A Rapid Prototype of an
IEEE802.11a Synchronizer

Examensarbete utfört i Elektroniksystem
av

Mattias Olsson

LITH-ISY-EX-3290-2002
Linköping, 2002
A Rapid Prototype of an IEEE802.11a synchronizer

Master’s thesis written at division of Electronics Systems at the Department of Electrical Engineering at Linköping university

Mattias Olsson

Reg. nr.: LITH-ISY-EX-3290-2002

Supervisor: Mikael Karlsson-Rudberg
Examiner: Kent Palmkvist

Linköping November 13, 2002
En snabbt framtagen prototyp för IEEE802.11a synkronisering
A Rapid Prototype of an IEEE802.11a Synchronizer

Mattias Olsson

Sammanfattning
Abstract
The first part of the thesis consists of a theoretical overview of OFDM, the effects of different imperfections like carrier frequency offset, timing offset and phase noise followed by a short overview of the IEEE802.11a standard for WLAN. The second part consists of an overview of a number of different techniques for synchronization that have been published. A technique based on correlation in the time domain is chosen and implemented as a floating-point model and later as a fixed-point model using Matlab, Simulink and Xilinx System Generator. The fixed-point model is then synthesized to an FPGA to verify that the design flow works and that a required clock frequency can be achieved.

Nyckelord
Keyword
wlan, IEEE802.11a, OFDM, synkronisering
Abstract

The first part of the thesis consists of a theoretical overview of OFDM, the effects of different imperfections like carrier frequency offset, timing offset and phase noise followed by a short overview of the IEEE802.11a standard for WLAN. The second part consists of an overview of a number of different techniques for synchronization that have been published. A technique based on correlation in the time domain is chosen and implemented as a floating-point model and later as a fixed-point model using Matlab, Simulink and Xilinx System Generator. The fixed-point model is then synthesized to an FPGA to verify that the design flow works and that a required clock frequency can be achieved.
Acknowledgement

I would like to thank my supervisor Mikael Karlsson-Rudberg for his help and advise during this project. I would also like to thank Kent Palmkvist, Per Löwenborg, Jonas Carlsson and all the other people at the division of Electronic Systems of the department of Electrical Engineering at Linköping University.

Above all my girlfriend Kristin has been invaluable. She has endured the reading of my report and given me many good comments!
### Abbreviations

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BPSK</td>
<td>Binary Phase Shift Keying</td>
</tr>
<tr>
<td>CFO</td>
<td>Carrier Frequency Offset</td>
</tr>
<tr>
<td>CP</td>
<td>Cyclic Prefix</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response (filter)</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GI</td>
<td>Guard Interval</td>
</tr>
<tr>
<td>ICI</td>
<td>Inter-Carrier Interference</td>
</tr>
<tr>
<td>IDFT</td>
<td>Inverse Discrete Fourier Transform</td>
</tr>
<tr>
<td>IFFT</td>
<td>Inverse Fast Fourier Transform</td>
</tr>
<tr>
<td>IQ</td>
<td>In-phase and Quadrature</td>
</tr>
<tr>
<td>ISI</td>
<td>Inter-Symbol Interference</td>
</tr>
<tr>
<td>LUT</td>
<td>Look-Up Table</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium-Access Controller</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>MMSE</td>
<td>Minimum-Mean-Square Error</td>
</tr>
<tr>
<td>NLS</td>
<td>Non-Linear Squares</td>
</tr>
<tr>
<td>OFDM</td>
<td>Orthogonal Frequency Division Multiplexing</td>
</tr>
<tr>
<td>PAR</td>
<td>Peak to Average Ratio</td>
</tr>
<tr>
<td>PHN</td>
<td>PHase Noise</td>
</tr>
<tr>
<td>ppm</td>
<td>Parts Per Million</td>
</tr>
<tr>
<td>QAM</td>
<td>Quadrature Amplitude Modulation</td>
</tr>
<tr>
<td>QPSK</td>
<td>Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RFO</td>
<td>Residual Frequency Offset</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SFO</td>
<td>Sampling Frequency Offset</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>WL</td>
<td>Word Length</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction 1
   1.1 The objectives ........................................... 2
   1.2 Structure of the report ................................. 2

2 The OFDM System Model 3
   2.1 Introduction ................................................ 3
   2.2 The model .................................................. 3
   2.3 Imperfections in an OFDM system ......................... 6
      2.3.1 The frequency offset .................................. 7
      2.3.2 The time offset ....................................... 10
      2.3.3 Sampling frequency offset ............................. 10
      2.3.4 Phase rotation ....................................... 12
      2.3.5 Power amplifier transients ......................... 13
      2.3.6 Other imperfections .................................. 13

3 An Overview Of IEEE802.11a 15
   3.1 Introduction ............................................... 15
   3.2 Specification ............................................... 16
   3.3 The preamble ............................................... 16
   3.4 Maximum allowed CFO .................................... 17

4 Different Approaches to Synchronization 19
   4.1 Introduction ............................................... 19
   4.2 Maximum likelihood approach ............................. 20
      4.2.1 ML estimation in AWGN channels using the CP ...... 21
      4.2.2 ML estimation in Rayleigh fading channels ....... 22
      4.2.3 ML time offset estimation using CP and pilot sub-carriers 23
      4.2.4 ML CFO estimation in the frequency domain ...... 24
   4.3 Minimum Mean-Squared Error timing estimation ....... 25
   4.4 Maximum-Correlation Criterion ........................... 25
CONTENTS

4.4.1 Maximum-Normalized Correlation timing- and CFO estimation .............................. 26
4.4.2 Modified Maximum-Normalized Correlation ........................................................... 26
4.4.3 Implementation of the MC algorithm ...................................................................... 27
4.5 Nonlinear squares CFO estimation ............................................................................. 27
4.6 Coarse and fine timing estimation .............................................................................. 28
4.7 Phase tracking using pilots ....................................................................................... 28
4.8 Conclusions ............................................................................................................... 31

5 Simulation and Implementation .................................................................................. 33
5.1 Rapid prototyping design flow ................................................................................ 33
5.2 Choosing algorithms to implement .......................................................................... 34
  5.2.1 Frequency offset estimation ................................................................................ 35
  5.2.2 Phase- and frequency tracking ........................................................................... 35
  5.2.3 Coarse- and fine timing offset estimation ......................................................... 35
5.3 Floating-point simulation model ................................................................................. 36
  5.3.1 Structure ............................................................................................................. 36
  5.3.2 Correlator ........................................................................................................... 37
  5.3.3 Angle calculator .................................................................................................. 38
  5.3.4 Frequency offset corrector .................................................................................. 39
  5.3.5 Controller ........................................................................................................... 39
  5.3.6 Results ................................................................................................................ 40
5.4 Fixed-point implementation ....................................................................................... 41
  5.4.1 Structure ............................................................................................................. 43
  5.4.2 Correlator ........................................................................................................... 44
  5.4.3 Angle calculator .................................................................................................. 44
  5.4.4 Frequency offset corrector .................................................................................. 45
  5.4.5 Controller ........................................................................................................... 45
  5.4.6 Finding suitable word lengths ............................................................................ 46
  5.4.7 Hardware reuse .................................................................................................. 49
  5.4.8 Synthesis ............................................................................................................ 49
  5.4.9 Simulation of the hardware implementation ....................................................... 50
  5.4.10 Performance ..................................................................................................... 50
5.5 Evaluation .................................................................................................................. 51

6 Conclusions .................................................................................................................. 53
  6.1 Ideas for future work ............................................................................................... 54

Appendix 1. Experiences from using System Generator v2.2 ........................................ 59
Appendix 2. Screenshots ................................................................................................... 61
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix 3. Blackbox Wrapper</td>
<td>63</td>
</tr>
<tr>
<td>Appendix 4. Matlab simulation scripts</td>
<td>67</td>
</tr>
<tr>
<td>Appendix 5. Tools that were used</td>
<td>69</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This final year report is part of a larger project at the Division of Electronic Systems at Linköping University. The purpose of this project is first to create a simulation model of a wireless local area network (WLAN) transmitter and receiver using Mathwork’s Matlab and Simulink. The report will focus on a standard called IEEE802.11a. This model is then to be refined and translated into a hardware description in VHDL (Very high speed integrated circuits Hardware Description Language) using a tool from Xilinx called System Generator. This hardware description is then to be implemented in an FPGA (Field Programmable Gate Array), which is a highly flexible programmable logic device. The possibility to perform parallel signal processing in an FPGA is one of the advantages of using an FPGA compared to a traditional DSP (Digital Signal Processor) solution.

The radio part of the IEEE802.11a standard for WLAN is based on a modulation technique called Orthogonal Frequency Division Multiplexing (OFDM), a technique used to achieve data transmission in a bandwidth-efficient way. This is accomplished by dividing the frequency band into several smaller sub bands with low individual data rates. If the sub bands are placed correctly they will not interfere with each other and cause Inter Carrier Interference (ICI).

One of the advantages of OFDM is that it is relatively robust against multipath reflections. Multipaths occur whenever a radio transmission can take several paths to reach its destination at slightly differing times. This means that the previous symbols will overlap with the current symbol and increase the noise. How the robustness to multipath arise will be treated in the coming chapters. The robustness does not apply to every aspect though, in reality
it has been realized that OFDM is rather sensitive to synchronization errors and especially to carrier frequency offset (CFO). The CFO is defined as the deviation in carrier frequency that the receiver experience. A vital part of an OFDM system is therefore the CFO estimator and corrector.

1.1 The objectives

The first goal of this report is to create an overview of the area of IEEE802.11a synchronization. The second goal has been to evaluate some of the algorithms using both floating point and fixed point arithmetic.

The reader of this report is assumed to have basic knowledge in electronics and telecommunications.

1.2 Structure of the report

The report is structured as follows. First, the OFDM system model is described, including some of the imperfections that it can experience. Then, the IEEE802.11a standard is briefly summarized to give necessary background information, followed by an overview of different methods and algorithms for synchronization. Some of the algorithms are chosen for implementation and evaluation. Finally, a few conclusions are drawn.
Chapter 2

The OFDM System Model

2.1 Introduction

The report will begin by describing how an OFDM (Orthogonal Frequency Division Multiplexing) signal is generated and by presenting some of the notation that will be used throughout the report.

The idea behind OFDM is to send data in parallel as $N$ densely packed sub carriers to achieve a bandwidth efficient- and multipath-resistant transmission. An OFDM signal could of course be created in several ways, but it is most commonly done in the frequency domain. Each symbol to be sent on a certain sub carrier is mapped onto a complex valued number, representing a certain amplitude and phase in the time domain. The constellations used are ranging from the simple BPSK (Binary Phase Shift Keying) to 64-QAM (Quadrature Amplitude Modulation) or higher. A constellation can be seen as a group of points in the complex plane which represents one or more bits of the data that is to be sent. See fig. 2.5 for a picture of the QPSK constellation.

