Synchronization Algorithms for OFDM Systems (IEEE802.11a, DVB-T)
Analysis, Simulation, Optimization and Implementation Aspects

vorgelegt von

Charbel El Hajjar

Erlangen, 2007
Als Dissertation genehmigt von
der Technischen Fakultät der
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Tag der Einreichung: 15. Oktober 2007
Tag der Promotion: 17. Dezember 2007
Dekan: Prof. Dr.-Ing. Johannes Huber
Berichterstatter: Prof. Dr.-Ing. Heinz Gerhäuser
Prof. Dr.-Ing. Wolfgang Koch
Acknowledgment

I would like to express my sincere thanks to my supervisor Prof. Dr.-Ing. Heinz Gerhäuser who gave me the opportunity to compete this thesis; I am grateful for the many hours and long meetings he spent with me for discussion. Prof. Dr.-Ing Wolfgang Koch was a great support being the co-referee; I hereby thank him for his fruitful comments. Many thanks go to Dipl.-Ing. Martin Speitel for his continuous support throughout this thesis. Moreover, I am deeply grateful to Dipl.-Ing. Dieter Peer as a friend and colleague; he never stopped motivating me. I would also like to thank Dipl.-Ing. Michael Schlicht and Dipl.-Ing. Karlheiz Ronge for their flexibility and support to me to pursue this dissertation. For the fruitful discussions, work and support, I owe many thanks to Dipl.-Ing. Holger Stadali, Dipl.-Phys. Bernhard Niemann, Dipl.-Ing. Amaia Anorga, Dipl.-Ing. Ralph Spitschka, Dipl.-Ing. Javier Arrospide and Mrs. Katharina Bronninger d’Aboim. Thanks to all my colleagues and friends at Fraunhofer IIS for the nice and productive working atmosphere. I would also like to thank my family and friends for their continuous care. Last but not least, I am deeply grateful to my wife for her support and patience throughout this thesis.

To my loving wife Meray, and my little sweet angel Carla
Abstract

Orthogonal Frequency Division Multiplexing (OFDM) or Multi-carrier Modulation (MCM) is a digital modulation technique that supports high-rate data with sufficient robustness to radio channel impairments (especially multi-path propagation). Due to that, it is emerging as the modulation technique used for the new generation of wireless communication systems (IEEE802.11a and DVB-T). However, one of the arguments against OFDM is that it is highly sensitive to synchronization errors. This raises up the need for optimum synchronization algorithms for OFDM applications such as IEEE802.11a and DVB-T.

In this thesis several synchronization algorithms are presented. We focus on the implementation aspects of synchronization algorithms and propose optimizations which lead to well performing and robust fixed point implementations. In addition, complexity and cost needed for such a project are analyzed leading to a model for classifying different algorithms depending on cost, time-to-market, and performance.

Chapter 1 is a short introduction. The use of OFDM in high data-rate communication systems is presented. In addition, the necessity of synchronization in OFDM systems opens way for the research done in the following chapters.

Chapter 2 is a general overview of OFDM systems. It deals with the basics of the OFDM technique in addition to the IEEE802.11a Wireless Local Area Network (WLAN) standard and the DVB-T standard which are explained briefly. Two important synchronization problems are presented: Timing and frequency errors and their consequences.

Chapter 3 gives an overview on synchronization principles of IEEE802.11a and DVB-T which will be used later in Chapter 4 and 5.

The above chapters give the needed background for the research done in this thesis. In addition, the following chapters deal with the newly performed results for the two OFDM applications: IEEE802.11a and DVB-T, which include algorithmic and implementational optimizations.
In chapter 4, IEEE802.11a synchronization is treated in details. This includes a fixed point implementation of an IEEE802.11a in SystemC. In addition, possible algorithmic optimizations aiming at decreasing the complexity of an IEEE802.11a synchronizer and maintaining a very well performance are presented.

In chapter 5 a similar treatment for the DVB-T system is presented. A fixed point implementation is studied, as well as algorithmic optimizations for increasing the performance and robustness of the DVB-T synchronizer with the lowest cost are discussed.

Cost analysis methods are presented in chapter 6. Effort and manpower for the studied algorithms are estimated. The new optimization ideas presented in the previous chapters are used to develop a new model for choosing the optimal synchronization algorithm depending on the cost, time-to-market, and performance desired.

The thesis concludes in chapter 7 which is a general overview summary of the work, opening new issues for further continuing research.
# Contents

1 Introduction 1

2 Overview of OFDM Systems 3
   2.1 Channels in a Multipath Environment 3
   2.2 What is OFDM? 5
   2.3 The IEEE802.11a Standard 8
   2.4 The DVB-T Standard 10
   2.5 Imperfections of OFDM Systems 12
      2.5.1 Time Offset 13
      2.5.2 Carrier Frequency Offset 14
      2.5.3 Sampling Frequency Offset 16
      2.5.4 Phase Offset 17

3 Synchronization Principles 19
   3.1 IEEE802.11a 19
      3.1.1 Autocorrelation Method 20
      3.1.2 Crosscorrelation Method 23
      3.1.3 Synchronization Procedure 24
   3.2 DVB-T 25
      3.2.1 Autocorrelation of the Guard Interval 26
      3.2.2 Crosscorrelation of Frequency-Domain Pilot Symbols 27

4 IEEE802.11a Synchronization 29
   4.1 Optimal Correlator Parameters 30
   4.2 Metrics 33
      4.2.1 Maximum Correlation (MC) 34
      4.2.2 Maximum Normalized Correlation (MNC) 34
      4.2.3 Minimum Mean Squared Error (MMSE) 36
      4.2.4 Maximum Likelihood (ML) 38
      4.2.5 Conclusions 42
   4.3 Simulation Model 42
4.3.1 Overview of the IEEE802.11a Transmitter .......................... 43
4.3.2 Imperfections .................................................................. 44
4.3.3 Synchronization Module .............................................. 45
  4.3.3.1 Coarse Synchronization ........................................ 45
  4.3.3.2 Fine Synchronization ............................................. 46
4.4 Fixed Point Implementation ............................................. 47
  4.4.1 SystemC for Fixed Point Implementation ......................... 48
  4.4.2 Correlator Implementation ......................................... 49
  4.4.3 MNC Metric in Fixed Point ....................................... 52
  4.4.4 CORDIC Algorithm ............................................... 53
  4.4.5 Coarse and Fine Synchronization .................................. 54
  4.4.6 Word Length Optimization ....................................... 55
  4.4.7 Implementation Loss Analysis ..................................... 55
4.5 Complexity Analysis ...................................................... 56
  4.5.1 MNC ................................................................... 58
  4.5.2 MC ................................................................... 58
  4.5.3 MMSE ............................................................... 59
  4.5.4 ML .................................................................. 59
4.6 Algorithmic Optimization ................................................... 60
  4.6.1 Correlator Approximations ......................................... 60
  4.6.2 Cramer Rao Bound .................................................. 63
    4.6.2.1 Log-Likelihood Function .................................... 64
    4.6.2.2 Cramer Rao Inequality ....................................... 64
    4.6.2.3 Cramer Rao Bound for This Application .................. 65
5 DVB-T Synchronization ......................................................... 67
  5.1 Synchronization Algorithms ............................................. 67
    5.1.1 Coarse Time Synchronization ................................... 67
    5.1.2 Frequency Synchronization ....................................... 70
      5.1.2.1 Pre-FFT Stage ............................................... 70
      5.1.2.2 Post-FFT Stage .............................................. 71
  5.2 Simulation Model .......................................................... 73
    5.2.1 DVB-T Transmitter ................................................ 73
    5.2.2 Channel ............................................................... 75
    5.2.3 DVB-T Receiver ..................................................... 77
      5.2.3.1 Pre-FFT Synchronization .................................. 77
      5.2.3.2 Post-FFT Synchronization .................................. 79
  5.3 Simulation Results .......................................................... 80
    5.3.1 Time Synchronization Evaluation ................................ 80
5.3.1.1 AWGN Channel .............................................. 84
5.3.1.2 Fixed Multipath Channel F1 ................................. 84
5.3.1.3 Mobile Channel ............................................. 85
5.3.2 Frequency Synchronization Evaluation ......................... 86
  5.3.2.1 Fractional Frequency Offset ............................... 87
  5.3.2.2 Integer Frequency Offset .................................. 89
5.4 Fixed Point Implementation ...................................... 96
  5.4.1 ML Algorithm ................................................. 97
  5.4.2 Choice of Optimal Wordlength ................................. 100
5.5 Algorithmic Optimization ....................................... 102
  5.5.1 Averaged Maximum Likelihood ............................... 103
  5.5.2 Weighted Maximum Likelihood (WML) ......................... 104
    5.5.2.1 Concept ..................................................... 104
    5.5.2.2 Simulation Results ...................................... 105
5.6 Complexity Analysis ........................................... 109
  5.6.1 ML Algorithm ................................................. 109
  5.6.2 WML Algorithm ............................................... 112
6 Cost Analysis ................................................................ 115
  6.1 Cost Estimating Models ........................................ 116
    6.1.1 Basic models for Effort Estimation ......................... 116
    6.1.2 Constructive Cost Model COCOMO ........................... 117
      6.1.2.1 COCOMO Models ......................................... 119
  6.2 Rayleigh Curve for Manpower Estimation ....................... 122
  6.3 Application of Costing Models on Synchronization Algorithms 124
    6.3.1 Application of COCOMO Estimation Model .................. 124
      6.3.1.1 IEEE802.11a synchronizer using MNC .................. 125
      6.3.1.2 DVB-T synchronizer using ML .......................... 125
      6.3.1.3 DVB-T synchronizer using WML ......................... 125
    6.3.2 Application of Rayleigh Curve .............................. 126
    6.3.3 Cost-Time-Performance Model (CTP) ......................... 129
      6.3.3.1 IEEE802.11a ............................................. 130
      6.3.3.2 DVB-T ..................................................... 132
7 Conclusions .................................................................. 133
A Background ................................................................ 137
  A.1 Gaussian Distribution ............................................ 137
  A.2 Rayleigh Distribution ............................................. 138
  A.3 Rice Distribution .................................................. 138
1 Introduction

As high data-rate communication systems tend to play an increasing role in our daily life, more and more research is being done on them. The vast number of applications involving high data-rate systems made it essential to achieve the best possible performance with the lowest possible cost.

Normally, these high data-rate systems suffer from the presence of multipath channels. This leads to the fact that the receiver is no longer able to distinguish the different symbols due to the delays occurring in each separate copy of the transmitted signal that arrives at the receiver. Thus complex equalizers are needed at the receiver end.

A solution to that problem is the use of OFDM. OFDM stands for Orthogonal Frequency Division Multiplexing. It is a digital modulation technology used in many of the high data-rate systems such as DAB (Digital Audio Broadcast), DVB-T (Digital Video Broadcast - Terrestrial), high speed telephone line communication such as xDSL (Digital Subscriber Line), and WLAN (Wireless Local Area Network).

One OFDM symbol is a set of a large number of orthogonal waves. The transmission is divided into smaller sub-bands, each having a low separate rate and is insensitive to dispersive multipath channels. Thus complex equalizers are not needed and a lower complexity is accomplished. Intersymbol Interference (ISI) is avoided by introducing a guard interval between successive symbols. This is the price to be payed for simple receiver structures.

Whereas OFDM solves the above mentioned problem, it introduces new problems itself. By using high numbers of narrow sub-carriers, the system becomes very sensitive to time and frequency offsets and thus a precise synchronization is needed.

Synchronization remains as one of the most important factors to think of while using OFDM. Many algorithms have been developed and analyzed in the literature, but still in the domain needs a lot to be done. This thesis aims at suggesting best synchronization solutions depending on priorities and trade-offs between cost (complexity), time-to-market and performance. Therefore, different synchronization techniques and
algorithms are studied, analyzed, simulated and optimized leading to a hardware implementation. Optimizations include algorithmic and implementational steps. We concentrate on optimization criteria keeping in mind the real implementation issue where many trade-offs are taken into account. To be able to analyze and optimize cost functions of the synchronization algorithms of OFDM systems, we have investigated 2 standards: IEEE802.11a and DVB-T. Each of the 2 models represents a different OFDM system (example: continuous mode transmission in the case of DVB-T, and burst mode transmission in the case of IEEE802.11a, etc... ). Thus, we have complemented our research with different OFDM models, starting with analysis, simulation, optimization, and ending in a fixed-point implementation of an optimized algorithm. This made it possible to perform hardware complexity study and an implementation loss analysis. After complexity estimation, manpower and effort estimations are done using different effort estimation methods. Therefore, having in mind the complexity estimation, the manpower estimation as well as the performance evaluation for different algorithms, we introduce what is called Cost-Time(-to-market)-Performance model which acts as a constellation to get the best solution depending on the priorities set by the developer.

This work concentrates on some open points in OFDM synchronization such as optimum implementation, optimization of complexity and cost needed and choosing the optimum synchronizer in different scenarios. However, there is still a lot to do on base of this work especially in the field of technology and innovation: cost analysis and manpower estimation approaches for such projects.
2 Overview of OFDM Systems

In this chapter, the multipath environment is shortly explained, followed by a general overview of OFDM. In addition to that, both investigated systems in this thesis are introduced: The IEEE802.11a standard and the DVB-T standard. The synchronization errors and problems facing OFDM systems are then summarized. The critical consequences of these synchronization errors raise the motivation behind this research work in order to analyze techniques and algorithms which correct them, optimize these algorithms for an efficient hardware implementation and establish an implementation with a good performance and low cost.

Fig. 2.1: Reference block diagram of an OFDM system.

Fig. 2.1 is the block diagram of an OFDM system showing the basic signals between transmitter and receiver. These signal notations will be used throughout this work. The synchronization block seen in this figure is dependent on the application under study (IEEE802.11a, DVB-T).

2.1 Channels in a Multipath Environment

In a wireless communication system, there are multipath channels between the transmitter and the receiver. Whereas the transmitter sends the modulated signal, there are multiple...

\[^{1}\text{All signals are considered in their equivalent complex baseband representation.}\]
indirect paths (non-line-of-sight) between the transmitter and the receiver beside the direct path (line-of-sight). These non-line-of-sight paths are due to three basic effects: reflection, refraction and diffraction. Fig. 2.2 gives an example of the different paths of a signal due to reflection.

Thus the received signal will be a superposition of signals each having a different delay and amplitude. Therefore, the resulting signal amplitude will vary depending how the individual signals add up (depending on their phase). Such an effect is known as fading. The transmission channel has a complex channel impulse response:

\[ h(\tau; t) = \sum_{i=1}^{N} h_i(t) \cdot \delta(\tau - \tau_i) \]  \hspace{1cm} (2.1)

which is a summation of delayed weighted Dirac impulses. \( N \) is the number of different paths, \( \tau_i \) stands for the propagation delay of path \( i \), and \( h_i(t) \) is the weighting factor of the path \( i \). If the channel has a direct path, the first path delay is called the propagation delay:

\[ \tau_p = \frac{d}{c} \]  \hspace{1cm} (2.2)

(\( d \) is the distance separating the transmitter from the receiver, and \( c \) is the speed of light).

The time between the first and the last path is called the maximum delay spread, which is \( \tau_{Max} = \tau_N - \tau_1 \). The last path is considered to be the one falling -10 dB under the strongest path. As defined in [Sk97], the root-mean square channel delay spread is:

\[ \sigma_\tau = \sqrt{\tau^2 - (\tau)^2} \]  \hspace{1cm} (2.3)

It is the second central moment of the power delay profile.

Back to (2.3),

\[ \tau^2 = \sum_{i=1}^{N} \frac{|h_i|^2 \tau_i^2}{\sum_{i=1}^{N} |h_i|^2} \]  \hspace{1cm} (2.4)
2.2. What is OFDM?

and

$$\tau = \frac{\sum_{i=1}^{N} |h_i|^2 \tau_i}{\sum_{i=1}^{N} |h_i|^2}$$  \hspace{1cm} (2.5)$$

The root-mean square channel delay spread can range from $20 - 50 \text{ ns}$ for small office environments, from $50 - 100 \text{ ns}$ for large buildings and $100 - 200 \text{ ns}$ for factory environments [Sk97]. In these environments usually only reflected paths are received, and these channels have a Rayleigh fading path characteristic and exponentially decaying power profile. This distribution changes into a Rice distribution if the direct signal is also received. In case the direct signal dominates, the distribution converges to a Gaussian with the strength of the direct signal as the central value.

As shown in [Kat96] the delay spread can take different values according to the type of terrain:

- Open area : $\sigma_\tau < 0.2 \mu s$
- Sub-urban area : $\sigma_\tau < 0.5 \mu s$
- Urban area : $\sigma_\tau < 3 \mu s$

If taken in the frequency domain, the channel coherence bandwidth is inversely proportional to $\sigma_\tau$:

$$B_c \sim \frac{1}{\sigma_\tau}$$  \hspace{1cm} (2.6)$$

A frequency-non-selective or flat channel is that where the signal has a bandwidth $B < B_c$. The result can be approximated by a multiplicative distortion leading to a time-dependent phase offset and an attenuation. A frequency-selective channel is that where the signal has a bandwidth $B > B_c$ and frequency components which fade differently. In this case, $\sigma_\tau$ is greater than the symbol duration $T_{SYM}$, and thus an equalizer is needed to avoid ISI in single carrier systems. In the case of IEEE802.11a, it is assumed that the channel impulse response is time-invariant during a burst. This is because in IEEE802.11a, the application is normally not mobile and the channel impulse response undergoes only slow changes over the time.

2.2 What is OFDM?

OFDM, an important special case of MCM (Multi-Carrier Modulation), stands for Orthogonal Frequency Division Multiplexing. The basic idea is to divide the dispersive channel into multiple low-rate non-dispersive channels.
In a single carrier system, different symbols overlap at the receiver if $\sigma_T > T_{SYM}$ in a multipath propagation. The delayed symbols of the non-line-of-sight paths which arrive at the receiver cause what is called Inter Symbol Interference (ISI). Thus the receiver is not capable of demodulating such a signal without the use of an equalizer. The equalizer introduces a high complexity to the system, and that’s why MCM is used to avoid this introduced high complexity.

The basic principle of an OFDM system is to send the complex information symbols by means of a set of sinusoids. Each sinusoid conveys the information corresponding to data symbols $c_k$ and can be regarded as a symbol of duration $T_{SYM}$. The superposition of these signals forms the received signal.

Fig. 2.3 shows an OFDM system comprising transmitter and receiver. At the transmitter side, the symbols $c_k$ are grouped in blocks of $N_{FFT}$ samples which are then processed by the IFFT. Therefore, the symbols $c_k$ can be regarded as the frequency content of the transmitted sequence. An FFT is performed at the receiver to recover the complex values. The orthogonality of the base signals used to construct the OFDM signal is the reason why the symbols $c_k$ do not interfere with each other in the time domain. In the time domain, the orthogonality condition states [BaSa]:

$$
\int_0^{T_{SYM}} g_i(t)g_k(t)dt = 0 \text{ for } i \neq k
$$

(2.7)

At the output of the IFFT, the obtained discrete sequence yields:

$$
s[n] = \sum_{k=0}^{N_{FFT}-1} c_k e^{j2\pi nk/N_{FFT}} \quad 0 \leq n < N_{FFT}
$$

(2.8)
Then, the continuous signal after the DAC will be the interpolation of $s[n]$:

$$s(t) = \sum_{k=0}^{N_{\text{FFT}}-1} c_k e^{j \omega_k t} \quad 0 \leq t < T_{\text{SYM}}$$

(2.9)

The base signals in our case are: $g_k(t) = e^{j \omega_k t}$ which satisfy the orthogonality condition shown in equation (2.7). The data stream is then split into multiple sub-bands with different sub-carrier frequencies as shown in Fig. 2.4. The spectrum is a sinc function $\text{sinc}(x) = \frac{\sin(x)}{x}$ because an OFDM symbol is multiplied with a rectangular FFT-window in the time domain and consequently a convolution of the discrete spectrum with the sinc function in the frequency domain. It is clear that in $T_{\text{SYM}}$ seconds $N_{\text{FFT}}$ symbols are sent, instead of 1, as in the single carrier system. The duration of the symbol is $N_{\text{FFT}}$ times longer. Therefore, the data rate is maintained with the advantage that the symbol is much longer. Thus choosing $N_{\text{FFT}}$ large enough, the symbol duration $T_{\text{SYM}}$ increases which helps avoiding ISI. It is true that the delayed path is still the same, yet the overlap is decreased and thus the effect of ISI is reduced to a large extent.

To achieve a complete ISI free transmission even for very large channel delay spreads, OFDM transmitters insert a Guard Interval (GI) (or sometimes called cyclic prefix) before each OFDM symbol as seen in Fig. 2.5. Some of the last samples of the time domain signal are copied and placed at the start of the OFDM symbol. The data rate is thereby reduced but the system is no more vulnerable to multipath channels in most cases. The GI could also be composed of zeros or another arbitrary signal, but the duplication from the end of the OFDM symbol preserves the periodicity and simplifies the time synchronization.
2.3 The IEEE802.11a Standard

IEEE802.11a is a Wireless Local Area Network (WLAN) standard [IEEE99] which has been approved by the Institute of Electrical and Electronic Engineers (IEEE). It supports speeds up to 54 Mbps and uses OFDM as the modulation technique. It operates in the 5.8 GHz spectrum (frequency band) which leads to a high absorption rate and consequently the signal is more blocked by walls than signals in the 2.4 GHz frequency band such as IEEE802.11b/g. The total bandwidth of the IEEE802.11a is 20 MHz with an occupied bandwidth of 16.6 MHz. Table 2.1 summarizes the most important system parameters for IEEE802.11a.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Frequency</td>
<td>$1/T_S$</td>
<td>20 MHz</td>
</tr>
<tr>
<td>System clock</td>
<td>$T_S$</td>
<td>0.05 μs (1 sample)</td>
</tr>
<tr>
<td>FFT Length</td>
<td>$N_{FFT}$</td>
<td>64</td>
</tr>
<tr>
<td>FFT period</td>
<td>$T_{FFT} = T_S \cdot N_{FFT}$</td>
<td>3.2 μs</td>
</tr>
<tr>
<td>Sub-carrier frequency spacing</td>
<td>$\Delta_F = 1/T_{FFT}$</td>
<td>312.5 kHz</td>
</tr>
<tr>
<td>Number of data sub-carriers</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>Number of pilot sub-carriers</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>GI duration</td>
<td></td>
<td>0.8 μs (16 samples)</td>
</tr>
<tr>
<td>OFDM symbol duration</td>
<td></td>
<td>4 μs (80 samples)</td>
</tr>
</tbody>
</table>

Table 2.1: IEEE802.11a system parameters.

Figure 2.6 shows a complete IEEE802.11a OFDM burst. It consists of a preamble, a signal frame and several data frames. The preamble is composed of the short training sequence and the long training sequence for an initial synchronization (acquisition). The short training sequences ($t_1, t_2, \ldots, t_{10}$) are formed of a repetition of 10 identical pseudo-noise training symbols each with 16 samples, that is a total length of 160 samples or 8 μs. 12 sub-carriers of the short training sequence are occupied. The short training sequence can be used for Automatic Gain Control (AGC) adjustment and symbol detection, but their main task is timing and coarse frequency synchronization. The long training sequence is composed of a Guard Interval (GI) with 32 samples followed by 2
identical pseudo-noise symbols \((T1, T2)\) each having a duration of 3.2 \(\mu s\). This training sequence has 52 occupied sub-carriers. It is commonly used for channel estimation and fine synchronization. The signal frame is formed of 64 samples and a GI with 16 samples. It contains information about the data rate and the length of the following data frames. The information from the MAC layer is contained in the data frame which has the same length and GI as the signal frame.

The IEEE802.11a transmitter uses a scrambler which scrambles the data field with a generator polynomial \(S(x) = x^7 + x^4 + 1\). Then the data is encoded with a convolutional code with different code rates depending on the desired data rate in the signal frame for the purpose of Forward Error Correction (FEC). The code rates are shown in Table 2.2. An Interleaver follows the encoder to separate the adjacent bits. This is done in 2 steps. A first permutation which ensures that the adjacent bits are mapped into non-adjacent sub-carriers while the second permutation ensures that the adjacent bits are mapped alternately into less but more significant bits of the constellation when 16 QAM or 64 QAM modulation is used.

As shown in Table 2.2, the OFDM sub-carriers are modulated by different modulation techniques depending on the data rate of the signal frame. In modulation, all constellations are normalized to obtain the same signal power with each modulation. A step of inserting 4 pilots onto the sub-carriers \(-21, -7, +7, +21\) follows in order to help tracking the phase at the receiver [IEEE99]. An Inverse FFT (IFFT) with length 64 transforms this signal into the time domain and finally the last 16 samples are copied to the beginning of the frame forming the GI which is also known as the Cyclic Prefix (CP).

At the receiver side, a pre-FFT synchronization (including timing and frequency synchronization) is done before removing the GI. Then the GI is removed and an FFT transforms the signal into the frequency domain where a post-FFT synchronization (including channel estimation and another fine frequency synchronization) is applied.

![Figure 2.6: A complete OFDM burst.](image)
Table 2.2: Data rates and corresponding modulation.

<table>
<thead>
<tr>
<th>Data rate (Mbits/s)</th>
<th>Modulation</th>
<th>Code rate (R)</th>
<th>Coded bits per sub-carrier</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>BPSK</td>
<td>1/2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>BPSK</td>
<td>3/4</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>QPSK</td>
<td>1/2</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>QPSK</td>
<td>3/4</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>16-QAM</td>
<td>1/2</td>
<td>4</td>
</tr>
<tr>
<td>36</td>
<td>16-QAM</td>
<td>3/4</td>
<td>4</td>
</tr>
<tr>
<td>48</td>
<td>64-QAM</td>
<td>2/3</td>
<td>6</td>
</tr>
<tr>
<td>54</td>
<td>64-QAM</td>
<td>3/4</td>
<td>6</td>
</tr>
</tbody>
</table>

After that, demodulation, de-interleaving and decoding are applied to the signal. The same scrambler as in the transmitter is then performed on the signal which then continues to the MAC layer.

2.4 The DVB-T Standard

Another example of OFDM systems is the DVB-T (Digital Video Broadcasting-Terrestrial). This section explains the basic structure of a DVB-T transmitter, and presents an overview of the system parameters. The basic structure of the DVB transmitter as given in [DVB01] is shown in Fig. 2.7.

The input data which are MPEG-2 compliant MUX packets, are randomized using a scrambler to achieve a dispersion of the energy. The scrambler polynomial is $1 + x^{14} + x^{15}$. After the energy dispersal, a Forward Error Correction (FEC) stage follows, which is divided into 2 parts. First, the outer coding is performed, using a Reed-Solomon RS (255, 239, t=8) code, which is followed by a byte-wise interleaving. Then, the standard defines an inner coding stage, which consists of a convolutional encoding of the protected packets that come out of the RS encoding stage. This convolutional coding is based on a mother code of rate 1/2, which can then be punctured to obtain a code rate of 2/3, 3/4, 5/6 or 7/8. The convolutional encoding is followed by a bit-wise interleaving and then by a symbol interleaving stage. Since the concentration of this work is on
synchronization the above-mentioned stages can be omitted.

Then, the bit stream is mapped into a symbol stream, using one of the three possible signal constellations specified in the standard: QPSK, 16-QAM and 64-QAM. All of these constellations follow a Gray mapping. The specific mappings are defined in [DVB01].

Later on, the signal has to be adapted to the DVB-T frame structure. Each OFDM super-frame is composed of 4 frames, and every frame is composed of 68 OFDM symbols. The length of these symbols is equal to the number of points of the IFFT. This number depends on the selected transmission mode, which can be 2k or 8k. Each OFDM symbol carries the information corresponding to $K_u$ useful data carriers. Some reference carriers are inserted in every symbol, these consist of 2 types:

- **TPS carriers**: these carriers convey the information corresponding to the transmission parameters, such as the transmission mode, the code rate, the signal constellation chosen or the length of the GI.

- **Pilot carriers**: these carriers are inserted following a Pseudo Random Binary Sequence (PRBS) which is also known to the receiver, and are normally used for channel estimation. However, they can also be used to perform some of the synchronization stages.

The TPS carriers are inserted in fixed positions over every OFDM symbol [DVB01]. There are $K_{TPS}$ carriers per OFDM symbol, and all of them conduct the same information. Each of these $K_{TPS}$ carriers is DBPSK modulated, according to the value of the same carrier in the previous symbol and to a 68-bit $TPS$ word which depends on the transmission parameters [DVB01].

We differentiate between two types of pilot carriers:

- **Continual pilots**: found in the same positions for all the symbols.

- **Scattered pilots**: found in different positions from symbol to symbol.

This can be seen in Fig. 2.8.

The pilot carriers are modulated according to a PRBS which is initialized for every new OFDM symbol. This sequence generates a new value no matter if the carrier is a pilot carrier or not.

The TPS carriers, as well as, the data carriers are transmitted at a normalized power, whereas the pilot carriers are transmitted at a boosted power level.

The carriers which are either reference or data carriers are termed as active. The number of active carriers, $K$, stays constant from symbol to symbol. Once the reference carriers have been inserted, the IFFT is performed to transform the carriers in the frequency domain to samples in the time domain. Last, after the IFFT is performed, the GI has
to be added, as it has been explained previously. The length of the GI is defined as a parameter in the DVB-T standard, and can take one of the following values (normalized by the length of the IFFT): 1/4, 1/8, 1/16 or 1/32.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Name</th>
<th>8k mode Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Frequency</td>
<td>$f_c$</td>
<td>$474 + i.8$ MHz, $i = 0, 1, 2, \ldots$</td>
</tr>
<tr>
<td>System Frequency</td>
<td>$f_s$</td>
<td>64/7 MHz</td>
</tr>
<tr>
<td>FFT Length</td>
<td>$N_{FFT}$</td>
<td>8192</td>
</tr>
<tr>
<td>GI Length</td>
<td>$L_g$</td>
<td>256, 512, 1024, 2048</td>
</tr>
<tr>
<td>GI duration</td>
<td>$T_g$</td>
<td>28, 56, 112, 224μs</td>
</tr>
<tr>
<td>Useful OFDM symbol duration</td>
<td>$T_u$</td>
<td>896μs</td>
</tr>
<tr>
<td>OFDM symbol duration</td>
<td>$T_{SYM}$</td>
<td>924, 952, 1008, 1120μs</td>
</tr>
<tr>
<td>Number of useful carriers</td>
<td>$K_u$</td>
<td>6048</td>
</tr>
<tr>
<td>Number of active carriers</td>
<td>$K$</td>
<td>6817</td>
</tr>
<tr>
<td>Sub-carrier spacing</td>
<td>1/$T_u$</td>
<td>1116</td>
</tr>
</tbody>
</table>

Table 2.3: DVB-T system parameters.

