UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Denise Costa Alves Tamagno

OFDM Frame and Frequency Synchronization in IEEE 802.15.4g: Algorithms and Hardware Implementation

Sincronização de Quadro e Frequência para OFDM no Padrão IEEE 802.15.4g: Algoritmos e Implementação em Hardware

Campinas

2018

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Denise Costa Alves Tamagno

OFDM Frame and Frequency Synchronization in IEEE 802.15.4g: Algorithms and Hardware Implementation

Sincronização de Quadro e Frequência para OFDM no Padrão IEEE 802.15.4g: Algoritmos e Implementação em Hardware

Thesis presented to the Faculty of Electrical and Computer Engineering of the University of Campinas, as part of the requirements for the degree of Master in Electrical Engineering, in the area of Telecommunications.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestra em Engenharia Elétrica, na Área de Telecomunicações e Telemática.

Supervisor/Orientador: Prof. Dr. Renato da Rocha Lopes

Co-supervisor/Coorientador: Dr. Eduardo Rodrigues de Lima

Este exemplar corresponde à versão final da tese defendida pela aluna Denise Costa Alves Tamagno, e orientada pelo Prof. Dr. Renato da Rocha Lopes.

Campinas

2018

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

T15o

Tamagno, Denise Costa Alves, 1988-
OFDM frame and frequency synchronization in IEEE 802.15.4g : algorithms and hardware implementation / Denise Costa Alves Tamagno. – Campinas, SP : [s.n.], 2018.

Orientador: Renato da Rocha Lopes.
Coorientador: Eduardo Rodrigues de Lima.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. OFDM. 2. Sincronização. 3. Sistemas de comunicação sem fio. I. Lopes, Renato da Rocha. II. Lima, Eduardo Rodrigues de. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Sincronização de quadro e frequência para OFDM no padrão IEEE 802.15.4g : algoritmos e implementação em hardware
Palavras-chave em inglês:
OFDM
Synchronization
Wireless communication systems
Área de concentração: Telecomunicações e Telemática
Titulação: Mestra em Engenharia Elétrica
Banca examinadora:
Renato da Rocha Lopes [Orientador]
Aldebaro Barreto da Rocha Klautau Júnior
Cláudio Ferreira Dias
Data de defesa: 18-07-2018
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidata: Denise Costa Alves Tamagno RA: 141082

Data da Defesa: 18 de Julho de 2018

Título da tese em inglês: "OFDM Frame and Frequency Synchronization in IEEE 802.15.4g: Algorithms and Hardware Implementation".

Título da tese em português: "Sincronização de Quadro e Frequência para OFDM no Padrão IEEE 802.15.4g: Algoritmos e Implementação em Hardware".

Prof. Dr. Renato da Rocha Lopes (Presidente, FEEC/UNICAMP)

Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior (Universidade Federal do Pará / Belém)

Prof. Dr. Cláudio Ferreira Dias (Universidade Estadual de Campinas / Campinas)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Dedico esta tese aos meus pais Sérgio e Márcia Alves.

Agradecimentos

Agradeço sempre primeiramente a Deus pelo dom da vida e à minha família, por sempre estimular meus estudos, apoiar todas as minhas decisões, acreditar na minha capacidade e torcer pelo meu sucesso. Agradeço aos meus amigos que convivem comigo, por tornarem meus dias mais alegres e leves, o que alivia minha mente e a prepara todo dia para novos desafios. Agradeço ao Instituto de Pesquisa Eldorado, por me permitir desenvolver este trabalho com recursos da empresa, especialmente ao meu líder de projeto e coorientador Eduardo Lima, por estimular todo o time de desenvolvedores no seu desenvolvimento acadêmico em paralelo com as atividades profissionais. E, finalmente, agradeço ao meu orientador Renato Lopes por acreditar nesta parceria e me orientar para eu concluir este trabalho nesta renomada Universidade.

"Há três caminhos para o fracasso: não ensinar o que se sabe, não praticar o que se ensina, não perguntar o que se ignora."

(São Beda)

Resumo

O objetivo deste trabalho é propor métodos de sincronização de quadro e de frequência de portadora para a camada física MR-OFDM do padrão IEEE 802.15.4g, começando pela pesquisa de algoritmos, passando pelas etapas de modelagem e simulação em alto nível, e finalmente implementando e avaliando os métodos propostos em *hardware*. A sincronização de quadro é o processo responsável por detectar o início do dado transmitido, ou seja, a primeira amostra válida do sinal de interesse. No caso de sistemas OFDM, onde o sinal transmitido é composto por um ou mais símbolos OFDM (cada símbolo sendo composto por uma quantidade fixa de amostras), o objetivo é detectar a borda ou janelamento de tais símbolos OFDM, ou seja, onde começa e termina cada um deles. A sincronização de frequência, por sua vez, consiste em estimar e compensar o erro de frequência de portadora, causado principalmente pelo descasamento dos osciladores do transmissor e do receptor. Com base em estudos preliminares, selecionamos o algoritmo de Minn para a detecção de quadro. Para a correção de erro de frequência, dividimos o processo em duas etapas, como é geralmente proposto na literatura: primeiro, o erro de frequência fracionário é estimado no domínio do tempo durante a detecção de quadro e compensado via rotação de sinal; após a conversão do domínio do tempo para o domínio da frequência, o erro de frequência inteiro é estimado e compensado utilizando um novo e simples algoritmo que será proposto e detalhado neste trabalho. Os algoritmos propostos foram implementados em *hardware* e uma plataforma de verificação baseada em FPGA foi criada para avaliar o seu desempenho. Os módulos implementados são parte de um projeto que está sendo desenvolvido no Instituto de Pesquisa Eldorado (Campinas) que tem como objetivo implementar em ASIC um transceptor compatível com o padrão IEEE 802.15.4g.