2.2 The model

Using an IDFT (Inverse Discrete Fourier Transform) device the $N$ complex symbols $X_k$ taken from the modulation constellations in the frequency do-
2.2. THE MODEL

main are transformed into $N$ time domain samples $x_n$ such that [1]

$$x_n = \sum_{k=0}^{N-1} X_k e^{j \frac{2\pi kn}{N}}, \text{ for } n = 0, 1, 2, \ldots, N - 1. \quad (2.1)$$

Note that the time domain signal consists of complex numbers.

Some systems reserve a number of sub carriers for pilot carriers. On these pilots a sequence of bits known both in the transmitter and the receiver is sent. This knowledge can be used to track the phase of the received signal and to estimate channel characteristics, although it also increases the overhead in the system since useful information can not be sent on them.

A complete OFDM symbol is then constructed by appending a copy of the last $N_g$ samples to the beginning of the symbol, expanding the length to $(N + N_g)$ samples. The whole symbol, see fig. 2.1, can then be written as the sequence \( \{x_{-N_g}, \ldots, x_{-1}, x_0, x_1, \ldots, x_{N-1}\} \). The extra samples in the beginning of the symbol are called the cyclic prefix (CP) or the guard interval (GI).

![Equal samples](Equal samples)

The cyclic prefix is copied.

Figure 2.1: The cyclic prefix.

Before the time domain baseband signal is sent, it is multiplied by a pulse shaping filter $p(t)$ which is equal to zero outside the symbol, resulting in

$$x(t) = \frac{1}{\sqrt{N}} \sum_{i=-\infty}^{\infty} \sum_{k=-N_g}^{N-1} X_{i,k} e^{j \frac{2\pi kn}{N} t} p(t - iT_{s}), \quad (2.2)$$

where $i$ is the symbol number, $T_s$ is the sample period and $T = (N + N_g)T_s$ is the total symbol time. In IEEE802.11a the pulse shaping window has slopes at the ends to prevent out-of-band frequency leaks, but here a window $p(t)$, which is equal to one during one symbol, will be used to simplify the calculations.
CHAPTER 2. THE OFDM SYSTEM MODEL

The time domain signals is now multiplied (mixed) with a carrier, filtered and sent through the channel (air).

![Diagram of OFDM signal spectrum](image)

Figure 2.2: The spectrum of an OFDM signal after it has been mixed with the carrier.

Now assume the channel is a multipath channel where each path \( l \) has \( h_l(t) \) as its impulse response and is delayed \( T_l \) seconds. Define the timing offset \( \tau_0 \) as the difference between the true symbol starting time and the estimated starting time. Also assume that the guard interval (GI) is larger than the maximum path delay \( T_{max} \) minus the timing offset \( \tau_0 \), i.e. \(-N_g T_s + T_{max} < \tau_0 \leq 0\), so that no Inter Symbol Interference (ISI) occurs. With ISI the interference caused by overlapping symbols in time is meant.

In the receiver the opposite operation is performed. The received signal is brought down to the baseband by multiplication and low pass filtering. Because of differences between the oscillators in the transmitter and receiver, Doppler shift, etc, each path in the received signal is affected by a Carrier Frequency Offset (CFO) \( \Delta f \). The received signal, affected by the CFO \( \Delta f \), the impulse response \( h_l(t) \) and the white Gaussian noise \( w(t) \), can be written as

\[
r(t) = \sum_{i=0}^{L_p-1} h_l(t) x(t - T_l) e^{j2\pi \Delta ft} + w(t).
\] (2.3)

The normalized frequency offset \( \epsilon \) will sometimes be used instead of \( \Delta f \) to simplify the expressions. Define \( \epsilon \) as \( \epsilon = N \Delta f T_s \).

By inserting (2.2) into (2.3) and setting \( t = n'T_s = (i(N + N_g) + n + \frac{\tau_0}{T_s})T_s \)
2.3. IMPERFECTIONS IN AN OFDM SYSTEM

the received time domain signal at symbol $i$ and sample $n$ can be written as

$$r_{i,n} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{l=0}^{L_p-1} h_{i,l}(nT_s + \tau_0) e^{j \frac{2\pi (\tau_0 - T_l)k}{N T_s}} X_{i,k} e^{j \frac{2\pi n (k + \Delta f N T_s)}{N}} e^{j 2\pi \Delta f \tau_0} + w_n,$$

(2.4)

where $h_{i,l}(nT_s + \tau_0) = h_l(i(M + L)T_s + nT_s + \tau_0)$ is a simplified expression for the channel impulse response.

The usage of a CP makes the system less sensitive to a timing offset and multipath delays. This cyclical extension ensures that there is always an integer number of cycles within the DFT interval, at least as long as the timing offset and multipath delay is small enough. See the next chapter for a more thorough treatment of the timing- and frequency offset.

For a simplified description of the transmission chain, see fig. 2.3. The synchronization block contains operations such as packet detection, timing estimation, CFO estimation, and CFO correction.

![Diagram of OFDM transmitter and receiver chain](image)

Figure 2.3: A simplified OFDM transmitter- and receive chain.

2.3 Imperfections in an OFDM system

In [2] some of the imperfections that can occur in a real OFDM system are mentioned.

First, the time and frequency dispersion introduces both Inter Carrier Interference (ICI) and Inter Symbol Interference (ISI). ICI is the disturbing noise coming from other surrounding sub carriers and ISI is the noise coming from earlier OFDM symbols.
CHAPTER 2. THE OFDM SYSTEM MODEL

Second, the nonlinearities and clipping distortion. The amplitudes transmitted on different frequencies might add up and result in a high peak-to-average power ratio (PAR). This requires amplifiers with wide input ranges and with linear amplification. Such amplifiers are difficult to build and therefore one might have to accept a certain degree of nonlinearity. In OFDM nonlinearities may cause ICI and ISI.

Third, there is external interference, for example neighboring transmitters. External interference might be included in a model in the form of colored noise.

There are several causes for frequency offset [2]. Most important is the difference between the oscillators in the transmitter and receiver, but Doppler shifts caused by movement and phase noise (PHN) introduced by non-linear channels will also cause frequency offset. PHN is also introduced by imperfections in the oscillators, especially in the small and inexpensive ones that are normally used in mobile applications.

PHN and residual frequency offset (RFO) will result in phase rotation. A coherent OFDM system needs information about the phase to be able to correctly decode the incoming symbols and therefore some kind of a phase tracking device is essential.

In the next sections the effects of time-, frequency- and phase offsets on the received signal (2.4) will be studied.

2.3.1 The frequency offset

To see how the time and frequency offsets affect the received signal take (2.4) and assume that the channel is time invariant. Let $H_{i,k} = \sum_{l=0}^{L_p-1} h_{i,l} e^{-j \frac{2\pi k T_l}{N T_s}}$ and (2.4) can be written as

$$r_{i,n} = \frac{e^{j2\pi f_{f_0} T_0}}{\sqrt{N}} \sum_{k=0}^{N-1} H_{i,k} e^{j \frac{2\pi k T_0}{N T_s}} X_{i,k} e^{j \frac{2\pi n (k + \Delta f N T_s)}{N}} + w_n. \quad (2.5)$$

From (2.5) it can be seen that the effect of the frequency offset $\Delta f$ is as if each sample $n$ is multiplied by $e^{j2\pi n \Delta f / T_s}$, causing a time varying rotation in the complex plane. The direction of the rotation is depending on the sign of the frequency offset $\Delta f$. If the CFO is not completely corrected a slow rotation with time will occur. Since a coherent system calculates the phase continuously, for example by using the known pilot sub carriers, the rotation
can be estimated and compensated if it is small [3]. Unfortunately, as it will be seen in the next section, the CFO will also cause errors that can not be completely compensated by simply rotating the phase in the frequency domain.

**The SNR with frequency offset**

In this section the effects on the received signals by a frequency offset will be studied. If the symbol number \(i\) is neglected and \(\tau_0 = 0\) the DFT of \(r_{i,n}\) calculated at the receiver is equal to

\[
R_k = \sum_{n=0}^{N-1} r_n e^{-j2\pi kn/N} = X_k H_{i,k} \left( \frac{\sin(\pi \epsilon)}{N \sin(\pi \epsilon/N)} \right) e^{j\pi\epsilon(N-1)/N} + S_k + W_k, \quad (2.6)
\]

where the term \(S_k\), which represent the ICI, can be written as

\[
S_k = \sum_{l=0}^{N-1} X_l H_k \left( \frac{\sin(\pi \epsilon)}{N \sin(\pi (l-k+\epsilon)/N)} \right) e^{j\pi\epsilon(l-k+\epsilon)/N}. \quad (2.7)
\]

\(X_{i,k}\) is the desired signal and \(W_k\) is the additive white noise. See fig. 2.4 for a more intuitive picture of what is happening when a frequency offset is present. It can be seen as if the frequency axis is shifted and the sampling of the spectrum is no longer performed at the center of the sub carriers. As it can be seen from (2.6) this results in a decreased amplitude and rotated phase, but most important: it results in ICI.

![Figure 2.4: The spectrum with a frequency offset.](image-url)
In [4] a Gaussian approximation of the effective SNR \( \gamma_{\text{eff}} \) is derived to be

\[
\gamma_{\text{eff}} = \frac{|S_0|^2 \gamma_s}{1 + \gamma_s \sigma_{\text{ICI}}^2},
\]

(2.8)

where \( \gamma_s = E[|X_k|^2]/\sigma^2 \) is the SNR for the \( k \)-th sub carrier in the absence of a CFO. The variance of the ICI can be given as

\[
\sigma_{\text{ICI}}^2 = \sum_{l=0, l\neq k}^{N-1} |S_{l-k}|^2.
\]

(2.9)

This approximation assumes that the ICI is a Gaussian-distributed random variable. Equation (2.8) can be used to calculate approximate error rates expressed in the terms of the \( Q(\cdot) \) function. The \( Q(\cdot) \) function is the integral of the Gaussian distribution up to a certain point. It turns out that the approximation works well for small CFO and especially for BPSK, but less well for constellations with more levels and higher CFO.

In [5] the effect of the frequency offset is studied by deriving another expression for the loss of SNR in each sub band in decibels. It can be expressed as

\[
D = 10 \frac{1}{\ln 10} \frac{1}{3} (\pi \epsilon)^2 \gamma_s,
\]

(2.10)

where \( \epsilon \) is the normalized frequency offset. This approximation is only valid for a small CFO since it is derived using \( \ln(1 + x) \approx x \). It turns out that for a negligible degradation of about 0.1 dB, the CFO needs to be less than 1% of the sub carrier spacing. This will be used as a rule of thumb for the maximum remaining CFO that can be accepted.

In [6] an alternative expression for the effective SNR is presented:

\[
\gamma_{\text{eff}} \geq \frac{\text{SNR}}{1 + 0.5947 \text{SNR} \sin^2 \pi \epsilon \left( \frac{\sin \pi \epsilon}{\pi \epsilon} \right)^2},
\]

(2.11)

where \( \text{SNR} = \sigma_s^2/\sigma_n^2 \).

