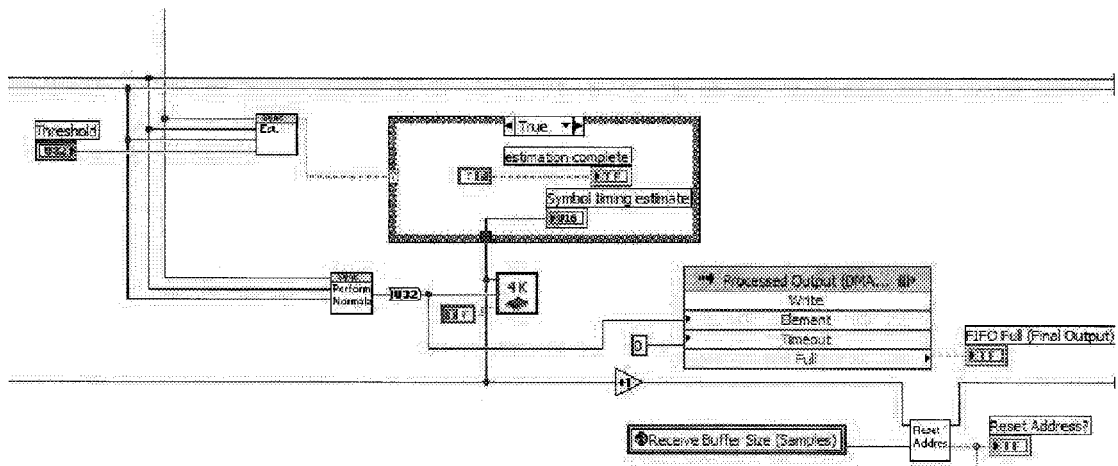


Figure 5.12 : Estimation and DMA transfer

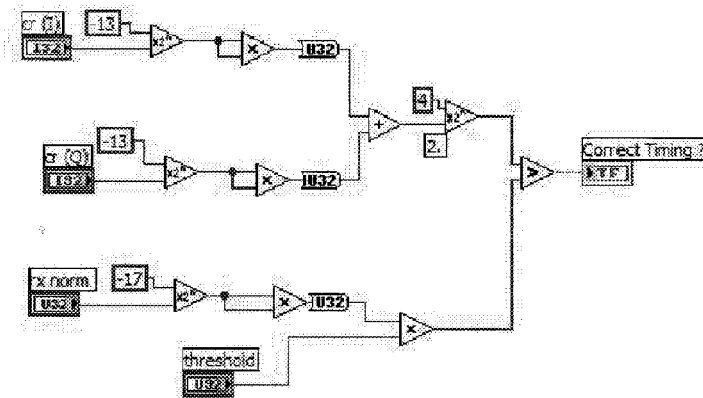


Figure 5.13 : Timing offset estimation

The front-panel view of the top-level FPGA VI is shown in Figure 5.14. As



discussed earlier, we can see the controls required for HOST/FPGA data transfers, indicators for communicating the state of the FIFOs and the FPGA VI, as well as parameters controllable from the HOST.

5.5 Synthesis Results

Table 5.1 shows the gate counts and resources required for the implementation on the Xilinx Virtex-II Pro XCV2P30 targeted using the LabVIEW FPGA module. The numbers in parentheses represent percentage.

Table 5.1 : FPGA synthesis results

Algorithm	Slices	Multipliers	Clock Freq.
Schmidl-Cox(ACFN16)	3769 (27)	20 (14)	78.09MHz
M_2 (ACFN32)	3604 (26)	20 (14)	73.93 MHz
AMDF	3428 (25)	4 (2)	84.66 MHz
M_3 (ACF)	2884 (21)	10 (7)	83.20MHz

For all the algorithms, our implementation makes use of the redundant information from previous iterations thereby reducing the computations needed for each new iteration significantly. This comes at the cost of increased control logic which is common for all the algorithms. This trade-off means that the impact of modifying the core estimation algorithm is reduced considerably as compared to an implementation that is not iterative in nature (and hence does not require two different modes of operation with unbalanced workloads).