Table 2.3 shows the most important parameters of a DVB-T system.

## 2.5 Imperfections of OFDM Systems

OFDM transmission suffers from some imperfections that have to be corrected at the receiver side. These imperfections or problems facing OFDM systems include: Time offset and carrier frequency offset which result in Intercarrier Interference (ICI) and ISI and these are corrected in the pre-FFT synchronization stage as explained before. In addition to these, OFDM transmissions suffer from phase offset whose correction is done in the post-FFT synchronization stage. Another synchronization error in OFDM systems...
is the sampling frequency offset. These errors will be presented in the following sections; however, this thesis is mainly concerned with time offset and carrier frequency offset which will looked at and analyzed in deep details throughout this work.

### 2.5.1 Time Offset

An OFDM frame includes several symbols and each consists of a certain number of samples depending on the OFDM application. The goal of the receiver is to find the beginning of the OFDM symbol. The basic problem is that the receiver does not know when the transmitter sends a new burst (for example in IEEE802.11a WLAN) because the transmitted signal $s[n]$ reaches the receiver delayed by $\tau$ in time. This timing offset is normalized to the system clock $T_s$, producing the normalized timing offset $\theta$ which has no unit and which is equal to the number of samples between the transmitted signal $s[n]$ and the received signal $s[n - \theta]$. In this work, we estimate integer values of $\theta$, even though it is possible that $\theta$ takes fractional values. Estimating a fractional $\theta$ could be done by upsampling doing an estimation step and then downsampling. However, this is beyond the scope of this work and therefore we consider $\theta$ to be an integer and thus the estimation resolution is 1 sample.

If two OFDM symbols are transmitted, one preceding the other, the receiver must pick the correct samples of the actual OFDM symbol and give it to the FFT so the FFT window should be correctly placed. If there is no synchronization, the beginning of the FFT window will be placed too early or too late. If placed too late for example, some samples will be collected from the actual OFDM symbol and the others will be collected from the following OFDM symbol; if no following OFDM symbol exists, the last samples are filled up with noise. If placed too early, samples from the previous OFDM symbol will be collected. Both cases result in ISI. In addition to this, ICI results also from an FFT window mis-alignment because the periodic signal does not anymore fit completely in the FFT window. To point out the effect of ICI, we suppose that the FFT window takes some samples from the actual OFDM symbol and the last samples are filled with zeros. Such a time sequence has no longer definite frequency, so the discrete spectrum will not have single peaks anymore but sidelobes occur as seen in Fig. 2.9. Consequently, the orthogonality is lost due to ICI.

However, when the time offset is within the GI, no ISI and ICI occurs (assuming no doppler effect), only a phase offset $\varphi$ which is proportional to the time shift occurs. If the DFT is applied to the cyclically shifted time sequence $s[n - \theta]$, the result is:

\[
DFT\{s[n - \theta]\} = S[k]e^{-j\frac{2\pi k}{NFFT}\theta}
\]
This shows that the offset has the formula:

$$\varphi = -\frac{2\pi k}{N_{FFT}} \theta$$  \hspace{1cm} (2.11)

which is not constant for all sub-carriers.

Taking the IEEE802.11a as an example, the coarse timing offset has to be corrected in an accuracy of 1 sample which is equal to 50 ns, thus a phase offset occurs with a very high probability, but no information is lost because the phase can be corrected by a post-FFT synchronization step and the embedded pilots that are placed on the sub-carriers $-21, -7, +7, +21$ in the data frame can be used. After time offset estimation, the residual error $|\hat{\theta} - \theta|$ should produce a phase offset which can be corrected and not larger, since in the post-FFT synchronization, a phase offset within a certain range only can be detected. As a result, the residual time offset is restricted and we have a maximum tolerable residual time offset as seen in [KH00]: $|\hat{\theta} - \theta| \leq \frac{N_{FFT}}{2 \Delta p} = 2.28$, where $\Delta p = 14$ which is the distance between the pilot carriers. Therefore, the acceptable range $|\hat{\theta} - \theta| \leq 2$ is used as a rule of thumb for IEEE802.11a in this work.

More about that will be seen in the separate chapters dealing with IEEE802.11a and DVB-T.

### 2.5.2 Carrier Frequency Offset

Multi-carrier systems are very sensitive to carrier frequency offset because the orthogonality between the sub-carriers is lost. Reasons for the carrier frequency offset include
the Doppler effect which occurs in mobile applications and the difference between the transmitter and the receiver oscillators. The Doppler effect depends on the application taking into account the velocity of the mobile. In an OFDM system, to transmit a signal over a long distance, it has to be converted up to a higher frequency. This is done by multiplying \( s(t) \) by \( e^{j2\pi f_C t} \) where \( f_C \) is the oscillator frequency of the transmitter. At the receiver side, the signal is shifted down to base band with the oscillator frequency \( f'_C \), that is by multiplying the signal by \( e^{-j2\pi f'_C t} \). This results in the frequency offset \( \Delta f_C \):

\[
\Delta f_C = f_C - f'_C
\]  

(2.12)

According to the IEEE802.11a standard [IEEE99], the relative deviation of the transmitted center frequency from the nominal frequency must not exceed \( \pm 20 \text{ ppm} \). If the same applies to the receiver, then the maximum carrier frequency offset is \( \pm 40 \text{ ppm} \) or \( \Delta f_{C,\text{max}} = \pm 232 \text{ kHz} \) with a carrier frequency of 5.8 GHz. Fig. 2.10 shows the maximum allowed frequency offset related to the sub-carriers spacing of \( \Delta F = 312.5 \text{ kHz} \). For simplification, the frequency offset is normalized to the sub-carrier spacing. We call

\[
\varepsilon = \frac{\Delta f_C}{\Delta F}
\]

(2.13)

the normalized frequency offset. The maximum frequency lock-in range for IEEE802.11a is illustrated in Fig. 2.10.

The carrier frequency offset has two negative effects: Time-variant phase offset common to all sub-carriers and ICI. Neglecting any channel distortion (\( h(t) = \delta(t) \)), the received signal has the ideal form:

\[
 r(t) = h(t) * s(t).e^{j2\pi.\Delta f_C t}
\]

(2.14)

where \( * \) denotes the convolution. This gives in the discrete time domain:

\[
 r[n] = h[n] * s[n].e^{j\frac{2\pi.n}{NFFT}.\varepsilon}
\]

(2.15)
In case of distortion free channel it is obvious that the received spectrum is a shifted version of the transmitted one. Equation 2.15 shows the phase offset

\[ \varphi = \frac{2\pi n}{N_{\text{FFT}}} \varepsilon \]  

(2.16)

In contrast to a time offset, the phase offset is equal to all sub-carriers but increases in time. This phase offset is corrected in the post-FFT synchronization stage.

The reason of ICI in the case of frequency offset is different from that in the case of time offset. With a time offset, the shape of the spectrum is changed, but in the case of carrier frequency offset, the spectrum is only shifted and the sub-carriers are still mutually orthogonal. In Fig. 2.11, the spectrum is shifted about \( \varepsilon = 0.2 \) to the right due to the carrier frequency offset. The receiver takes the value at the normal frequency left of the received maximum. The decreasing amplitude is negligible up to \( \varepsilon < 0.1 \) but the influence from the neighbor sub-carriers is already significant.

It is worth to note that the rule of thumb for estimating frequency in this work depends on SNR degradation which is caused by frequency offsets. The residual frequency offset should cause only a negligible SNR degradation. This will be discussed further in details for each system.

### 2.5.3 Sampling Frequency Offset

As mentioned before, sampling frequency offset is not the concern of this thesis. It is mentioned here just for completion. In the IEEE802.11a standard, it is written that the
system clock \( T_S \) and the transmit frequency \( f_C \) should be derived from the same reference oscillator. This oscillator has an offset from the receiver oscillator; this causes the ADC in the receiver to sample at a wrong sampling frequency. This results in time-varying phase offset and ICI. According to [NA99], this effect is negligible due to the short duration of a burst.

2.5.4 Phase Offset

The phase offset is also mentioned here only for completion and is not a main point in this work. An IEEE802.11a WLAN uses the absolute phase information to demodulate the received signal in contrary to the differential modulation schemes like DPSK where only the phase difference is evaluated. These differential systems have some negative effects like a loss in SNR of about 3 dB [NA99] and the disability to use complex multi-amplitude constellations such as 64-QAM because they have difficulties in detecting amplitude shifts. An IEEE802.11a receiver has to track the phase continuously and that’s why pilots are used. As discussed before, phase offsets occur due to many reasons. These reasons include carrier phase offset, phase noise, channel impulse response, residual frequency offset, residual time offset.

As mentioned before, the difference between transmitter and receiver oscillators cause not only frequency offset but also phase offset. This is what’s known as carrier phase offset. Phase noise has similar reasons as the carrier frequency offset. It is found in every analog oscillator and results from the imperfections of the analog components. This phase noise causes not only phase offset but also plays a role in causing carrier frequency offset and ICI. The channel impulse response is one reason of the phase offset. It has to be corrected by channel estimation.

After time and frequency synchronization, a residual time offset and a residual frequency offset still exist and they will cause ISI, ICI as well as a phase offset. In case of frequency offset, this phase offset is constant to all sub-carriers but depends on time, while the phase offset resulting from time offset is time-invariant but not constant in frequency as mentioned earlier.
2. Overview of OFDM Systems
3 Synchronization Principles

This chapter gives an overview of the most common synchronization methods used in Literature. In the following chapters, we will concentrate on implementation aspects such as computational complexity and design area (gates) in order to optimize the state-of-the-art algorithms according to special parameters we will later discuss. The later optimization will lead us to a fixed-point implementation and a deep analysis of the hardware complexity and cost will be done.

Back to the state of the art, this chapter concentrates on the synchronization techniques for OFDM, namely IEEE802.11a and DVB-T. Only digital methods are taken into account as the work is concerned with the digital part. As IEEE802.11a is a burst mode transmission, the synchronization has to be done as fast as possible. In continuous mode transmissions like in DVB-T, synchronization time is not as critical as in the burst-mode. This is because it is possible to do an averaging over some OFDM symbols.

When speaking of synchronization, most algorithms are based on two important principles: Autocorrelation and crosscorrelation. In the autocorrelation process, the signal is correlated with itself, whereas in the crosscorrelation, the signal is correlated with a stored pattern known to the receiver.

Fig. 3.1 shows a basic transmission model. The signal is sent from the transmitter, propagates through the channel and arrives later at the receiver. The signal undergoes disturbances depending on the channel it propagates in. The delay and noise which corrupt the signal while it propagates through the channel (example multi-path propagation) should later be corrected and eliminated at the receiver end. A major part of that process is the time and frequency offset synchronization.

3.1 IEEE802.11a

In this section, the synchronization methods of IEEE802.11a will be presented. The two mentioned concepts: Autocorrelation as well as crosscorrelation are shown in Fig. 3.2. The Autocorrelation can be either performed on the GI or on the training symbols. Both
concepts will be hereby explained in detail.

<table>
<thead>
<tr>
<th>Auto correlation</th>
<th>Cross correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guard Interval</td>
<td>Training Symbols</td>
</tr>
<tr>
<td>Training Symbols</td>
<td>Training Symbols</td>
</tr>
<tr>
<td>Training Symbols (special correlation signals)</td>
<td>Training Symbols</td>
</tr>
</tbody>
</table>

![Figure 3.2: Autocorrelation and crosscorrelation methods.](image)

### 3.1.1 Autocorrelation Method

In statistics, the autocorrelation function of a stochastic process is used for describing the similarity of the process at different points in time. In signal processing, the autocorrelation function is used for analysing functions or series of values, such as time domain signals. It is a measure of how well a signal matches a time-shifted version of itself. First of all, let’s define the mathematics of the autocorrelation.

\[
c[d] = \frac{1}{l} \sum_{i=0}^{l-d-1} x^*[i] \cdot x[i+d]
\]  

(3.1)

where \(x[n]\) is a complex-valued sequence of length \(l\). \(c[d]\) is defined as the estimation of the autocorrelation function. In our application, we are interested in the maximum of the autocorrelation function, and therefore we neglect the normalizing term \(\frac{1}{l}\). Thus, the equation becomes:

\[
c[d] = \sum_{i=0}^{l-d-1} x^*[i] \cdot x[i+d]
\]  

(3.2)
In the equation is the shifting variable, and thus $c[d]$ gets its maximum at $d = 0$. In our application, we are only interested in some values of $d$ where $c[d]$ can get its maximum. In each application the shifting variable $d$ gets a constant value $N$. Thus (3.2) becomes.

$$c[N] = \sum_{i=0}^{l-N-1} x^*[i] \cdot x[i + N]$$  \hspace{1cm} (3.3)

However, $c[d]$ is time-dependent and we search for the time instant where $c[d]$ is maximum, therefore we introduce the discrete time index $n$ and (3.3) then reads:

$$c[n, N] = \sum_{i=0}^{l-N-1} x^*[n + i] \cdot x[n + i + N]$$  \hspace{1cm} (3.4)

Let $L = l - N$, then (3.4) becomes:

$$c[n, N] = \sum_{i=0}^{L-1} x^*[n + i] \cdot x[n + i + N]$$  \hspace{1cm} (3.5)

In the following, (3.5) will represent the autocorrelation equation used in simulation and implementation. For the aim of coarse time synchronization of IEEE802.11a, the autocorrelation of the training symbols of the received signal is performed. It is important to note that autocorrelation methods can only be applied if the noise-free signal is periodic and the period is an integer multiple of the sampling interval. This is valid for training symbols of the IEEE802.11a [IEEE99].

Thus applying the above derivation, the autocorrelation equation becomes:

$$c[n, N_{PER}] = \sum_{i=0}^{N_{TRAIN}-1} r^*[n + i] r[n + i + N_{PER}]$$  \hspace{1cm} (3.6)

$r[n]$ is the received time-domain signal. The term $N_{PER}$ is called the periodicity factor. In the case of the short training symbols, this periodicity factor is equal to 16, whereas it is 64 in the case of long training symbols. $N_{TRAIN}$ is the term called training symbol length and it is equal to the periodicity factor or an integer multiple of it. In the case of short training symbols, $N_{PER} + N_{TRAIN} = 160$. Chapter 4 discusses the settings of these two terms. In the following, a mathematical justification is presented of why the autocorrelation method is widely used. Throughout the derivation, $N$ and $L$ (instead of $N_{PER}$ and $N_{TRAIN}$) will be used since the autocorrelation method is not only valid for training symbols of IEEE802.11a. This will be the case for the autocorrelation of the GI in DVB-T. It is to be noted that noise and multi-path are not taken into account in the mathematical derivation below.

As mentioned earlier, the signal at the receiver end suffers from a frequency and an initial
phase offset. The correlation signal can be written as:

$$c[n, N] = \sum_{i=0}^{L-1} |r^*[n + i]||e^{-j2\pi N_{FFT}\epsilon} r[n + i + N]||e^{j2\pi(n+i+N)/N_{FFT}} + j\varphi$$  (3.7)

It follows that:

$$c[n, N] = e^{j2\pi N_{FFT}\epsilon} \sum_{i=0}^{L-1} |r[n + i]|$$  (3.8)

For the sequences that satisfy the condition:

$$|r[n + i]| = |r[n + i + N]| \quad \forall i = 0, \ldots, L - 1$$  (3.9)

we can write:

$$c[n, N] = e^{j2\pi N_{FFT}\epsilon} \sum_{i=0}^{L-1} |r[n + i]|^2$$  (3.10)

In the case of IEEE802.11a ($N = N_{PER}$ and $L = N_{TRAIN}$), the condition (3.9) is satisfied when $n = \theta$. If we focus the attention at the magnitude of the correlation signal, we can see that at $n = \theta$, all the terms of the summation add in phase. As a result, the maximum of the correlation is achieved exactly at the time offset.

At the estimated timing offset $\hat{\theta}$, (3.10) becomes:

$$c[\hat{\theta}, N] = e^{j2\pi N_{FFT}\epsilon} \sum_{i=0}^{L-1} |r[\hat{\theta} + i]|^2$$  (3.11)

After calculating the argument of (3.11), we can obtain the normalized frequency offset by:

$$|\epsilon| = \frac{N_{FFT}}{2\pi N_{PER}} \arg c[\hat{\theta}, N]$$  (3.12)

The maximum lock-in range is $|\epsilon| \leq \frac{N_{FFT}}{2N_{PER}}$. Since $N_{PER} = 16$ for short training symbols and $N_{FFT} = 64$, then the maximum normalized frequency offset is $|\epsilon| \leq 2$ while for the long training symbols, $N_{PER} = N_{FFT} = 64$ and thus $|\epsilon| \leq 1/2$.

Autocorrelation of the GI was discussed in [Bvd98] for multi-path channels. The GI must be widely unconsumed (ISI free) in fading channels to get a reliable synchronization. But this is difficult because the main role of the GI is to avoid ISI. An accurate estimation with the GI requires averaging over a large number of OFDM symbols. This is possible in continuous mode transmission such as DVB-T but not suitable for burst mode transmission. On the other hand, the IEEE802.11a standard specifies the training symbols (short and long) to perform the synchronization. Autocorrelation with training symbols has some advantages: Compared to GI method, the training symbols method gives a faster and a more accurate estimate. In addition to
that, there are more synchronization methods in digital systems but they are not very convenient for IEEE802.11a. Such methods include post-FFT timing synchronization which is not very applicable for burst mode transmission because an FFT causes a long delay and in burst mode transmission, synchronization has to be done fast.

### 3.1.2 Crosscorrelation Method

Crosscorrelation is a method by which we correlate a preamble stored in the receiver with the received data stream. Let $N_{\text{TRAIN}}$ denote the length of the training symbol, and $s[i]$ for $i = 0, \ldots, N_{\text{TRAIN}}$ be the samples of the ideal preamble. Then, the output of the crosscorrelator can be written as:

$$x[n] = \sum_{i=0}^{N_{\text{TRAIN}}-1} s^*[i]r[n+i]$$  \hspace{1cm} (3.13)

The stored signal pattern in the receiver is not affected by noise. This makes the crosscorrelation advantageous to autocorrelation in AWGN channels. The receiver works as a matched filter. Fig. 3.3 shows the output of correlating the preamble with a matched filter $N_{\text{TRAIN}} = 16$. The ten short training symbols generate the 10 peaks each arising at the end of every short training symbol (no noise).

![Crosscorrelation Diagram](image)

**Figure 3.3:** 10 peaks arising from the crosscorrelation in IEEE802.11a.

This method has several disadvantages. First of all, the crosscorrelator has a high complexity and no iterative calculation can be performed. This will be discussed deeper in chapter 4. In addition, the difference between the stored preamble and the received one can be very large due to multi-path propagation, noise, and non-linear components.
in the transmitter. Consequently, the output of the correlator will no longer be a peak. If the undistorted received signal \( r[n+i] \) is ideally correlated with the preamble in the receiver, we get \( x[n] = r[n] * s^*[-n] \). But in reality, the channel which suffers from multi-path propagation with the channel impulse response \( h[n] \) affects the received signal. The received signal suffers from a frequency offset and has the form:

\[
r[n] = h[n] * s[n]e^{j2\pi nNFFT\varepsilon}
\]

(3.14)

Assuming a distortion free channel,

\[
h[n] = \begin{cases} 
e^{j\phi} & \text{for } n = 0; \\ 0 & \text{else.}\end{cases}
\]

After (3.13), \( x[n] \) will be:

\[
x[n] = e^{j\phi} \sum_{i=0}^{N_{\text{TRAIN}}-1} \left|s[i]\right|\left|s[n+i]\right|e^{j2\pi\varepsilon nNFFT}
\]

(3.15)

If the condition

\[
\left|s[i]\right| = \left|s[n+i]\right| \quad \forall i = 0, \ldots, N_{\text{TRAIN}} - 1
\]

(3.16)

holds then, (3.15) becomes

\[
x[n] = e^{j\phi} \sum_{i=0}^{N_{\text{TRAIN}}-1} \left|s[n+i]\right|^2e^{j2\pi\varepsilon nNFFT}
\]

(3.17)

The condition (3.16) is satisfied when \( n = \theta \). If the correct time offset \( \hat{\theta} = \theta \) is found and \( n \) is substituted in (3.17) by \( \theta \), we will still get an exponential function depending on \( i \) inside the sum. It is difficult especially for non constant envelope signals to directly calculate the frequency offset from the crosscorrelation.

The autocorrelation can be computed recursively which makes it far less complex than crosscorrelation. A disadvantage of using the autocorrelation is the overhead due to inserting the training symbols which reduces the bandwidth efficiency. However, as seen in the IEEE802.11a standard, these training symbols are already present in the IEEE802.11a signal and thus can be used [IEEE99]. Further discussion on the choice of the correlator and the suitable parameters is provided in chapter 4.

3.1.3 Synchronization Procedure

In the pre-FFT synchronization, two estimators will be discussed: Frequency synchronization which includes both coarse and fine, and time synchronization. As discussed before, the short training symbols are used for coarse frequency synchronization with lock-in range \( |\varepsilon| \leq 2 \). Then the long training symbols are used for a fine synchronization
step which corrects the remaining offset at most to $|\varepsilon| \leq 1/2$. The time synchronization estimates the normalized time offset $\theta$ with an accuracy of one sample. A fine time synchronization is necessary if the coarse synchronization is not able to obtain a constraint of the maximum tolerable time offset $|\theta| \leq 2$. Synchronization can be divided into three steps: Detection, acquisition and tracking. The detection is necessary for power saving if the receiver can switch to IDLE when no signal is present. This work does not consider detection. Recognition of a new burst is done within the coarse synchronization unit. The frequency and the time offset errors as well as the unknown channel impulse response are errors that have to be estimated initially in the acquisition stage. In the tracking phase, some small deviations are corrected. This is mainly the phase shift that results from the channel impulse response and the residual synchronization errors and the tracking of the phase offset is done with the pilots as mentioned before.

An overview of the basic principles of the pre-FFT synchronization algorithm treated in this work is presented in Fig. 3.4. An autocorrelation of the received signal using short training symbols is done and the correlator signal will have a triangular peak-like shape which is essential to obtain a time offset estimation. In order to get a clear maximum, the parameters of the autocorrelation $N_{PER}$ and $N_{TRAIN}$ should be correctly chosen. This will be further discussed in chapter 4.

The correlator output is then fed into a Metric block which produces a metric signal $m[n]$ depending on the specific metric used. The metrics treated in this work are the MC (Maximum Correlation), MNC (Maximum Normalized Correlation), ML (Maximum Likelihood) and MMSE (Minimum Mean Squared Error). In-depth study of these metric and further optimizations are discussed later in the thesis (chapters 4 and 5). The metric signal is robust against distortions and SNR variations. The maximum of the metric signal provides the time offset. At this point, finding the arg of the correlated signal will result in the angle signal and consequently an estimation for the coarse frequency offset. This is done by averaging over several samples to get a better accuracy. For the fine synchronization, a similar process is performed except that the autocorrelation is calculated using the long training symbols. The metric signal is not evaluated because the time offset has been already calculated using the short training sequence. The fine frequency offset is evaluated using the correlator signal.

### 3.2 DVB-T

The aim of this section is to explain the principles used for synchronization in DVB-T systems. Generally, they are also based on autocorrelation and crosscorrelation concepts.
3. Synchronization Principles

The mathematical principle of the autocorrelation which was shown in section 3.1 is also applicable in the case of DVB-T. Nevertheless, the application of the autocorrelation method is done on the Guard interval of the DVB-T signal. Thus, $N = N_{FFT}$ and $L = L_g$ where $L_g$ is the length of the guard interval (also known as cyclic prefix). As a result of this, the autocorrelation equation reads:

$$c[n] = \sum_{i=0}^{L_g-1} r^*[n+i]r[n+i+N_{FFT}]$$  \hspace{1cm} (3.18)

It takes $L_g+N_{FFT}$ samples from the received signal and correlates the first $L_g$ samples with the $L_g$ samples located $N_{FFT}$ samples apart. The condition (3.9) is met when the correlation window is aligned with the guard interval as seen in Fig. 3.5. Therefore, the
maximum of the correlation is achieved at $n = \theta$. ($\theta$ being the time offset).

Similar as in the IEEE802.11a, the autocorrelation can be used for the frequency estimation by calculating the argument of the correlation at the time offset:

$$\varepsilon = \frac{1}{2\pi} \arg c[\hat{\theta}] \quad (3.19)$$

For the estimation of $\varepsilon$ to be unambiguous, $|\arg c[\hat{\theta}]|$ has to be less than $\pi$. So the maximum allowed value of $|\varepsilon|$ should be less than $1/2$. Thus the autocorrelation of the GI is only suitable to estimate the fractional part of the frequency offset. The integer part is estimated by two post-FFT methods. The first one is an autocorrelation of 2 consecutive symbols and the second is an autocorrelation of the pilots of 1 symbol. These pilots are defined in section 3.2.2.

The different metrics using the autocorrelation signal will be shown in chapter 5.

### 3.2.2 Crosscorrelation of Frequency-Domain Pilot Symbols

The DVB-T standard provides some pilot carriers which can be used for the purpose of time as well as frequency synchronization. This is different to the data-aided (DA) methods because these pilots do not form a whole symbol in the time domain. They are frequency domain pilots inserted in some of the carriers of every symbol while useful data is inserted in the rest of the carriers. After the IFFT in the transmitter, all the samples will carry pilot and data content until the signal reaches the receiver and an FFT is done to separate the two contents. As a result, crosscorrelation instead of autocorrelation is used.

Denote by $S$ the set of all continual pilot carriers in one symbol. Then:
where $c_k$ is the frequency symbol transmitted on the kth sub-carrier. The sent signal $s[n] = d[n] + m[n]$ where $d[n]$ is the data component and $m[n]$ is the pilot component of $s[n]$.

For the synchronization, a crosscorrelation between $m[n]$ and the received signal is performed. The metrics used with this crosscorrelation are then presented in chapter 5.
4 IEEE802.11a Synchronization

This chapter describing in detail the IEEE802.11a synchronization procedure which was introduced in chapter 3. In addition, it deals with the different optimizations applied to the IEEE802.11a synchronization in order to decrease complexity on one side and keep a good performance on the other side. These optimization steps include: optimum correlator parameters, optimum correlator implementation, as well as using different estimations for calculating the correlator signal in order to decrease the complexity. These optimizations will be taken into account in finding the best trade-off for an IEEE802.11a synchronizer. IEEE802.11a synchronization includes coarse timing and frequency synchronization as well as fine frequency synchronization. The synchronization procedure is based on the autocorrelation calculation. In coarse time and frequency synchronization the autocorrelation of short training symbols is used, while for fine frequency synchronization the autocorrelation of long training symbols is used. Therefore, the parameters chosen for the autocorrelation calculation play a major role in the synchronization procedure. As discussed in chapter 3 the synchronization procedure requires the calculation of a metric signal to increase the estimator robustness against SNR variations. The choice of a suitable metric will be further discussed.

This chapter is organized as follows: First, the optimum correlator parameters are studied. After that, four possible metrics (MC, MNC, MMSE and ML) are studied in details and the most suitable metric for our application is chosen. The simulation model is then introduced and the floating point simulation results are shown. A section describing the fixed point implementation in SystemC follows. After discussing the floating and fixed point implementations of the synchronization algorithms a complexity analysis is done which is a base in the background for forming the CTP constellation of the IEEE802.11a synchronizer in chapter 6. Finally, a complexity optimization of the synchronization algorithm is studied which involves the use of a simplified correlator signal calculation.
4.1 Optimal Correlator Parameters

As seen in section 3.1.3, the autocorrelator parameters $N_{TRAIN}$ and $N_{PER}$ have to be chosen correctly to obtain a high robust synchronization and to cover the frequency lock-in-range. In this section the choice of these parameters is discussed in details. Section 3.1.1 describes the autocorrelation method used for IEEE802.11a synchronization. The autocorrelation equation applied on the IEEE802.11a is given by (3.6) with the two variables $N_{TRAIN}$ and $N_{PER}$. The estimated time offset $\hat{\theta}$, is the point where the autocorrelation reaches its maximum. According to (3.11), the maximum amplitude of the absolute value $|c[\hat{\theta}]|$ is

$$|c[\hat{\theta}]| = N_{TRAIN} \sigma_r^2$$

(4.1)

where $\sigma_r^2$ is the received power. Thus, proportional to the received signal power and training symbol length ($N_{TRAIN}$). Using (3.12), the maximum frequency lock-in-range will be given by

$$|\varepsilon| < \frac{N_{FFT}}{2N_{PER}}$$

(4.2)

Therefore, the parameter $N_{PER}$ determines the frequency lock-in-range. As seen in chapter 2, in IEEE802.11a a maximum frequency offset $= \pm 2$ could occur. Therefore, $N_{PER}$ should be chosen such that this maximum offset can be detected. In IEEE802.11a the frequency is estimated by two steps: the first step is by a coarse estimation covering the whole range, that is detecting offsets within $|\varepsilon| \leq 2$. The second step is by refining the estimated offset and estimating a residual offset within $|\varepsilon| \leq \frac{1}{2}$. It is worth to note that in DVB-T it is different. The frequency offset can take any value, and the frequency synchronization is done in two steps: one for estimating the fractional part of the frequency and the other for estimating the integer frequency. More details will be discussed in the following chapter.