In [4] an expression for the bit-error probability in OFDM with BPSK-modulation is derived. The expression is

\[
P_b = \frac{1}{2} - \sum_{n \in \mathbb{N}_0} \sin[n \omega_0 \Re(S_0)] \frac{e^{-(1/2)w_0^2 \sigma^2}}{n \pi} \prod_{l=1}^{N-1} \cos(n \omega_0 \Re(S_l)) + \epsilon(0, \omega_0),
\]

(2.12)
2.3. IMPERFECTIONS IN AN OFDM SYSTEM

where $w_0 = 2\pi/T$ and $N_0 = \{1, 3, \ldots\}$ is the set of all positive odd numbers.

An expression for the probability of error in 16-QAM has also been derived in [4], but it is rather complex and will not be transcribed here. The technique used in [4] might be possible to use to derive a similar expression for 64-QAM.

Compensation for frequency offset

The most natural way to compensate for frequency offset would be to simply change the frequency in the receiver. The compensation can, however, be done digitally by multiplying each incoming sample by $e^{-j2\pi n \Delta f T_s}$. Note the minus sign which counteract the rotation caused by the CFO.

2.3.2 The time offset

The time offset has a different effect. It can be seen as if each complex OFDM symbol is multiplied by $e^{j 2\pi k \tau_0}$ where $k$ is the sub channel. The multiplication causes a sub channel dependent rotation in the complex plane with the largest rotation on the edges of the frequency band [7]. See fig. 2.5 for a picture of how the QPSK constellation is affected. As long as the timing offset is small enough this rotation can be estimated and corrected in the channel estimator [8] or in the phase tracker.

If the timing offset is larger than the guard interval minus the delay spread the sub carriers lose their orthogonality. This is because the DFT is no longer calculated using the samples within a symbol or the CP, instead it is overlapping with either earlier or later symbols. The delay spread is the time between the main path and the path that arrives last.

The synchronizations in time and frequency are closely related to each other [2]. A smaller relative frequency offset will occur if the distance between the sub carriers is increased, but the symbol length will get shorter and the relative timing error will become larger.

2.3.3 Sampling frequency offset

Sampling frequency offset (SFO) occurs mainly because of the tolerances in the quartz oscillators controlling the sample and hold device. In [9] the effects
of SFO is studied. Using the theory from [9] eq. (2.2) can be written as

\[ x_i(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_{i,k} e^{j \frac{2 \pi n k}{N} (1 + \epsilon_s)}, \]  

(2.13)

where \( \epsilon_s \) is the relative sampling period error \( \epsilon_s = \frac{T_s}{\Delta f_s} \). Assuming no CFO, no timing offset and no multipath, the output from the DFT becomes

\[ R_i(k) = e^{-j \frac{\pi}{N} n-k} \frac{\sin(\pi k \epsilon_s)}{N \sin(\frac{\pi k \epsilon_s}{N})} X_{i,k} + W(k) + \]

\[ + \frac{1}{N} \sum_{n=0, n \neq k}^{N-1} X_{i,k} \frac{\sin(\pi n \epsilon_s)}{\sin(\frac{\pi n (1 + \epsilon_s)}{N})} e^{j \pi n \epsilon_s \frac{N-1}{N} e^{-j \frac{\pi}{N} (n-k)}}, \]

where \( W(k) \) is the Gaussian noise. From this it can be seen that three different things occur. First the amplitude is reduced, secondly there is a phase rotation and thirdly there is ICI caused by the loss of orthogonality. If this is compared with the effects caused by a CFO, it can be seen that the phase shift and amplitude reduction is no longer equal for all sub carriers, but otherwise the effects are rather similar.

In [2] it is concluded that a non-synchronized sampling system is much more sensitive to frequency offset.
2.3. IMPERFECTIONS IN AN OFDM SYSTEM

2.3.4 Phase rotation

There are two major causes for phase rotation: residual frequency offset and phase noise. The RFO and the PHN will cause a phase rotation common to all the sub carriers, which can be estimated and corrected. Unfortunately the PHN will also have another higher frequency, Gaussian-like, behavior that can not be compensated.

IEEE802.11a uses coherent reception of the signals. Coherent reception means that the information is embedded in the amplitude and phase of the signal and therefore a phase tracking device is needed in the receiver. High-rate transmissions using 64-QAM or higher are especially sensitive to phase errors. Algorithms performing phase tracking will be covered in subsequent chapters.

Residual frequency offset

In reality there will always exist a small residual frequency offset even after the CFO has been corrected. This RFO will cause a rotating phase that can be modeled as [10]:

$$\phi_{RFO}^m(n) = \phi_{RFO}^m(n-1) + 2\pi \Delta f (N_g + n), 0 \leq n \leq N - 1,$$

with $\phi_{RFO}^0(n) = 2\pi \Delta f (N_g + n)$. This phase rotation will be no different compared to other common phase rotations and will be taken care of by the phase tracking device.

Carrier phase noise

Random carrier PHN is caused by imperfections in the oscillators of the transmitter and receiver or non-linear channels or amplifiers. The PHN has two effects. First it causes an equal random phase rotation of all the sub-carriers. Secondly, ICI will occur due to the loss of orthogonality between the sub-carriers, that depends on the bandwidth of the PHN [3].

The degradation in SNR can be estimated as [2]

$$D \approx \frac{11}{6 \ln 10} \left(4\pi N \beta T_s\right) \frac{E_s}{N_0}$$

where $\beta$ (in Hz) is the bandwidth of the power density spectrum of the carrier generator, $N$ is the number of sub carriers.
CHAPTER 2. THE OFDM SYSTEM MODEL

The phase rotation caused by the PHN can be modeled as a stochastic Wiener process [10]:

\[ \phi_{PHN}^m(n) = \phi_{PHN}^{m-1}(N - 1) + \sum_{l=0}^{N_0 + n} \theta_l, 0 \leq n \leq N - 1 \]  

(2.16)

with \( \phi_0^{PHN}(n) = \sum_{l=0}^{N_0 + n} \theta_l \), where \( \theta_l \) are stationary, zero-mean Gaussian samples with variance \( \sigma_{\theta}^2 \).

2.3.5 Power amplifier transients

Wireless devices often use batteries and therefore have to conserve power. This can for example be done by turning off components like the power amplifier in the RF (Radio Frequency) transmitter when it is not used [11]. However, it takes a while for the power amplifier to stabilize its gain after it has been switched on again. This time dependent gain introduces ICI and errors in Viterbi metrics computation. The Viterbi algorithm is used to find the bits that are most likely to have been transmitted. The metric is, in the simplest case, the number of positions where the bit values differ. In [11] an algorithm to decrease the effects of the transients are presented. The idea behind it is to use the known pilots to calculate new values for the equalization taps and the Viterbi metric based on the pilots. It is also show in simulations that the effect of the power amplifier transient is efficiently reduced with this algorithm.

2.3.6 Other imperfections

Other imperfections might occur that have not been treated here. One of them is IQ imbalance which is a result of gain mismatch between the I and Q amplifiers and deviation from the normal \( \pi/2 \) phase difference between the I- and Q-mixer. The IQ imbalance can be avoided almost completely by using a receiver with digital IQ conversion.
2.3. IMPERFECTIONS IN AN OFDM SYSTEM
Chapter 3

An Overview Of IEEE802.11a

3.1 Introduction

In 1999 the IEEE802.11a standard for WLAN was established. Until recently it has mainly been used in USA, but it is now coming to other parts of the world.

The radio interface is packet based and uses OFDM to achieve high data rates. Unfortunately, as it has been seen in the previous chapter, OFDM is rather strict when it comes to synchronization.

The IEEE802.11a MAC (Medium Access Unit) unit uses a technique called carrier sense multiple access, collision avoidance (CSMA/CA). The transmitter first listens to see if anyone is transmitting. If this is not the case it sends a short request-to-send package containing, among other things, the length of the unsent packet. The transmitter waits for the response before it starts transmitting. Other transmitters within reach also receive the request-to-send packet and then know how long the transmission will take.

In the following sections the key frequency- and timing parameters of the IEEE802.11a specification will be summarized.
3.2 Specification

The radio transmission takes place in a 20 MHz wide frequency band divided into 64 sub bands. 48 of these sub bands are used for transmission and 4 are used as pilot sub carriers. The remaining 12 sub bands are unused.

The IEEE802.11a receiver is coherent, which means that the phase has to be estimated before the decoding takes place. There is no way to know the phase of the transmitter except using the received data to calculate an estimate. The pilots are normally used to estimate the phase.

In the table below are some of the timing related parameters from the IEEE802.11a standard [12].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{SD}$ : Number of sub carriers</td>
<td>48</td>
</tr>
<tr>
<td>$N_{SP}$ : Number of pilot sub carriers</td>
<td>4</td>
</tr>
<tr>
<td>$N_{ST}$ : Number of sub carriers, total</td>
<td>52</td>
</tr>
<tr>
<td>$f_s$ : sampling frequency</td>
<td>20 MSample/s</td>
</tr>
<tr>
<td>Sub carrier frequency spacing</td>
<td>0.3125 MHz (=20 MHz/64)</td>
</tr>
<tr>
<td>$T_{FFT}$ : IFFT/FFT time</td>
<td>3.2 $\mu$s</td>
</tr>
<tr>
<td>$T_{GI}$ : GI duration</td>
<td>0.8 $\mu$s ($T_{FFT}/4$)</td>
</tr>
<tr>
<td>$T_{GI2}$ : Training symbol GI duration</td>
<td>1.6 $\mu$s ($T_{FFT}/2$)</td>
</tr>
</tbody>
</table>

Table 3.1: Timing related parameters.

3.3 The preamble

Each IEEE802.11a packet is preceded by a preamble, a sequence of samples whose purpose is to allow detection, synchronization and training. The standardized preamble structure, in the time domain, is shown to the left in fig. 3.1.

The first half of the preamble consists of ten identical short symbols. Each short symbol consists of 16 samples. The second half of the preamble consists of two identical long symbols, each 64 samples long, preceded by a 32 sample CP. The symbols are designed so that the correlation between two subsequent samples is minimal. Figure 3.2 shows the amplitude of the preamble in the time domain.
CHAPTER 3. AN OVERVIEW OF IEEE802.11A

Figure 3.1: The IEEE802.11a preamble in the beginning of a packet [12].

3.4 Maximum allowed CFO

The standard specifies a maximum oscillator frequency error of 20 ppm (parts per million) of the carrier frequency. If the transmitter and receiver have errors with inverse signs the observed total error will be 40 ppm. If a carrier frequency of approximately 5.3 GHz is assumed this translates to a $\Delta f_{\text{max}}$ that is 212 kHz.

Using (2.10) it can be seen that for a 0.1 dB degradation the maximum CFO is about 1% of the distance between the sub carriers or about 0.58 ppm. This will be used as a rule of thumb for the maximum tolerable CFO error.