Comparing AMDF with ACFN16 and ACFN32 (auto-correlation based metric with the normalization being done using 16 and 32 samples, respectively), we note that

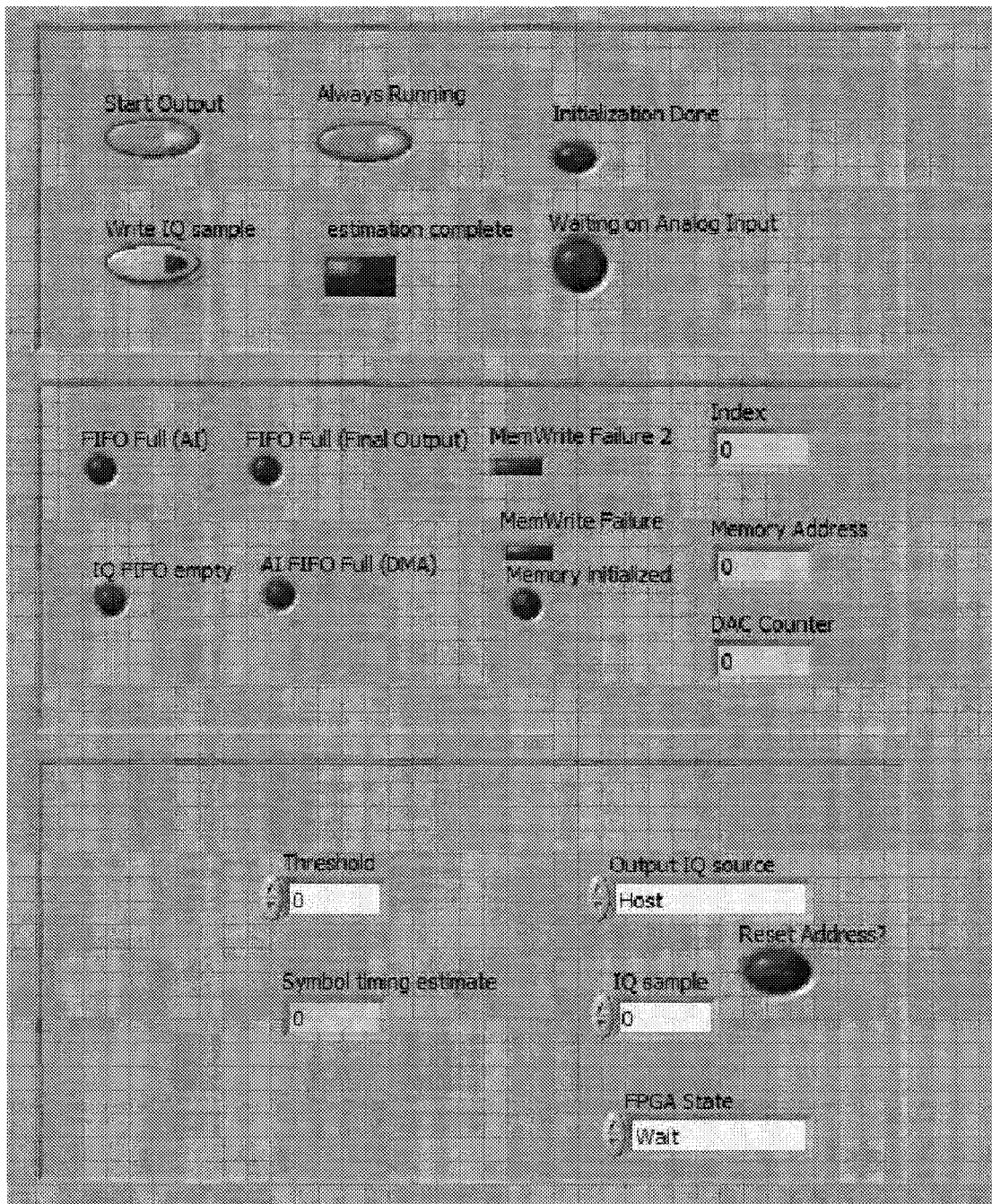


Figure 5.14 : Front panel view of FPGA VI

the AMDF-based method gives a factor of 5 reduction in the number of multipliers. With the complexity of the wireless receiver ramping up to support higher data-rates and complex modulation/transmission schemes, we believe that the multipliers are central to most receiver signal processing algorithms. This is supported by the increasing support for multipliers by key FPGA vendors. In terms of slice utilization, we get a slightly lower utilization for the AMDF method compared with ACFN16 and ACFN32.