If the autocorrelation is applied on the guard interval as seen in section 3.1.1, possible values for the correlator parameters are $N_{TRAIN} = 16$ and $N_{PER} = 64$. $N_{TRAIN} = 16$ is the maximum value possible for $N_{TRAIN}$, since the GI in an IEEE802.11a signal has a length of 16 samples and it is repeated after 64 samples [IEEE99]. Fig. 4.1 presents the autocorrelation on the GI using the maximum for $N_{TRAIN}$.

In this case 64 complex-valued delay elements are needed in the hardware implementation. Substituting the value of $N_{PER}$ in (4.2) will lead to frequency estimation in the lock-in-range $|\varepsilon| \leq \frac{1}{2}$. This case is helpful in detecting the fractional frequency offset in DVB-T, however it is not sufficient for coarse frequency estimation in IEEE802.11a. In the case of short training symbols we have many possibilities to choose the parameters. One of the possibilities is $N_{TRAIN} = 16$ and $N_{PER} = 16$, where the length of a short
4.1. Optimal Correlator Parameters

Figure 4.1: Autocorrelation parameters of the guard interval.

The training symbol is 16. This involves 16 complex-valued delay taps. The output signal which has a flat shape is represented with the solid line in Fig. 4.2. The maximum is obtained for 128 samples (8 short training symbols). This is because the correlator for these 8 short training symbols has the same 32 values in its delay line. Such a flat signal is not accurate for estimating the timing offset since the challenge is to find the end of the flat signal. However, in some publications [Ols] this method is used to find a primary timing offset estimation. Nandula in [NG03] proposes using the autocorrelation with $N_{TRAIN} = 16$ and $N_{PER} = 16$, supported by the crosscorrelation to help detecting the end of the flat part. However, by using additionally the crosscorrelation the computational complexity increases. An alternative of this is to use suitable parameter values such that the correlator does not contain the same values in the delay line. In this case the correlator output will be again a peaked like triangle and not a flat signal, which is more useful in detecting the timing offset.
To achieve this there are two possibilities for choosing the parameters. One possibility is to choose $N_{TRAIN} = N_{PER} = 80$, i.e., combining five short training symbols together to a large one and therefore increasing the training symbol length to 80 samples. In this case the correlator will need 180 complex-valued delay taps. The output of the correlator in this case is shown in Fig. 4.3.

![Correlator signal with $N_{TRAIN} = N_{PER} = 80$](image)

Figure 4.3: Correlator signal with $N_{TRAIN} = N_{PER} = 80$

The disadvantage in this case is the long periodicity factor $N_{PER}$. Since the value of $N_{PER}$ affects the frequency estimation range as seen in (3.19). Therefore, when using the short training symbols it is optimum to have a $N_{PER}$ not exceeding 16. This arises the third possibility where $N_{TRAIN} = 144$ and $N_{PER} = 16$. In this case the two correlation parts overlap but we still need 160 delay taps. Fig. 4.4 shows the three alternatives for the parameters for autocorrelation of short training symbols.

![Three possibilities for autocorrelation of short training symbols.](image)

Figure 4.4: Three possibilities for autocorrelation of short training symbols.

The correlator output for $N_{TRAIN} = 144$ and $N_{PER} = 16$ is shown in Fig. 4.5
4.2 Metrics

The IEEE802.11a synchronization procedure was discussed in section 3.1.3. The correlator signal is fed into a metric block to obtain a metric signal. This metric signal is only an optimized correlation which increases the robustness of synchronization. The metric block seen in Fig. 3.4 can be one of the following metrics.

- Maximum Correlation (MC)
• Maximum Normalized Correlation (MNC)
• Minimum Mean Squared Error (MMSE)
• Maximum Likelihood (ML)

A short overview on these metrics is given in the following section. In addition a comparison between the four criteria is done which leads to the choice of the optimal metric for IEEE802.11a synchronization. Actually the MC method has a low complexity because the output of the correlator is the output of the metric, but the other three metrics include further signal information and therefore have a higher complexity. These algorithms are examined as floating-point models and their performance is analyzed based on the acquisition failure probability $P(|\hat{\theta} - \theta| \leq m)$, which is the probability that the synchronizer fails for more than $m$ samples. As seen in section 2.5.1, the acceptable range for the timing offset is $|\hat{\theta} - \theta| \leq 2$, is used as a rule of thumb for IEEE802.11a in this work.

### 4.2.1 Maximum Correlation (MC)

The Maximum Correlation is given in [KP01]. It is a simple scheme defined as follows:

$$\hat{\theta}_{MC} = \arg \max_{n} (m_{MC}[n]) = \arg \max_{n} (|c[n]|)$$  \hspace{1cm} (4.3)

In (4.3) $|c[n]|^2$ can also be used instead of $|c[n]|$ since both $|c[n]|$ and $|c[n]|^2$ take maximum at the same argument. The complexity of these metrics and possible simplifications will be discussed later in this chapter. The correlator output has a wide range of values because it is not normalized to the input signal power. Therefore, it is difficult to find a fixed threshold for detecting a maximum. Fig. 4.7 shows the histogram of the estimated time offset at different SNR values using the MC metric. In addition the acquisition failure probability is shown for $m = 0, 1, 2$ in Fig. 4.8. These figures are given for a static AWGN channel. It is clear that as the SNR increases, the performance of the metric increases.

### 4.2.2 Maximum Normalized Correlation (MNC)

The MNC metric is an optimization of the MC metric, unlike the MC metric it takes into consideration the power of the input signal. Therefore it decreases the range of the output values, thus making it easier to find a threshold for detecting the maximum. The signal power is measured and used as normalization factor. The MNC was discussed by
4.2. Metrics

Figure 4.7: Histogram for the estimated timing offset using the MC metric.

Figure 4.8: Acquisition failure probability versus $SNR$ for the MC metric.

Schmidl in [ScC97]. The power of the signal is calculated by:

$$p[n] = \sum_{i=0}^{N_{POW} - 1} |r[n + i]|^2$$  \hspace{1cm} (4.4)$$

where $N_{POW}$ is the length of the window for power calculation. The normalization should be done such that the metric signal remains unity for ideal conditions (no additive noise, and no multipath). The total power of the signal is the sum of the received signal power and the power of noise. Therefore, the power sum seen in (4.4) is equal $\sigma_r^2 + \sigma_n^2$. Timing
estimation using the MNC metric is defined as follows:

\[ \hat{\theta}_{\text{MNC}} = \arg \max_n (m_{\text{MNC}}[n]) = \arg \max_n \left( \frac{|c[n]|}{|p[n]|} \right) \]  \hspace{1cm} (4.5)

The autocorrelation of a signal determines the power of the periodic part in the signal and for ideal conditions the noise is neglected. For a channel without additive noise, the ratio in (4.5) is unity.

To avoid unnecessary calculation, the ratio \( \frac{|c[n]|}{|p[n]|} \) can be replaced by \( \frac{|c[n]|^2}{|p[n]|^2} \) which saves the use of two square roots for the calculation of \( |c[n]| \) and \( |p[n]| \), this will be discussed in more details in the complexity analysis section. The number \( N_{\text{POW}} \) in (4.4) can be used equal to \( N_{\text{TRAIN}} \) and in this case the correlator and the power sum calculator use the same delay line. Definitely taking into account the power sum calculation increases the complexity of the metric. The block diagram of the MNC metric is shown in Fig. 4.9. As for the MC, the MNC metric performance is analyzed according to the acquisition

\[ \text{Figure 4.9: Block diagram of generating the MNC metric.} \]

failure probability. Fig. 4.10 shows the histogram of the MNC metric for different SNR values. Fig. 4.11 shows the acquisition failure probability of the time offset estimation as a function of SNR for \( m = 0, 1, 2 \).

These simulations are done for a static AWGN channel. As already discussed before, the autocorrelation considers only the repetitive part of the received signal whereas the power sum includes everything. So if the received signal contains white Gaussian noise then the SNR of the signal can be calculated on the basis of the MNC metric as follows:

\[ \text{SNR} = \frac{\sigma_r^2}{\sigma_n^2} \approx \frac{|c[\hat{\theta}]|}{p[\hat{\theta}] - |c[\hat{\theta}]|} = \frac{m_{\text{MNC}}[\hat{\theta}]}{1 - m_{\text{MNC}}[\hat{\theta}]} \]  \hspace{1cm} (4.6)

The calculated SNR is only valid at the found time offset \( n = \hat{\theta} \), this is useful for the Maximum Likelihood (ML) metric discussed later in this chapter.

### 4.2.3 Minimum Mean Squared Error (MMSE)

A third metric described in this chapter is the MMSE. In general, the MMSE is not suitable for burst mode transmission like the IEEE802.11a. But in here the MMSE metric is used
as an introduction to the ML metric. Using the MMSE metric signal, the time offset is estimated as follows:

\[ \hat{\theta}_{\text{MMSE}} = \arg\max_n \left( m_{\text{MMSE}}[n] \right) \]
\[ = \arg\max_n (2 \cdot |c[n]| - (p[n] + p[n + N_{\text{PER}}])). \]  

(4.7)  
(4.8)

Denote by \( \tilde{p}[n] = p[n] + p[n + N_{\text{PER}}] \). A block diagram for calculating the MMSE metric signal is shown in Fig. 4.12. Another version of the MMSE is used in [M00] where the metric signal defined in (4.8) is negated and the minimum of the metric output estimates
the timing offset. However, through out this thesis only searching the maximum will be used.

![Diagram](image)

Figure 4.12: Block diagram for generating the MMSE metric.

Since complexity is always a very important issue, a simplified less complex version of the MMSE was found where the power can be calculated by omitting the redundant calculation of $p[n+N_{PER}]$. This is possible because both signals overlap if the determined correlator parameters defined in Section 4.1 are used. The simplified MMSE is defined as follows:

$$\hat{\theta}_{MMSE-SIM} = \arg \max_n (m_{MMSE-SIM}[n]) = \arg \max_n (|c[n]| - p[n])$$  \hspace{1cm} (4.9)

Here the square roots for the calculation of $|c[n]|$ and $p[n]$ can not be saved as in the case of MNC. In contrast to all other metrics, the MMSE output is negative since the correlator signal contains a negligible noise part and therefore it is smaller than the power signal. When no signal is received the metric signal increases, since in this case the correlator output and the power signal are both very low and the difference is marginal. As a result the MMSE suffers from false alarms. This makes the MMSE not suitable for burst transmission. No simulations of the MMSE are done in this thesis.

### 4.2.4 Maximum Likelihood (ML)

The ML estimator is a well known estimator in Literature. It is used for a variety of applications one of which is the timing estimation in OFDM systems. As the name indicates the ML estimator intends to get the parameter with the maximum likelihood. Beek in his paper [BS96], introduced the ML estimator used for OFDM synchronization. As all metrics the ML metric is calculated depending on the correlator output. The timing offset is estimated by finding the maximum of the metric signal as follows:

$$\hat{\theta}_{ML} = \arg \max_n (m_{ML}[n]) = \arg \max_n (2 \cdot |c[n]| - \rho \cdot (p[n] + p[n+N_{PER}])))$$  \hspace{1cm} (4.10)
4.2. Metrics

Where $\rho$ is a weighting factor which depends on the $SNR$ value. It is defined as:

$$\rho = \frac{SNR}{SNR + 1} \quad (4.11)$$

By comparing (4.8) and (4.10) we can see that the ML is very similar to the MMSE. In other words, the MMSE is an ML estimator with $\rho = 1$. The block diagram of the ML estimator is shown in Fig. 4.13. The ML metric needs the $SNR$ value in order to calculate the metric signal. Therefore, the division block in Fig. 4.13 is needed. It was explained in the previous section that the power calculation can be simplified by removing the redundant calculation of $p[n + N_{PER}]$. As a result, the ML can be simplified into ML-SIM whose block diagram is shown in Fig. 4.14. The equation of the simplified ML then reads:

$$\hat{\theta}_{ML-SIM} = \arg \max_n (m_{ML-SIM}[n])$$

$$= \arg \max_n (|c[n]| - \rho \cdot (p[n])) \quad (4.12)$$

The performance of the ML metric is analyzed according to the acquisition failure probability. Fig. 4.15 shows the histogram of the error of time offset estimation for different $SNR$ values. In addition the acquisition failure probability for 3 values of $m$ of the ML
Figure 4.15: Histogram of the estimated timing offset obtained using the ML metric.

Figure 4.16: Acquisition failure probability for the ML metric.

The ML estimator suffers from some drawbacks when applied to burst mode transmission systems like the IEEE802.11a. One basic drawback is that the ML has a high probability of making a false alarm similar to the MMSE metric. If $SNR$ is high, i.e., $\rho \rightarrow 1$, the ML metric approaches the MMSE metric and the metric signal will have the form:

$$m[n] = (|c[n]| - p[n])$$

(4.13)

metric using a static AWGN channel is shown in Fig. 4.16.
If there is a silent period, i.e., no signal then \( c[n] \) will go to 0. Similarly if the noise level is low then the power sum \( p[n] \) will also go to 0. As a result the metric signal \( m[n] \) will also approach 0. As we mentioned in 4.2.3 the metric signal defined in (4.13) gets always negative values. Therefore, the value of \( m[n] \) approaching 0 can be detected as a maximum, thus resulting in a false alarm since no signal is present. Therefore, a period without signal may be mistaken as a start of a new burst. In the ideal case when a signal is present but with a high \( SNR \) value, the \( c[n] \) will approach the \( p[n] \) and \( m[n] \approx 0 \). In this case the detected maximum is the beginning of a new burst. The ML false alarm problem can be solved when an AGC is used as seen in Fig. 4.17. An additional solution for the false alarm problem known from literature is to define a threshold for the signal power and do not perform ML estimation if the signal power is less than the defined threshold.

Another drawback of ML: The \( SNR \) has to be estimated before the signal metric is achieved, this is no problem in continuous mode as it will be seen in the case of the DVB-T, since the \( SNR \) is tracked continuously. However, in the burst transmission the receiver first has to guess an \( SNR \) value when a new burst arrives and is then able to calculate the current \( SNR \), but when the next burst arrives this calculated value is outdated.

We aim now on choosing one of the metrics described above for the IEEE802.11a synchronization. The choice of the metric depends on the burst mode transmission characteristic of the IEEE802.11a wireless LANs, i.e., the choice is not the same for the IEEE802.11a and the DVB-T since the DVB-T as seen in Section 2.7 has a continuous
mode transmission. Fig. 4.18 shows the acquisition failure probability for \( m = 2 \) of the three metrics MC, MNC and ML.

![Graph showing acquisition failure probability for MC, MNC, and ML metrics.](image)

Figure 4.18: Comparison of the acquisition failure ratios of MC, MNC and ML.

### 4.2.5 Conclusions

It was discussed before that the ML metric suffers from false alarms in the burst mode transmission which make it not superior in the IEEE802.11a synchronization. Even though in Fig. 4.18 it appears to be performing better. In addition the MC metric suffers from the dependency on the power of the received signal. Therefore, the MNC metric is chosen for the simulations and the fixed point implementation of the IEEE802.11a synchronizer. In addition some complexity reasons for the choice of the MNC will be explained in Section 4.5

### 4.3 Simulation Model

In this section we aim on building an IEEE802.11a WLAN synchronizer. Section 3.1.3 gave an overview on the synchronization structure and the use of the metric signal. In this section, an overview on the simulation model used will be given and the synchronization algorithm will be further explained. Fig. 4.19 shows the Top level module used for our simulations. Details about each block of this module will be given in the coming sections. In addition, the simulation results will be shown.
4.3. Simulation Model

4.3.1 Overview of the IEEE802.11a Transmitter

The task of the transmitter module is to create the defined preamble and an arbitrary number of frames according to the IEEE802.11a standard. Long and short training symbols are read from a Table specified in [IEEE99]. The data is then transformed into the time domain via an IFFT and extended to a defined length of 320 samples altogether. The parameters used in the transmitter module are specified in table 4.1. The frames are generated by a random bit generator, and are then mapped in a modulator according to a selectable modulation type that is defined by the parameter 'OFDM modulation type'. The value 6 means that a 64-QAM modulation is performed. Then, reordering of the 64 samples follows since only 52 of them are occupied. For this purpose, 48 samples are read from the preceding stage and rearranged as seen in Fig. 4.20. This corresponds to the...
### 4.3.2 Imperfections

![Figure 4.21: Block diagram of imperfections module.](image)

In our simulation module, the imperfections block is used to introduce the frequency and time offsets that need to be estimated by the algorithm later on. It was taken unchanged from an earlier simulation model at Fraunhofer IIS. Therefore, the role of the imperfections block is to reproduce a transmission channel as realistic as possible. For the simulations that are seen throughout this chapter, the inserted time offset is $\theta = 100$. In addition, the frequency offset is $\epsilon = 0.25$, actually $\epsilon$ can take values with arbitrary accuracy, however only values within the range $|\epsilon| \leq 2$ make sense because IEEE802.11a preambles are only able to detect a frequency offset within this range. After inserting the offsets, a multipath channel should be specified. All together, 6 channels can be chosen within the imperfection block. The channel choice is set by a parameter $ch_{mode}$. If $ch_{mode}$ is set to 0, this means that there is no multipath propagation and the channel is an AWGN channel. The other 5 channels model different multipath channels. It is important in this procedure to exhibit a constant group delay and a constant amplification.
4.3.3 Synchronization Module

The synchronization algorithm proposed in this work consists of two synchronization phases. A coarse synchronization phase and a fine synchronization phase as seen in Fig. 4.22

![Diagram of synchronization modules](image)

Figure 4.22: Top level of the synchronization algorithm.

The coarse synchronization module estimates the timing offset and coarse frequency offset. Then the corrected signal is passed to the fine synchronization module to estimate the fine frequency offset. The structures of the two synchronization modules will be discussed in the following.

4.3.3.1 Coarse Synchronization

Fig. 4.23 shows the structure of the coarse synchronization module. The received signal is first given to the "Metric" block, where one of the metrics (MC, MNC, ML) discussed in Section 4.2 is calculated.

![Diagram of coarse synchronization module](image)

Figure 4.23: Coarse Synchronization Module.

In our discussion about metrics, we found that MNC is the most suitable metric as seen in Section 4.2.5. Therefore, the MNC metric will be further used for our simulation Model. In the coarse synchronization, autocorrelation of the short training symbols is performed. The "Metric" block outputs three signals used for the synchronization. One of which is the metric signal \( m[n] \) given to the "Argmax" block. The "Argmax"
block searches for a maximum of the \( m[n] \) and it waits \( N_{\text{argmax}} \) samples to assure that the calculated maximum is the true maximum. The argument at which the metric signal takes the maximum will be the estimate of the timing offset. Therefore the ”Argmax” block outputs the time offset. In addition to \( m[n] \), the ”Metric” block outputs the correlator signal \( c[n] \) and the delayed received signal \( r[n - N_{\text{delay}}] \). The delayed signal \( r[n - N_{\text{delay}}] \) is fed into the ”Frequency Correction” block, and if that is disabled then directly to the fine synchronization stage. In order to obtain a coarse frequency estimation and correct the received signal, the correlator signal \( c[n] \) is used. The signal \( c[n] \) is fed into the ”Angle” block where the angle signal is calculated as seen in Fig. 3.4. Actually only the angle at the maximum is correct because only then the correlator is filled up with the complete short training sequence, but due to noise an averaging over \( N_{\text{average}} \) values after the maximum increases the accuracy of estimation. It is important to note that the \( N_{\text{average}} \) values should be the ones before the maximum of the metric signal is achieved. After the maximum is achieved the correlator starts to collect values from the long preamble. Therefore, averaging over values before the maximum will prevent this error. The two ”Enable” blocks are controlled by the ”Argmax” block. Their task is to enable the frequency correction at the estimated timing offset.

The ”Delay” block introduced after the ”Angle” block is needed to provide correct timing with the succeeding blocks. Another ”Delay” block would be introduced to delay \( r[n - N_{\text{delay}}] \) if the variable \( N_{\text{argmax}} \) does not satisfy the condition \( N_{\text{delay}} = N_{\text{argmax}} - 1 \). This is needed to assure that the upper ”Enable” block activates the throughput when the long preamble starts. Therefore, the recommended parameters are \( N_{\text{argmax}} = 161 \) and \( N_{\text{average}} = 4 \), while the correlator parameters used in our simulations are \( N_{\text{PER}} = 16 \), \( N_{\text{TRAIN}} = 144 \) and \( N_{\text{delay}} = 160 \). The \( N_{\text{average}} \) parameter was chosen such that it results in the minimum variance for the frequency offset.

### 4.3.3.2 Fine Synchronization

In the Fine Synchronization procedure, only the fine frequency offset is estimated. The time is already corrected and the incoming signal \( r[n] \) starts with the long training symbols. In the coarse synchronization block a coarse frequency correction may be applied to the signal. Fig. 4.24 shows the fine frequency synchronization block. The received signal \( r[n] \) is given to the ”Correlator” block to get the correlator signal. Note that here the correlator has different parameters than the one for coarse synchronization, because now we are correlating the long training symbols. Therefore, according to Section 4.1 \( N_{\text{PER}} = 64 \), \( N_{\text{TRAIN}} = 96 \) and \( N_{\text{delay}} = 160 \). The ”Angle” block is also used for calculating the angle signal. Similar to the coarse synchronization case, averaging over some samples is needed for the fine frequency estimation. The correlator takes a maximum at the end of the long training symbols that is sample 159. The detection accuracy could be increased by
 avering over some values before the maximum. Fig. 4.25 plots the acquisition failure probability for both coarse and fine frequency estimation.

![Figure 4.25: Simulation results for coarse and fine frequency estimation](image)

### 4.4 Fixed Point Implementation

All the simulations that have been done until now are floating point simulations where the accuracy of numbers was irrelevant. In real systems, usually a finite word length is used, therefore when the target is a hardware implementation, then a fixed point implementation is necessary. The fixed point implementation is done in SystemC. A receiver usually contains an Automatic Gain Control (AGC) and an Analog-to Digital Converter (ADC). The AGC has the task to adjust the received signal to a certain power level in order to achieve an optimal saturation level and to highly avoid clipping by the
ADC. In our simulation, we include an AGC block before the synchronization block and after the imperfections are introduced to the signal. The AGC block does a simple multiplication to adjust the power level of the incoming signal to $-10\,dB$ full scale. This power level is chosen according to [PeCW]. Before going into implementation details, a short overview on the fixed-point types and the syntax used is given in the following section.

### 4.4.1 SystemC for Fixed Point Implementation

The fixed point implementation algorithm discussed in this chapter is done using SystemC. Therefore, an explanation of the fixed point types valid in SystemC and the parameters used for our algorithm are given in this section. The main source of information for the fixed-point types is [SysC]. [YaR] gives a general view on the fixed-point arithmetic. Four fixed point types are available in SystemC. Two of which need static arguments that have to be known at compile time, these are

- $\text{sc}_{\text{fixed}} \langle \text{wl}, \text{iwl}, \text{qmode}, \text{o mode}, \text{nbits} \rangle$
- $\text{sc}_{\text{ufixed}} \langle \text{wl}, \text{iwl}, \text{qmode}, \text{o mode}, \text{nbits} \rangle$

The representations are signed and unsigned respectively. The other fixed point types having dynamic arguments are

- $\text{sc}_{\text{fix}} \langle \text{list of options} \rangle$
- $\text{sc}_{\text{ufix}} \langle \text{list of options} \rangle$

that can be defined at run time. The fixed point format consists of the following parameters:

- $\text{wl}$: total word length (sign bit, integer part, fractional part)
- $\text{iwl}$: integer word length
- $\text{qmode}$: quantization mode
- $\text{o mode}$: overflow mode
- $\text{nbits}$: number of saturated bits

Quantization which is the procedure of converting a real number into a finite binary word is performed by the analog to digital converter at the receiver. The quantization error is usually modeled by added noise that decreases the signal-to-noise ratio. The
quantization mode is specified by the parameter $q_{mode}$. SystemC offers seven quantization modes. In our fixed point algorithm we use the quantization mode which rounds towards $-\infty$. Whereas the parameter $o_{mode}$ specifies the behavior when an overflow occurs. A saturated overflow mode specified in the SystemC is used as a default mode in this work. Since the default parameters are used, only the parameters $wl$ and $iwl$ should be specified.

An example of a signed fractional representation $x.xxx$ is the definition $sc_{fixed} < 4, 1 >$.

4.4.2 Correlator Implementation

Before going into explaining the fixed point implementation of the MNC metric, the implementation of the correlator is discussed in this section. Since the correlator plays a big role in the synchronization procedure, it is important to choose the less complex correlator implementation. Let us define a delay line with total Delay of $N_{\text{delay}}$ which has $N_{\text{delay}} - 1$ taps, the index $n = 0$ represents the current value. The input of the delay line is drawn on the right side while the output on the left as it is seen in Fig. 4.27

The autocorrelation equation is given in the form:

$$c[n, N_{\text{PER}}] = \sum_{i=0}^{N_{\text{TRAIN}}-1} r^*[n + i] \cdot r[n + i + N_{\text{PER}}].$$  \hspace{1cm} (4.14)

The choice of the parameters $N_{\text{TRAIN}}$ and $N_{\text{PER}}$ are discussed in section 4.1. Three possible cases can be distinguished in the choice of these parameters:
1. $N_{\text{PER}} > N_{\text{TRAIN}}$
2. $N_{\text{PER}} = N_{\text{TRAIN}}$
3. $N_{\text{PER}} < N_{\text{TRAIN}}$

The first case happens in a guard interval correlation, while the other two cases are commonly used with training symbols correlation.

In general, the autocorrelator can be implemented in three different ways:

- Basic implementation
- Double multiplier
- Single multiplier

Therefore, a trade off between the three implementations is needed in order to get the optimum way of implementation in terms of complexity efficiency. The basic implementation is only the basic formula of the autocorrelation seen in (4.14) without any simplification. As a consequence, this implementation is too complex because we need $N_{\text{TRAIN}}$ complex multipliers. While in the other two implementations only two or one complex multipliers are used. In the following, the notation will be changed: $c_n$ corresponds to $c[n]$ and similarly, $r_n$ corresponds to $r[n]$. The Double multiplier implementation is based on an recursive method for calculating the autocorrelation function given as:

$$c_n = c_{n-1} + r_n^* \cdot r_{n-N_{\text{PER}} - r_{n-N_{\text{TRAIN}}} \cdot r_{n-N_{\text{TRAIN}}-N_{\text{PER}}}} \quad (4.15)$$

The principle is simple, it is based on adding the values $r_n^* \cdot r_{n-N_{\text{PER}}}$ that arrive the delay line and subtracting the values $r_{n-N_{\text{TRAIN}}} \cdot r_{n-N_{\text{TRAIN}}-N_{\text{PER}}}$ that leave the delay line. At every time the delay in shifted once and a new value enters while an old value leaves the delay line, so all the other multiplications are redundant. The implementation of this method needs only two complex-valued multipliers, and not $N_{\text{TRAIN}}$ multipliers as needed in the basic case. In addition, the accumulator in this implementation is less complex than that in the basic implementation, because it has to add only 3 input values. However, this needs one more complex delay tap than the basic implementation, but this is not so important because the delay line is large anyway. One additional register is needed in this case in order to save the old value of the correlation.

The single multiplier implementation involves a second delay line ($d_n$) which stores the value in the multiplier and then subtracts from the previous correlator output according to:

$$d_n = r_n^* \cdot r_{n-N_{\text{PER}}} \quad (4.16)$$
As it is clear this method needs only a single multiplier, and the delay taps are the same as the other methods, because even if a new delay line is introduced, the main delay line is decreased and therefore ending with the same number of delay taps. The disadvantage of this method is that at the end the overall delay is decreased which will cause the insertion of an additional delay block in the coarse synchronization model as discussed in section 4.3.3.1. The overall delay between the signal input and the "Enable" block seen in Fig. 4.23 should be equal to $N_{\text{argmax}}$ samples. As a result, the double multiplier method is less complex although it uses two complex multipliers instead of one. As a result, the dual multiplier will be used in the Metric block. Fig. 4.28 shows the 3 different correlator implementations in the case $N_{\text{PER}} > N_{\text{TRAIN}}$, while Fig. 4.29 shows the three implementations for $N_{\text{PER}} < N_{\text{TRAIN}}$.

The possibility of recursive calculation of the autocorrelation makes it more preferable than crosscorrelation for synchronization algorithms, since the crosscorrelation can not
be computed recursively (see A.6). Due to this recursive calculation the autocorrelator was implemented more efficiently with less complexity.

4.4.3 MNC Metric in Fixed Point

The correlator is used to implement the fixed point MNC metric block. The double multiplier is used for the autocorrelation in order to reduce the complexity. The MNC metric block is implemented in systemC using the parameters given before. In the fixed point implementation we deal only with real values, therefore the recursive calculation of the autocorrelation has to be split up into real and imaginary parts. The real part is given by:

\[
\begin{align*}
    c'_n &= c'_{n-1} + \\
    &+ r'_n \cdot r'_{n-N_{\text{PER}}} + r''_n \cdot r''_{n-N_{\text{PER}}} \\
    &- (r'_{n-N_{\text{TRAIN}}} \cdot r'_{n-N_{\text{TRAIN}}-N_{\text{PER}}} + r''_{n-N_{\text{TRAIN}}} \cdot r''_{n-N_{\text{TRAIN}}-N_{\text{PER}}})
\end{align*}
\]
and the imaginary part:
\begin{align}
c''_n &= c''_{n-1} + \\
&+ r'_{n-N_{PER}}^* \cdot r''_{n-N_{PER}} - r''_{n-N_{PER}} \cdot r'_{n-N_{PER}} \\
&- (r'_{n-N_{TRAIN}} \cdot r''_{n-N_{TRAIN}} - N_{PER} - r''_{n-N_{TRAIN}} \cdot r'_{n-N_{TRAIN}} - N_{PER}).
\end{align}

In addition, a recursive calculation for the power is given by:
\begin{align}
p_n &= p_{n-1} + r_n^* \cdot r_n - r_{n-N_{TRAIN}}^* \cdot r_{n-N_{TRAIN}} \\
&= p_{n-1} + (r'_{n})^2 + (r''_{n})^2 - (r'_{n-N_{TRAIN}})^2 - (r''_{n-N_{TRAIN}})^2.
\end{align}

The calculation of the autocorrelation and the power sum is done using the same delay line. The MNC metric block outputs the correlator signal \(c[n]\), the metric signal \(m[n]\), the delayed received signal \(r[n-N_{delay}]\) and the power signal \(p[n]\). These output signals are with fractional format with the corresponding bit widths \(fwl_c\), \(fwl_m\), \(fwl_r\) and \(fwl_p\). The term \(fwl\) is equivalent to \(wl - iw\).