IEEE802.11a uses error correction based on convolutional codes [3]. For the code used in IEEE802.11a the minimum Hamming distance between two codewords is 10 bits and the constraint length is 7 bits. Hamming distance is the number of positions where two code words differ. In a convolutional code the output bits depend on a certain number of previous input bits, namely the constraint length. According to [3] this means that if hard decision decoding is used the code can correct up to 4 bits within a group of about 3 to 5 times the constraint length. When soft decision decoding is used to increase the performance, the term coding gain is often used. The coding gain is a measure of how much the transmitted power can be reduced without increasing the error rate. An asymptotic coding gain at high SNR for a convolutional code can be calculated as [3]

\[
\text{coding gain} = 10 \log_{10}(\text{rate} \cdot \text{free distance}) \tag{3.1}
\]

Using the equation for a free distance of 10 and the rate $1/2$ gives an asymptotic coding gain of 7.0 dB. Simulations in [3] for a system operating in the 12 Mbit/s mode show that a coding gain of 5.5 dB can be achieved in reality for a normal SNR level.
To get an idea of how much CFO the error correcting codes can handle approximations can be made. If a hard decision decoder is assumed, it can, in the worst case, correct four bits during three times the constraint length. This gives a worst case bit error rate of roughly $4/(3 \cdot 7) = 0.1905$. 

Figure 3.2: *The amplitude of the IEEE802.11a preamble in the time domain.*
Chapter 4

Different Approaches to Synchronization

4.1 Introduction

In a packet based system like IEEE802.11a it is very important to get a quick estimate of the timing and CFO, otherwise the whole packet might get lost.

The common CFO estimation algorithms can be divided into two main groups: algorithms that are executed in the time domain and algorithms that are executed in the frequency domain.

In the time domain it is common to use the CP to calculate a CFO estimate. One disadvantage with that is that the CP will almost certainly be disturbed by ISI, since one of the purposes for putting it there in the beginning was to protect the data from ISI.

CFO estimation can also be performed in the frequency domain, after the calculation of the FFT. However, in the IEEE802.11a case the CFO estimate is needed early in the receiving process. CFO estimation in the time domain will cause less delay since the FFT does not have to be calculated before the estimate calculation.

In the following sections a few of the common techniques for time and frequency synchronization will be studied. It starts with Maximum Likelihood (ML), then it continues with a look at Minimum-Mean-Squared-Error (MMSE) and Maximum-Correlation (MC). Some more complex methods that are not necessarily well suited for implementation will also be presented.
4.2 Maximum likelihood approach

The principle of maximum likelihood was introduced by Fisher (1912) and is a method for parameter estimation. The idea behind it is to make the parameters $\theta$ as likely as possible by maximizing the joint probability function $f_y(\theta; y_1, y_2, \ldots, y_N)$ when the observed $N$ values are given by $y^*_N$. A reasonable estimator is then given by [13]

$$\hat{\theta}_{ML}(y^*_N) = \arg \max_{\theta} f_y(\theta; y^*_N) \quad (4.1)$$

According to [14] ML-estimators are usually consistent and often result in an estimate with a smaller variance compared to other non-biased estimators. It is not certain that a ML estimator is non-biased, but it can usually be corrected to become non-biased.

With a non-biased estimator the estimate becomes equal to the true value after an ‘infinite’ number of samples have been observed.

The Cramér-Rao inequality

It is interesting to see that there exists a theoretical lower bound on the mean-square error matrix $P$ that can be obtained with non-biased estimators. The following relation is called the Cramér-Rao inequality (C-R).

$$E[P] = E[\hat{\theta}(y^N) - \theta_0][\hat{\theta}(y^N) - \theta_0]^T \geq M^{-1} \quad (4.2)$$

where

$$M = E\left[ \frac{d}{d\theta} \log f_y(\theta; y^N) \right] \left[ \frac{d}{d\theta} \log f_y(\theta; y^N) \right]^T \bigg|_{\theta=\theta_0} \quad (4.3)$$

The matrix $M$ is called the Fisher information matrix.

Several authors have suggested maximum likelihood (ML) frequency offset estimators. The following section is a summary of the reasoning in [15], where a joint time- and frequency offset estimator is derived in the case of an Additive White Gaussian Noise (AWGN) channel.
4.2.1 ML estimation in AWGN channels using the CP

An ML timing- and CFO estimator is derived in [15]. The log-likelihood function can, under the assumption that the received samples \( r(k) \) are Gaussian, be written as

\[
\Lambda(\theta, \epsilon) = |\gamma(\theta)| \cos(2\pi \epsilon + \angle \gamma(\theta)) - \rho \Phi(\theta)
\]

where

\[
\gamma(m) = \sum_{k=m}^{m+N_g-1} r(k) r^*(k + N),
\]

is the complex correlation between \( N_g \) samples \( N \) samples apart and

\[
\Phi(m) = \frac{1}{2} \sum_{k=m}^{m+N_g-1} |r(k)|^2 + |r(k + N)|^2
\]

and

\[
\rho = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2} = \frac{SNR}{SNR + 1}.
\]

The value \( \Theta \) that maximizes (4.4) can be found to be

\[
\hat{\Theta}_{ML} = \arg\max_{\Theta} \{ |\gamma(\Theta)| - \rho \Phi(\Theta) \}.
\]

The normalized CFO estimate can then be calculated as

\[
\hat{\epsilon}_{ML} = -\frac{1}{2\pi} \angle \gamma(\hat{\Theta}_{ML})
\]

Since the CFO estimator depends on the angle of (4.5) it will be periodic and therefore the upper limits on the CFO that can be estimated is

\[
|\Delta f| \leq \frac{1}{N T_s} = \Delta f_{\text{max}}
\]

where \( N \) is the delay between the correlated samples and \( T_s \) is the sampling time. If the frequency offset is greater than \( \Delta f_{\text{max}} \) the resulting estimate will be unable to detect the part of the CFO that consists of an integer number times the distance between the carriers. For the short training symbols in the IEEE802.11a preamble the maximum detectable frequency error is 625 kHz and for the long symbols it is 156.25 kHz. As it was seen in Ch. 3 the
maximum CFO allowed is 212 kHz which is within the range of the short symbol, but not within the range of the long symbol. Thus, as soon as the coarse estimate is calculated it can be used to start correcting for the CFO, before the long symbols is received. However, frequency domain techniques that can estimate the integer part of the CFO exists. These algorithms will not be considered here because, as it was seen, the estimation range of the above algorithm is sufficient.

Using the knowledge of the pulse-shaping function

In [2] a method that also exploits the knowledge about the pulse-shaping function to create a CFO- and timing estimate is presented. The algorithm is related to the one presented in [15], but is of minor interest in IEEE802.11a since IEEE802.11a has a pulse-shaping window that only shapes the outermost samples. In simulations it is shown that a pulse-shaping function that shapes four samples on each side decreases the variance of the timing estimate significantly, whereas the performance is roughly the same for the CFO estimator [2].

4.2.2 ML estimation in Rayleigh fading channels

Rayleigh fading is not taken into account in the algorithm above. In a practical situation there might be so much fading that the performance becomes poor.

In [16] an ML estimator is derived that also depends on the autocorrelation of the received signal and thus indirectly on the autocorrelation of the transmitted signal.

Assume that $M$ consecutive samples are observed, i.e. $r_n$ where $k \leq n \leq k + M - 1$ and

$$r_n = s_n e^{j2\pi \epsilon/N} + w(n),$$

where $s_n$ are the samples at the receiver without the fractional frequency offset $\epsilon$.

Define $R_{ss}$ as

$$R_{ss} \equiv R_s + \sigma^2 I$$

where $R_s$ is the autocorrelation of $s_n$ and $\sigma^2$ is the variance of the added white noise.
Let $\tilde{a}_{m,n}$ be the elements of $R_{ss}^{-1}$. With these assumptions the log-likelihood cost function can be shown to be [16]

$$\Lambda = \sum_{m=1}^{M} \sum_{n=1}^{M} r^*(m)r(n)\tilde{a}_{m,n}e^{j2\pi(m-n)\epsilon/N} \quad (4.13)$$

Finding the CFO that maximizes (4.13) is rather difficult and therefore a simpler estimator is proposed in [16], namely

$$\hat{\epsilon}_N = \frac{1}{2\pi}(\pi \text{sign}(\theta_N) - \theta_N). \quad (4.14)$$

where $\theta_N$ is given by $\theta_N = \angle \sum_{n=1}^{2L} r^*(n+N)r(n)\tilde{a}_{n+N,n}$.

The proposed estimator above requires knowledge of the channel statistics, i.e. the noise power, Doppler spread, delay spread and multipath intensity profile, which makes it difficult to use it as an initial CFO estimator.

In [17] another algorithm for timing synchronization in a dispersive channel environment is presented. The structure is quite different with a linear pre-filter and a correlating filterbank. As before, the channel has to be known or at least estimated.

### 4.2.3 ML time offset estimation using CP and pilot sub-carriers

In [18] an extension to the algorithm in [15] is presented that also incorporates the known pilot sub-carriers in the time offset estimation process. The reasoning behind it is summarized below.

The time offset is estimated by maximizing a log-likelihood function over all possible values of $\theta$. To derive the log-likelihood function first assume a situation with $N$ sub carriers of which $N_p$ is occupied with pilot carriers. In an AWGN channel the received signal can be written as

$$r(k) = s(k - \theta) + m(k - \theta) + w(k) \quad (4.15)$$

where $\theta$ is the time offset, $s(k)$ is the received data, $m(k)$ is the pilots and $w(k)$ is the white noise with variance $\sigma^2_w$. Furthermore assume that $s(k)$ is a zero-mean Gaussian process with variance $\alpha \sigma^2_x$ where $\alpha = (N - N_p/N)$. Define

$$\rho = \frac{\alpha \sigma^2_x}{\alpha \sigma^2_x + \sigma^2_w} = \frac{\alpha \text{SNR}}{\alpha \text{SNR} + 1} \quad (4.16)$$
and use the knowledge of \(m(k)\) and the statistical properties of \(r(k)\), the log-likelihood function can be shown to be

\[
\Lambda(\theta) = \rho \Lambda_{cp}(\theta) + (1 - \rho) \Lambda_p(\theta)
\]

where

\[
\Lambda_{cp}(\theta) = \Re\left\{ \sum_{k=\theta}^{\theta+L-1} r^*(k)r(k+N) \right\} - \frac{\rho}{2} \sum_{k=\theta}^{\theta+L-1} |r(k)|^2 + |r(k+N)|^2
\]

depends on the cyclic prefix and

\[
\Lambda_p(\theta) = (1+\rho)\Re\left\{ \sum_{k=\theta}^{\theta+L-1} r^*(k)m(k-\theta) \right\} - \rho\Re\left\{ \sum_{k=\theta}^{\theta+L-1} (r(k)+r(k+N))^*m(k-\theta) \right\}
\]

depends on the pilot sub-carriers.

In this algorithm \(\Lambda_{cp}(\theta)\) gives an unambiguous but coarse estimate. \(\Lambda_{cp}(\theta)\) together with \(\Lambda_p(\theta)\), which has several very distinct peaks, result in an unambiguous and distinct peak in the log-likelihood function. The unambiguosness of \(\Lambda_p(\theta)\) also depends on the locations of the pilot sub-carriers.