The last scheme, M_3 (ACF without any normalization) has a much lower slices utilization than the other schemes. It still requires more than twice the number of multipliers needed for AMDF-based estimation, though. Furthermore, as discussed in Chapter 3, this scheme performs quite poorly compared to AMDF and ACFN32.

Finally, we observe that AMDF and M_3 can be run at a faster clock frequency than the other two methods. This result translates directly from their lower computation complexity and facilitates supporting higher-data rate systems.

In conclusion, if we compare the two methods with the best estimation performance, we note that the AMDF-based method reduces the resource utilization significantly while allowing for a faster clock rate as compared to M_2 (ACFN32).

Chapter 6

Conclusions and Future Work

In this thesis, we have addressed the problem of symbol timing synchronization for WLAN OFDM systems. Specifically, a simple preamble-based timing estimation method was proposed that exhibits a low computational complexity and results in performance comparable to the widely used autocorrelation-based approach. In addition to the complexity/performance analysis, we have applied the Golomb sequence, a sequence with low autocorrelation properties to preamble-based synchronization and shown simulation results quantifying the performance improvements over the short symbol sequence used in IEEE 802.11a WLAN systems. The preamble can be stored as a look-up table in the hardware and therefore does not require any changes in existing hardware implementations. An experimental field-programmable gate array implementation of timing estimation using the National Instruments NI-5640R hardware and LabVIEW software environment was also presented that utilizes fixed-point arithmetic.

In terms of future work, as pointed out earlier in this thesis, there is some literature on preamble-design for OFDM systems based on the optimization of a multi-objective function (for example, low PAPR, low autocorrelation, sensitivity to AGC, etc.). This thesis leads us towards exploring the design space for a suitable preamble sequence as an interesting problem for future work. With OFDM technology being combined with adaptive modulation schemes, multi-antenna systems, etc., the preamble design can have a major impact on the overall system performance.

In terms of hardware architecture and implementation, as the NI-5640R tools mature further along with increased support on the LabVIEW FPGA module, the hardware implementation can be adapted to utilize these advances and support higher data-rates required for future wireless communication systems.

Bibliography

- [1] National Instruments, “NI-5640R Transceiver Help,” Date taken: July 15th, 2007.
- [2] “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5 GHz Band,” 1999. Piscataway, NJ: IEEE P802.11/D7.0.
- [3] “Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer ETSI, Technical Report: TS 101 475 v1.1.1,” ETSI, April 2000.
- [4] “HiperLAN/2-The broadband radio transmission technology operating in the 5GHz frequency band,” HiperLAN/2 Global Forum, 1999.
- [5] Heiskala, J., Terry J., *OFDM Wireless LANs: A Theoretical and Practical Guide*. Sams publishing, 2001.
- [6] Speth, M., Fechtel, S.A., Fock, G., Meyr, H., “Optimum receiver design for wireless broad-band systems using OFDM:I,” *IEEE Transactions on Communications*, vol. 47, pp. 1668–1677, Nov 1999.
- [7] Speth, M., Classen, F., Meyr, H., “Frame synchronization of OFDM systems in frequency selective fading channels,” in *IEEE Vehicular Technology Conference*, vol. 3, pp. 1807–1811, May 1997.