### 4.4.4 CORDIC Algorithm

This section provides an overview on the CORDIC (Coordinate Rotation Digital Computer) algorithm and how it is used in the fixed point implementation of the IEEE802.11a synchronizer. The cordic algorithm is usually used to compute trigonometric, hyperbolic and logarithmic functions. The basic principle of the cordic algorithm is to rotate a vector in particular steps. That is a complex valued vector \(r = x + j \cdot y\) is rotated by the transform \(r' = r \cdot e^{j \phi}\). The idea is to execute the rotation in \(N\) steps each with a predetermined angle \(\phi_i\), this angle decreases with every iteration \(i\) and is set so that \(\tan \phi_i = \pm 2^{-i}\). The advantage is that the multiplication with \(2^{-i}\) can be implemented by a single shift operation. The iterative rotation can be expressed as:
\begin{align}
x_{i+1} &= [x_i - y_i \cdot d_i \cdot 2^{-i}] \\
y_{i+1} &= [y_i + x_i \cdot d_i \cdot 2^{-i}]
\end{align}
where \(d_i = \pm 1\) and it determines in which direction is the rotation. The final angle after \(N\) iterations will amount to:
\begin{align}
\phi_N &= \sum_{i=0}^{N-1} d_i \phi_i
\end{align}

In this work, the cordic algorithm is used for two purposes. On the one hand, it is used to calculate the angle of the complex-valued correlator signal to estimate the frequency offset. The mode of the cordic algorithm in this case is referred to as vectoring mode. On the other hand, the cordic algorithm is used to correct the frequency offset by rotating the received signal. In this case the algorithm is working in the rotation mode. According to [KC93] a cordic unit with \(N + \log_2(N)\) fractional bits in the internal data path and \(N\) iterations is sufficient to obtain a precision of \(N\) bits in the external data paths.
4.4.5 Coarse and Fine Synchronization

All the floating point modules are now built in the fixed point using a finite word length. Fig. 4.30 shows the coarse synchronization block. The cordic block in vectoring mode substitutes the angle calculation, where it calculates the phase of the correlated signal given by:

\[
\phi = \frac{2\pi N_{\text{PER}}}{N_{\text{FFT}}} \hat{\epsilon}
\]

Fig. 4.30: The fixed point representation of the coarse synchronization block

The fine synchronization block is shown in Fig. 4.31. The fixed-point module of the correlator in Fig. 4.31 is the same as the one found in the MNC fixed point block. The frequency correction is done using the cordic algorithm in rotation mode. Another possibility for the frequency correction is a complex-valued multiplication by \( e^{\frac{2\pi i}{N_{\text{FFT}}}} \). The optimal choice depends on the specific hardware implementation.

Fig. 4.31: The fixed point representation of the fine synchronization block
4.4.6 Word Length Optimization

In the coarse synchronization block, the cordic algorithm has the correlator signal $c[n]$ as an input. The input of the cordic algorithm has a word length of $N_{VEC} + \log_2(N_{VEC}) + 2$. The value of $N_{VEC}$ is chosen from a look-up table according to a predetermined angle. This angle is determined depending on how much accuracy is needed. In our work, we have chosen the normalized frequency offset to be within $|\varepsilon| \leq 0.01$. So for maximum frequency offset we calculate the upper bound of the angle:

$$\phi = \varepsilon \cdot \frac{2\pi \cdot N_{PER}}{N_{FFT}}$$

which is equal in this case $\phi \approx 0.9^\circ$. Then from the look up table we find that the needed $N_{VEC} = 8$. Therefore, $c[n]$ with a wordlength $wl = 13$ bits ($N_{VEC} + \log_2(N_{VEC}) + 2$) is enough to get the accuracy needed.

4.4.7 Implementation Loss Analysis

The main parameters to be adjusted in the fixed-point model to get results similar to the floating point implementation are the fractional bits $fw_{lm}$ of the metric signal and the number of iterations $N_{VEC}$ of the cordic algorithm. This is because all other parameters depend on these two. Fig. 4.32 plots the acquisition failure probability for the time offset for different values of $fw_{lm}$. It is clearly seen that the accuracy increases with each bit and for the value $fw_{lm} = 12$ it achieves nearly the floating point results.

Figure 4.32: Acquisition failure probability of time estimation for different word lengths
Therefore, also in simulations it is shown that \( w_{l_m} = 13 \) bits is enough to get the accuracy needed. As a result \( w_{l_p} = 13 \) bits. This implies that the received signal should be with a minimum wordlength \( w_{l_r} = 6 \) bits to achieve the required accuracy. Table 4.2 summarizes the required word lengths for the signals in this algorithm.

<table>
<thead>
<tr>
<th>Data signal</th>
<th>Representation</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r[n] )</td>
<td>(&lt; f w_{l_r} + 1, 1 &gt;)</td>
<td>(&lt; 6, 1 &gt;)</td>
</tr>
<tr>
<td>( c[n] )</td>
<td>(&lt; f w_{l_c} + 1, 1 &gt;)</td>
<td>(&lt; 13, 1 &gt;)</td>
</tr>
<tr>
<td>( p[n] )</td>
<td>(&lt; f w_{l_p} + 1, 1 &gt;)</td>
<td>(&lt; 13, 1 &gt;)</td>
</tr>
<tr>
<td>( m[n] )</td>
<td>(&lt; w_{l_m}, 1 &gt;)</td>
<td>(&lt; 13, 1 &gt;)</td>
</tr>
</tbody>
</table>

Table 4.2: Word lengths used in the synchronization algorithm.

Fig. 4.33 shows the acquisition failure probability for the coarse frequency estimation using different values for \( N_{VEC} \). It is clear that the value \( N_{VEC} = 9 \) is needed to approach the floating point results. Whereas it is seen in Fig. 4.34 that only \( N_{VEC} = 8 \) is needed to approach the floating point results in the fine synchronization stage. In fine synchronization, an additional iteration does not increase the accuracy.

4.5 Complexity Analysis

In this section a study of the complexity of an IEEE802.11a synchronizer is done. The synchronization algorithm of IEEE802.11a is based on calculating the correlator signal
and from which a metric signal is evaluated using one of the Metrics described in section 4.2. Even though in this study the MNC metric was chosen for the fixed point implementation, a complexity analysis for all metrics is done.

The Complexity of the algorithms depends highly on the correlator implementation chosen. In Section 4.4.2, the double multiplier implementation of the correlator was chosen. Therefore, for the correlator 8 Multipliers, 8 Adders and 2 Registers are needed per output value. In addition, $2 \times 160$ real valued delay taps are needed. For estimating the gates needed, $10 \text{ gates/bit}$ is used as a rule of thumb. Table 4.3 summarizes the gates needed for the correlator implementation. The autocorrelator will need approximately 23380 gates.

Figure 4.34: Simulations for different cordic rotations for the fine frequency estimation.
4.5.1 MNC

The MNC metric discussed in Section 4.2.2 includes in addition to the autocorrelation calculation, the calculation of the power sum as seen in (4.5). Squaring the correlation signal and the power signal is done in order to reduce the complexity of the MNC metric by saving two square root units. The complexity of computing the MNC metric signal, assuming that the correlator signal \( c[n] \) is already calculated, is summarized in Table 4.4. Therefore, the MNC metric would require a total of 14180 gates other than the correlator.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Bits</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mults</td>
<td>6 * 6</td>
<td>1440</td>
</tr>
<tr>
<td>3 Mults</td>
<td>13 * 13</td>
<td>5070</td>
</tr>
<tr>
<td>4 Adds</td>
<td>13</td>
<td>520</td>
</tr>
<tr>
<td>1 Adds</td>
<td>26</td>
<td>260</td>
</tr>
<tr>
<td>1 Regs</td>
<td>13</td>
<td>130</td>
</tr>
<tr>
<td>1 Div</td>
<td>26 * 26</td>
<td>6760</td>
</tr>
</tbody>
</table>

Table 4.4: Complexity of the MNC metric after calculation of \( c[n] \).

Therefore, all together 14180 + 23380 = 37560 gates to compute \( m_{MNC}[n] \) are needed.

4.5.2 MC

The Maximum Correlation (MC) metric discussed in Section 4.2.1 has definitely a lower complexity than the MNC metric, since it includes only the correlator as seen in (4.3). Having the correlator signal the MC metric will need only 2 Mults and 1 Add to compute \( |c[n]|^2 \). The complexity is seen in Table 4.5. Therefore, the MC metric signal will need a total of 3640 + 23380 = 27020 gates, which is much less than the number of gates needed for the MNC. Nevertheless, the MNC metric is preferred on the MC metric due to its robustness against the power variations of the input signal due to the normalization involved in its calculation.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Bits</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Mults</td>
<td>13 * 13</td>
<td>3380</td>
</tr>
<tr>
<td>1 Adds</td>
<td>26</td>
<td>260</td>
</tr>
</tbody>
</table>

Table 4.5: Complexity of the MC metric after calculation of \( c[n] \).
4.5.3 MMSE

The MMSE can be calculated in a simplified way as seen in (4.9). The complexity required to calculate the metric signal assuming the correlated signal \( c[n] = c'[n] + jc''[n] \) is calculated as seen in Table 4.6. The total number of gates needed to calculate the MMSE metric signal is approximately \( 5730 + 23380 = 29110 \) gates. But in addition two square-root units are needed to calculate \( |c[n]| \) and \( p[n] \), since in this metric the square-root units can not be replaced by squaring. These would increase the complexity dramatically.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Bits</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mults</td>
<td>6 * 6</td>
<td>1440</td>
</tr>
<tr>
<td>2 Mults</td>
<td>13 * 13</td>
<td>3380</td>
</tr>
<tr>
<td>4 Adds</td>
<td>13</td>
<td>520</td>
</tr>
<tr>
<td>1 Adds</td>
<td>26</td>
<td>260</td>
</tr>
<tr>
<td>1 Regs</td>
<td>13</td>
<td>130</td>
</tr>
</tbody>
</table>

Table 4.6: Complexity of the MMSE metric after calculation of \( c[n] \).

4.5.4 ML

In addition to the MMSE metric, the ML metric given in (4.10) has also a simplified version seen in (4.12). The complexity is summarized in Table 4.7 assuming that the correlator signal is calculated. The total number of gates needed to calculate the ML metric signal amounts to \( 14310 + 23380 = 37690 \) gates, in addition to the two square-root units which will increase the complexity more.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Bits</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mults</td>
<td>6 * 6</td>
<td>1440</td>
</tr>
<tr>
<td>3 Mults</td>
<td>13 * 13</td>
<td>5070</td>
</tr>
<tr>
<td>5 Adds</td>
<td>13</td>
<td>650</td>
</tr>
<tr>
<td>1 Adds</td>
<td>26</td>
<td>260</td>
</tr>
<tr>
<td>1 Regs</td>
<td>13</td>
<td>130</td>
</tr>
<tr>
<td>1 Div</td>
<td>26 * 26</td>
<td>6760</td>
</tr>
</tbody>
</table>

Table 4.7: Complexity of the ML metric after calculation of \( c[n] \).
4.6 Algorithmic Optimization

The synchronization algorithm discussed throughout this chapter is based on autocorrelation calculation. In this section, an optimization of this algorithm by using a simplified correlator in order to decrease the complexity is studied. In addition, the performance of these algorithm are compared to the ML estimator which is a realization of the Cramer Rao bound.

4.6.1 Correlator Approximations

Let us remember the autocorrelation equation mathematically. Using the two parameters $L$ and $N$ for generalization, the autocorrelation is given by:

$$c[n, N] = \sum_{i=0}^{L-1} r^*[n + i]r[n + i + N].$$  

(4.24)

$c[n]$ takes complex values, therefore for a fixed point implementation, the correlation is calculated as two parts, the real part $c'[n]$ and imaginary part $c''[n]$. The magnitude of $c[n]$ is then calculated as:

$$|c[n]| = \sqrt{(c'[n])^2 + (c''[n])^2}.$$  

(4.25)

where $c'[n]$ is the real part of $c[n]$ and $c''[n]$ is the imaginary part. In this section the following two approximations for $|c[n]|$ are evaluated.

- $|c[n]| \approx |c'[n]| + |c''[n]|$
- $|c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|)$ with $\alpha = 1$ and $\beta = \frac{1}{4}$

The correlator can be implemented in different ways, but it was seen before that the double multiplier implementation is the most efficient and less complex. By the double multiplier correlator we get $c'[n]$ and $c''[n]$. Using these approximations for calculating $|c[n]|$ a square root unit is saved. In the following, the different metric signals (MC, MNC and ML) are calculated using the two approximations for $|c[n]|$. In our simulations the metric signals where evaluated for different $SNR$ values and two types of channels: AWGN and Rayleigh. In the following we will only show the metric signals for a Rayleigh channel and $SNR = 0 \text{ dB}$. Fig. 4.35 shows the MC metric signal for a rayleigh channel with $SNR = 0 \text{ dB}$. The metric signal calculated using the two approximations takes maximum at the same point as when using the exact magnitude $|c[n]|$ or the magnitude squared $|c[n]|^2$. Taking the maximum at the same point is sufficient for our application, since we are interested in the sample at which the metric signal $m[n]$ takes maximum and not in the value of the maximum of $m[n]$. The MNC metric signal $m_{MNC}[n]$ using the two approximations is shown in Fig. 4.36 for a rayleigh channel with $SNR = 0 \text{ dB}$. The
4.6. Algorithmic Optimization

Figure 4.35: MC metric for a Rayleigh channel with SNR = 0dB.

Figure 4.36: MNC metric for a Rayleigh channel with SNR = 0dB.

MNC metric signal $m_{MNC}[n]$ calculated using the two approximations take maximum at the same sample as by using the exact value $|c[n]|$. Similarly for the ML metric as seen in Fig. 4.37. In the ML metric, calculating $|c[n]|^2$ instead of $|c[n]|$ is not possible because the metric is not a ratio value where we can square the numerator and denominator and still get maximum at the same point. Therefore, using any of these approximations is more helpful and is the only solution for the ML metric to decrease computational complexity and save a square root unit. In addition, as seen in Fig. 4.38 the approximation $|c[n]| \approx |c'[n]| + |c''[n]|$ solves the false alarm problem for high SNR ($SNR = 10dB$).
In order to evaluate the approximations compared to the exact magnitude calculation, the variance values for the different calculations w.r.t $SNR$ are calculated. Fig. 4.39 plots the variance vs. $SNR$ values for an AWGN channel. In these simulations 1000 OFDM bursts are used. In addition Fig. 4.40 plots the variance versing the $SNR$ for a Rayleigh channel.

In the following a description of the Cramer Rao bound as a lower limit for the variances when using a certain estimation is given. The problems of using the Cramer
Rao bound for the following application and the proposed alternative are then discussed.

### 4.6.2 Cramer Rao Bound

In statistics, the Cramer-Rao bound expresses a lower bound on the variance of estimators of a deterministic parameter. In its simplest form, the bound states that the variance of any unbiased estimator is at least as high as the inverse of the Fisher information. The idea of this section was to compare the different estimations of the correlation function to the Cramer Rao bound.
4.6.2.1 Log-Likelihood Function

Let $U$ be a random vector represented by the observed sample vector $u$, where the sample vector $u$ has $u_1, u_2, \ldots, u_M$ for its elements, and $\lambda$ is a parameter vector with $\theta_1, \theta_2, \ldots, \theta_K$ as elements. Let $f_U(u|\lambda)$ denote the conditional joint probability density function of the random vector $U$.

The method of maximum-likelihood is based on the principle that we should estimate the parameter vector $\lambda$ by its most plausible values, given the observed sample vector $u$. In other words, the maximum-likelihood estimators of $\lambda_1, \lambda_2, \ldots, \lambda_K$ are those values of the parameter vector for which the likelihood-function is at maximum, where the likelihood-function which is denoted by $l(\lambda)$ is equal to the conditional joint probability density function $f_U(u|\lambda)$. That is

$$l(\lambda) = f_U(u|\lambda)$$

(4.26)

In many cases it turns out to be more convenient to work with the logarithm of the likelihood function rather than with the likelihood function itself. Thus using $L(\lambda)$ to denote the log-likelihood function, we write:

$$L(\lambda) = \ln[l(\lambda)] = \ln[f_U(u|\lambda)]$$

(4.27)

The parameter vector for which the likelihood function $l(\lambda)$ is at the maximum is exactly the same as the parameter vector for which the log likelihood function $L(\lambda)$ is at its maximum.

4.6.2.2 Cramer Rao Inequality

Let $U$ be a random vector with conditional joint probability density function where $u$ is the observed sample vector with elements $u_1, u_2, \ldots, u_M$ and $\lambda$ is a parameter vector with elements $\lambda_1, \lambda_2, \ldots, \lambda_K$. Using the definition of (4.27) for the log-likelihood function $L(\lambda)$ in terms of the conditional joint probability density function we form the K-by-K matrix:

$$J = \begin{bmatrix}
E \left[ \frac{\partial^2 L}{\partial \lambda_1^2} \right] & E \left[ \frac{\partial^2 L}{\partial \lambda_1 \partial \lambda_2} \right] & \cdots & E \left[ \frac{\partial^2 L}{\partial \lambda_1 \partial \lambda_K} \right] \\
E \left[ \frac{\partial^2 L}{\partial \lambda_2 \partial \lambda_1} \right] & E \left[ \frac{\partial^2 L}{\partial \lambda_2^2} \right] & \cdots & E \left[ \frac{\partial^2 L}{\partial \lambda_2 \partial \lambda_K} \right] \\
\vdots & \vdots & \ddots & \vdots \\
E \left[ \frac{\partial^2 L}{\partial \lambda_K \partial \lambda_1} \right] & E \left[ \frac{\partial^2 L}{\partial \lambda_K \partial \lambda_2} \right] & \cdots & E \left[ \frac{\partial^2 L}{\partial \lambda_K^2} \right]
\end{bmatrix}$$

(4.28)

The matrix $J$ is called the Fisher’s information matrix. Let $I$ denote the inverse of Fisher’s information matrix $J$. Let $I_{ii}$ denote the $i$th diagonal element of $I$.

Let $\hat{\lambda}_i$ be any unbiased estimate of the parameter $\lambda_i$, based on the observed sample
vector $u$. We may then write:

$$\text{var} \left[ \hat{\lambda}_i \right] \geq I_{ii}, \quad i = 1, 2, \ldots, K \quad (4.29)$$

The above equation is called the Cramer Rao inequality.

### 4.6.2.3 Cramer Rao Bound for This Application

For this application, namely time offset estimation, the purpose was to complement the plots of the variances with the Cramer Rao bound. We have achieved different variance values for the different correlation calculations at different $SNR$ values. Nevertheless, the Cramer Rao Bound was not feasible. To simplify the discussion of the Cramer Rao derivation, we concentrate on estimating a single parameter (Time offset) depending on observations. Consequently, different observations lead to different estimates. Thus, the Cramer Rao bound is approximated by:

$$CRB(\lambda) = \frac{1}{E \left[ \frac{\partial L(\lambda)}{\partial \lambda} \right]^2} \quad (4.30)$$

Unfortunately, application of this bound to practical synchronization problems is difficult due to the necessity of computing $L(\lambda)$ (log-likelihood function) which is seldom feasible. First of all, to estimate the time offset in our work, we first do the autocorrelation of the received signal as a first step, then the correlation signal is passed to the Metric Block (example MNC) to estimate the time offset. In our concrete application, the following problems face the calculation of the Cramer Rao bound:

- We could not get a closed form of the likelihood function which could be later derivated.

- We are performing coarse time offset estimation, so the accuracy is 1 sample (time-offset-estimation-error = 0, 1, 2, etc...). At high $SNR$ and with an AWGN channel, the simulated variance was 0 (error-free for the 1-sample resolution). Thus, the cramer rao as a lower bound in not applicable in this case.

In addition, the cramer rao is a theoretical bound which is not feasible and the ML estimator is the best realization for implementing the cramer rao. Therefore, in the implementation the best we can reach is the ML estimator then comparing with the ML...
estimator is an alternative for the cramer rao bound. Fig. 4.41 plots the variances of the approximations at different SNR values for an AWGN channel compared to the ML estimator for 1000 bursts. Fig. 4.42 plots the same as in Fig. 4.41 but for simulations using 100000 bursts.

Figure 4.41: Comparison with the ML for AWGN channel using 1000 bursts.

Figure 4.42: Comparison with the ML for AWGN channel using 100000 bursts.
5 DVB-T Synchronization

In this chapter DVB-T synchronization algorithms are developed and analyzed. Simulation results are shown for different scenarios, in addition to fixed point implementation results. An important aspect of the DVB-T transmission is that the signal is continuous at the contrary to the IEEE802.11a signal which is transmitted in a burst mode and this make it possible to use techniques like averaging over many symbols in the time domain. The principles of synchronization in DVB-T were discussed in Section 3.2, which are based on either autocorrelation or crosscorrelation.

This chapter is organized as follows: In the next section an overview on the synchronization algorithms (pre-FFT and post-FFT) for DVB-T systems will be given. A description of the simulation model used to test the algorithms follows. After that the simulation results for different channel types are given. Then, the fixed point implementation of the ML algorithm is discussed. A section dealing with algorithmic optimization is followed, in which an approach to improve the performance of the ML algorithm, by using a weighted sum of several OFDM symbols is described. The complexity analysis of the suggested algorithms is studied in the last section.

5.1 Synchronization Algorithms

As seen in chapter 2, the DVB-T receiver suffers from time and frequency offsets. Therefore, synchronization techniques are used for both timing and frequency synchronization.

5.1.1 Coarse Time Synchronization

Coarse time synchronization methods are developed in the time domain, i.e., they are classified as pre-FFT synchronization methods. As seen in Section 3.2.1, autocorrelation of the GI is a technique used for timing synchronization. The idea of coarse time synchronization is the same as in IEEE802.11a. It is based on calculating a metric signal which depends on a correlator output. Finding the argument at which this metric takes a maximum gives an estimate for the timing offset. As seen before the autocorrelation is expressed by

\[ c[n, N_{FFT}] = \sum_{i=0}^{L_g-1} r^*[n+i]r[n+i+N_{FFT}] . \]  \hspace{1cm} (5.1)
The absolute value of \( c[n] \) can be used as a metric for estimating \( \theta \), which was described in Section 4.2.1 by the Maximum Correlator (MC) metric. The implementation of the correlator was discussed in Section 4.4.2. Depending on the implementation method used the parameters of the MC metric that can be achieved are: \( N_{FFT} + L_g \) delay taps and 1 or 2 complex multipliers depending on which iterative implementation is used. These would result in a total delay of the received signal of \( N_{FFT} \) if the single multiplier is used and \( N_{FFT} + L_g - 1 \) if the double multiplier implementation is used. According to the DVB-T standard [DVB01] the number \( L_g \) (length of GI) should take one of the following values: \( N_{FFT} \frac{4}{16} \), \( N_{FFT} \frac{8}{16} \), \( N_{FFT} \frac{16}{16} \), \( N_{FFT} \frac{32}{16} \), this defines the complexity of the correlator. A disadvantage of this method in its application to DVB-T is the usage of the GI samples and this is because these samples can be corrupted by ISI especially if we have multipath propagation which is highly probable in DVB-T systems. In addition, the MC metric suffers from the problem of not taking into account the power of the signal, which provokes the use of the Maximum Normalized Correlation (MNC) metric given in section 4.2.2. The power sum is defined by

\[
p[n] = \sum_{i=0}^{L_g-1} |r[i+n]|^2.
\]  

(5.2)

Since the MNC depends on the incoming \( SNR \) of the signal then it performs better than the MC especially if the incoming signal power varies fast. This metric was chosen as the best metric in the IEEE802.11a synchronization. In addition to the above two metrics, the ML metric which also takes into account the \( SNR \) value can be used for DVB-T synchronization. The ML criterion based on the autocorrelation of the GI can be expressed as:

\[
M_{GI}[n] = |c[n]| - \rho \cdot p[n]
\]

(5.3)

The parameter \( \rho \) given by (4.11) is directly dependent on the \( SNR \). For calculating the \( SNR \) value the correlator signal and the power sum are used. The correlation calculates the power of the the periodic part of the received signal, i.e., the part with no noise, while the power sum calculates the power of the whole received signal. As a result, the \( SNR \) can be estimated as [M00]:

\[
SNR \approx \frac{|c[\hat{\theta}]|}{p[\hat{\theta}] - |c[\hat{\theta}]|}
\]

(5.4)

To get an estimation of the \( SNR \) as accurate as possible, the time offset \( \hat{\theta} \) should be first estimated. However, in the ML metric we need \( SNR \) in order to be able to estimate the timing offset. Therefore the \( SNR \) depending on the previous estimation can be used. This is different to the IEEE802.11a synchronization, since in burst transmission, the previous \( SNR \) is considered outdated. But in continuous transmission this is possible and this makes the ML metric yield the best results in continuous mode transmission.
In addition to the autocorrelation of the GI, another method was proposed in [ScC97] which is based on autocorrelation of the time domain training symbols. It has the advantage that it is not dependent on the channel characteristics as long as the channel delay spread is not greater than the GI. The principle is based on inserting one symbol whose first half is random and the second half is identical to the first half. The autocorrelation is performed with a shift of \( N = \frac{N_{\text{FFT}}}{2} \), then one of the metrics described above can be used to estimate the time offset. But this method is not possible for DVB-T synchronization, since the DVB-T standard does not provide training symbols in the time domain.

The DVB-T standard provides however frequency domain pilots which can be used for synchronization as seen in 3.2.2. These pilots do not form a complete symbol in the time domain. These pilots are inserted in the frequency domain in the transmitter and after the IFFT all the subcarriers will carry pilot and data contributions. The crosscorrelation is used with the autocorrelation in order to perform timing synchronization. Denote by \( S \) the set of all continual pilot carriers in one symbol (\( S_c \) is the set of other carriers). We can write:

\[
d_c[n] = \sum_{k \in S} c_k e^{j2\pi nk/N_{\text{FFT}}} \\
d[n] = \sum_{k \in S_c} c_k e^{j2\pi nk/N_{\text{FFT}}}
\]

where:
- \( c_k \): frequency symbol transmitted on the \( k \)th subcarrier.
- \( d[n] \): data component of the transmitted signal.
- \( d_c[n] \): continual pilot component of the transmitted signal.

Therefore, the sent signal has the form

\[
s[n] = d[n] + d_c[n].
\]

Thus, the metric proposed in [LB02] is given by

\[
M[\theta] = \rho \cdot M_GI[\theta] + (1 - \rho)M_p[\theta]
\]

The metric \( M_GI \) is exactly the ML metric defined in (5.3). The Metric \( M_p \) is realized by two crosscorrelation functions defined as:

\[
M_p[\theta] = (1 + \rho) \cdot S_1 - \rho \cdot S_2
\]

where \( S_1 \) and \( S_2 \) are defined as:

\[
S_1 = \left| \sum_{n=0}^{\theta+L_p-1} r^*[n] \cdot d_c[n - \theta] \right| \\
S_2 = \left| \sum_{n=0}^{N} (r[n] + r[n + N])^* \cdot d_c[n - \theta] \right|
\]
It is clear from the equations above that for high SNR ($\rho$ goes to 1) $M$ is the ML metric. However for low SNR the crosscorrelation metric $M_p$ prevails. This metric is a very complex metric and this is because it involves the calculation of an ML metric in addition to two crosscorrelation equations. However, the cross-correlation can not be calculated iteratively which adds more complexity to the metric.

One important drawback of the crosscorrelation method is its sensitivity to frequency offset $\varepsilon$. Since due to a frequency offset the samples corresponding to the received signal are corrupted, but the samples saved in the receiver are not affected. Therefore, the correlation of the two sequences may not reach a maximum when they are aligned. This will be shown in the simulations.

Not only the continual pilots can be used for this metric but also the scattered pilots given in the DVB-T standard or even both kinds of pilots. This is advantageous to the algorithm performance due to the increase of number of pilots used, but on the other hand it increases the complexity.

5.1.2 Frequency Synchronization

In addition to the timing offset, we aim in this thesis at estimating the frequency offset of DVB-T systems. The DVB-T system suffers from a frequency offset $\Delta_{F_{req}}$ which is divided into an integer part and a fractional part.

$$\Delta_i + \Delta_f$$

(5.12)

where $\Delta_i$ is the multiple part of the sub-carrier spacing $\Delta_i = \varepsilon_i \Delta F$, and $\Delta_f = \varepsilon_f \Delta F$ is the fractional part. $\Delta F$ is the subcarrier spacing. After normalization the frequency offset will be given as:

$$\varepsilon = \varepsilon_i + \varepsilon_f$$

(5.13)

where $\varepsilon_i$ stands for the integer part of the normalized frequency offset and $\varepsilon_f$ is the fractional part. The frequency offset estimation for DVB-T systems is done by two stages: A pre-FFT stage where the $\varepsilon_f$ is estimated, and a post-FFT stage where $\varepsilon_i$ is estimated. In the following a detailed description of the pre-FFT and post-FFT stages will be given.

5.1.2.1 Pre-FFT Stage

The pre-FFT synchronization stage is performed in the receiver before the FFT, where the received signal is still in the time-domain. The pre-FFT frequency offset estimation is similar to the frequency offset estimation in the IEEE802.11a. It depends on the correlated signal, the correlator output takes a maximum when the correlator window is occupied.
by a complete OFDM symbol, i.e., the GI is correlated with the part it is taken from. Therefore, at the calculated time offset the frequency offset can be estimated:

$$\varepsilon = \frac{1}{2\pi} \arg(c[\hat{\theta}]).$$ \hspace{1cm} (5.14)

As seen in the IEEE802.11a case, the frequency estimation is done after the time offset is corrected. It is clear that the maximum allowed value of $|\varepsilon|$ is given by:

$$|\varepsilon| < \frac{1}{2}$$ \hspace{1cm} (5.15)

Therefore, it is obvious that using this method we are only able to estimate the fractional part of the frequency offset as it is said in the beginning of this section. In Section 5.1.1 other procedures were discussed for the estimation of the timing offset, these procedures can also be used for the frequency estimation.