In [18] a more robust estimator is also derived. This is accomplished by taking the absolute value of the terms in the log-likelihood function instead of the real value and by using \(\rho\) as a fixed design parameter. This algorithm is not ML, but it has a much smaller variance than the original algorithm. This depends on the fact that if a non zero RFO is present a phase rotation will occur. By using the absolute value instead of the real part only, the measure will be phase independent.

Simulation results are also presented in [18] and it can be seen that the synchronization performance is improved when also taking the pilot sub-carriers into account. It can also be seen from (4.16) that in a situation with low SNR and hence a small \(\rho\) the estimator depends mostly on the pilots and that the opposite is the case for a high SNR situation.

### 4.2.4 ML CFO estimation in the frequency domain

In [6] the following ML-estimator for the frequency domain is derived

\[
\hat{\epsilon} = \frac{1}{2\pi} \arctan\left( \frac{\sum_{k=0}^{N-1} \Im[R_{2,k}R_{1,k}^*]}{\sum_{k=0}^{N-1} \Re[R_{2,k}R_{1,k}^*]} \right),
\]

(4.20)
using two successive symbols in the frequency domain.

The estimator above is used in [19] to calculate a coarse estimate using the short training symbols in the IEEE802.11a preamble. The fine estimate is performed after the demodulation by correlating the demodulated output and the expected output and finding the maximum and shifting the position accordingly.

### 4.3 Minimum Mean-Squared Error timing estimation

In [20] different approaches to timing offset estimation is studied. One of them is based on the Minimum Mean-Squared Error (MMSE) criterion, which is the same as maximizing the similarity probability: \( P\{\hat{n} = n\} \).

In [20] the MMSE estimator is shown to be

\[
\hat{\Theta}_{\text{MMSE}} = \arg \max_\Theta \{ |\gamma(\Theta) - \Phi(\Theta) | \} \tag{4.21}
\]

The difference between (4.21) and (4.5) is that \( \rho \) is equal to 1 all the time, avoiding the need to calculate the SNR. In [20] it is seen in simulations that the MMSE approach is asymptotically identical to the ML approach for moderate noise. The CFO offset is calculated the same way as in (4.9).

### 4.4 Maximum-Correlation Criterion

In [21] a synchronization technique based on the maximization of the correlation (MC) between the CP and the data part of the OFDM symbol is presented. The timing estimate is found using

\[
\hat{k} = \arg \max_k |S_k|. \tag{4.22}
\]

The CFO estimate using the MC criterion is

\[
\hat{\epsilon}_{\text{MC}} = -\frac{1}{2\pi} \angle S_k \tag{4.23}
\]

If the envelope is constant and the noise is moderate this criterion could be compared with (4.21), but since this is not the case in OFDM this approach
must be sub optimum [20]. In the following chapter some variations of the maximum-correlation criterion will be studied.

4.4.1 Maximum-Normalized Correlation timing- and CFO estimation

In [22] Shmidl and Cox present an algorithm named Maximum-Normalized Correlation. The timing offset is found by maximizing

\[
\hat{m}_{\text{max}} = \arg \max_{m} \left( \frac{|\gamma(m)|^2}{P(m + L)^2} \right)
\]  

(4.24)

where \( \gamma(m) \) is given by the equation (4.5) and

\[
P(m) = \sum_{k=m}^{m+L-1} |r(k)|^2.
\]  

(4.25)

An estimate of the CFO can be found using (4.9) near the timing point found using (4.24). The variance of the estimate is calculated, using a method from [6], to be

\[
\text{Var}[\hat{\epsilon}] = \frac{1}{4\pi^2 \cdot L \cdot \text{SNR}}.
\]  

(4.26)

4.4.2 Modified Maximum-Normalized Correlation

In [23] a modification of the timing synchronization algorithm in [22] is presented. According to [23] algorithms like ML and MMSE are well suited for continuous OFDM applications, but tend to increase the probability for false alarm when no data is transmitted for an unknown time interval. To address the problems encountered in a burst mode applications the following timing estimator is proposed in [23].

\[
\hat{m}_{\text{max}} = \arg \max_{m} \left( \frac{1}{N_g + 1} \sum_{t=-N_g}^{0} \frac{2|\gamma(m)|^2}{P(m)^2} \right),
\]  

(4.27)

where \( P(m) \) is equal to (4.25) and \( N_g \) is the number of CP samples.
4.4.3 Implementation of the MC algorithm

In [24] the ML algorithm described in [15] is simplified and implemented in 10 DSPs, a FPGA and in an ASIC (Application Specific Integrated Circuit). The simplification that is done is to remove the energy term and then they arrive at an algorithm similar to the one based on the Maximum-Correlation criterion. It is concluded that the computing power of an FPGA or an custom designed ASIC is needed to make the implementation practical.

4.5 Nonlinear squares CFO estimation

In [25] a CFO estimator for IEEE802.11a based WLAN is derived. It is based on the minimization of a nonlinear squares (NLS) cost function. When the additive noise is white and Gaussian the estimate is equivalent to an ML-estimate conditioned on $x_S(1, n)$, which contains the samples of the first short symbol, at the transmitter. The estimate is

\[
\hat{\epsilon} = \arg \max_{\epsilon} \frac{1}{N_S(M_S - 1)} \sum_{n=0}^{N_S-1} |a_S^H(\epsilon)y_S(n)|^2
\]  

where $a_S(\epsilon) = [1 \ e^{j2\pi \epsilon} \ldots e^{j2\pi(M_S-2)\epsilon}]^T$, $y_S(n)$ is a vector of the received sample $n$ in all short symbols, $M_S = 10$ is the number of short symbols and $N_S = 16$ is the number of samples in each short symbol. The letter $H$ denotes transposition and complex conjugation.

In [25] it is said that (4.28) can be computed efficiently by applying a one-dimensional FFT to $y_S(n)$. Unfortunately, padding with zeros is needed to achieve high estimation accuracy.

For the long symbols a closed expression of the CFO estimate can be shown to be

\[
\hat{\epsilon} = -\frac{1}{2\pi N_L} \arg \left( \sum_{n=0}^{N_L-1} y_L(0, n)y_L^*(1, n) \right)
\]  

where arg(·) denotes argument.
4.6 Coarse and fine timing estimation

In [3] an algorithm for packet detection is presented. It works as follows. Calculate the correlation for the short training symbols and the received energy continuously. Divide these two with each other and trigger when the result grows towards one. Ideally the energy and correlation will be equal when the short training symbols are received. Precautions have to be taken to prevent division with small energy value, otherwise error amplification will occur. In a practical implementation the division can be avoided by using the following comparison instead:

\[
\text{correlation} > \text{threshold} \cdot \text{energy} \quad (4.30)
\]

A fine time estimate can later be found by correlating continuously with the known long symbols and then trying to find the maximizing instant. An approximation of where this instant should be located can be found using the packet detection. It can be shown that this correlation can be heavily quantized and that it works quite well using only the sign bit of the known sequence and the received signal. In an ideal situation in IEEE802.11a the correlation of the quantized long symbol gives 126 as the result. Knowing the maximum value allows us to chose a reasonable threshold level.

Thanks to the CP it is not critical if the timing estimate is a little too early as long as the multipath delay is small. On the other hand, if the timing estimate is too late ICI will occur. In a well designed system the CP will be long enough to take care of the normal multipath delay and more. This margin can be used by systematically shifting the symbol timing point inside the CP with a few samples. In [3] a 4-6 sample shift is presented as a rule of thumb for an IEEE802.11a system.

4.7 Phase tracking using pilots

Phase tracking is needed to estimate the phase of the received signal before it is passed through to the QAM demapper. Phase rotation has several sources, for example PHN, RFO, timing offset or moving targets.

In IEEE802.11a four sub carriers are reserved for pilots to help the receiver in the reception process. These pilots can be used to estimate the phase rotation by comparing the phase of the known pilots and the phase of the measured
pilots. If the angle differences between the transmitted and received pilots are plotted it might look like fig. 4.1. By studying equation (2.5) it can be seen that each sub carrier is multiplied with $e^{j2\pi k\tau_0/NT_s}$ if a time offset $\tau_0$ is present. In the figure this is represented by the slope of the line. It can also be seen that in a situation with CFO each symbol will be multiplied with $e^{j2\pi n\Delta fT_s}$, where $n$ is the symbol number, causing rotation common to all carriers. This is represented by the offset $m$ in the figure. By calculating the rate of change in the offset $m$ the CFO estimate can be further enhanced.

Using least square line fitting the slope $k$ and offset $m$ can be estimated. Another method to find the slope would be to divide each angle with its sub carrier index and then simply calculate the average. The estimate can be averaged over several OFDM symbols to decrease the variance. The offset can be calculated similarly by simply taking the average of the angles.

One problem with using the pilots for phase tracking is that the angles are limited and will be expressed modulo $2\pi$. For the line fitting or average calculation to give correct results some kind of a phase unwrapper must be used.

In [26] an extended version of the above method is presented that performs better in an environment where several different transmitters with different phase noise power are communicating. When the phase noise power is small
and the thermal noise is dominating, a slow FIR filter with many taps is used. The opposite is the case when the phase noise power is large. In that situation a faster FIR filter with fewer taps is shown in simulations to perform better. The algorithm is described below:

Let $\zeta$ be the phase rotation for a pilot sub carrier. The average phase rotation within a symbol can then be written as

$$\theta_i = \frac{1}{L} \sum_{l=1}^{L} \zeta_{i,l}$$

(4.31)

A rough estimate of the phase noise power can then be calculated as

$$\phi = \left| \frac{1}{N} \sum_{i=1}^{N} \theta_i \right|$$

(4.32)

This estimate can be used to determine how many taps the FIR filter should use:

$$M = \begin{cases} M_s & \phi \geq \text{th} \\ M_l & \phi < \text{th} \end{cases}$$

(4.33)

where th is the threshold. Finally, the averaged phase rotation is calculated as

$$\theta_{i,\text{ave}} = \frac{1}{M} \sum_{j=0}^{K-1} \theta_{i-j}.$$  

(4.34)

It is shown in [26] that the algorithm performs well, although it is simple.

To compensate for the phase offset the estimate can be used to adjust the channel estimate so that the phase of each sub carrier becomes correct. How a channel corrector can be implemented will not be covered in this report.

In [27] an interesting non-pilot-based algorithm for tracking the CFO is presented. It can be summarized as follows:

1. Calculate the CFO estimate and start compensating for it.
2. Select a number of sub channels whose channel transfer function magnitudes exceeds a chosen threshold. This is because small values in the transfer function will cause amplification of errors in the channel equalizer and it is not desirable that the errors increase.
3. Calculate the angle difference between the output from the selected sub channels after the channel equalizer and a following slicer. Buffer the difference for $N_w$ OFDM symbols.
CHAPTER 4. DIFFERENT APPROACHES TO SYNCHRONIZATION

4. Use an phase unwrapper to unwrap the phase difference after the buffer. Find the gradient of the unwrapped phase. This is done every $N_w$ symbols.