- [8] Speth, M., Fechtel, S., Fock, G., Meyr, H., "Optimum receiver design for OFDM-based broadband transmission:II. A case study," *IEEE Transactions on Communications*, vol. 49, pp. 571–578, Apr 2001.
- [9] Yang, H., "A road to future broadband wireless access: MIMO-OFDM-based air interface," *IEEE Communications Magazine*, vol. 43, pp. 53–60, Jan. 2005.
- [10] Vaughan-Nichols, S.J., "OFDM: back to the wireless future," *Computer*, vol. 35, pp. 19–21, Dec 2002.
- [11] Batra, A., Balakrishnan, J., Aiello, G.R., Foerster, J.R., Dabak, A., "Design of a multiband OFDM system for realistic UWB channel environments," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 9, pp. 2123–2138, 2004.
- [12] Cabric, D., Brodersen, R.W., "Physical layer design issues unique to cognitive radio systems," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 2, no. 11-14, pp. 759–763, 2005.
- [13] Schmidl, T.M., Cox, D.C., "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, pp. 1613–1621, Dec 1997.
- [14] Minn, H., Zeng, M., Bhargava, V.K., "On timing offset estimation for OFDM systems," *IEEE Communications Letters*, vol. 4, pp. 242–244, Jul 2000.
- [15] Park, B., Cheon, H., Kang, C., Hong, D., "A novel timing estimation method for OFDM systems," *IEEE Communications Letters*, vol. 7, pp. 239–241, May 2003.

- [16] Van de Beek, J.J., Sandell, M., Borjesson, P.O., "ML estimation of time and frequency offset in OFDM systems," *IEEE Transactions on Signal Processing*, pp. 1800–1805, July 1997.
- [17] Chia-Horng Liu, "On the design of symbol timing recovery for WLAN OFDM systems," in *IEEE International Symposium on Spread Spectrum Techniques and Applications*, pp. 184–188, Aug.-2 Sept. 2004.
- [18] Li Hui, Bei-Qian Dai, Lu Wei, "A Pitch Detection Algorithm Based on AMDF and ACF," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, 2006.
- [19] Fette, B., Gibson, R., Greenwood, E., "Windowing functions for the average magnitude difference function pitch extractor," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 5, pp. 49– 52, Apr 1980.
- [20] A. Fort et al., "A performance and complexity comparison of auto-correlation and cross-correlation for OFDM burst synchronization," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, p. 341, 2003.
- [21] Pacheco, R.A., Serinken, N., et. al., "Bayesian Frame Synchronization Using Periodic Preamble for OFDM-Based WLANs," in *IEEE Signal Processing Letters*, pp. 524–527, July 2005.
- [22] Gadhiok , M., Cavallaro, J.R., "Preamble-based Symbol Timing Estimation for Wireless OFDM Systems," in *Asilomar Conference on Signals, Systems, and Computers*, Nov. 2007.
- [23] Van Zelst, A., Schenk, T.C.W., "Implementation of a MIMO OFDM-based wire-

- less LAN system,” *Signal Processing, IEEE Transactions on*, vol. 52, pp. 483–494, Feb. 2004.
- [24] Lou Feifei, “Channel estimation OFDM,” *MS Thesis, Rice University*, Dec 2005.
- [25] Karkooti M., “Semi-Parallel Architectures For Real-time LDPC Coding,” *MS Thesis*, May 2004.
- [26] Karkooti, M., Cavallaro, J.R., “Semi-parallel reconfigurable architectures for real-time LDPC decoding,” in *International Conference on Information Technology: Coding and Computing, Proceedings*, vol. 1, pp. 579–585, April 2004.
- [27] Yang, S., Karkooti, M., Cavallaro, J.R., “VLSI Decoder Architecture for High Throughput, Variable Block-size and Multi-rate LDPC Codes,” in *IEEE International Symposium on Circuits and Systems*, pp. 2104–2107, May 2007.
- [28] Radosavljevic, P., De Baynast, A., Karkooti, M., Cavallaro, J.R., “Multi-Rate High-Throughput LDPC Decoder Tradeoff Analysis Between Decoding Throughput and Area,” in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, Sept. 2006.
- [29] Zhang N., Golomb S.W., “Polyphase sequence with low autocorrelations,” *IEEE Transactions on Information Theory*, vol. 39, pp. 1085–1089, May 1993.
- [30] Bumiller, Gerd and Lampe, Lutz, “Fast Burst Synchronization for Power Line Communication Systems,” *EURASIP Journal on Advances in Signal Processing*, 2007. Article ID 12145.