5.1.2.2 Post-FFT Stage

The post-FFT algorithms are used for the correction of the integer part of the frequency offset. The time offset and the fractional frequency offsets are first corrected at the receiver and then the partially corrected received signal is given to the FFT. After the FFT, the remaining integer frequency offset is estimated and corrected. Therefore, the frequency offset is defined by $\varepsilon = \varepsilon_i$. All methods described in this section make use of the continual pilots inserted in the DVB-T according to the DVB-T standard [DVB01].

An approach for integer frequency estimation is presented in [SF03], it is based on calculating the absolute value of the correlation between the pilot sub-carriers of one OFDM symbol and the previous OFDM symbol.

$$M[l] = \left| \sum_{k \in S} c_{m,k+l} \cdot c_{m-1,k+l}^* \right|$$ \hspace{1cm} (5.16)

where:

$c_{m,k}$: $k$-th sub-carrier of the $m$-th received symbol

$l$: carrier shift

$S$: set of pilot carriers indices.

It was explained in chapter 2 that the integer frequency offset causes a shift in the spectrum of the signal. That is, data transmitted at the sub-carrier $k$ will show up at the sub-carrier $k + l$. So if the multipliers are placed at the theoretical positions of the pilots, when the pilots are in these positions a maximum will be achieved because the pilots add up in phase. While if data are transmitted in these carriers, the correlator output will tend to zero. It was seen before that the time offset causes a phase offset in the frequency
domain. Therefore, this metric is sensitive to small time offsets. If the time synchronization procedure is not done perfectly, the summation in (5.16) will not add up in phase. As a result, a maximum will not be achieved even if the pilot carriers are in their positions.

An alternative approach for solving the above mentioned problem is using the metric defined as follows:

$$M[l] = \sum_{k \in S} |c_{m,k+l} \cdot c_{m-1,k+l}^*|. \quad (5.17)$$

In this metric the absolute value computation is done inside the summation which assures that the terms will add up coherently. It depends on the fact that the pilot carriers have boosted power, i.e., its summation will again be larger than the summation when information data is in these sub-carriers. In order to reduce the complexity of this metric the absolute value squared is calculated in order to avoid the use of a square root as seen in (5.18):

$$M[l] = \sum_{k \in S} |c_{m,k+l} \cdot c_{m-1,k+l}^*|^2. \quad (5.18)$$

Another algorithm for estimating the integer frequency offset is proposed in [BJY04]. It uses the crosscorrelation technique instead of the autocorrelation used in the previous approach. As seen before, the autocorrelation of the GI is subject to artifacts caused by multipath propagation and this is avoided by using the crosscorrelation technique. It is based on correlating the received carriers with pilot carriers saved in the receiver, which are not subject to ISI and to time offset. The metric is defined by:

$$M[l] = \sum_{k \in S} |c_{m,k+l} \cdot p_k^*| \quad (5.19)$$

where

$$p_k: \text{pilot carrier corresponding to index } k.$$

The $p_k$ are known to the receiver, since the modulation of the pilot is done using PRBS, the values $p_k$ are the same, and in the particular case of DVB-T they either take the value of $\frac{4}{3}$ or $-\frac{4}{3}$. One more advantage of this method is that a delay line of $N_{FFT}$ is not needed because the metric does not need to use the previous OFDM symbol. Since the absolute value of the pilots stays constant, (5.19) is equivalent to:

$$M[l] = \sum_{k \in S} |c_{m,k+l}|. \quad (5.20)$$

Again squaring the term inside the summation will decrease the complexity and lead to the following metric

$$M[l] = \sum_{k \in S} |c_{m,k+l}|^2. \quad (5.21)$$
5.2 Simulation Model

In this section the building blocks of the simulation model are presented, describing briefly the function of every block inside the model. Then in the following subsections each module is explained in details. Fig. 5.1 shows the top level block diagram of the DVB-T system.

The DVB-T transmitter module generates a stream of OFDM frames according to the structure defined in the DVB-T standard. The channel module has a double function, on the one hand, the time delay $\theta$ and the frequency offset are inserted. On the other hand, the multi-path propagation effect is modeled, with different channel impulse responses. In the DVB-T receiver module the time and frequency synchronization is done. Coarse time and fractional frequency estimation is done in the time domain, i.e., pre-FFT. While, the integer frequency synchronization is done in the frequency domain, i.e., post-FFT. Finally, the evaluation module calculates the quality of the simulated algorithms by taking information from the obtained results, namely the variance of estimation.

5.2.1 DVB-T Transmitter

In order to be able to test the algorithms for synchronization of DVB-T, a DVB-T transmitter was built following the standard [DVB01]. It generates a stream of frames which comply with the DVB-T standard. The first step in the transmitter is generating the MPEG-2 transport packets. In the simulations random data have been used. By this we skip the step of randomizing the data with a scrambler. After the packets are generated they have to be adapted to the channel characteristics of the DVB-T standard. First, we convert the random bits into modulated symbols, with one of the following signal constellations: QPSK, 16-QAM, 64-QAM. The average energy of the constellation is normalized so that the modulated signals satisfy:

$$E\{c_k \cdot c_k^*\} = E\{|c_k|^2\} = 1$$  \hspace{1cm} (5.22)

These symbols will be the data carriers of the OFDM symbols. For (2k mode) the number of useful data carriers = 1512, while for (8k mode) = 6048. After that, the pilot and TPS
carriers are inserted. This is done in the (Pilot, TPS insertion) block seen in Fig. 5.2. The carriers are sorted by a sorting method in order to find the carrier number that corresponds to a pilot or a TPS carrier. Starting from an index of zero, it outputs a null symbol if the index corresponds to a pilot or TPS index, otherwise it reads a new symbol from the input and outputs it. Then the index is increased by one and the sorting is done again. After sorting is finished the symbol stream is ready for modulation of the reference signals. First, the modulation of the pilot carriers is done. In case the incoming carrier is a pilot carrier, then it is modulated with the value:

\[
Re\{c_k\} = \frac{4}{3} \cdot 2 \cdot (\frac{1}{2} - w_k)
\]  
\[
Im\{c_k\} = 0
\]

(5.23)  
(5.24)

where \(\{w_k\}\) are the PRBS defined in the standard. Note that the pilot carriers are transmitted with boosted power:

\[
E\{c_k \cdot c_k^*\} = \frac{4}{3}
\]

(5.25)

They are always placed on the real axis, i.e., the phase can be either 0 or 180 degrees. Once the pilot sub-carriers have been modulated, the TPS carriers modulation is performed. If the input carrier is a TPS carrier, DBPSK modulation is done in such a way that the carriers convey the information about the transmission parameters. The information is stored in a 67-bit word whose bits vary according to the transmission parameters. The relevant parameters in this work are, txmode, GIlength and modtype. This word is error-protected since the last 14 bits are redundant, which are calculated using BCH code. The code generator polynomial is:

\[
h(x) = x^{14} + x^9 + x^8 + x^5 + x^4 + x^2 + x + 1.
\]

(5.26)

The redundancy bits are the remainder of dividing the information bits by the code generator polynomial in modulo-2 arithmetic. The code is generated by the TPS module before reading any sample from the input, because it only depends on the transmission
parameters which are known a-priori. On the other hand, the bits \( s_1 - s_{53} \) depend not only on transmission parameters but also on frame number inside the super-frame. Therefore, the encoding has to be performed 4 times. For the 67-bit word the DBPSK is performed as the following:

If \( s_l = 0 \), \( \text{Re}\{c_{m,l,k}\} = \text{Re}\{c_{m,l-1,k}\} \), and \( \text{Im}\{c_{m,l,k}\} = 0 \)

If \( s_l = 1 \), \( \text{Re}\{c_{m,l,k}\} = -\text{Re}\{c_{m,l-1,k}\} \), and \( \text{Im}\{c_{m,l,k}\} = 0 \)

Note that in these equations: \( m \) denotes the frame number and \( l \) the OFDM symbol index within each frame. An initialization is done which is specified in the standard, it depends on the PRBS that are calculated in the Pilot insertion stage. The TPS carriers are also placed on the real axis, but not sent in a boosted power:

\[
E\{c_k \cdot c_k^*\} = 1. \tag{5.27}
\]

It is worth to note that the TPS carriers which belong to the same symbol carry the same information.

The last task done in the transmitter is the OFDM modulation. It generates the baseband discrete representation of the transmitted signal, which is then transformed into an analog signal by the use of a DAC before sent over air. According to the DVB-T standard the continual sent signal has the equation:

\[
s(t) = \text{Re}\{e^{j2\pi f_c t} \sum_{m=0}^{\infty} \sum_{l=0}^{67} \sum_{k=0}^{K_{\text{max}}} c_{m,l,k} \cdot \Psi_{m,l,k}(t)\}. \tag{5.28}
\]

Where \( \Psi_{m,l,k}(t) \) is the set of basis functions also defined in the standard. The OFDM modulation is done by three steps. First the frequency carriers are reorganized and prepared to the input of the IFFT, which then transforms the frequency carriers to a time domain signal. It reads the carriers \( K \) by \( K \), and reorder them as indicated in the standard. After that it pads the sequence with the necessary number of zeros to get the necessary FFT length. The FFT-length \( N_{\text{FFT}} \) is chosen depending on the transmission mode: (2048 in 2k-mode) and (8192 in 8k-mode). To generate the GI and due to periodic nature of the signal, the last \( L_g \) samples are just copied and appended to the front of the sequence.

### 5.2.2 Channel

After the DVB-T stream is formed in the DVB-T transmitter it is sent to the receiver, via the channel. In our simulations the channel is modeled by the channel module which is shown in Fig. 5.3. First the imperfections are applied to the signal. The time
offset is modeled as a delay in the received signal. It is done by inserting $\theta$ zeros in the complex sample stream. After the time offset, the frequency offset is inserted. As explained in chapter 2, the frequency offset $\varepsilon$ causes a rotation equal to $e^{2\pi\varepsilon/N_{FFT}}$ of the time domain signal. In DVB-T systems we have a continuous transmission which causes the accumulation of the rotation from one OFDM symbol to the next. Three different transmission paths were simulated in this work. These paths are modeled by the block (Multipath propagation).

First channel simulated is the **AWGN channel**. In this case the (Multipath propagation) does nothing, because the (AWGN channel) block inserts Gaussian noise to the signal before it arrives to the receiver. This module already exists as a Co-Centric System Studio module, which offers the possibility of choosing the desired SIR at its output. Another channel simulated is the **fixed channel F1** which is specified in [DVB01]. This channel presents a Ricean power delay profile, and its expression is:

$$r(t) = \rho_0 \cdot s(t) + \sum_{i=1}^{20} \rho_i \cdot e^{-j2\pi\theta_i} s(t - \tau_i)$$

where

$$\rho_0 = \sqrt{10 \cdot \sum_{i=1}^{20} \rho_i^2}$$

The Ricean factor defined as:

$$K = \frac{\rho_0^2}{\sum \rho_i^2}.$$  \hspace{1cm} (5.31)

is set in our simulations to $5 \, dB$.

A third channel simulated in this work is the **Rayleigh mobile channel**. This mobile environment is simulated with a channel consisting of 6 Rayleigh taps. The mobile is assumed to be moving at velocity of 120 km/h. The movement of the receiver will cause the amplitudes of the taps in the CIR to vary with time. The power and delays of the different taps are shown in Table 5.1.
Table 5.1: Delay and power of the Rayleigh Mobile Channel.

<table>
<thead>
<tr>
<th>Path</th>
<th>Relative Delay (μs)</th>
<th>Power (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>-1.5</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>-4.5</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>-7.5</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>-8</td>
</tr>
<tr>
<td>6</td>
<td>17.2</td>
<td>-17.7</td>
</tr>
</tbody>
</table>

5.2.3 DVB-T Receiver

The synchronization procedures that were discussed in Section 5.1 take place at the receiver side of the DVB-T system. Imperfections were inserted to the DVB-T stream while it is transmitted via the channel. At the receiver side we aim on estimating these imperfections (time offset and frequency offset). As seen before the synchronization procedure in DVB-T discussed in this chapter is done by two stages, Pre-FFT and Post-FFT as seen in Fig. 5.4.

Figure 5.4: Building blocks of the DVB-T receiver module.

5.2.3.1 Pre-FFT Synchronization

The Pre-FFT synchronization module estimates the time offset and the fractional frequency offset, while using the time domain signal. After detection of the beginning of a symbol the GI can be removed, in addition the correction of the fractional frequency offset is done within this block. The output of this block will then only suffer from an integer frequency offset.

Fig. 5.5 shows the block diagram of the Pre-FFT synchronizer. The metric signal using any of the metrics discussed before is calculated. As in the IEEE802.11a synchronizer, the metric block outputs three signals: A delayed form of the received signal, \( r[n - n_0] \), where \( n_0 = N_{FFT} + L_g + 1 \) is the delay of the correlator. A metric signal \( m[n] \) which is given to a (Max search) block. The (Max search) aims on finding the argument at which this metric signal takes a maximum. It then waits for \( N_{argmax} \) values to be sure that this is the correct maximum. As a result it outputs the flag (max-valid=1). This flag will
activate the module (Enable) which until then is ignoring all samples that arrive. After it receives the max-valid=1 flag this block will ignore then only the coming $L_g - N_{\text{argmax}} + 1$ samples, because they will still correspond to the GI. Similar to the IEEE802.11a synchronizer the metric block outputs the correlator signal $c[n]$ which is given to the (fractional epsilon estimation) block which estimates the fractional frequency offset only by calculating the angle of the incoming complex samples and divides it by $2\pi$. The values are then delayed by $N_{\text{argmax}}$ samples. Therefore when the (Max search) block outputs the flag (max-valid=1) the fractional frequency correction module will get the $\hat{\epsilon}$ so that it performs the fractional frequency correction. When the fractional frequency correction is to be done, the GI interval will be already removed from the signal. Therefore the rotation of the first incoming sample will not be zero, but $\frac{2\pi \hat{\epsilon} L_g}{N_{\text{FFT}}}$, where $\hat{\epsilon}$ is the estimated fractional frequency offset of the first symbol. Then the rotation at the beginning of each symbol is calculated recursively from the previous estimation as follows:

$$\Phi_{im} = \Phi_{im-1} + 2\pi \hat{\epsilon} + 2\pi \hat{\epsilon} \frac{L_g}{N_{\text{FFT}}}. \quad (5.32)$$

After that, every new sample of the m-th symbol is rotated additionally by $\frac{2\pi \hat{\epsilon}}{N_{\text{FFT}}}$.

The (Metric) block seen in Fig. 5.5 calculates the metric signal depending on one of the metrics discussed before: Maximum Correlation (MC), Maximum Normalized Correlation (MNC), Maximum Likelihood (ML) and ML using the frequency domain pilots (ML-pilots).

All the metrics depend basically on the calculation of the correlated signal. The optimum correlator is discussed in chapter 4. In addition an iterative implementation was introduced, which decreases the complexity of the metric block. For more information about the correlator implementation refer to section 4.4.2.
The block diagrams of the three metrics MC, MNC, and ML were discussed in details in chapter 4. Therefore here we will only explain the block diagram of the ML-pilots metric.

**ML using pilots in the frequency domain (ML-pilots):** The block diagram of this metric is seen in Fig. 5.6. It is composed of the ML metric in addition to the cross-correlation of the received signal with the IFFT of the pilots as seen before.

![Figure 5.6: Building blocks of the ML-pilots metric.](image)

**5.2.3.2 Post-FFT Synchronization**

It has been explained previously in this chapter that the post synchronization stage is in charge of the integer frequency synchronization. This can be done as seen in Fig. 5.7.

![Figure 5.7: Post-FFT synchronizer blocks.](image)

It is similar to the pre-FFT synchronization principle. It is based on calculating the Metric signal and searching for the maximum. But the metric signal calculated in this case is one of the following:
• Autocorrelation of the pilots of 2 consecutive OFDM symbols: depending on the phase property (5.16).

• Autocorrelation of the pilots of 2 consecutive OFDM symbols: depending on the power property (5.17).

• Crosscorrelation of the received OFDM symbol using the saved pilots in the receiver (5.19).

It was seen before that the third metric described above is similar in the case of DVB-T to taking the power of the carriers which are at a given moment in the positions of the multipliers. The (Max search) block is the same as in the pre-FFT synchronizer, but here it has different parameters. The detection threshold and the window needed to validate the maximum are chosen differently depending on the specific metric.

5.3 Simulation Results

In this section, the simulation results obtained using the described algorithms will be presented. We evaluate the algorithms at different SNR values and for the three different channel types: AWGN, F1 and mobile. The AWGN channel is considered the reference since if an algorithm is not useful for AWGN channels then it will not be useful for channels with multipath propagation.

5.3.1 Time Synchronization Evaluation

The algorithms used for the estimation of the time offset will be evaluated in this section. All the simulations were done under an 8k transmission mode and for 16-QAM modulation. The GI inserted has a length of $1/8$ of the OFDM symbol according to [DvbE]. In order to perform these simulations a time offset of 100 samples was inserted in addition to frequency offset of value 5.3. The simulations are done for an SNR of value 6 dB.

**Autocorrelation of GI:** Figures 5.8, 5.9 and 5.10 show the metric signals using the autocorrelation of the GI, $M_{GI}[n]$.

It is clear that all the metrics show distinct peaks, with a distance of one OFDM symbol (9216 samples) in between. The first peak appears at $n = 9315$. Therefore all these metrics can be used for the estimation of $\theta$.

It is clear that the ML metric suffers from false alarm which is seen by the peak at the time instant zero. This is because at the instant zero both the correlation and the
power sum are zero, which cases the metric signal to be also zero. But in continuous mode transmission this can be neglected since it occurs only at the beginning of the signal. This false alarm drawback is more annoying in the burst mode transmission (IEEE802.11a) as described in chapter 4.

**Cross-correlation of frequency domain pilot symbols:** Fig. 5.11 shows the metric signal $M[\theta]$ calculated using frequency domain pilots (lower graph) given by (5.8). Whereas the upper graph shows the contribution of the pilots to the metric signal which
is defined as $M_p[n]$ in (5.8). In this graph only time offset is taken into account.

The $M_p$ metric cannot be used alone, because it gives distinct peaks at different positions. However, it gives more distinct peaks when used in combination with the GI autocorrelation. This makes the estimation error less probable. On the other hand, this metric suffers from a serious problem when it is used under a certain frequency offset. Fig. 5.12 shows the same signal as in the above figure but with a frequency offset $\varepsilon = 1.0$. The distinct peaks disappear, and another peaks appear but not at instants when a complete OFDM symbol is inside the correlation window. Therefore, this metric is only
Figure 5.12: Metric signal using pilot cross-correlation with $\varepsilon = 1.0$.

useful if the correction of the $\varepsilon$ has been carried at an earlier stage or the frequency offset is negligible. As a result, the valid metrics are the ones based on the autocorrelation of GI. Now we will evaluate these metrics for different environments.

It was seen before that if the start of the estimated FFT-window falls within a certain interval then it does not suffer from ICI or ISI. While, if it is not the case, the OFDM symbols will suffer from ICI and ISI, which can be approximated by Gaussian noise with power:

$$\sigma_n^2 = \sum_i |h_i|^2 \left( \frac{\delta_i}{N_{FFT}} - \left( \frac{\delta_i}{N_{FFT}} \right)^2 \right)$$

(5.33)

according to [SFF01], where $\delta_i$ depends on $\theta$. The maximum allowable error will depend on the ICI or ISI noise power that the system can deal with and also on the channel characteristics as seen in (5.33). This will be the condition which limits the maximum allowable time offset. The metrics are evaluated using the following two criteria.

- Measuring the probability that the error of estimation $|\theta - \hat{\theta}|$ is greater than the maximum tolerable time offset.

- Calculating the standard deviation of the estimation error.

The simulations are done for 1000 OFDM symbols.
5.3.1.1 AWGN Channel

The three metrics MC, MNC and ML are evaluated for the AWGN channel. Fig 5.13 shows the probability \(|\theta - \hat{\theta}| > 2\). Using the ML metric, the probability decreases exponentially as the SNR increases. This is not the case of the MNC and MC which show a flat graph after some SNR. For the ML metric for \(SNR > 16\,dB\) no failure occurs for the 1000 bursts and therefore the probability is limited to less than \(10^{-3}\). This would lead to the conclusion that the ML metric in case of continuous transmission performs better than MC or MNC. The standard deviation for the AWGN channel is shown in Fig. 5.14.

![Figure 5.13: Probability of |\theta - \hat{\theta}| > 2 for AWGN channel.](image)

5.3.1.2 Fixed Multipath Channel F1

In the previous results for an AWGN channel, the MC metric performs bad compared to the ML and MNC metrics. Therefore, in testing multipath channels, the MC metric is excluded because it is not expected to perform better than in an AWGN channel. The two metrics ML and MNC, are tested on a Fixed multipath channel (F1) whose CIR was seen in (5.29). The probability that the error is larger than 7 samples is plotted in Fig. 5.15. It is seen that the ML metric drops to a very low probability < \(10^{-3}\) for an \(SNR > 15\,dB\). The performance of the estimator decreases compared to the AWGN case and therefore the maximum tolerable offset value is relaxed to 7. In addition, the standard deviation is shown in Fig. 5.16. It is clear that even in multipath channel the ML metric outperforms the MNC metric.
5.3. Simulation Results

5.3.1.3 Mobile Channel

The two metrics ML and MNC are tested for a mobile channel, where the receiver is said to be moving with a velocity of 120 km/hr. The ML metric outperforms the MNC metric even in mobile environment as seen in Fig. 5.17. It is seen that the $p(|\text{error}| > 5)$ does not reach $10^{-3}$ as for the previous environments. The standard deviation of ML and MNC for the mobile channel are seen in Fig. 5.18.

The standard deviation of the ML metric in a mobile channel is almost 3 for high SNR, where for F1 it reaches slightly above 1. This means that for the mobile channel
Figure 5.16: Standard deviation of $|\theta - \hat{\theta}|$ for F1 channel.

Figure 5.17: Probability of $|\theta - \hat{\theta}| > 5$ for mobile channel.

a more complex estimator maybe needed. Optimization of the ML metric is discussed in section 5.5.

5.3.2 Frequency Synchronization Evaluation

In this section the different algorithms used for frequency estimation will be evaluated. Since different approaches are used to evaluate the fractional and integer frequency offset, we will describe in separate sections the performance of the algorithms for fractional
Figure 5.18: Standard deviation of $|\theta - \hat{\theta}|$ for mobile channel.

frequency offset as well as integer frequency offset.

5.3.2.1 Fractional Frequency Offset

The methods described in this work for the estimation of fractional frequency offset depend on calculating the angle of the correlated signal. So the three metrics ML, MNC and MC would yield the same results because they contain the same correlated signal. The estimator is only valid at the instants were the correlator window is completely occupied by an OFDM symbol. The estimator has a lock-in range between $-0.5$ and $0.5$. In order to evaluate the performance of the estimators some requirements are set in order to see if the estimators are able to obtain the required accuracy.

As seen in [Mo94] the frequency offset causes an $SNR$ degradation lower bound defined by:

$$SNR \geq SNR_i \frac{\left(\frac{\sin \pi \varepsilon}{\pi \varepsilon}\right)^2}{1 + 0.5947 \cdot SNR_i (\sin \pi \varepsilon)^2}$$

(5.34)

where $SNR_i$ stands for the initial $SNR$ value. Therefore, for an allowable $SNR$ degradation, a maximum tolerable error can be chosen. Another equation given in [SF99] expresses the maximum tolerable frequency offset given a certain $SNR$ and allowable $SNR$ degradation ($SNR_{deg}$).
\[ \varepsilon < \frac{\sqrt{3}}{\pi} \sqrt{\frac{1}{SNR_i} \left( 1 - \frac{1}{SNR_{deg}} \right)} \]  

(5.35)

If the required performance of our synchronizer is set to a maximum SNR degradation of 0.1 dB, then according to (5.34) we get: For \( SNR_i = 12\, dB \), \( |\varepsilon| < 0.016 \); while for \( SNR_i = 15\, dB \), \( |\varepsilon| < 0.012 \). Now if considering the same performance \( SNR_{deg} = 0.1\, dB \) and using (5.35) we get: For \( SNR_i = 12\, dB \), \( |\varepsilon| < 0.0209 \); while for \( SNR_i = 15\, dB \), \( |\varepsilon| < 0.014 \). Therefore, as a rule of thumb we can say that for \( SNR_i = 12\, dB \) the maximum allowable fractional frequency estimation error is \( |\varepsilon| < 0.015 \), while for \( SNR_i = 15\, dB \) the maximum allowable fractional frequency estimation error is \( |\varepsilon| < 0.01 \).

The \( SNR \) under which the synchronizer will be working is given by the requirements of the DVB-T receiver. It depends on the transmission parameters. However, for the transmission parameters given before we can approximate that for an AWGN and a fixed multipath channel (F1) an \( SNR \) of 12 dB is required while for a mobile channel an \( SNR \) of 15 dB is required. Therefore, for an AWGN channel or F1 channel the condition to meet would be:

\[ |\varepsilon_{max}| < 0.015. \]  

(5.36)

While, for a mobile channel, which presents fading taps, the condition to meet is

\[ |\varepsilon_{max}| < 0.01. \]  

(5.37)

In the following the simulation results for the AWGN and mobile channel types will be shown.

**AWGN Channel:** Fig. 5.19 plots the standard deviation of \(|\varepsilon_f - \hat{\varepsilon}_f|\) for different \( SNR \) values, for the three metrics (MC, MNC and ML). The plots of the three metrics coincide since the three metrics have the same correlated signal. As seen before the synchronizer in an AWGN channel does not need to work below 12 dB, therefore it is clearly seen that the deviation of the estimation error for this \( SNR \) is about 0.0015 which is well below the limit set in (5.36). This means that the fractional frequency estimator in the AWGN channel performs very good and needs no additional correction.

**Mobile Channel:** To study the performance of the ML metric in fractional frequency estimation in multipath environment a mobile channel is simulated. Fig. 5.20 plots the standard deviation of the estimation error in a mobile channel. We observe that for \( SNR = 15\, dB \) the standard deviation of the estimation error is about 0.021 which is above the limit set in (5.37). This means, according to (5.34) that if the ML estimator is used, the synchronizer will degrade the \( SNR \) in about 0.4 dB instead of 0.1 dB. Therefore, an
5.3. Simulation Results

Figure 5.19: Standard deviation of the estimator error $|\epsilon_f - \hat{\epsilon}_f|$ in AWGN channel.

Figure 5.20: Standard deviation of the estimator error $|\epsilon_f - \hat{\epsilon}_f|$ in a mobile channel.

additional technique is needed in this case to improve the performance of the synchronizer. This will be further discussed in section 5.5.

5.3.2.2 Integer Frequency Offset

As seen in section 5.1.2.2, five algorithms are given for the estimation of the integer frequency offset. Namely these are: auto-pilots-phase (5.16), auto-pilots-power (5.17), auto-pilots-power-squared (5.18), pilots-magnitude (5.20) and pilots-power (5.21). The performance of these algorithms can be seen by just looking at the metric signal, if
a peak is present at the expected position (i.e., yields a perfect estimation). The algorithm named auto-pilots-phase did not perform well in simulations where the timing estimation is not perfect. For OFDM frames where $\theta$ differs from the previous frame, the correlation terms do not add up in phase and no maximum is achieved. For the other four algorithms, simulations are done for an AWGN channel, fixed multipath channel, and mobile channel, with two $SNR$ values: very low $SNR = 0 \text{dB}$ and $SNR = 10 \text{dB}$ which is the lowest $SNR$ at which the synchronizer is expected to work. In the figures 5.21 till 5.41 the index $i$ is given by $i = k + m \cdot 8192$.

**AWGN channel:** Fig. 5.21 shows the auto-pilots-power-squared metric signal for $SNR = 10 \text{dB}$. Similarly Fig. 5.22 plots the metric auto-pilots-power described in (5.21) for $SNR = 10 \text{dB}$. These metrics are obtained using the ML metric for the pre-FFT synchronization stage. Distinct peaks separated by 8192 time instants which is equal to $N_{FFT}$ can be seen. Therefore, for an AWGN channel, with the given parameters of transmission, both algorithms give exact estimation of the integer frequency for a synchronizer operating minimum at 10 dB.

![Figure 5.21: Auto-pilots-power-squared metric signal for AWGN channel with $SNR = 10 \text{dB}$.](image)

The performance of the above algorithms is analyzed also for $SNR = 0 \text{dB}$, which is an extreme case because according to the DVB-T standard, the DVB-T system should be always working at more than 3 dB. The metric signals (5.18) and (5.21) for this case are seen in Figures 5.23 and 5.24 respectively. The metrics based purely on the power do not give a reliable estimation since for some symbols no distinct peak is seen. This is a logical consequence due to the fluctuations suffered by the pilot carriers because of the additional
5.3. Simulation Results

Figure 5.22: Pilots-power metric signal for AWGN channel with $SNR = 10 dB$.

noise. Therefore, the metric signal seen in Fig. 5.23 based on the autocorrelation property works better than that based purely on the power seen in Fig. 5.24.

Figure 5.23: Auto-pilots-power-squared metric signal for AWGN channel with $SNR = 0 dB$.

Fig. 5.25 shows the auto-pilots-power metric (5.17) without squaring. We can notice that the peaks in this case are more clear than in Fig. 5.23. This means that the squaring degrades the performance in the low $SNR$. As a result, for low $SNR$ the metric described in (5.17) should be used even though it is with the highest complexity.

**Fixed channel (F1):** Similar to the AWGN channel, the metrics are tested in Fixed
Figure 5.24: Pilots-power metric signal for AWGN channel with $SNR = 0 dB$.