5. Once every $N_w$ symbols, use the gradient to calculate a new CFO correction term.

The phase unwrapping algorithm can be expressed as

$$\hat{\theta}_i = \hat{\theta}_{i-1} + \alpha \text{SAW}(\theta_i - \hat{\theta}_{i-1})$$  \hspace{1cm} (4.35)

where SAW(·) is a sawtooth function which limits the output to $\pm \pi/4$ and $\alpha$ is parameter that controls the variance of the unwrapped noise. The unwrapping algorithm was taken from [28].

4.8 Conclusions

Since this report focuses on IEEE802.11a the situation is rather different compared to what is found in a general OFDM system. First of all, it is packet based, which gives certain requirements that is not present, for example, in a broadcast application like digital television. Secondly, the IEEE802.11a packet has a preamble with short and long training sequences that are sent in the beginning of each packet. This preamble is designed for synchronization and CFO- and channel estimation and is therefore well suited for this purpose.

Several of the algorithms presented in this chapter are not really suited for usage in a packet based system. Some of them make use of a channel estimate that is not existing in the beginning of the packet and some are too complex and have high requirements on processing speed. IEEE802.11a was created in 1999 and therefore almost every algorithm presented before that time is aimed at continuous transmission of symbols.

In a general OFDM system without a preamble it is usually critical when the CFO estimate is calculated. This is different in IEEE802.11a. Since any two samples 16 samples apart in the first half of the preamble are identical any 32 samples can be taken and then the first 16 of them can be viewed as the CP. If a sliding window is used a CFO estimate that is not dependent on where it is sampled is found, as long as it is within certain limits. Fig. 4.2 describes how this is done. The mean value of all the CFO estimates can be calculated to get an estimate with lower variance. The same technique can also be used for the second part of the preamble.
4.8. CONCLUSIONS

The correlation between 16 samples that are 16 samples apart

Equal samples

16 16 16 16

Time

The correlation between 16 samples that are 16 samples apart

Figure 4.2: The structure of the preamble makes it easy to calculate the frequency offset.

Packet detection can be done using an energy measurement technique, but it is normally better to use all the available information about the structure of the preamble. By comparing the received energy and the correlated short training symbols in the preamble the beginning of the packet can be found.

Phase tracking is essential and is probably easiest performed using the pilot symbols. A phase unwrapper is needed to get correct results.
Chapter 5

Simulation and Implementation

Once again, the main goal of this project is to find suitable algorithms for synchronization that adheres to the IEEE802.11a standard. The result is supposed to be incorporated into a larger design which implements the whole standard from the MAC interface to the baseband in an FPGA and perhaps even in an ASIC.

Another goal is to investigate whether it is possible to use high level tools like Simulink for simulation and code generation and still get acceptable performance.

The phase tracking implementation will not be treated and it is assumed to have been taken care of elsewhere.

The next section will begin by describing the design flow that was used. The algorithm is then implemented using both floating-point and fixed-point arithmetic.

5.1 Rapid prototyping design flow

Rapid prototyping is discussed in [29], using IEEE802.11a as an example. The differences between the project in [29] and this project are basically the tools used. Below, in fig. 5.1, is a summary of their design flow after it has been adapted to suit this project.

The design flow is not strictly one-way. Results from each step can be used to change parameters and design on other levels.
5.2. CHOOSING ALGORITHMS TO IMPLEMENT

To verify that the implementation works as it is supposed to do, data output from the evaluation board can be compared to the data obtained from the simulations. The synchronizer was not evaluated in a real FPGA, but eventually it will be when it is integrated into the complete IEEE802.11a receiver.

5.2 Choosing algorithms to implement

In a packet based system like IEEE802.11a it is important that the time- and frequency synchronization and training are performed quickly and within the preamble. This forces us to favor algorithms with short delay and that are easily implemented. In the following sections suitable algorithms are discussed.
5.2.1 Frequency offset estimation

Practically every algorithm for calculating an CFO estimate in the time domain is based on measuring the phase difference experienced over time using either known data or the CP.

The best algorithm from a complexity viewpoint is to calculate the angle of the delayed and correlated samples as it is expressed for example in (4.9). It can be used, with different delays, for both the short and long training symbols in the preamble. The short symbols are needed for the estimator to get the required working range.

In an IEEE802.11a system the CFO estimation is normally done using the short and long training symbols in the preamble, sometimes followed by CFO tracking. The resolution of the short symbol estimate is not critical since small enough errors will be taken care of by the estimation using the long symbols.

It is possible to get sufficient accuracy using only the short training symbols, especially when the received signal has a high SNR and many symbols are used to average the estimates.

There exist other frequency based CFO estimators, but they have no advantage, at least not in the IEEE802.11a case with its preamble.

5.2.2 Phase- and frequency tracking

A phase tracker is needed to decrease the effects of RFO and PHN. The embedded pilots can be used to estimate the phase rotations. See section 4.7 for a description of a suitable algorithm using the pilots.

Frequency offset tracking may also be done using the pilots. From the slope of the phase offset variations between subsequent symbols the CFO can be tracked and compensated.

5.2.3 Coarse- and fine timing offset estimation

The first coarse timing estimation that has to be done is the packet detection. This can be done in several ways, for example with a windowed moving average correlation with the IEEE802.11a short training symbols [29] or by observing the relation between the received energy and the short training
symbol correlation. The second algorithm is preferred since it results in a criterion that is normalized to a value between 0 and 1.

Other criterions might be needed to ensure that a packet is really received, but that needs further investigation.

5.3 Floating-point simulation model

The chosen CFO estimation algorithm was first implemented in Simulink to verify that it worked as it was supposed to. To generate input data for the simulation a package of Matlab routines written by Mikael Karlsson-Rudberg was used. The channel model used was a simple multipath channel with additive white noise, but without Rayleigh fading.

In a later stage a partially complete Simulink model was used to generate packets. The simulation results are from simulations using datasets generated by the same Matlab routines as above.

5.3.1 Structure

In fig. 5.2 the structure of the CFO estimator and corrector is presented. In the beginning the incoming samples pass through the rotator unchanged and into the correlator. The output from the correlator is then transferred into the angle calculator which computes the argument of the complex signal. The mean angle is then calculated and finally a CFO estimate found. This first estimate is then used to decrease the CFO before the new estimate is calculated using the long training symbols.

Figure 5.2: The structure of the frequency estimator and corrector.


5.3.2 Correlator

The most central part of the CFO estimator is the correlator. The structure of the correlator is shown in fig. 5.3.

![Figure 5.3: The structure of the correlator.](image)

The correlator delay is different for the short and long training symbols. To save space and implementation cost the correlator can be adapted to the different delays during run time. This can be done by using multiplexers and by designing the control signals so that zeroes are shifted in to clear the shift registers, see fig. 5.4.

![Figure 5.4: The structure of the adaptive correlator.](image)

The running mean device calculates the mean value continuously for a certain number of past samples. To reduce the number of additions that has to be performed during each sample the calculations are performed iteratively, see fig. 5.5.

![Figure 5.5: The structure of the running mean calculator.](image)

In fig. 5.6 the amplitude of the output from the correlator in the time domain is shown. The first, almost square, peak is the correlated short training
signals and the second peak part is the long training symbols. The last parts are the correlated data symbols.

Figure 5.6: *The amplitude of the correlated signal vs. time.*

### 5.3.3 Angle calculator

The angle calculator is easily implemented in Simulink using a block that converts incoming complex valued numbers to a polar representation. An example of the output from the angle calculator can be seen in fig. 5.7. The left part is the calculated argument from the correlation of the short symbols and the right part is the argument from the long symbols. After the first CFO estimate the rotator starts to correct for the offset, that is why the angles are smaller for the long symbols.

The angles that come from the angle calculator contain noise that can be decreased by averaging. This is done in a running mean calculator over 32 samples. The output from the mean angle calculator can be seen in fig. 5.8.
5.3.4 Frequency offset corrector

The incoming samples can easily be rotated in the complex plane by a complex multiplication with a complex exponential with an increasing or decreasing angle.

5.3.5 Controller

Each sub block in the CFO estimator and rotator is equipped with control- and enable signals.

A simple ROM-based finite state machine was chosen to implement the controller, see fig. 5.9. A signal from the packet detector tells the sample counter to start counting. When the sample number equals the delayed next change point the program counter changes to the next address in the ROM which outputs the changed control signals and the new next change point. When the controller is not needed anymore it is reset and stopped until the next packet arrives. In fig. 5.10 the timing for the different control signals is
shown. The meanings of the control signals are found in table 5.1. As can be seen the correlator works in two different modes, correlation 16 or 64 samples apart. The only difference between the modes is the length of the delays in the correlator and in the running mean calculator. Zeros can be shifted into the delays by activating the reset signals.

5.3.6 Results

The floating-point model was simulated to measure the performance. In fig. 5.11 the absolute CFO error is plotted for different SNR.

The CFO estimator will perform differently depending on whether the packet is detected early or late. In fig. 5.12 the absolute CFO error is plotted against the packet detection error.
5.4 Fixed-point implementation

Xilinx has a product called System Generator which consists of a set of additional blocks for Mathwork’s Simulink. The blocks can be used for fixed point simulations in Simulink. The main reason, however, for using System Generator is that synthesizable VHDL easily can be generated from the simulation model. One of the objectives with the project is to determine how well this solution is compared to traditional ‘hand made’ solutions.

This project and hence the report has been targeted at finding algorithms and techniques for synchronization in IEEE802.11a. Due to time limitations only the CFO estimation and correction were implemented in hardware.

In [29], which is a report from a project similar to this, two important design
5.4. FIXED-POINT IMPLEMENTATION

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>corr-mode</td>
<td>Tell the correlator to correlate samples 16 or 64 samples apart</td>
</tr>
<tr>
<td>corr-delay</td>
<td>Enable the correlator delay</td>
</tr>
<tr>
<td>corr-mean</td>
<td>Enable the correlator running mean</td>
</tr>
<tr>
<td>corr-reset</td>
<td>Clear the correlator delay and running mean</td>
</tr>
<tr>
<td>ang-mode</td>
<td>Select CFO estimator factor for 16 or 64 samples</td>
</tr>
<tr>
<td>ang-mean</td>
<td>Enable the angle running mean</td>
</tr>
<tr>
<td>ang-reset</td>
<td>Clear the angle running mean</td>
</tr>
<tr>
<td>sample</td>
<td>Sample the estimate from the running mean and load the corrector with the new value.</td>
</tr>
</tbody>
</table>

Table 5.1: The meaning of the control signals.

Figure 5.11: The absolute error vs. the SNR for the floating point model.

requirements were emphasized. The first is that a highly pipelined architecture is needed to reach the required clock frequencies. In System Generator many of the building blocks can optionally use pipelining to run faster. In [29] it was also noted that the IEEE802.11a standard defines the IFFT/FFT period to be 3.2µs. The IFFT/FFT core that will be used in this project (the same as in [29]) requires 192 clock cycles to finish a 64-point complex fixed point computation which means that the clock frequency has to be at least 60
MHz. The second design requirement that was emphasized was that high fan outs can become a problem. Signals such as reset and chip enable can easily have to drive several hundred registers or combinational logic which limits the speed. Instead, when possible, such nets should be split into several nets with lower fan out to reduce the critical path delays.

5.4.1 Structure

The structure of the fixed-point implementation is basically the same as in the floating point implementation. See Appendix 2 for a screenshot of the structure taken from System Generator.

Assume that the incoming in phase and quadrature samples have a resolution of 9 bits in 2’s-complement.

Unfortunately the current version of System Generator has no abstraction for complex numbers and can not handle complex multiplication directly. Instead, a complex multiplicator has to be built using the available basic
5.4. FIXED-POINT IMPLEMENTATION

multiplication and addition blocks.

5.4.2 Correlator

The correlator is implemented as in the simulation model using a iterative approach. The memory usage is rather high, in total 256 delay elements are needed since two memory positions are needed to store each complex and real value.

An alternative implementation of the shift registers would be to use a dual port RAM and a ring counter to address the RAM. The benefit would be that no data has to be shifted, just the addresses have to be updated. The reason why the shift registers still are chosen is that the shift register block in System Generator is automatically replaced with a core that is optimized for FPGAs made by Xilinx. It is assumed that those cores will perform better.

5.4.3 Angle calculator

The angle between the imaginary- and real part of the correlated value is needed before the CFO estimate can be calculated, see for example (4.9). This involves an arctan calculation, which can be implemented using a Cordic rotator [30].

In System Generator it is possible to include external components written in VHDL by insertion of a certain black box-block into a Simulink subsystem. The behavior of the block can then be modeled using the ordinary Simulink blocks to make simulation work.

A free Cordic core can be found on [www.opencores.org](http://www.opencores.org) and it is documented in [31]. The implementation contains a rectangular to polar converter that can be used to calculate the angle. The core is pipelined in a number of steps that depends on the desired accuracy.

In the documentation accompanying the core some synthesis results are summarized. With a pipeline of 15 stages it uses 704 slices in a XC2S100-6 FPGA and can be run at 93 MHz.

When the core is simulated it can be seen that it introduces some quantization noise. See for example [32] for a thorough treatment of quantization errors in Cordic implementations.
Another simple solution would be to use a LUT (Look Up Table) containing the angles for different arctan values. Despite this the Cordic implementation was chosen, partly because it could serve as a test of the black box functionality in System Generator and partly because of the big LUT that would be needed to achieve a resolution high enough.

### 5.4.4 Frequency offset corrector

The System Generator package made by Xilinx contains a block called Direct Digital Synthesizer (DDS) which can be used to implement a Voltage Controlled Oscillator (VCO). The DDS is loaded with an angle incrementor that will be added to the internal accumulator after each sample. The current accumulator value is then used to find the correct complex value in a LUT. This value is then multiplied with the incoming samples to correct for the CFO. See fig. 5.13 for the structure of the CFO corrector.

![Figure 5.13: The frequency offset corrector.](image)

One disadvantage of the above solution is that another complex multiplicator is needed. An alternative solution would be to use another Cordic-unit to do the phase rotation. The free Cordic core used for the angle calculation can rather easily be transformed into a complex rotator instead. In this implementation another complex multiplicator was chosen.

### 5.4.5 Controller

The controller is implemented using a Moore state machine and a ROM for the control signals. Each address in the ROM represents the values for the control signals at each state.

The scheduling of the different control signals has to be done a little differently in the System Generator implementation compared to the floating point simulation. The angle calculator is implemented using a pipelined Cordic core and therefore causes a delay. This means that the control signals to the devices following the angle calculator has to be delayed accordingly.
5.4. FIXED-POINT IMPLEMENTATION

5.4.6 Finding suitable word lengths

To make the hardware implementation feasible it has to use fixed point arithmetic instead of floating point arithmetic. The size, cost and speed of a system is dependent on the word length (WL) used. There are two main approaches to WL optimization: one is based on simulations and one is based on analytical approach. Combinations of those two approaches can also be employed.

The simulation based approach: The results are compared to a reference model. Each WL is either changed exhaustively or changed heuristically according to some kind of error criterion until a minimum is found. This can lead to long simulations if it converges slowly.

In [33] a method for WL optimization is presented. The heuristic algorithm can be summarized as follows:

1. Design a reference system without signal overflows and quantization effects to verify that the algorithm works as it is supposed to do.
2. Define a performance measure such as signal-to-quantization-noise ratio.
3. Define a cost function such as area or gates.
4. Group signals that need to have the same WL to decrease the number of variables.
5. Determine the minimum WL for each group that fulfills the requirements on the performance measure. Consider one group at a time and let the other groups have a large WL.
6. Finish if the requirements are fulfilled.
7. Increase all WLs with one until the requirements are fulfilled.
8. Decrease the WL of one group at a time as much as possible. Start with the group that decreases the cost function the most, then the second most, etc.

Exhaustive WL optimization can also be performed as soon as the minimum WL for each group has been determined. It is stated in [33] that the additional hardware cost for the heuristic algorithm is usually less than 5% of that for the optimal exhaustive search. The number of simulations is linearly proportional to the number of groups.
One of the problems with WL optimization in a high-level environment is that such things as hardware assignment and scheduling normally are performed on a lower abstraction level. Simple optimization can result in two units using different WL even when there is more to gain by partly using the same hardware for both of them. One approach to handle this is presented in [34]. The idea behind it is to assign the minimum WLs, perform hardware assignment and scheduling and then find the optimal WLs.

**The analytical approach:** When using the analytical approach one tries to calculate the rounding error analytically and then determine the optimum WL.

WL optimization is a whole area of research in itself. In our relatively small case it was easier to use a more heuristic approach.

**WL optimization in the CFO estimator and corrector**

One of the advantages of Simulink is that every number, even inside the blocks, can be defined as variables. These variables can then be changed in for example simulation scripts. This way several different optimization algorithms can be used.

System Generator has one block called 'Quantization Error' which can be helpful when performing WL optimization. This block outputs the difference between the quantized signal and the floating point signal. The variance of this difference can be seen as a measure of the amount of noise introduced.

The design was simulated for different word lengths. It was assumed that the width of the input samples was nine bits. The following rather simple heuristic method was used:

1. Find relations between the word lengths in different parts of the design to decrease the number of variables in the system.

2. Decide on a weighting function. In this case the remaining absolute CFO error was used, but another function could be the quantization error.

3. Vary one word length at a time.

4. Study how the weighting function changes for each new value. Try to find the point when an increased word length does not result in a better performance.
5.4. FIXED-POINT IMPLEMENTATION

5. Validate the choice with another data set. If it is still a good choice continue by varying another variable.

6. When the word lengths can not be shortened any further, validate the final choice with fresh data sets.

Fig. 5.14 is an example of how the absolute CFO error changes as the word length of the angle calculator changes. The angle output width was chosen to be 20 bits.

![Bar chart](image)

Figure 5.14: The absolute CFO error for different angle calculator word lengths.

In [35] two extra bits are used in the receiver ADC (Analog-to-Digital Converter) compared to the transmitter DAC (Digital-to-Analog Converter) to accommodate for nonlinear effects and imperfect Adaptive Gain Control (AGC) at the receiver. In the same article artificial digital clipping in the transmitter is also studied. It turns out that at a price of a rather small degradation in performance the number of output bits in the transmitter can be decreased by introducing clipping.

Another choice that has to be made is how overflows should be treated and
if rounding or truncation should be used in the arithmetic units. In this case
the variance of the estimate could almost be halved if rounding was used
instead of truncation in the correlator running-mean device.

In fig. 5.15 the final word lengths are shown. The Cordic implementation
that was used takes in 16 bit 2’s-complemented numbers and puts out a 20
bit unsigned number representing the angle.

5.4.7 Hardware reuse

In the design there are some rather obvious places where it would be ben-
eficial to reuse the hardware. The only difference, for example, between
the correlator for the short symbols and the long symbols is the difference
in delay. By using multiplexers a switch between the delay lengths can be
implemented and the delay registers can be reset by shifting in zeros.

5.4.8 Synthesis

The generated VHDL implementation was synthesized using Leonardo Spec-
trum by Exemplar to verify that it is possible to get the required perfor-
ance. It was realized that just because the design can be run without
problems in Simulink it is not sure that it can be synthesized directly. Some-
times Leonardo complained that cores that had been generated could not be
used in a Virtex2, but after the design had been altered a little in Simulink
it worked all right.

Some of the important synthesis results are listed in table 5.2. Even though
the maximally allowed frequency has to be regarded as an approximation it
should be safe to assume that the design can be run at the 60 MHz that is
required by the Xilinx 64-point Virtex2 FFT core. The design only included
a controller, a complex rotator and a CFO estimator and was targeted for a
5.4. FIXED-POINT IMPLEMENTATION

Xilinx Virtex2-xc2v1000-bg575. In a situation where the design is allowed to use less space it is likely that the performance would be worse.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum frequency</td>
<td>136.537 MHz</td>
</tr>
<tr>
<td>SLICEs</td>
<td>534 out of 5120 (10%)</td>
</tr>
<tr>
<td>RAMB16s</td>
<td>2 out of 40 (5%)</td>
</tr>
<tr>
<td>MULT18X18s</td>
<td>4 out of 40 (10%)</td>
</tr>
<tr>
<td>RAM usage</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2: The results of the synthesis.

The synthesis tool also generates a post-synthesis VHDL representation that can be used in a VHDL simulator to verify that the design behaves correctly. Such a simulation was not performed.

5.4.9 Simulation of the hardware implementation

In fig. 5.16 the mean CFO error for different SNRs is plotted for the final hardware implementation. Each SNR was averaged over six datasets. The irregularities are a result of the stochastic nature of the datasets, but the trend is clearly that a higher SNR gives a better estimate, at least up to a certain level.

Normally the packet is detected within a few samples, but it can be interesting to see how the performance of the CFO estimator and corrector degrades as the timing offset changes. A number of simulations were performed using different timing offset. The absolute residual CFO error in ppm is shown in fig. 5.17.

5.4.10 Performance

The System Generator Reference Guide presents some guidelines to increase the performance of synthesized FPGA designs. One of them is that inputs and outputs should be registered. Another is that delay elements should be included within the design where it is possible to do so. The delay elements decreases the length of the paths and allows the design to be run at a higher frequency at the cost of a longer latency.
Saturation logic and rounding have performance costs and should not be used more than necessary. The places where it is more beneficial to use rounding and saturation logic have to be identified, either heuristically or analytically.

### 5.5 Evaluation

In this chapter a simulation model for an IEEE802.11a synchronizer has been created. The CFO estimator and corrector have also been synthesized to an FPGA using Xilinx System Generator.

A number of simulations have been performed to measure the ability to tackle a frequency offset. The channel used to generate the data sets did not incorporate Rayleigh fading, only a few multipaths and Gaussian noise, but the simulations should still give a rather realistic measure of the performance.