Figure 5.25: Auto-pilots-power metric signal for AWGN channel with $SNR = 0 dB$.

channel (F1) for $SNR = 10 dB$ and $SNR = 0 dB$. Figures 5.26, 5.27, 5.28 and 5.29 show the four metrics for $SNR = 10 dB$. For $SNR = 10 dB$ the four metrics yield good estimations, which means that the corruption of the GI samples due to multipath propagation did not heavily affect the post-FFT estimators. Therefore, a threshold can be applied to get the positions of the maxima.

The metric signals for $SNR = 0 dB$ are seen in Figures 5.30 till 5.33. The metrics seen in Figures 5.32 and 5.33 which are based on the magnitude of the pilots, did not
5.3. Simulation Results

Figure 5.26: Auto-pilots-power metric signal for F1 channel with $SNR = 10 dB$.

Figure 5.27: Auto-pilots-power-squared metric signal for F1 channel with $SNR = 10 dB$.

show a good performance. While the first two metrics which rely on the autocorrelation property give better results. Once more, the squaring makes the metric less reliable and therefore even for the F1 channel in low $SNR$ the auto-pilots-power (5.17) metric is preferable.

Mobile channel: In a mobile channel for $SNR = 10 dB$ all metrics work well as seen in Figures 5.34 till 5.37. However, in the mobile channel the value of the maximum varies strongly due to the fading effect and therefore, fixed threshold could not be anymore
5. DVB-T Synchronization

Figure 5.28: Pilots-magnitude metric signal for F1 channel with $SNR = 10\, dB$.

Figure 5.29: Pilots-power metric signal for F1 channel with $SNR = 10\, dB$.

used for detecting the maxima.

On the other hand, for $SNR = 0\, dB$, all the metrics perform well, even the metrics that depend purely on the pilots magnitude (power) which did not perform good in AWGN and F1 environments.

We can conclude from the above, that if the synchronizer will work with a minimum $SNR$ of $10\, dB$ according to the German Parameters [DVB01], the best suitable metric
5.3. Simulation Results

Figure 5.30: Auto-pilots-power metric signal for F1 channel with $SNR = 0 dB$.

Figure 5.31: Auto-pilots-power-squared metric signal for F1 channel with $SNR = 0 dB$.

for the integer frequency estimation is the (pilots-power), being the less complex and with a very well performance. While if the synchronizer is expected to work in low $SNR$, then the suitable metric is the (auto-pilots-power) even though it is the most complex metric for a hardware implementation.
5. DVB-T Synchronization

Figure 5.32: Pilots-magnitude metric signal for F1 channel with $SNR = 0 dB$.

Figure 5.33: Pilots-power metric signal for F1 channel with $SNR = 0 dB$.

5.4 Fixed Point Implementation

This section describes a fixed point implementation for the pre-FFT synchronizer for DVB-T systems using the ML metric. Since according to the floating point simulations, the ML metric seems to perform superior to MC and MNC. In addition, an ML synchronizer can be extended to more robust synchronizers which depend on more than one OFDM symbol. This will be further explained in section 5.5.
5.4. Fixed Point Implementation

Figure 5.34: Auto-pilots-power metric signal for a mobile channel with $SNR = 10\, dB$.

Figure 5.35: Auto-pilots-power-squared metric signal for a mobile channel with $SNR = 10\, dB$.

5.4.1 ML Algorithm

Fig. 5.42 shows the building blocks of the fixed point ML model. The floating point model has the same blocks, except that it is done with complex signals, while in HW implementation only real signals are used. Therefore, the complex signals are represented by two lines, i.e., two real signals.

In Fig. 5.42, the word lengths are also specified. The integer word lengths are determined depending on the range of values taken by the signals. The received signal
Figure 5.36: Pilots-magnitude metric signal for a mobile channel with $SNR = 10\, dB$.

Figure 5.37: Pilots-power metric signal for a mobile channel with $SNR = 10\, dB$.

takes values between $-0.125$ and $0.125$. Therefore, the first two bits after the point need not be used because they are always zero and the integer word length is set to $iwl = -2$. A similar analysis is applied to the rest of the signals. The fractional word length values have to be tuned, more details about the specific fractional word lengths for every signal will be given in the next section. The notations $u$ and $s$ in Fig. 5.42 stand for unsigned and signed variables.
5.4. Fixed Point Implementation

Figure 5.38: Auto-pilots-power metric signal for a mobile channel with $SNR = 0\, dB$.

Figure 5.39: Auto-pilots-power-squared metric signal for a mobile channel with $SNR = 0\, dB$.

Correlation and Power Sum

In this block the correlation and the power sum of the received signal are calculated. The correlation and the power sum are implemented in the same module as seen in Fig. 5.43 in order to save a delay line of length $(N_{FFT} + L_g = 9216)$ delay taps. The implementation of the correlation is done recursively according to the concept of the double multiplier as seen in section 4.4.2. Note that for calculating the correlation, not all the fractional bits of the received signal $(fwl_r)$ are used, but only some of them $(fwl_{c_i})$. On the other hand, two implementations are proposed for the power sum calculation.
5. DVB-T Synchronization

More details about the different implementations of power sum and absolute value will be discussed in section 5.6.

5.4.2 Choice of Optimal Wordlength

In this section, the loss of performance of the ML synchronizer due to the change from floating to fixed point will be analyzed. After the tuning of the initial and intermediate fractional word lengths, the list of used fractional word lengths is given in Table 5.2.

The parameter still to be tuned is the fractional word length of the metric signal.
5.4. Fixed Point Implementation

Figure 5.42: Fixed point module of the ML algorithm.

Figure 5.43: Correlation and Power Sum block.

<table>
<thead>
<tr>
<th>$fwl_{ci}$</th>
<th>$fwl_{pa}$</th>
<th>$fwl_c$</th>
<th>$fwl_p$</th>
<th>$fwl_{p-p}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>20</td>
<td>21</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.2: Fractional word length values.

Fig. 5.44 shows the probability that the estimation error is more than 2 for different values of $fwl_m$ for an AWGN channel. As it is shown the minimum fractional word
length of the metric signal ($fwl_m$) to obtain a reasonable result similar to the floating point is $fwl_m = 15$.

It was seen before that for AWGN channel the expected $SNR$ is about $12 \, dB$. For $fwl_m = 17$ the curve is very close to the floating point curve especially around $SNR = 12 \, dB$. In addition, it should be noted that in changing to a fixed point algorithm, an approximation was used for calculating the absolute value of the signal. This degrades also the performance of the algorithm. But as seen in the above figure, this degradation is negligible because the two curves converge to one curve for $SNR > 10 \, dB$.

5.5 Algorithmic Optimization

The Maximum Likelihood (ML) estimator has been proved to be one of the best estimation method for continuous mode transmission systems like DVB-T. It was shown by the simulation results that the ML metric outperforms the MC or MNC metrics for DVB-T. However, the accuracy of the ML metric decreases for mobile channels, i.e. when a moving receiver is used. According to the $SNR$ degradation that the system can support, a more complex estimator is needed. Van de Beek proposed in his paper [BB97] an extension to the ML estimator based on an observation of several OFDM symbols. Yet, this method has a drawback that the complexity of the stored history increases linearly with the number of OFDM symbols used. In this section, we propose a new approach which also takes into account the previously received OFDM symbols at a least price in complexity. Using the stored history of the received symbols, this approach proves to outperform the simple ML metric without a considerable increase.
in complexity. Such approaches which take into account several OFDM symbols are only possible in continuous mode transmission, where the time required for estimation is not as determining as in burst mode transmission. In this section, we focus on the Pre-FFT synchronization stage of DVB-T since it is the stage where the ML algorithm is used for estimating the coarse timing offset and fractional frequency offset. In the following a description of the averaged ML algorithm proposed by Beek in [BB97] is given followed by a detailed study of the proposed Weighted ML (WML). The better performance of the WML is illustrated by simulation results for the mobile channel environment.

5.5.1 Averaged Maximum Likelihood

An approach to increase the reliability of the ML algorithm, proposed in [BB97, BS96], is based on using multiple OFDM symbols in the estimation process. This approach is only possible in a continuous transmission, where the time required for the estimation is not as critical as in a burst-mode transmission. The estimator in this case is obtained by performing $M$ correlations and $M$ power-sums for $M$ consecutive OFDM symbols. The $M$ correlations are added up to form a composed autocorrelation $\Gamma[n]$ as well as the $M$ power-sums are added up to form a composed power-sum $\Phi[n]$. The averaged metric signal is thus defined as:

$$\Gamma[n] = \sum_{m=0}^{M-1} c[n + m \cdot (N_{FFT} + L_g)]$$

$$\Phi[n] = \sum_{m=0}^{M-1} p[n + m \cdot (N_{FFT} + L_g)]$$

$$m_{AML}[n] = |\Gamma[n]| - \rho \Phi[n]$$

where $L_g$ is the Guard Interval length. To store one OFDM symbol, $3 \cdot (N_{FFT} + L_g)$ delay taps would be required: 2 of which are needed for the real and imaginary parts of the complex correlation signal, and 1 for the real power signal. This algorithm would thus require $3 \cdot M (N_{FFT} + L_g)$ delay taps in order to store the $M$ consecutive OFDM symbols which are needed for the calculation of $m_{AML}[n]$. Approximating the hardware complexity to store 1 OFDM symbol, we assume a 10 bit word-length in the fixed-point implementation. In the DVB-T case (8k mode), each delay line consists of $N_{FFT} + L_g = 9216$ delay taps. Using a RAM, we can approximate the needed storage capacity for 1 OFDM symbol as follows: 10 bits/delay tap $\times$ 9216 delay taps = 92160 bits or 90 kbits. Estimating the gate complexity of a delay tap using a register chain, we can use the following rule of thumb: 10 gates/bit $\times$ 10 bits/delay tap = 100 gates/delay tap. As a result, storing 1 OFDM symbol would then require around 921600 gates or
gate-equivalence. Therefore, the complexity of the storage unit necessary for calculating $m_{AML}[n]$ would dramatically increase if $M$ increases. The complexity grows linearly with $M$. However, it is desired to increase the reliability of the ML algorithm without dramatically increasing the complexity, i.e., deriving an approach whose complexity is independent of the number of OFDM symbols used.

### 5.5.2 Weighted Maximum Likelihood (WML)

The WML algorithm is an extension of the ML approach which leads to a higher performance, without a high increase in complexity. We have published this approach in [EL07]

#### 5.5.2.1 Concept

The block diagram of the WML is seen in Fig. 5.45.

![Block diagram of the WML metric.](image)

The 'Correlation & Power Sum' block outputs the current correlation and power signals (i.e. of the current OFDM symbol). The 'Corr_accumulator' block calculates a weighted correlation depending on a weighting factor $W > 1$. This block stores an accumulated correlation history of all previous OFDM symbols. When a new OFDM symbol is processed, the stored history is updated accordingly. In the weighting process, the new-coming OFDM symbol is assigned a weight of $\frac{1}{W}$ while the accumulated correlation history gets a weight of $\frac{W-1}{W}$. This procedure takes into account the history of weighted correlations without the need to save several OFDM symbols since the stored history is updated for every received symbol. Mathematically, the output of the 'Corr_accumulator' block is written as:

$$
\psi_{i,k}[n] = \frac{1}{W}(c[n] + (W - 1)\psi_{i,k-1}[n])
$$

(5.41)
where \( i \) denotes the sample number and \( k \) the OFDM symbol number. In a similar procedure, the ‘Pow\( \sum\)accumulator’ block calculates a weighted accumulation of power sums given as:

\[
\psi^p_{i,k}[n] = \frac{1}{W}(p[n] + (W - 1)\psi^p_{i,k-1}[n]).
\]  

(5.42)

where \( p[n] \) is the power sum. The Metric block thus yields

\[
m_{WML}[n] = |\psi^c[n]| - \rho \psi^p[n].
\]  

(5.43)

The timing offset \( \theta \) is estimated by finding the instant at which the metric signal \( m_{WML} \) takes its maximum value. The frequency offset \( \varepsilon \) is then estimated by calculating the argument of \( \psi^c[n] \) at the estimated timing offset \( \hat{\theta} \).

5.5.2.2 Simulation Results

In the following section the performance of the WML metric will be compared to that of the ML metric in a mobile channel environment. The simulation results will be given for the coarse timing offset estimation and fractional frequency estimation.

Coarse Time Estimation

As explained before, finding the sample at which the metric signal gets a maximum yields the estimate of the timing offset. Fig. 5.46 shows the metric signal \( m_{WML}[n] \) of the WML for three different values of \( W \).

It can be seen that the metric signal \( m_{WML}[n] \) using different values of \( W \) yields distinct peaks at a distance difference of 9216 samples. The first peak is obtained at the instant \( n = 9315 \) which equal to \( (N_{FFT} + L_g + \theta - 1) \). The indicator that is used to evaluate the quality of this metric is the standard deviation of the estimation error \( |\theta - \hat{\theta}| \).

The number of simulated OFDM symbols, i.e., the number of estimations is set to 10000. As mentioned above, these simulations are done for a mobile channel. Fig. 5.47 shows the standard deviation \( \sigma_\theta \) of the estimation error \( |\theta - \hat{\theta}| \) when the ML metric is used as well as the WML for different values for \( W \). Using the ML metric, \( \sigma_\theta \) is almost 3 at high SNR for a mobile channel. Earlier simulations have shown that at the same conditions for a fixed channel, \( \sigma_\theta \) is slightly above 1. Therefore, depending on the SNR degradation that the system can support, a more complex estimator is needed.

Using the WML metric with \( W = 3 \), the standard deviation approaches a value of 2 even for low SNR values \((2 - 4 \text{ dB})\). The simulations shown in Fig. 5.47 prove that WML has more accuracy than ML in estimating the timing offset. Achieving an accuracy of less than 2 samples at low SNR makes the approach robust in fading channels. However, it is important to know how large the design parameter \( W \) could be chosen. The
5. DVB-T Synchronization

Figure 5.46: Weighted Maximum Likelihood (WML) metric signal with g\(W = 4\), r\(W = 3\) and b\(W = 2\).

Figure 5.47: Standard Deviation of \(|\theta - \hat{\theta}|\) for ML and WML.

Hardware complexity is the same independent of the value of \(W\). However, we have the limitation for the assumption that \(\theta\) is constant over a certain time window. In WML, the correlation \(\psi_{c,k,n}^f|\) is an accumulated value, where each adding term loses its weight according to the formula \((\frac{W-1}{W})^x\). The variable \(x\) denotes the number of OFDM symbols that have been received after a certain term of interest was added. Obviously, there is a moment when the contribution to the summation becomes negligible. The larger \(W\) is, the more consecutive symbols contribute significantly to the accumulation. For \(W = 2\)
we could approximate that 5 symbols contribute significantly to the accumulation, while for $W = 3$ we can approximate the number of effective OFDM symbols with 7. Fig. 5.48 shows the performance of the WML approach for increasing values of $W$ in the mobile channel described earlier. As $W$ increases, the enhancement of the estimation is clearly established.

![Figure 5.48: Standard Deviation of $|\theta - \hat{\theta}|$ for WML.](image)

**Fractional Frequency Estimation**

As seen before, the fractional frequency estimation is done by calculating the angle of the correlation at the instant where the metric signal takes a maximum. In the WML the correlation is obtained from the accumulation of the weighted correlations. The estimator presents values that range from $-0.5$ to $+0.5$, which limits the lock-in range of the estimator to these values. The $SNR$ under which the synchronizer will be working is determined by the requirements of the DVB-T receiver to obtain a Quasi Error Free (QEF) transmission. As seen before, for the mobile channel, the minimum required $SNR$ is $15\, dB$, which means that the maximum tolerable offset is $|\varepsilon_{max}| = 0.01$.

Fig. 5.49 plots the standard deviation of the frequency estimation error as a function of $SNR$ for the ML algorithm and the WML. It can be seen that the ML estimator does not meet the condition that the error should be below 0.01 for $SNR = 15\, dB$, and the standard deviation takes the value of 0.021. This means that using the ML estimator will degrade the $SNR$ by around 0.4 $dB$ which is larger than the maximum allowable degradation of the synchronizer 0.1 $dB$. The WML with $W = 2$ has a significantly better performance.
5. DVB-T Synchronization

Figure 5.49: Standard deviation of $|\varepsilon_f - \hat{\varepsilon}_f|$ for a mobile channel using ML and WML.

Performance than the ML estimator. The standard deviation of $|\varepsilon_f - \hat{\varepsilon}_f|$ reaches a value of 0.008 for $SNR = 15 dB$, which is below 0.1 dB. The increase of $W$ to 3 does bring an improvement, although it is not as big as the one achieved by the first change. Fig. 5.50 presents the improvement achieved in the frequency estimation error by increasing $W$ for the WML algorithm.

Figure 5.50: Standard deviation of $|\varepsilon_f - \hat{\varepsilon}_f|$ for a mobile channel WML for different values of $W$. 

5.6 Complexity Analysis

In the following section, a study of the complexity of the fixed point implementation of the ML algorithm is given. In addition, possible implementations of the WML are studied. The complexity of the WML is compared to that of AML. The fixed point implementation of the ML algorithm is seen in Fig. 5.42. In order to study the complexity of the algorithm, the possible implementation methods for the correlation and the power sum are studied. In addition, the implementation of the absolute value is analyzed.

5.6.1 ML Algorithm

Correlation

As seen in section 4.4.2, the correlator can be implemented recursively in order to get the less complex hardware implementation. In the DVB-T synchronizer the correlator is implemented in the same way as for IEEE802.11a.

Power Sum

The power sum should be calculated over the complete correlation window. This can be achieved by two ways. The first possibility is to use the iterative implementation illustrated in Fig. 5.51, in addition a delay line should be used to store the $N_{FFT}$ previous power sum values as seen in Fig. 5.52.

![Figure 5.51: Iterative implementation of the power sum.](image)

The other method to avoid using a delay line is to calculate the power sum iteratively over the whole correlation window. This is shown in Fig. 5.53. The first implementation...
would save 4 squaring units and the accumulator would require 4 inputs less than for the second choice. On the other hand, the delay tap of the feedback should be substituted by a delay line of length $N_{FFT}$. The additional delay line would be of higher complexity and thus the second option of calculating the Power sum over the whole correlation window calculated iteratively would be favorable. The power sum and the correlation are implemented in the same module to save a delay line of length $N_{FFT} + L_g = 9216$ delay taps as seen in Fig. 5.43. The complexity of the ‘Correlation & Power sum’ module is given by:

- 2 delay lines each of length $N_{FFT} + L_g = 2 \times (8192 + 1024) = 18432$ delay taps
- 3 additional delays (for the feed-back of the correlation & power sum)
- 16 multipliers (8 for the squaring operation)
5.6. Complexity Analysis

- 3 adders (two 5-input adders and one 9 input-adder)
- 1 shifter (for the division by 2)

Using a separate delay line for power sum would allow to store in shift register the values of the power \(|r[n]|^2\), instead of received samples, saving 6 multipliers. Thus with our implementation, we save 9216 delay taps at an expense of 6 multipliers. On rule-of-thumb calculations, we can estimate the gate complexity of the 9216 delay taps as follows (assuming a word length of 10 bits): 9216 delay taps \(\times 10\) gates/bit \(\times 10\) bits/delay tap = 921600 gates. For the multipliers: 6 multipliers \(\times 10\) gates/bit \(\times (10 \times 10)\) bits/multiplier = 6000 gates. Therefore, implementing correlation and power sum in one module is much favorable saving around 921600 – 6000 = 915600 gates.

**Absolute Value**

An exact implementation of the 'Absolute Value' module would require the use of a square root unit which is highly complex. Nevertheless, it was seen in section 4.6 two different approximations can be used for the absolute value calculation. The two approximations used gave reasonable results. In the implementation of the ML algorithm for DVB-T the approximation

- \(|z| \approx \alpha \max(|z'|, |z''|) + \beta \min(|z'|, |z''|)\) with \(\alpha = 1\) and \(\beta = \frac{1}{4}\)

is used. \(z'\) and \(z''\) are the real and imaginary parts respectively. The values of \(\alpha\) and \(\beta\) can be differently chosen, but this choice is easy to implement in hardware since a multiplication by 0.25 can be implemented by a shift of 2 bits to the right.

The implementation of the absolute value module is shown in Fig. 5.54. The complexity

![Diagram of absolute value implementation](image)

Figure 5.54: Implementation of the absolute value calculation.
<table>
<thead>
<tr>
<th>Adders</th>
<th>Multipliers</th>
<th>Shifters</th>
<th>Delay Taps</th>
<th>Comparators</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>17</td>
<td>2</td>
<td>(2 \cdot (N_{FFT} + L_g) + 3 = 18435)</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5.3: Complexity of the ML algorithm.

- 3 comparators
- Two 1-bit shifters
- One 2-input adder

Therefore, the overall complexity requirements of the ML algorithm is given in Table 5.3

5.6.2 WML Algorithm

Two blocks should be added to the ML module in order to obtain a WML module. The fixed point implementation of the WML is shown in Fig. 5.55.

There is more than one realization to implement the ‘Corr_accumulator’ and ‘Pow_sum_accumulator’ modules of the WML algorithm in hardware. One alternative could be to use a RAM module to store the history. In that case, an address counter would be needed to generate the ‘read’ and ‘write’ addresses. Another alternative is to use a delay line of registers where reading and writing would need additional multiplexer, demultiplexer and an address counter to be performed. This implementation is shown in Fig. 5.56. Only the real part is depicted in this figure, nevertheless same architecture.
The implementation of these new modules would require

- Three 2-input adders
- 3 memory register lines of length $N_{FFT} + L_g$
- 3 counters
- 3 multipliers
- 3 dividers: for the division to be simpler, it is recommended that $W$ takes an even value
- 3 multiplexers of $N_{FFT} + L_g$ inputs
- 3 demultiplexers of $N_{FFT} + L_g$ outputs

The memory register lines will be the most complex additions. Nevertheless, the length of these lines does not grow with $W$. The multiplexers and demultiplexers with the large number of inputs and outputs greatly increase the computational complexity. A solution to this is to use a delay line (a shift register) instead of the registers used in this implementation. Fig. 5.57 shows a realized alternative using a shift register as a delay line to store the history. This avoids the extra use of the multiplexer, demultiplexer, and address counter because the ‘read’ and ‘write’ ports are then fixed to the first and last
registers respectively. The delay line has a length of $N_{FFT} + L_g$ taps, which is equal to the length of one OFDM symbol.

For every sample of the OFDM symbol, the history of the accumulated correlations is multiplied by $W - 1$, and then added to the correlation of the current OFDM sample. The result of this summation is divided by $W$ as stated in (5.41). The output $\psi_{i,k}^c[n]$ is then fed-back to the delay line at the ‘write’ port. This updates and shifts the delay line by 1 tap. After $N_{FFT} + L_g$ steps, the total delay line is updated. In the fixed-point implementation, two structures as in Fig. 5.57 are needed for the ‘Corr_accumulator’ block since the correlation signal is a complex one. The ‘Pow_sum_accumulator’ defined in (5.42) is implemented using the same structure seen in Fig. 5.57. Nevertheless, this block requires only one structure since the power signal takes only real values. According to this proposed structure, the number of delay taps needed to compute $m_{W,ML}[n]$ is independent of $W$. If $W$ is chosen as a $2^n$ number (2, 4, etc...), the implementation of the ‘Multiplier $W - 1$ ‘ block can be optimized and realized in a form of an ‘Adder’. At the same time, the ‘Divider $W$’ block can be realized in a form of a ‘Shifter’ which will solve the problem of implementing a divider in hardware. The simulations shown before show that as $W$ increases, the algorithm delivers a more accurate estimation while the number of delay taps remains constant. This was not the case with the averaged Maximum Likelihood where the complexity is directly proportional to the number of OFDM symbols $M$ to be averaged as seen earlier.

Figure 5.57: Implementation structure of the Corr_accumulator for WML using a shift register storage.
6 Cost Analysis

In the following chapter, an overview on cost and effort required for a project is given. The modeling of different cost functions is shown. Effort needed for a project can be estimated using different methods. Effort estimation methods can be classified into two classes. One class is based on estimating the effort using only the length of source code in lines if code (LOC), but independent of the project parameters. Examples of this class are the Walston-Felix Model, Bailey-Baisili Model and the COCOMO basic model. These effort estimation models are however not very accurate, since the only parameter is the length of code. Therefore, the product attributes, computer attributes, personnel attributes and project attributes are not taken into account. Another class of effort estimation models are based on parameters which can be tailored to different project modes. An example is the COCOMO costing model [B081]. In this chapter a detailed explanation of the two classes of costing models is given. In Literature, these costing models are mainly used for software projects. I have considered transferring and applying them to our project, which is a hardware application. We assume that writing code (SystemC, vhdl, etc...) for hardware is not of big difference than writing code for software projects. Nevertheless, we neglect special hardware-oriented effort such as timing constraints, place-and-route, etc..., which are not present in software. Due to the fact that all calculations are done for the different implementations in our project, comparing relative results becomes possible.

The Norden-Rayleigh curve for estimation of manpower and/or staffing needed for a project will be also explained. Depending on the effort needed for a project, the Norden-Rayleigh curve shows how the manpower needed for a project is distributed over time. Nevertheless, integrating the Rayleigh curve leads to the cumulative Rayleigh distribution which plots the distribution of effort over time.

Effort estimation methods and manpower distribution curves are used for cost analysis of given projects. Therefore, these methods will be further used in order to estimate the cost needed by some of the synchronization algorithms discussed in chapters 4 and 5.
6.1 Cost Estimating Models

In this section a general overview on the typical cost models found in literature will be given. An algorithmic effort/cost model can be built by analyzing the costs and attributes of complete projects. A mathematical formula is used to predict costs. As already mentioned before, the effort estimation models are two classes. In the first class (basic model), the mathematical formula used for effort estimation is based on an estimate of the project size. However, in the other class estimates for number of programmers, hardware capabilities, time needed for a project, etc ... are included.

Since the parameters for the basic models of effort estimation are independent of the project itself, they are considered unrealistic. What differs from a project to the other is not only the length of code (LOC), but also other conditions which play a role in the effort needed for a project. In order to refine the estimation of effort, an new effort estimation model was developed by Boehm in his book (The Software Engineering Economics) [B081]. This cost model is known as COCOMO (Constructive Cost Model). The parameters of COCOMO change depending on the project type, staffing and other factors. These factors are modeled by cost drivers which are modeled by a multiplier in the effort estimation equation. A detailed explanation about the COCOMO will follow.

The above estimation models are known in literature as cost models. Another type of estimation models are the constraint models. Constraint models demonstrate the relationship over time between the two parameters of effort, duration or staffing level. The most commonly used constraint model in literature is the Norden-Rayleigh curve for manpower distribution. Norden observed that the Rayleigh distribution provides a good approximation of the manpower curve [PN97]. Further detailed explanation of the Norden-Rayleigh manpower distribution is given throughout this section.

6.1.1 Basic models for Effort Estimation

A typical cost model is derived using regression analysis on data collected from past software projects. Therefore, it is built by analyzing the cost/effort of completed projects. In the basic models for effort estimation a mathematical formula is used to predict effort based only on project size (kilo lines of code (kLOC)). Kitchenham described in [Ki90] 13 different models which have been developed from empirical observations. The models of estimating effort say that as size increases more effort is needed because of the need for larger teams, more complex management and so on. However, effort/cost does not increase linearly with project size, that is why most algorithmic estimation models have an exponential component.
Effort can be expressed by the following general formula

\[ E = A + B \times PM^S. \]  \hspace{1cm} (6.1)

A, B and S are empirically derived constants. While PM is a size metric (lines of code (kLOC)). E is the effort in man months.

The most commonly used cost models obeying equation (6.1) are:

- **Walston-Felix Model**: The Walston-Felix Model is given by

  \[ E = 5.2 \times (kLOC)^{0.91} \]  \hspace{1cm} (6.2)

- **Bailey-Basili Model**: The Bailey-Basili Model is given by

  \[ E = 5.5 + 0.73 \times (kLOC)^{1.16} \]  \hspace{1cm} (6.3)

The constant S is usually close to 1 but reflects the increasing effort required for large projects. E is estimated in man month.

### 6.1.2 Constructive Cost Model COCOMO

As seen in the previous sections, the basic effort estimation methods like Walston-Felix model or Bailey-Basili model depend has the length of code as the only parameter for effort estimation. The constants are derived empirically upon studying many projects throughout time. However, these constants are fixed and do not change from one project to the other. Therefore, the conditions under which the project is being executed, or the quality of the staff and material needed for this project are not playing any role in the estimation of effort or cost needed for the project. This fact results in the unreliability of these estimation methods.

The Constructive Cost Model (COCOMO) is a well-documented software effort or costing model, whose parameters can be tailored to particular modes of working. After Boehm published this costing model in his book *Software Engineering Economics* in 1981, it became the most widely used software project cost estimation model throughout the world. Therefore, most publications concerning cost/effort estimation are based on this model [SB00, SB99, Su00].

COCOMO is a static model with a hierarchy of three estimation models, the basic model, the intermediate model and the detailed model. The basic model gives a rough
estimation of the effort needed for the project. The intermediate model refines the estimation by the use of cost drivers. The cost drivers show the impact of factors like product, hardware, personnel and project attributes on the effort/cost of a project. In addition, in the intermediate model effort is adjusted to take into account any schedule constraints. These models will be further explained in details throughout this section.

In the three models, the size of the project is the main input. Another factor used in all three models is the ”mode” of project development, i.e. the way the project is handled. This factor presents the relation between the way a project is handled and the effort needed for this, at the contrary to the basic effort estimation methods seen in section 6.1.1.

The COCOMO takes into consideration the following three different project modes.

- **Organic Mode**: Projects which fall under the organic mode are those which are developed by a small experienced team (up to six members). In such projects, the communication between the team members is considered to be easy. The project are small to medium sized. Environment, hardware and tools are stable.

- **Embedded Mode**: Projects under this type involve more than 12 members, which communication is not achieved easily between the team members. Projects that fall under this mode are medium to large sized and are developed under tight conditions.

- **Semi Detached Mode**: These are projects whose properties fall between those of Organic and Embedded modes.

Before going into a detailed explanation of the three estimation models of the COCOMO costing model, an overview on the cost drivers is given.

**Cost Drivers**: Cost drivers are factors that influence the effort estimation. Cost drivers are of four broad categories, product attributes, computer attributes, personnel attributes and project attributes. Therefore, the cost drivers reflect the influence of different conditions on the effort needed for a project. In the following a summary of some cost drivers is presented.

**Product attributes:**

- RELY  Required system reliability
- CPLX  Complexity of system modules
- DOCU  Extent of documentation required
- DATA  Size of database used
- RUSE  Required percentage of reusable components

Computer attributes:

- TIME  Execution time constraints
- PVOL  Volatility of development platform
- STOR  Memory constraints

Personnel attributes:

- ACAP  Capability of project analyst
- PCON  Personal continuity
- PEXP  Programmer experience in project domain
- PCAP  Programmer capability
- AEXP  Analyst experience in project domain
- LTEX  Language and tool experience

Project attributes:

- TOOL  Use of software tools
- SCED  Development schedule compression
- SITE  Extent of multi-site working and quality of site communication

More details about the cost drivers, their rating and their influence on the effort estimation are given in the appendix.

6.1.2.1 COCOMO Models

As already mentioned before, the constructive costing model is a static model with the hierarchy of three estimation models. In the following an overview of the three models is given with the constants used for each project mode.

**The Basic Model:** The basic model is the simplest model since it does not include any cost drivers. It uses just the size as basis of estimation. The COCOMO basic model is similar to the costing models explained in section 6.1.1. This basic model gives a rough estimation of effort using the equation

\[ E = a \times (\text{Size}^b). \]  

(6.4)

In addition the development time can be estimated using the equation

\[ DT = c \times (E^d). \]  

(6.5)

The constants \( a, b, c \) and \( d \) are calibrated using past project data according to the project development mode. The values of these constants specified in table 6.1 are based on
The Intermediate Model: The intermediate model refines the basic model by using cost drivers. As already seen before, the use of cost drivers adjusts the effort estimation taking into account the project conditions, time constraints and the team working on the project. The intermediate model uses the nominal cost equation for effort estimation and then applies a factor based on the cost drivers to it. Therefore, the equation is given by:

\[ E = a \times (\text{Size}^b) \times M, \quad (6.6) \]

where \( M \) is the impact of the cost drivers and the size is given in kLOC. Each cost driver is rated using a scale varying from very low to extra high. Then, the values of the different cost drivers are multiplied together to give the overall multiplier \( M \). Table 6.2 summarizes the values given for the different ratings of some cost drivers (See the appendix A.7 for the rating of the cost drivers).

It can be noticed that for cost drivers like CPLX (Complexity of system modules), as the rating is higher the multiplier increases from values less than 1 to values greater than 1. A multiplier less than 1 means that the effort is decreased due to this condition, while a multiplier greater than 1 means that the effort increases due to this condition. If CPLX is rated as very low this means that straight line code is needed and thus the

<table>
<thead>
<tr>
<th>mode</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>2.4</td>
<td>1.05</td>
<td>2.5</td>
<td>0.38</td>
</tr>
<tr>
<td>Semi-detached</td>
<td>3.0</td>
<td>1.12</td>
<td>2.5</td>
<td>0.35</td>
</tr>
<tr>
<td>Embedded</td>
<td>3.6</td>
<td>1.2</td>
<td>2.5</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Table 6.1: Constant values of basic mode.

<table>
<thead>
<tr>
<th></th>
<th>very low</th>
<th>low</th>
<th>nominal</th>
<th>high</th>
<th>very high</th>
<th>extra high</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>0.75</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.40</td>
<td>1.40</td>
</tr>
<tr>
<td>DATA</td>
<td>0.85</td>
<td>0.85</td>
<td>1.00</td>
<td>1.08</td>
<td>1.16</td>
<td>1.16</td>
</tr>
<tr>
<td>CPLX</td>
<td>0.70</td>
<td>0.85</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td>1.65</td>
</tr>
<tr>
<td>TIME</td>
<td>1.0</td>
<td>1.0</td>
<td>1.00</td>
<td>1.11</td>
<td>1.30</td>
<td>1.66</td>
</tr>
<tr>
<td>STOR</td>
<td>1.0</td>
<td>1.0</td>
<td>1.00</td>
<td>1.06</td>
<td>1.21</td>
<td>1.56</td>
</tr>
<tr>
<td>ACAP</td>
<td>1.46</td>
<td>1.19</td>
<td>1.00</td>
<td>0.86</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td>AEXP</td>
<td>1.29</td>
<td>1.13</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>TOOL</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td>SCED</td>
<td>1.23</td>
<td>1.08</td>
<td>1.00</td>
<td>1.04</td>
<td>1.10</td>
<td>1.10</td>
</tr>
</tbody>
</table>

Table 6.2: Cost driver multipliers.

project data collected from multiple sources.

<table>
<thead>
<tr>
<th>mode</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>3.2</td>
<td>1.05</td>
</tr>
<tr>
<td>Semi-detached</td>
<td>3.0</td>
<td>1.12</td>
</tr>
<tr>
<td>Embedded</td>
<td>2.8</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Table 6.3: Constants of the intermediate model.

standard effort is multiplied by 0.70, therefore it decreases. On the other hand, if CPLX is rated as high then highly nested structured programming operators are needed. Thus, the standard effort is multiplied with 1.15 and increased.

However, for other cost drivers like ACAP (Capability of project analyst), as the rating increases the multiplier decreases and therefore effort decreases. If ACAP is rated with very high, this means that the analysts of the project as a team have a high capability and as a result effort is multiplied by 0.71 and decreased. At the contrary, when the analysts have a very low capability effort is multiplied by 1.46 and increased.

It could happen that for a given cost driver, one of the ratings is not available like for DATA the rating very low is not available so it defaults to low and the rating extra high is not available so it defaults to very high. In the appendix a detailed explanation on rating the different cost drivers in the COCOMO estimation model is given.

The constants $a$ and $b$ in equation (6.6), are different than those for the basic model. Table 6.3 gives the values of $a$ and $b$ for the three different project modes.

After the total effort is calculated using equation (6.6), the development time can be calculated using the same equation (6.5) used for the basic model.

**The Detailed Model:** The detailed model is the most refined model of the three. This model is based on the fact that different cost drivers have different impact on the phases of a project. It also uses the product hierarchy of a project to refine the estimation. By giving the effort estimation broken up by phases, estimation is more meaningful in planning a project.

Below are tabulated the phase wise effort distribution percentages for the three project development modes.

The project classification is based on the size of the code (kLOC) according to the following:
6. Cost Analysis

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Small</th>
<th>Intermediate</th>
<th>Medium</th>
<th>Large</th>
<th>Very large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>5.7%</td>
<td>5.7%</td>
<td>5.7%</td>
<td>5.7%</td>
<td>5.7%</td>
</tr>
<tr>
<td>Design</td>
<td>15.1%</td>
<td>15.1%</td>
<td>15.1%</td>
<td>15.1%</td>
<td>15.1%</td>
</tr>
<tr>
<td>Code &amp; unit test</td>
<td>64.2%</td>
<td>61.3%</td>
<td>58.5%</td>
<td>55.7%</td>
<td></td>
</tr>
<tr>
<td>Integration and system test</td>
<td>15.1%</td>
<td>17.9%</td>
<td>20.8%</td>
<td>23.6%</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Effort distribution according the project mode: Organic mode

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Small</th>
<th>Intermediate</th>
<th>Medium</th>
<th>Large</th>
<th>Very large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>6.5%</td>
<td>6.5%</td>
<td>6.5%</td>
<td>6.5%</td>
<td>6.5%</td>
</tr>
<tr>
<td>Design</td>
<td>15.9%</td>
<td>15.9%</td>
<td>15.9%</td>
<td>15.9%</td>
<td>15.9%</td>
</tr>
<tr>
<td>Code &amp; unit test</td>
<td>59.8%</td>
<td>57%</td>
<td>54.2%</td>
<td>51.4%</td>
<td>48.6%</td>
</tr>
<tr>
<td>Integration and system test</td>
<td>17.8%</td>
<td>20.6%</td>
<td>23.4%</td>
<td>26.2%</td>
<td>29.0%</td>
</tr>
</tbody>
</table>

Table 6.5: Effort distribution according the project mode: Semi-detached mode

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Small</th>
<th>Intermediate</th>
<th>Medium</th>
<th>Large</th>
<th>Very large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>7.4%</td>
<td>7.4%</td>
<td>7.4%</td>
<td>7.4%</td>
<td>7.4%</td>
</tr>
<tr>
<td>Design</td>
<td>16.7%</td>
<td>16.7%</td>
<td>16.7%</td>
<td>16.7%</td>
<td>16.7%</td>
</tr>
<tr>
<td>Code &amp; unit test</td>
<td>55.6%</td>
<td>52.8%</td>
<td>50.0%</td>
<td>47.2%</td>
<td>44.4%</td>
</tr>
<tr>
<td>Integration and system test</td>
<td>20.4%</td>
<td>23.1%</td>
<td>25.9%</td>
<td>28.7%</td>
<td>31.5%</td>
</tr>
</tbody>
</table>

Table 6.6: Effort distribution according the project mode: Embedded mode

6.2 Rayleigh Curve for Manpower Estimation

Projects follow a characteristic curve for their manpower loading. Project staff increase (decrease) is not abrupt and staff levels are not flat throughout the life cycle. As known from literature, staffing in projects is a continuous function which increases as the project progresses, peaking at the development time. It again decreases in the system testing and maintenance phase until it dies out. This leads to the idea of the distribution of manpower as a Rayleigh curve known as (Norden-Rayleigh curve) [PN97].
The Norden-Rayleigh curve represents manpower as a function of time. Norden observes that the Rayleigh distribution provides a good approximation of the manpower curve for various hardware development processes. Peter Norden was at IBM when he worked on how to model research and development projects. For this he studied the characteristics of projects. Norden described a project as a set of unsolved problems and as the project progresses, these problems are solved.

Norden in his study described a project as:

- The number of problems is finite though unknown.
- The project effort is the effort needed to solve these problems.
- Team working on the project solves these problems.
- The number of people working is approximately proportional to the number of problems ready for solution.
- Problem solving effort is aimed at removing problems from the finite list of problems. The distribution of problems is therefore the Poisson distribution type.

The Norden-Rayleigh distribution for manpower is given by

\[ m(t) = 2 \times E \times a \times t \times e^{-a \times t^2}. \] (6.7)

where \( m(t) \) is the manpower curve over time, \( E \) is the total effort in man month and \( a \) is a constant called shape parameter. The parameter \( a \) is determined by the time at which the curve reaches its maximum value. The Rayleigh curve can be applied to any project, the effort \( E \) can be calculated by using the COCOMO estimation model explained in section 6.1.2. The parameter \( a \) determines whether the peak of the curve is sharp or not.

For a curve that peaks at \( t_d \) months the value of \( a \) is given by

\[ a = \frac{1}{2 \times t_d^2}. \] (6.8)

where \( t_d \) is known as the development time in months which is the time needed to reach the full operational capability of the project. The larger the value of \( a \) the earlier the peak occurs and the sharper the peak.

The cumulative manpower curve can be obtained by integration of the equation 6.7, leading to the equation

\[ C(t) = E(1 - \exp(-a \times t^2)). \] (6.9)
For a mean manpower \( m_0 \) it can be seen that

\[
m_0 \times t_d \times e^{0.5} = E
\]  

(6.10)

Therefore, the overall effort required by a project can be derived if the peak time and the manpower of a project are known.

The development effort of a project is the effort needed to reach the full operational capability of the project. Since \( t_d \) is the development time then the development effort is the integration of \( m(t) \) from \( t = 0 \) to \( t = t_d \). This is then given as:

\[
D = E(1 - e^{-0.5}) = 0.39E.
\]  

(6.11)

Thus, the development effort is 39% of the total effort needed for the project.

6.3 Application of Costing Models on Synchronization Algorithms

The above discussed methods for effort and manpower estimation can be applied to IEEE802.11a or DVB-T synchronization algorithms discussed in chapters 4 and 5. Considering coding and testing a synchronizer block as a project, the effort needed for such a project can be estimated using the COCOMO estimation model. After the effort is estimated, the distribution of the manpower needed for the project can be estimated using the Rayleigh curve (6.7). In addition, the cumulative Rayleigh curve (6.9) shows the effort needed at every stage of the project. In our study on IEEE802.11a synchronization algorithms in chapter 4, the IEEE802.11a synchronization algorithm using MNC showed the best performance. Therefore, the IEEE802.11a synchronizer using MNC will be considered as a project to which effort will be estimated. In addition, for DVB-T, synchronization using ML and WML algorithms are chosen since they have better performance than others.

6.3.1 Application of COCOMO Estimation Model

The COCOMO estimation model discussed above will be used for estimating effort in the three synchronization projects.

- IEEE802.11a synchronizer using MNC (burst mode transmission)
- DVB-T synchronizer using ML (continuous mode transmission)
6.3. Application of Costing Models on Synchronization Algorithms

- DVB-T synchronizer using WML (continuous mode transmission)

The three projects can be considered as organic mode projects, therefore, the parameters of the organic mode will be used. The estimations for the lines of code used below are taken from our simulation model.

6.3.1.1 IEEE802.11a synchronizer using MNC

The estimated length of code for this project is 1.5 kLOC.

- Basic model:
  \[ E = 2.4 \times (1.5)^{1.05} = 3.67 \text{ man month} \]  
  \text{(6.12)}

- Intermediate model:
  \[ E = 3.2 \times (1.5)^{1.05} \times M_1 \]  
  \text{(6.13)}

  where \( M_1 = 0.72 \) which the result of multiplying the various cost drivers. Thus,
  \[ E = 3.52 \text{ man month} \]

6.3.1.2 DVB-T synchronizer using ML

The estimated length of code for this project is 1.9 kLOC.

- Basic model:
  \[ E = 2.4 \times (1.9)^{1.05} = 4.7 \text{ man month} \]  
  \text{(6.14)}

- Intermediate model:
  \[ E = 3.2 \times (1.9)^{1.05} \times M_2 \]  
  \text{(6.15)}

  where \( M_2 = 0.985 \). Thus, \( E = 6.18 \text{ man month} \)

6.3.1.3 DVB-T synchronizer using WML

The estimated length of code for this project is 2.2 kLOC.

- Basic model:
  \[ E = 2.4 \times (2.2)^{1.05} = 5.49 \text{ man month} \]  
  \text{(6.16)}

- Intermediate model:
  \[ E = 3.2 \times (2.2)^{1.05} \times M_3 \]  
  \text{(6.17)}

  where \( M_3 = 1.11 \). Thus, \( E = 8.12 \text{ man month} \)
The calculation of the multipliers depends on rating the cost drivers for each project. The difference between the multipliers is resulted from the different rating for the three projects. The multipliers $M_1$ and $M_2$ are less than 1, thus the refined effort could be less than the one expected in the basic model as in the case of IEEE802.11a using MNC. It could also be that the refined effort greater than the one estimated in the basic model as in the case of DVB-T using ML, (even $M_2 < 1$). This is because the basic and intermediate models have different values of $a$ and $b$. At the other hand, for DVB-T synchronizer with WML $M_3$ is greater than 1. Thus, the refined effort is greater than the effort estimated by the basic model. This is due to the complexity of the modules and that kLOC > 2.

6.3.2 Application of Rayleigh Curve

Using the effort estimated by the intermediate COCOMO for the three projects in section 6.3.1, the Norden-Rayleigh manpower distribution curves are plotted using equation (6.7). In addition, the cumulative Rayleigh curve is plotted to show the effort distribution over time.

For $E = 3.52$ manmonths, the manpower distribution for an IEEE802.11a synchronizer using MNC is given by the Rayleigh distribution shown in Fig. 6.1. Fig. 6.2 shows the corresponding effort distribution over time.

![Figure 6.1: Manpower distribution for IEEE802.11a synchronizer using MNC (t_d in months).](image-url)
Similarly the manpower distribution over time for a DVB-T synchronizer using ML and

![IEEE synchronizer with MNC](image)

**Figure 6.2:** Cumulative Rayleigh for IEEE802.11a synchronizer using MNC ($t_d$ in months).

WML are shown in Figures 6.3 and 6.4.

In addition, the effort distribution for the two DVB-T synchronization algorithms is

![DVB-T synchronizer using ML](image)

**Figure 6.3:** Manpower distribution for DVB-T synchronizer using ML ($t_d$ in months).

plotted in Figures 6.5 and 6.6.
Figure 6.4: Manpower distribution for DVB-T synchronizer using WML ($t_d$ in months).

In order to see manpower distribution more clearly for larger projects, a theoretical example is considered which is building a dual (IEEE802.11a and DVB-T) synchronizer. Such a project is assumed to need effort $E = 36$ man month.

Figures 6.7 and 6.8 show the manpower and effort distributions respectively.
6.3 Application of Costing Models on Synchronization Algorithms

Figure 6.6: Cumulative Rayleigh for DVB-T synchronizer using WML ($t_d$ in months).

Figure 6.7: Manpower distribution for a dual synchronizer ($t_d$ in months).

6.3.3 Cost-Time-Performance Model (CTP)

In the previous sections, effort and manpower were estimated for the algorithms studied under this thesis. Thinking in production, cost, time and performance are the three basic factors that affect the choice of a certain algorithm. Since in real life if a product
6. Cost Analysis

should be produced, not always the algorithm with the highest performance will be used. This is due to the fact that in many cases the highly performing algorithm is very costly or requires a lot of time. Therefore, a compromise between these three factors is always preferred.

In this section I develop a model which I call the CTP (Cost-Time-Performance) model. The idea of using the RGB model in the background of the CTP axes is to clarify the idea of continuous transition from region to region. This model is used to show a trade-off between these three factors: Cost, time and performance of a project. Fig. 6.9 shows the CTP model.

In the following the application of the CTP constellation on IEEE802.11a synchronizer and DVB-T synchronizer will be given.

6.3.3.1 IEEE802.11a

In our project, the IEEE802.11a synchronization algorithm was studied and discussed according to two different criteria:

- Metrics: MC, MNC and ML
- Correlator signal calculation: magnitude of $|c[n]|$, approximation1: $|c[n]| \approx |c'[n]| + |c''[n]|$ and approximation2: $|c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|)$
These three criteria result in many different combinations to implement the synchronizer. Each of the combinations lies in some region of the CTP-model.

If cost is the most important, then time and then performance, we move either in the region R1 or on axis (XM). Therefore, in this case the algorithm we suggest is the MC metric with double multiplier and the first approximation $|c[n]| \approx |c'[n]| + |c''[n]|$. Similarly, if performance is the most important, then time and then cost, we move either in the region R4 or on axis (XU). In this case the algorithm suggested is (MNC, basic impl., $|c[n]|$). Following this strategy all algorithms investigated in this thesis can be put in their corresponding regions as follows.

- **R1**: MC, $|c[n]| \approx |c'[n]| + |c''[n]|$
- **R2**: MC, $|c[n]|$
- **R3**: MNC, $|c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|)$
- **R4**: MNC, $|c[n]|$
- **R5**: ML, $|c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|)$
6. Cost Analysis

• R6: MC, $|c[n]| \approx |c'[n]| + |c''[n]|$

However, according to our study the algorithm which lies in the center of the CTP constellation is using the MNC metric, the double multiplier implementation and the approximation $|c[n]| \approx |c'[n]| + |c''[n]|$. This is considered the best trade-off between the three factors. The trade-off can be done using any three criteria which could affect the performance, the cost and the time-to-market of the synchronizer.

6.3.3.2 DVB-T

A similar study is done on the proposed algorithms for the DVB-T synchronizer. The DVB-T algorithms are chosen according to the following criteria:

• Metrics: MC, MNC, ML and WML.

• Correlator signal calculation: magnitude of $|c[n]|$, approximation1: $|c[n]| \approx |c'[n]| + |c''[n]|$ and approximation2: $|c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|)$

• Integer frequency estimation methods: auto-pilots-power, auto-pilots-power-squared, pilots-magnitude and pilots-power.

Based on these criteria the CTP-constellation for the DVB-T algorithms is as follows:

• R1: MNC, $|c[n]| \approx |c'[n]| + |c''[n]|$, pilots-power

• R2: MNC, $|c[n]|$, pilots-power

• R3: ML, $|c[n]|$, pilots-power

• R4: WML, $|c[n]|$, auto-pilots-power

• R5: ML, $|c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|)$, auto-pilots-power

• R6: ML, $|c[n]| \approx |c'[n]| + |c''[n]|$, pilots-power

In addition, the optimum trade-off in the case of DVB-T is with the algorithm (WML, $|c[n]| \approx |c'[n]| + |c''[n]|$, pilots-power).

The constellation of the CTP model depends on the complexity and manpower estimations for each case, using the different optimizations. This constellation serves as the background for choosing the best synchronization algorithm depending on priorities and trade-offs between complexity, time and performance. Thus, in this work, the algorithmic performance, cost and manpower were brought under one umbrella.


7 Conclusions

In this thesis, a study on synchronization algorithms for IEEE802.11a WLAN and DVB-T systems was done. It included time and frequency synchronization. The implementation aspects were mostly investigated. The aim is an efficient fixed point implementation in SystemC of an IEEE802.11a synchronizer as well as a DVB-T synchronizer, which can lead to a high-level synthesis. Therefore, algorithmic optimizations in terms of complexity are done for the two applications (IEEE802.11a and DVB-T). In addition, some methods for estimating effort and manpower in projects are investigated, then tuned to be used for estimating effort in such projects.

In chapters 2 and 3 the OFDM technique is introduced as used in the IEEE802.11a standard and the DVB-T standard. In addition, the synchronization errors in both systems and the synchronization procedure are explained. This is the introduction to the other chapters in which the synchronization problem is investigated in more details.

Chapter 4 deals with the synchronization of the IEEE802.11a WLAN. Three different metrics are studied for an IEEE802.11a synchronizer. These are the MC, MNC and ML metrics. The MC metric has the drawback that it does not take into account the power of the signal. Therefore, setting a threshold for detecting the maximum of the metric signal in this case is very difficult due to the large variety of signals. This arises the idea of using the MNC metric where the power of the signal is used to normalize the metric signal. The MNC metric shows to perform very well. It is preferred over the ML metric due to some drawbacks of ML like false alarm in the burst mode transmission. In addition, the ML metric requires the estimation of the current $SNR$ which is not possible before estimating the time offset. On the other hand, the already calculated $SNR$ is outdated. Therefore, the MNC metric is chosen for the fixed point implementation. The correlator is a main block, therefore it is more optimized for an efficient implementation. Three ways for implementing it are investigated. The basic implementation and two iterative methods using one or two multipliers respectively. The implementation using two multipliers is preferred on the other two due to complexity reasons.

Nevertheless, more optimization is done on the IEEE802.11a synchronizer and more precisely on the correlator output. The performance of the algorithms are tested for
approximating the correlator signal by two ways:

- \[ |c[n]| \approx |c'[n]| + |c''[n]| \]
- \[ |c[n]| \approx \alpha \max(|c'[n]|, |c''[n]|) + \beta \min(|c'[n]|, |c''[n]|) \]

Both approximations gave good results. So if a low complexity is needed then the first approximation is preferred. In addition, this chapter contains a complexity analysis of the different possible implementations. This complexity analysis is then used in chapter 6.

Analysis of synchronization algorithms for DVB-T systems is done in chapter 5. The difference now is that the DVB-T systems have a continuous mode transmission. Pre-FFT as well as post-FFT synchronization is performed for DVB-T. The pre-FFT synchronization is coarse time synchronization and fractional frequency offset estimation. While integer frequency offset estimation is done in the post-FFT stage. The pre-FFT synchronization procedure is similar to that of the IEEE802.11a. The same metrics are tested, however for the DVB-T systems, the ML metric is preferred. It performs very well and does not suffer from the drawbacks as in the burst mode transmission like IEEE802.11a WLAN. For post-FFT synchronization, several algorithms which make use of the continual pilots given by the DVB-T standard are tested. Simulations are done on three types of channels: AWGN, F1 (fixed multipath) and mobile channel with a receiver moving at 120 km/hr. For low \( SNR \) signals, the metric using the autocorrelation (without squaring) is preferred, since it seems that by squaring, the performance of the metric decreases in the case of a mobile channel. The ML metric is chosen for the fixed point implementation, however it does not give the needed accuracy in estimation especially in the mobile channel environment. This arises the need for a more complex estimator to avoid high \( SNR \) degradation. Since the acceptable \( SNR \) degradation is 0.1 \( dB \), and the ML will cause around 0.4 \( dB \) \( SNR \) degradation.

In this thesis, a new algorithm namely Weighted Maximum Likelihood (WML) is proposed. It makes use of the several OFDM symbols but still does not increase the complexity dramatically. The simulations in the mobile channel show that the WML with a weighting factor of \( W = 2 \) already increases the performance by around 60%. Since for an \( SNR = 15 \, dB \), the ML algorithm results in a standard deviation of 0.021 for the fractional frequency, while the standard deviation for WML with \( W = 2 \) is 0.008. A weighting factor of 2 would mean that 5 symbols contribute significantly to the accumulation. However, storing the history OFDM symbols would be an issue in the complexity. Therefore, an efficient way for implementing the WML is proposed which is based on using a shift register as a delay line for storing the history. This avoids the extra use of a multiplexer, demultiplexer and address counter. In addition, the complexity of
each of the algorithms is analyzed.

This work opens new prospects for further investigations in the domain of synchronization of OFDM systems. Similar investigations can be done on fine time synchronization, channel estimation as well as phase tracking. In addition, high level synthesis can be done for the already investigated algorithms, making use of the implementation optimizations.

The thesis concludes by a cost analysis done in chapter 6. This includes effort and manpower estimation. Several effort estimation methods are presented. However, for estimating effort needed for the synchronization algorithms, the COCOMO model was used. A CTP-model is introduced which helps in choosing the appropriate algorithm depending on the cost, time-to-market and performance restrictions. We consider our cost analysis as the first step-stone for such applications.

The cost analysis performed in this work is an introductory step to relate the technical research to some innovation and technology management issues. It should provide the first step in the direction of cost analysis for different technical alternatives in future research work, being in the domain of OFDM synchronization algorithms or any other scientific field.
A Background

A.1 Gaussian Distribution

The Gaussian distribution, also called the normal distribution, is an important family of continuous probability distributions. It is defined by two parameters, the mean ($\mu$) and the standard deviation ($\sigma$). Its probability density function is given by:

$$f(x; \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$  \hspace{1cm} (A.1)

It is often called the bell curve because the graph of its probability density has a bell shape as seen in Fig. A.1.

Figure A.1: Gaussian distribution pdf for different $\mu$ and $\sigma$. 

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$\sigma^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>0</td>
<td>5.0</td>
</tr>
<tr>
<td>-2</td>
<td>0.5</td>
</tr>
</tbody>
</table>
A.2 Rayleigh Distribution

The Rayleigh distribution is a continuous probability distribution, whose density function is:

\[ f(x|\sigma) = \frac{x \exp\left(\frac{-x^2}{2\sigma^2}\right)}{\sigma^2} \quad (A.2) \]

It usually arises when a two-dimensional vector has its two orthogonal components normally and independently distributed. Fig. A.2 shows the probability density functions of a two dimensional normal distribution and a Rayleigh distribution. \( R \approx \text{Rayleigh}(\sigma) \) if \( R = \sqrt{X_1^2 + X_2^2} \) where \( X_1 \approx N(0, \sigma^2) \) and \( X_2 \approx N(0, \sigma^2) \) are two normal distributions.

![Pdf of 2-D normal distribution and Rayleigh distribution.](image)

Figure A.2: Pdfs of two-dim normal distribution and Rayleigh distribution.

A.3 Rice Distribution

The Rice distribution is a continuous probability distribution whose probability density function is:

\[ f(x|\nu, \sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-(x^2 + \nu^2)}{2\sigma^2}\right) I_0\left(\frac{x\nu}{\sigma^2}\right) \quad (A.3) \]

where \( I_0 \) is a bessel function. For \( \nu = 0 \) The Rice distribution changes to a Rayleigh distribution.

A.4 Doppler Effect

The Doppler effect is the change in frequency and wavelength of a wave as perceived by an observer moving relative to the source of the wave. The change of frequency depends
if the waves propagate in a medium or if no medium is needed for the wave propagation.

For waves that travel through a medium (sound, ultrasound, etc...) the relationship between observed frequency $f'$ and emitted frequency $f$ is given by:

$$f' = \left(\frac{\nu}{\nu \pm \nu_s}\right)f \quad (A.4)$$

where $\nu$ is the speed of waves in its medium, and $\nu_s$ is the velocity of the source. While for waves that travel at the speed of light, such as radio waves, the relationship between observed frequency $f'$ and emitted frequency $f$ is given by:

- Change in frequency:
  $$\Delta f = \frac{f\nu}{c} \quad (A.5)$$

- Observed frequency:
  $$f' = f + \frac{f\nu}{c} \quad (A.6)$$

where $f$ is the transmitted frequency, $\nu$ is the velocity of the transmitter relative to the receiver, $c$ is the speed of light.

### A.5 Rayleigh Fading

The Rayleigh fading models the effect of a propagation environment on a radio signal, for example in wireless devices. It assumes that the magnitude of the radio signal that has passed through a communications channel will fade according to a Rayleigh distribution. Rayleigh fading is most applicable when there is no line of sight between the transmitter and receiver. How rapidly the channel fades will be affected by how fast the receiver and/or transmitter move.

### A.6 Why is Crosscorrelation Not Iterative

The crosscorrelation is computed by:

$$x[n] = \sum_{i=0}^{N_{TRAIN}-1} s^*[i]r[n+i] \quad (A.7)$$

Where $s^*[i]$ are stored in the receiver. The samples of the received signal $(r[n], \ldots, r[n + N_{TRAIN} - 1])$ depend on the index $n$. 