The floating-point model performs better than the fixed-point model, as expected, but not that dramatically. In [36] it is stated that an SNR larger
5.5. Evaluation

![Figure 5.17: The timing offset degradation for the fixed point model.](image)

than 25 dB is needed for 64-QAM reception and it can be seen from the simulations that the mean RFO after correction is less than 0.1 ppm, which corresponds to 0.2% of the distance in frequency between two sub carriers. If the 1% rule of thumb is used this means that the number of bits in the fixed-point model can be reduced until the rule of thumb no longer holds.
Chapter 6

Conclusions

The first part of this report is an overview of the area of IEEE802.11a synchronization and CFO estimation and correction in particular. The effects of different non-idealities, for example time- and frequency offset, were studied.

A number of techniques for synchronization were treated. Some of the algorithms are more suited for continuous, broadcast type transmissions, but the ideas can rather often be used in burst mode systems like IEEE802.11a too. One of the simplest and most straightforward methods was chosen for implementation and simulation. The method is based on measuring the phase difference between the cyclic prefix and the symbol during the short and long training symbols in the packet preamble. One of the drawbacks of the chosen method, though, is that it does not take the channel impulse response into account. Another drawback is that it does not, at least not in its current form, perform any CFO tracking while receiving the packet. An overview of tracking techniques has been included in the report, but no simulations have been performed.

To verify the fixed-point performance the algorithm has been modeled using a product from Xilinx called System Generator. System Generator consists of a number of additional blocks to Mathwork’s Simulink. As expected the fixed-point implementation gives a performance degradation compared to a floating-point implementation due to quantization noise. One often has to make a trade off between the added quantization noise and the word length. The simulations that can be performed using System Generator can be a valuable tool when the word lengths are to be optimized.

The main reason to use System Generator is that the models can be directly translated into synthesizable VHDL and hence reduce the time to create a
6.1. IDEAS FOR FUTURE WORK

working realtime signal processing application. This process usually works without problems, but sometimes one has to be careful how things are done, otherwise the design might not become synthesizable.

6.1 Ideas for future work

Cadence also has a tool for high level simulation and VHDL generation which is called Signal Processing Worksystem (SPW). SPW has been used by at least one research team ([29]) to implement IEEE802.11a in an FPGA. It would be interesting to compare the final results from System Generator with the same design implemented using SPW.

The next step is to integrate the design into the more complete ‘baseband’ receiver model that is being designed in the larger project at Electronics Systems Division.

In the future it would be interesting to implement the synchronizer using more asynchronous design techniques, even if no apparent advantages in the synchronizer case can be seen right now. Almost the whole synchronizer, apart from the tracking part and the CFO corrector, is only active during the beginning of a packet and can be completely shutdown after their work is done. It might be possible to find use for the synchronizer resources in other parts of the receiver, but that remains to be investigated.

On the algorithmic level there are also issues that remain to be treated. A synchronizer that is more resistant to multipath effects would be good. This could perhaps be accomplished by using the channel estimate, the cyclic prefix and the pilots during the reception of the packet together in a combined method. The achievable level of accuracy has to be further investigated.
Bibliography


Appendix 1. Experiences from using System Generator v2.2

During this project System Generator was used to implement a synthesizable design. Most of the time this worked well, but below we list a few things worth noting.

- Avoid using busses since they might cause problems at the synthesis step.
- The quantize-error block can sometimes cause problems when the design is synthesized. If they are used they might have to be removed before the synthesis.
- No blocks can have the same name, even if they are sub blocks. (At least in the earlier versions.)
Appendix 2. Screenshots

Figure 1: A screen shot of the CFO estimator and corrector.
Figure 2: A screen shot of the running mean calculator in the correlator.
Appendix 3. Blackbox Wrapper

External VHDL files can be included in a System Generator project using the so called blackbox building block. For this to work, a separate simulation model has to be implemented in Simulink. This model will then be replaced by a wrapper that will interface with the external building blocks written in VHDL.

In the CFO estimator an arctan calculator was needed and for this purpose a free Cordic core was integrated into the design. The Cordic algorithm can be used to calculate arctan. Below is the VHDL code of the wrapper that was used to interface with the core.

-- Anglecalc.vhd

-- Wrapper between the Rectangular–To–Polar CORDIC converter and the code generated by System Generator

-- Written by Mattias Olsson June 2002

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use work.conv_pkg.all;
use work.all;

entity Anglecalc is
  generic(
    Amplitude_arith: integer := xSigned;
    Amplitude_bin_pt: integer := 19;
    Amplitude_width: integer := 20;
    Angle_arith: integer := xSigned;
    Angle_bin_pt: integer := 19;
  )
end entity Anglecalc;
Angle_width: integer := 20;
Imagin_arith: integer := xlSigned;
Imagin_bin_pt: integer := 0;
Imagin_width: integer := 16;
Realin_arith: integer := xlSigned;
Realin_bin_pt: integer := 0;
Realin_width: integer := 16;
AngleEn_arith: integer := xlUnsigned;
AngleEn_bin_pt: integer := 0;
AngleEn_width: integer := 1
);

port ( Amplitude: out std_logic_vector(Amplitude_width − 1 downto 0);
Angle: out std_logic_vector (Angle_width − 1 downto 0);
Imagin: in std_logic_vector (15 downto 0);
Realin: in std_logic_vector (15 downto 0);
ce: in std_logic;
clk: in std_logic;
clr: in std_logic;
AngleEn: in std_logic_vector (AngleEn_width − 1 downto 0) );

end Anglecalc;

architecture structural of Anglecalc is

component r2p_corproc

port(
    clk : in std_logic;
    ena : in std_logic;
    Xin : in signed (15 downto 0);
    Yin : in signed (15 downto 0);

    Rout : out unsigned (19 downto 0);
    Aout : out signed (19 downto 0)
);

end component;

signal Xin : signed (16 − 1 downto 0); — The real value
signal Yin : signed (16 − 1 downto 0); — The imaginary value
signal Aout : signed (Angle_width − 1 downto 0); — Angle calculated — by the cordic

signal Rout : unsigned (19 downto 0); — Radius calculated
signal ena : std_logic;

signal counter : unsigned (4 downto 0);

begin — structural
  cordic : r2p_corproc port map (  
    clk => clk ,  
    ena => ena ,  
    Xin => Xin ,  
    Yin => Yin ,  
    Aout => Aout ,  
    Rout => Rout );

realdata : process ( ce , clk , clr )
begin
  if ( clr = '1' ) then
    Xin <= ( others => '0' );
  elsif ( rising_edge ( clk ) ) then
    if ( ce = '1' ) then
      Xin <= signed ( Realin );
    end if;
  end if;
end process realdata ;

imagdata : process ( ce , clk , clr )
begin
  if ( clr = '1' ) then
    Yin <= ( others => '0' );
  elsif ( rising_edge ( clk ) ) then
    if ( ce = '1' ) then
      Yin <= signed ( Imagin );
    end if;
  end if;
end process imagdata ;

— Do not enable the Cordic if the clr signal is active.

enablesig : process ( AngleEn ( 0 ) , clr )
begin
  if ( clr = '1' ) then
    ena <= '0';
  else
ena <= AngleEn(0);
end if;
end process enablesig;

-- Only output valid angles.
valid: process (ce, clk, clr, Aout)
begin
if (clr = '1') then
    counter <= (others => '0');
    Angle <= (others => '0');
elsif (rising_edge(clk)) then
    if (counter = 21) then
        Angle <= std_logic_vector(Aout);
    elsif (ce = '1' and ena = '1' and counter /= 21) then
        Angle <= (others => '0');
        counter <= counter + 1;
    end if;
end if;
end process valid;
end structural;
Appendix 4. Matlab simulation scripts

Below is the script that was used to measure the performance of the algorithm for different levels of SNR.

% simulate3.m
%
% Mattias Olsson 2002

T=21; % The number of delays in the angle calculator
DDSBits = 12; % The number of output bits from the DDS

Databits = 9; % The number of bits in the input samples
Inbits = 9; % The number of output bits from the corrector
Corrshift = 1; % The number of right shifts after the complex multiplication in the correlator
Corrmeanshift = 5; % The number of right shifts in the correlator running mean
CMultbits = Inbits*2 + 1 - Corrshift; % Number of output bits from the complex mult in the correlator
Meanbits = CMultbits + 6 - Corrmeanshift - 3; % Number of bits from the correlator running mean
Accubits = Meanbits; % Antalet bitar ut åfrn korrelatorn

Anglebits = 20; % Antalet bitar ut åfrn ävinkelberknngen
MAnglebits = Anglebits - 5; % Antalet bitar åfrn ämedelvordesbildaren
MAngleaccu = Anglebits; % Antalet bitar i medelvinkeln
Deltabits = MAngleaccu - 4; % Antalet bitar i ädeltavrdet

range = 0:2:30;
setspsnr = 3;
ppmerrvec = zeros(1, length(range));
for i = 1:length(range),
    for k = 1:setspsnr,
        disp(i)
        fname = sprintf('data/data%d-%d', i, k);
        load(fname); % Load the data set
        sim('xest19'); % Start the simulation
        ppmerrvec(i) = ppmerrvec(i) + ppmerr(end);
    end
end

bar(range, ppmerrvec/setspsnr)
xlabel('SNR(dB)');
ylabel('Absolute CFO error(ppm)');
title('Mean absolute CFO error vs SNR for the fixed point simulation model');
Appendix 5. Tools that were used

During this project the following software tools have been used:

Matlab 6.1
Simulink 4.1
Xilinx System Generator v2.2
WLAN-kit by Mikael Karlsson-Rudberg
\LaTeX
XEmacs ver 21.1
XFig
Leonardo Spectrum
ModelSim
Index

Angle calculator, fixed point, 46
Angle calculator, floating point, 40
CFO, 9
CFO estimation, frequency domain, 26
Controller, fixed point, 47
Controller, floating point, 41
Corrector, fixed point, 47
Corrector, floating point, 41
Correlator, fixed point, 46
Correlator, floating point, 39
CP, 6
Cramér-Rao inequality, 22
Cyclic prefix, 6
Fixed point implementation, 43
Frequency offset, 9
Hardware implementation, 43
Hardware reuse, 51
IEEE802.11a, introduction, 17
IEEE802.11a, specification, 17
Maximum likelihood, 22
Maximum-Correlation criterion, 27
Maximum-Normalized Correlation, 28
Minimum mean-squared error criterion, 27
MMSE, 27
Nonlinear squares, 29
OFDM, 5
OFDM, spectrum, 7
Packet detection, 29
Packet detection, 33
Performance, fixed point, 52
Phase noise, 9, 14
Phase tracking, 30
Pilots, 5
Power amplifier transients, 15
Preamble, 18, 26
Rapid prototyping, 35
Rayleigh fading channel, 24
Residual frequency offset, 9
Residual frequency offset, 14
Results, floating point, 42
RFO, 9
Simulation, fixed point, 52
Synthesis, 51
Time estimation, CP and pilots, 25
Word length, 48

70
The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

©Mattias Olsson