1. At the time index $n - 1$, we have in the delay line the values $(r[n - 1], \ldots, r[n + N_{TRAIN} - 2])$. Therefore, the crosscorrelation at the index $n - 1$ is given by

$$x[n - 1] = s^*[0] \cdot r[n - 1] + s^*[1] \cdot r[n] + s^*[2] \cdot r[n + 1] + \ldots + s^*[N_{TRAIN} - 1] \cdot r[n + N_{TRAIN} - 2]$$  \hfill (A.8)

2. At the time index $n$, we have in the delay line the values $(r[n], \ldots, r[n + N_{TRAIN} - 1])$. Therefore, the crosscorrelation at the index $n$ is given by

$$x[n] = s^*[0] \cdot r[n] + s^*[1] \cdot r[n + 1] + s^*[2] \cdot r[n + 2] + \ldots + s^*[N_{TRAIN} - 1] \cdot r[n + N_{TRAIN} - 1]$$  \hfill (A.9)

Comparing (A.9) and (A.10), it is clearly seen that, at $n$, new values are multiplied by the values $s^*[i]$ stored in the receiver. The multiplication terms that are added to compute $x[n]$ are totally different from those added to compute $x[n - 1]$. $s[i]$ are stored in the receiver, at a new time instant, a new value is received. But the multiplications are independent of those of $x[n - 1]$. So we have to multiply and calculate from the beginning. On the other hand, when a new value is received and autocorrelation is calculated, the samples are shifted in the delay line, and because we multiply received values with each other, the new $c[n]$ can be calculated recursively from $c[n - 1]$ by adding the corresponding multiplication with the newest $r[n]$ and subtracting the corresponding multiplication of the oldest value in the delay line. This means we cannot calculate $x[n]$ from $x[n - 1]$ recursively as done in the autocorrelation.

### A.7 Cost Drivers

**Required Reliability (RELY)**

This reflects the extent that a software product can be expected to perform its intended functions satisfactorily. The rating can be determined from the list below.

- **Very Low**: The effect of a software failure is simply the inconvenience incumbent on the developers to fix the fault.
- **Low**: The effect of a software failure is a low level, easily-recoverable loss to users.
- **Nominal**: The effect of a software failure is a moderate loss to users, but a situation for which one can recover without extreme penalty.
- **High**: The effect of a software failure can be a major financial loss or a massive human inconvenience.

- **Very High**: The effect of a software failure can be the loss of human life.

- **Extra High**: No rating - defaults to Very High.

**Database Size (DATA)**

This is the relative database size to be developed where size refers to the amount of data to be assembled and stored in non-main storage:

\[
D/P = \frac{\text{Database size in bytes or characters}}{\text{Program size in SLOC}}.
\]

- **Very Low**: No rating - defaults to Low.
- **Low**: \( D/P < 10 \)
- **Nominal**: \( 10 \leq D/P \leq 100 \)
- **High**: \( 100 \leq D/P \leq 1000 \)
- **Very High**: \( D/P > 1000 \)
- **Extra High**: No rating - defaults to Very High.

**Product Complexity (CPLX)**

Complexity is assessed as the subjective average of four types of control functions: control, computation, device-dependent, or data management operations.

*Control Operations:*

- **Very Low**: Straight-line code with a few non-nested structured programming operations: DOs, CASEs, IF-THEN-ELSEs. Simple predicates.

- **Low**: Straight forward nesting of structured programming operators. Mostly simple predicates.

- **Nominal**: Most simple nesting. Some intermodule control. Decision tables.

- **High**: Highly nested structured programming operators with many compound predicates. Queue and stack control. Considerable intermodule control.
• **Very High**: Reentrant and recursive coding. Fixed-priority interrupt handling.

• **Extra High**: Multiple resource scheduling with dynamically changing priorities. Microcode-level control.

**Computational Operations:**

• **Very Low**: Evaluation of simple expressions

• **Low**: Evaluation of moderate level expressions

• **Nominal**: Use of standard math and statistical routines, basic matrix and vector operations.

• **High**: Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns.

• **Very High**: Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations

• **Extra High**: Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data.

**Device-Dependent Operations:**

• **Very Low**: Simple read and write statements with simple formats.

• **Low**: No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level. No cognizance of overlap.

• **Nominal**: I/O processing includes device selection, status checking and error processing.

• **High**: Operations at the physical I/O level (physical storage address translations; seeks, reads, etc). Optimized I/O overlap.

• **Very High**: Routines for interrupt diagnosis, servicing, masking. Communication line handling.

• **Extra High**: Device timing-dependent coding, microprogrammed operations.
**Data Management Operations:**

- **Very Low**: Simple arrays in main memory.
- **Low**: Single file sub-setting with no data structure changes, no edits, no intermediate files.
- **Nominal**: Multi-file input and single file output. Simple structural changes, simple edits.
- **High**: Special purpose subroutines activated by data stream contents. Complex data restructuring at the record level.
- **Very High**: A generalized, parameter-driven file structuring routine. File building, command processing, search optimization.
- **Extra High**: Highly coupled, dynamic relational structures. Natural language data management.

**Execution Time Constraint (TIME)**

This reflects the degree of execution time constraint imposed upon a software product. The rating is expressed in terms of available execution time expected to be used.

- **Very Low**: No rating - defaults to Nominal.
- **Low**: No rating - defaults to Nominal.
- **Nominal**: $\leq 50\%$ use of available execution time.
- **High**: $70\%$ use of available execution time.
- **Very High**: $85\%$ use of available execution time.
- **Extra High**: $95\%$ use of available execution time.

**Main Storage Constraint (STOR)**

This reflects the percentage of main storage expected to be used by the software product and any subsystems consuming the main storage resources. Main storage refers to direct random access storage such as disks, tapes, or optical drives.
- **Very Low**: No rating - defaults to Nominal.
- **Low**: No rating - defaults to Nominal.
- **Nominal**: \( \leq 50\% \) use of available storage.
- **High**: 70\% use of available storage.
- **Very High**: 85\% use of available storage.
- **Extra High**: 95\% use of available storage.

**Analyst Capability (ACAP)**

Analysts participate in the development and validation of requirements and preliminary design specifications. They consult on detailed design and code activities. They are heavily involved in integration and test. The ratings for analyst capability are expressed in terms of percentiles with respect to the overall population of software analysts. The major attributes to be considered are ability, efficiency, thoroughness, and the ability to communicate and cooperate. This evaluation should not include experience (that is captured in other factors) and should be based on the capability of the analysts as a team rather than individuals.

- **Very Low**: 15th percentile
- **Low**: 35th percentile
- **Nominal**: 55th percentile
- **High**: 75th percentile
- **Very High**: 90th percentile
- **Extra High**: No rating - defaults to Very High

**Applications Experience (AEXP)**

This represents the level of equivalent applications experience of the project team developing the software product.

- **Very Low**: \( \leq 4 \) month experience.
- **Low**: 1 year of experience.
• **Nominal**: 3 years of experience.

• **High**: 6 years of experience.

• **Very High**: 12 years of experience

• **Extra High**: No rating - defaults to Very High

**Use of Software Tools (TOOL)**

This represents the degree to which software tools are used in developing the software product.

• **Very Low**: Basic tools, e.g. assembler, linker, monitor, debug aids.

• **Low**: Beginning use of more productive tools, e.g. High-Order Language compiler, macro assembler, source editor, basic library aids, database aids.

• **Nominal**: Some use tools such as real-time operating systems, database management system, interactive debuggers, interactive source editor.

• **High**: General use of tools such as virtual operating systems, database design aids, program design language, performance measurement and analysis aids, and program flow and test analyzer.

• **Very High**: General user of advanced tools such as full programming support library with configuration management aids, integrated documentation system, project control system, extended design tools, automated verification system.

• **Extra High**: No rating - defaults to Very High

**Schedule Constraint (SCED)**

This represents the level of constraint imposed on the project team developing a software product. Ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort.

• **Very Low**: 75% of nominal.

• **Low**: 85% of nominal.

• **Nominal**: 100%
• **High**: 130% of nominal.

• **Very High**: 160% of nominal.

• **Extra High**: No rating - defaults to Very High
B Symbols and Acronyms

In order to enhance the readability, this appendix summarizes the abbreviations, the naming conventions of variables and the mathematical notations used throughout the thesis.

B.1 Variables

\( n \)  
\( k \)  
\( t \)  
\( \tau \)  
\( \theta \)  
\( \hat{\theta} \)  
\( B \)  
\( \Delta f_c \)  
\( \Delta F \)  
\( \varepsilon \)  
\( \hat{\varepsilon} \)  
\( \phi \)  
\( \nu \)  
\( T_{SYM} \)  
\( T_{FFT} \)  
\( N_{FFT} \)  
\( N_{PER} \)  
\( N_{TRAIN} \)  
\( N_{POW} \)  
\( N_{delay} \)  
\( E_s \)  
\( E_B \)  
\( N_0 \)  
\( s[n] \)  
\( r[n] \)  
\( h[n] \)  

- \( n \): discrete time index
- \( k \): discrete frequency index
- \( t \): continuous time index
- \( \tau \): continuous time delay
- \( \theta \): normalized time offset
- \( \hat{\theta} \): estimated normalized time offset
- \( B \): occupied bandwidth
- \( \Delta f_c \): continuous carrier frequency offset
- \( \Delta F \): subcarrier spacing
- \( \varepsilon \): normalized frequency offset
- \( \hat{\varepsilon} \): estimated normalized frequency offset
- \( \phi \): phase offset
- \( \nu \): phase velocity
- \( T_{SYM} \): symbol clock
- \( T_{FFT} \): OFDM Symbol Period
- \( N_{FFT} \): FFT-length
- \( N_{PER} \): periodicity interval
- \( N_{TRAIN} \): training symbol length
- \( N_{POW} \): length of power calculation window
- \( N_{delay} \): number of delay taps
- \( E_s \): energy per symbol
- \( E_B \): energy per bit
- \( N_0 \): one-side noise power spectral density of AWGN
- \( s[n] \): complex-valued discrete-time transmit signal
- \( r[n] \): complex-valued discrete-time receive signal
- \( h[n] \): complex-valued discrete-time channel impulse response
B. Symbols and Acronyms

- $c[n]$ complex-valued discrete-time auto-correlator signal
- $x[n]$ complex-valued discrete-time cross-correlator signal
- $p[n]$ discrete-time power sum
- $m[n]$ discrete-time metric signal
- $\sigma^2$ variance
- $N$ number of propagation paths
- $d$ distance between transmitter and receiver
- $\tau_{MAX}$ maximum delay spread
- $\Delta_r$ delay spread
- $B_c$ coherence bandwidth
- $c_k$ frequency symbol transmitted on the k subcarrier
- $T_S$ system clock
- $K_u$ number of useful data carriers
- $\Delta_p$ distance of pilot carriers
- $f_c$ oscillator frequency
- $L_g$ length of guard interval

B.2 Units

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ns$</td>
<td>Nanosecond</td>
</tr>
<tr>
<td>$\mu s$</td>
<td>Microsecond</td>
</tr>
<tr>
<td>$Mbps$</td>
<td>Megabit per second</td>
</tr>
<tr>
<td>$GHz$</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>$ppm$</td>
<td>parts per million</td>
</tr>
<tr>
<td>$dB$</td>
<td>decibel</td>
</tr>
</tbody>
</table>

B.3 Operators

- $arg(.)$ Finds the angle
- $e^x$ Calculates the exponential
- $|.|$ Calculates the magnitude
- $max(.)$ Finds the maximum
B.4 Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGC</td>
<td>Automatic Gain Control</td>
</tr>
<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>CF</td>
<td>Crest Factor</td>
</tr>
<tr>
<td>CIR</td>
<td>Channel Impulse Response</td>
</tr>
<tr>
<td>CP</td>
<td>Cyclic Prefix</td>
</tr>
<tr>
<td>CTF</td>
<td>Channel Transfer Function</td>
</tr>
<tr>
<td>DA</td>
<td>Data Aided</td>
</tr>
<tr>
<td>DAB</td>
<td>Digital Audio Broadcasting</td>
</tr>
<tr>
<td>DBPSK</td>
<td>Differential Bi Phase Shift Keying</td>
</tr>
<tr>
<td>DD</td>
<td>Decision Directed</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DSL</td>
<td>Digital Subscriber Line</td>
</tr>
<tr>
<td>DSSS</td>
<td>Direct Sequence Spread Spectrum</td>
</tr>
<tr>
<td>DVB</td>
<td>Digital Video Broadcasting</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>GI</td>
<td>Guard Interval</td>
</tr>
<tr>
<td>HiperLAN2</td>
<td>High Performance Radio Local Area Network type 2</td>
</tr>
<tr>
<td>ICI</td>
<td>Intercarrier Interference</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineer</td>
</tr>
<tr>
<td>IFFT</td>
<td>Inverse Fast Fourier Transform</td>
</tr>
<tr>
<td>ISI</td>
<td>Intersymbol Interference</td>
</tr>
<tr>
<td>MC</td>
<td>Maximum Correlation</td>
</tr>
<tr>
<td>MCM</td>
<td>Multi-Carrier Modulation</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>MMSE</td>
<td>Maximum Mean-Squared Error</td>
</tr>
<tr>
<td>MNC</td>
<td>Maximum Normalized Correlation</td>
</tr>
<tr>
<td>MUX</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>NDA</td>
<td>Non Data Aided</td>
</tr>
<tr>
<td>NFO</td>
<td>Normalized Frequency Offset</td>
</tr>
<tr>
<td>NTO</td>
<td>Normalized Time Offset</td>
</tr>
<tr>
<td>OFDM</td>
<td>Orthogonal Frequency Division Multiplex</td>
</tr>
<tr>
<td>PER</td>
<td>Packet Error Rate</td>
</tr>
<tr>
<td>PN</td>
<td>Pseudo Noise</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>PRBS</td>
<td>Pseudo Random Binary Sequence</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>RS</td>
<td>Read Solomon</td>
</tr>
<tr>
<td>SCF</td>
<td>Simulation Control File</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WML</td>
<td>Weighted Maximum Likelihood</td>
</tr>
</tbody>
</table>
C German Translations

The following german translations of the title (Section C.1), the abstract (Section C.2), the table of contents (Section C.3), and the introduction (Section C.4), are a mandatory requirement for the dissertation at the Faculty of Engineering of the University of Erlangen-Nuremberg.

C.1 Titel

Synchronisationsalgorithmen für OFDM Systeme (IEEE802.11a, DVB-T)
Analyse, Simulation, Optimierung und Implementierungsaspekte

C.2 Zusammenfassung

”Orthogonal Frequency Division Multiplexing” (OFDM), oder ”Multi-carrier Modulation” (MCM) ist ein leistungsfähiges digitales Übertragungsverfahren. Es erlaubt hohe Datenraten mit ausreichendem Schutz gegen Störungen, wie sie typischerweise bei der Übertragung über drahtlose Kanäle auftreten, insbesondere bei Mehrwegeempfang durch Reflexionen im Übertragungskanal. Durch diese Eigenschaft entwickelte sich OFDM zunehmend als die Modulationstechnik, die sich für die neueste Generation digitaler drahtloser Übertragungssysteme am besten eignet (IEEE802.11a und DVB-T). Ein Nachteil beim Einsatz von OFDM besteht jedoch darin, dass diese Modulationstechnik sehr empfindlich auf Störungen bei der Synchronisation reagiert. Dies begründet eine Notwendigkeit, Synchronisationsalgorithmen für OFDM Anwendungen wie IEEE802.11a und DVB-T zu optimieren.

In der vorliegenden Arbeit werden unterschiedliche Algorithmen zur Synchronisation vorgestellt. Die Arbeit konzentriert sich auf Implementierungsaspekte solcher Algorithmen für IEEE802.11a und DVB-T Anwendungen. Es werden solche Optimierungsstrategien vorgeschlagen, die zu leistungsfähigen und robusten Realisierungen in Festkommaarithmetik führen. Darüber hinaus wird ein Kostenmodell entwickelt, in welches auch die Aspekte Komplexität und Kosten für die Realisierung einfließen. Weiterhin werden die verschiedenen Algorithmen klassifiziert und bezüglich Kosten (Cost), Produkteinführungszeit (Time-to-market) und Leistungsfähigkeit (Performance)
in die Bewertung einbezogen.

Kapitel 1 enthält eine kurze Einführung in die Theorie von OFDM zur schnellen Datenübertragung. Die Notwendigkeit von Synchronisation wird besonders herausgestellt.


Kapitel 3 geht im Detail auf Prinzipien und Besonderheiten ein, die bei der Synchronisation in IEEE802.11a- und DVB-T Empfängern auftreten. Es bildet die Basis für Kapitel 4 und 5.

Die nachfolgenden Kapitel beschäftigen sich mit den neuen Ergebnissen, die Optimierungen sowohl bei Algorithmen als auch bei der Implementierung für beide Applikationen (IEEE802.11a und DVB-T) zum Ergebnis haben.

Kapitel 4 konzentriert sich auf Details bei der Implementierung für IEEE802.11a Systeme. Es schließt die Realisierung einer Festkommaimplementierung mit SystemC ein. Darüber hinaus werden Ansätze aufgezeigt, wie Algorithmen optimiert werden können, wenn das Ziel die Verringerung der Komplexität ist, ohne gleichzeitige Reduktion der Leistungsfähigkeit.

Kapitel 5 zeigt einen ähnlichen Ansatz für DVB-T Systeme. Auch hier wird eine Festkommaarithmetik untersucht und es werden Optimierungen der Algorithmen vorgestellt. Diese zielen zum Einen auf ein verbessertes Störverhalten, zum Anderen auf eine Implementierung bei minimalen Kosten.

Kapitel 6 umfasst Methoden zur Kostenanalyse. Aufwand und Arbeitsleistung werden für die untersuchten Algorithmen geschätzt.
C.3 Inhaltsverzeichnis

1 Einleitung 1

2 OFDM Systeme: Überblick 3
   2.1 Mehrwegekanäle .................................. 3
   2.2 Was ist OFDM ...................................... 5
   2.3 Der IEEE802.11a Standard ........................ 8
   2.4 Der DVB-T Standard ................................ 10
   2.5 Beeinträchtigungen in OFDM Systeme ............... 12
      2.5.1 Zeitfehler .................................... 13
      2.5.2 Trägerfrequenzfehler .......................... 14
      2.5.3 Abtastratenfehler ............................. 16
      2.5.4 Phasenfehler .................................. 17

3 Prinzipien der Synchronisation 19
   3.1 IEEE802.11a ...................................... 19
      3.1.1 Autokorrelationsmethode ....................... 20
      3.1.1 Kreuzkorrelationsmethode ....................... 23
      3.1.3 Synchronisationsprozedur ....................... 24
   3.2 DVB-T ........................................... 25
      3.2.1 Autokorrelation des Guard Intervals ............ 26
      3.2.2 Kreuzkorrelation von Pilotsymbolen ............ 27

4 IEEE802.11a Synchronisation 29
   4.1 Optimale Korrelatiosparameter ..................... 30
   4.2 Metriken .......................................... 33
      4.2.1 Maximum Correlation (MC) ....................... 34
      4.2.2 Maximum Normalized Correlation (MNC) ........... 34
      4.2.3 Minimum Mean Squared Error (MMSE)) ........... 36
      4.2.4 Maximum Likelihood (ML) ....................... 38
      4.2.5 Schlussfolgerungen ............................ 42
   4.3 Simulationsmodell .................................. 42
      4.3.1 IEEE802.11a Sender: Überblick .................. 43
      4.3.2 Beeinrächtigungen ............................. 44
      4.3.3 Synchronisationsmodul .......................... 45
         4.3.3.1 Grobe Synchronisation ....................... 45
         4.3.3.2 Feine Synchronisation ....................... 46
   4.4 Festkommaimplementierung .......................... 47
4.4.1 SystemC Für Festkommaimplementierung ................. 48
4.4.2 Implementierung des Korrelators .................. 49
4.4.3 Festkomma MNC Metrik ................................ 52
4.4.4 CORDIC Algorithmus .................................. 53
4.4.5 Grobe und Feine Synchronisation .................... 54
4.4.6 Wortlängenoptimierung .................................. 55
4.4.7 Implementierungsverlust Analyse .................... 55
4.5 Komplexitätsanalyse ........................................ 56
4.5.1 MNC .................................... 58
4.5.2 MC .................................... 58
4.5.3 MMSE .................................... 59
4.5.4 ML .................................... 59
4.6 Algorithmische Optimierung ................................ 60
4.6.1 Korrelator Approximationen .................. 60
4.6.2 Cramer Rao Schranke .................................. 63
   4.6.2.1 Log-Likelihood Funktion .................. 64
   4.6.2.2 Cramer Rao Ungleichung .................. 64
   4.6.2.3 Cramer Rao Schranke für diese Applikation ... 65
5 DVB-T Synchronisation ........................................... 67
5.1 Synchronisationsalgorithmen ................................ 67
   5.1.1 Grobe Zeitsynchronisation .................. 67
   5.1.2 Frequenzsynchronisation .................. 70
      5.1.2.1 Pre-FFT Phase .................. 70
      5.1.2.2 Post-FFT Phase .................. 71
5.2 Simulationsmodell ........................................... 73
   5.2.1 DVB-T Sender ................................ 73
   5.2.2 Kanal .................................. 75
   5.2.3 DVB-T Empfänger ................................ 77
      5.2.3.1 Pre-FFT Synchronisation .......... 77
      5.2.3.2 Post-FFT Synchronisation .......... 79
5.3 Simulationsergebnisse ........................................ 80
   5.3.1 Auswertung der Zeitsynchronisation .............. 80
      5.3.1.1 AWGN Kanal .......................... 84
      5.3.1.2 Fester Mehrwegekanal F1 ............ 84
      5.3.1.3 Zeitvarianter Mobilfunfkanal ......... 85
   5.3.2 Auswertungen der Frequenzsynchronizierung .... 86
      5.3.2.1 Fraktaler Frequenzversatz ............ 87
      5.3.2.2 Ganzzahliger Frequenzversatz ...... 89
5.4 Festkommaimplementierung ........................................... 96
  5.4.1 ML Algorithmus ...................................................... 97
  5.4.2 Wahl der optimalen Wortlänge .................................... 100
5.5 Algorithmische Optimierung .......................................... 102
  5.5.1 Averaged Maximum Likelihood .................................. 103
  5.5.2 Weighted Maximum Likelihood ................................... 104
    5.5.2.1 Konzept ......................................................... 104
    5.5.2.2 Simulationsergebnisse ....................................... 105
5.6 Komplexitätsanalyse .................................................. 109
  5.6.1 ML Algorithmus ...................................................... 109
  5.6.2 WML Algorithmus .................................................... 112

6 Kostenanalyse ........................................................... 115
  6.1 Kostenschätzungsmodelle ............................................ 116
    6.1.1 Basismodelle für Aufwandsschätzung .......................... 116
    6.1.2 Constructive Cost Model COCOMO ................................ 117
      6.1.2.1 COCOMO Modelle .............................................. 119
  6.2 Rayleigh Kurve zur Schätzung des Arbeitsaufwands .............. 122
  6.3 Anwendung von Kostenmodellen auf Synchronisationsalgorithmen . 124
    6.3.1 Anwendung des COCOMO Schätzungsmodell .................... 124
      6.3.1.1 IEEE802.11a Synchronisator mit MNC ....................... 125
      6.3.1.2 DVB-T Synchronisator mit ML ............................... 125
      6.3.1.3 DVB-T Synchronisator mit WML ............................ 125
    6.3.2 Anwendung der Rayleigh Kurve ................................ 126
    6.3.3 Cost-Time-Performance Modell (CTP) .......................... 129
      6.3.3.1 IEEE802.11a .................................................. 130
      6.3.3.2 DVB-T ......................................................... 132

7 Schlussfolgerungen ....................................................... 133

A Hintergrund .............................................................. 137
  A.1 Gaussverteilung ....................................................... 137
  A.2 Rayleighverteilung .................................................... 138
  A.3 Riceverteilung ........................................................ 138
  A.4 Dopplereffekt .......................................................... 138
  A.5 Rayleighfading ......................................................... 139
  A.6 Warum ist Kreuzkorrelation nicht iterativ ....................... 139
  A.7 Kostentreiber .......................................................... 140
B Symbole und Abkürzungen
B.1 Variablen ........................................ 147
B.2 Einheiten ........................................ 148
B.3 Operatoren ....................................... 148
B.4 Akronyme ....................................... 149

C Deutsche Übersetzungen
C.1 Titel ............................................. 151
C.2 Zusammenfassung ............................... 151
C.3 Inhaltsverzeichnis ............................. 153
C.4 Einleitung ....................................... 157

Literaturverzeichnis ............................... 159
C.4 Einleitung


Eine Lösung dieses Problems ist der Einsatz von orthogonalem Frequenzmultiplex, abgekürzt OFDM (Orthogonal Frequency Division Multiplexing). Dieses digitale Modulationsverfahren wird in vielen Systemen mit hohen Datenraten eingesetzt, beispielsweise DAB (Digital Audio Broadcast), DVB-T (Digital Video Broadcast - Terrestrial), DSL (Digital Subscriber Line) und WLAN (Wireless Local Area Network).


Den genannten Vorteilen von OFDM stehen Nachteile durch die hohe Anzahl von Trägern gegenüber. Das System wird dadurch sehr empfindlich gegenüber Zeit- und Frequenzverschiebungen, weshalb eine hochpräzise Synchronisation notwendig wird.


Die Arbeit konzentriert sich auf einige ungelöste Probleme zur Optimierung der Synchronisation bei OFDM Systemen hinsichtlich der Komplexität der Implementierung, und Kriterien zur Wahl der optimalen Synchronisierung bei verschiedenen Szenarien. Basierend auf dieser Arbeit sind weitere Untersuchungen sinnvoll, da künftige Technologien und weitere Innovation sich auf die Kostenanalyse und Aufwandsabschätzungen auswirken werden.
Bibliography


[Ber] Bernardin, C. P.: RF System Design: Practical Considerations - Course EE 4365, Univ. of Texas, Dallas, USA


[BL02] Berno, A.; Laurenti, N.: Time and Frequency Synchronization Algorithm for Hiperlan/2(2002), University of Padova (Italy), Springer, V. 2345


[Dlu02] Dlugaszewski, Z.: Comparison of Frequency Offset Estimation Methods for OFDM Burst Transmission in the Selective fading Channels., Institute of Electrical and Telecommunication (2002), Poznan University of Technology, Poland


[DVB01] ETSI EN 300 744 Digital Video Broadcasting (DVB) Framing structure, channel coding and modulation for digital terrestrial television (2001)

[DvbE] www.dvb.org


[Hea] Heath, R. Jr.: *Wireless OFDM Systems*, University of Texas, Austin, USA


[Int00] Intini, A. L.: *Orthogonal Frequency Division Multiplexing for Wireless Networks - Standard IEEE802.11a*, University of California, USA (2000)


[LeC] Lee, D.; Cheun, K.:  *Recent Results in NCTU WLAN Baseband processor (Slides)*  
Department of Electronics Engineers, NCTU, Taiwan

[LiLG01] Li, J.; Liu, G.; Giannakis, G.B.:  *Carrier frequency offset estimation for OFDM-based WLANs*  


[L LN02] Lasanen, M.; Rautio, J.; Nissil, M.:  *Timing Synchronization of the WIND-FLEX OFDM Prototype,*  
In: VTT Electronics (2002), Finland

[LT98] Liu, H.; Tureli, U.:  *A High-Efficiency Carrier Estimator for OFDM Communications.*  

Delft University of Tech., The Nederlands


New York: John Wiley and Sons, 1998

[MM99] Morelli M. and Mengali U.  *Improved Frequency Offset Estimator for OFDM Applications,*  
In: IEEE Communications Letters (1999), Vol. 3, no. 3

[Mo94] Moose, P. H.:  *A Technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction,*  
In: IEEE Transactions on Communications (1994), Vol. 42, no. 10


Univ. of Erlangen-Nuremberg, Germany, Shaker Verlag 2000
[NA99] Nee, van R.; Awater, G.; Morikura, M.; Takanashi, H.; Webster, M.; Halford K. W.:  
*New High-Rate Wireless LAN Standards*, In: IEEE Communications Magazine (1999), Vol. 37, pp. 82-88

[NC02] Negi, R.; Cioffi, J.M.; Blind OFDM symbol Synchronization in ISI Channels 
IEEE Transactions on Communications (2002), Vol. 50, no. 9

[NG03] Nandula, S.; Giridhar, K.:  
*Robust Timing Synchronization for OFDM Based Wireless Lan Systems* In: Conference on Convergent Technologies for Asia-Pacific Region (2003), Vol. 4, pp. 1558-1561

[NP] Nikitopoulos K. and Polydoros A.  
*Post-FFT Fine Frame Synchronization for OFDM systems* Department of Physics, University of Athens, Greece

[Ols] Olsson M.:  

[PeCW] Peng, C.-S.; Cho, C.-H.; Wen, K.-A.:  
*Frame Synchronization and Digital AGC for OFDM based WLAN* National Chiao Tung University, Taiwan

[Pet] Peterson, N.:  
*Peak Power Reduction in Multicarrier Systems* Dissertation, Lund University, Sweden

[PN97] Krishnakurmar Pillai, V. S. Sukumaran Nair:  

[PMN95] Pollet, T.; Van Bladel, M.; Moeneclaey, M.:  

[Pr01] Proakis, J.G.:  

[RaF] Rabano-Rubio, F.-J.:  
*Design of a Software-based Baseband Processor for a Digital Broadcasting System* University of Navarra, Spain

[Ra03] Rajagopal S.:  
*OFDM: Synchronization Issues* Final Report - EE 359, Stanford University, Fall 2003

[Ram01] Ramasami, V. C.:  

[RB01] Ramasubramanian, K.; Baum, K.:  


[ShF] Shizuno, T.; Suyama, S.; Suzuki, H.; Fukawa, K.: An Accurate OFDM Timing Recovery with Time Offset Compensation by Channel Estimation Results for Mobile Packet Transmission Institute of Tech., Tokia, Japan


[Su00] Sunita Chulani: *COCOMO II* Wiley Software Engineering Encyclopedia, Fall 2000