Abstract

The objective of this work is proposing methods of frame and frequency synchronization for the MR-OFDM PHY of IEEE 802.15.4g standard, starting with the research of state-of-the-art algorithms, passing through modeling, high-level simulations, and finally implementing and evaluating the proposed methods in hardware. Frame synchronization is the process responsible for detecting the beginning of transmitted data and, in the case of OFDM systems, the border of each OFDM symbol, while frequency synchronization consists of estimating and compensating the Carrier Frequency Offset (CFO) caused mainly by a mismatch between the transmitter and receiver oscillators. Based on the initial studies, we selected Minn's algorithm for frame detection. For the CFO correction, we split the process into two steps, as commonly proposed in the literature: first, the Fractional CFO is estimated in the time domain during the frame detection and compensated via signal rotation; after the conversion from time to frequency domain, the Integer CFO is estimated and compensated with a novel and simple algorithm that will be detailed in this work. The proposed algorithms were implemented in hardware and inserted in an FPGA-based verification platform for performance measurement. The implemented modules are part of a project that is under development at Eldorado Research Institute (Campinas) and aims to implement in ASIC a transceiver compliant to the IEEE 802.15.4g standard.

List of Figures

List of Tables

Acronyms and Abbreviations

ADC Analog to Digital Converter

AWGN Additive White Gaussian Noise

CFO Carrier Frequency Offset

CORDIC Coordinate Rotation Digital Computer

CP Cyclic Prefix

CSI Channel State Information

CTF Cumulative Distribution Function

DFT Discrete Fourier Transform

FCFO Fractional Carrier Frequency Offset

FFT Fast Fourier Transform

FIFO First-In First-Out

FIR Finite Impulse Response

FPGA Field Programmable Gate Array

FSK Frequency Shift Keying

FSM Finite-State Machine

HDL Hardware Description Language

ICFO Integer Carrier Frequency Offset

ICI Inter Carrier Interference

IDFT Inverse Discrete Fourier Transform

ISI Inter Symbol Interference

JTAG Joint Test Action Group

LOS Line of Sight

LTF Long Training Field

LUT Look Up Table

MA Multiply-Accumulate

MAC Medium Access Control

MAE Mean Absolute Error

MCS Modulation and Coding Scheme

ML Maximum Likelihood

MR- Multi-Rate and Multi-Regional

MSE Mean Squared Error

O-QPSK Offset Quadrature Phase Shift Keying

OFDM Orthogonal Frequency Division Multiplexing

PDF Probability Density Function

PDP Power Delay Profile

PHR PHY Header

PHY Physical Layer

PPDU PHY Protocol Data Unit

PSDU Packet Service Data Unit

PSK Phase Shift Keying

QAM Quadrature Amplitude Modulation

RTL Register Transfer Level

SCO Sampling Clock Offset

SHR Synchronization Header

SNR Signal-to-Noise Ratio

SOF Start Of Frame

STF Short Training Field

STO Symbol Timing Offset

SUN Smart Utility Network

TDL Tapped Delay Line

TOA Time Of Arrival

Contents

1 Introduction

Smart Utility Networks (**SUNs**) play a key role in the context of smart grids: they enable multiple applications to operate over shared network resources, support two-way communications among measurement and control devices of a utility system, and frequently cover widespread areas with a large number of outdoor devices [1]. In order to provide the communication range, robustness, and coexistence required for this class of application, the IEEE 802.15.4g standard [1] specifies three alternate Physical Layers (PHYs) in addition to those of IEEE 802.15.4 standard [2], as well as the needed Medium Access Control (MAC) layer modifications. The proposed SUN PHYs are named: Multi-Rate and Multi-Regional Frequency Shift Keying (**MR-FSK**); Multi-Rate and Multi-Regional Orthogonal Frequency Division Multiplexing (**MR-OFDM**); and Multi-Rate and Multi-Regional Offset Quadrature Phase Shift Keying (**MR-O-QPSK**). As the name suggests, the MR-OFDM PHY, which is the focus of this work, is based on OFDM. More details on this PHY will be presented in Chapter 2.

The OFDM modulation has been widely covered in the literature, therefore we will just review a few concepts in order to contextualize our problem. For more details on this modulation scheme, refer to [3]. The basic idea of the OFDM modulation is dividing the available channel band into several narrow sub-bands and modulating the data into adjacent and orthogonal subcarriers. The encoded bits are mapped to symbols by any mapping scheme, such as Phase Shift Keying (PSK) or Quadrature Amplitude Modulation (QAM), and those mapped symbols are grouped and distributed along a finite number of tones (subcarriers), building an OFDM symbol. In an OFDM transmitter, each OFDM symbol is converted from frequency to time domain via Inverse Discrete Fourier Transform (IDFT), which will be covered afterwards. In order to deal with delay spread of wireless channels [3], a Cyclic Prefix (CP) corresponding to a replication of the last portion of each OFDM symbol is usually pre-appended to it. Besides, preamble (sequence of symbols in the beginning of the frame) and pilots (sparse pre-defined mapped symbols among mapped data symbols within OFDM symbols) may be added to the frame (composed of one or more OFDM symbols) in order to aid the synchronization process in the receiver.

Generally, in wireless communication systems, the receiver performance is degraded by several time and frequency impairments, such as oscillator mismatch, unknown propagation delay between the transmitter and the receiver, Doppler effect and phase noise. The consequence is that the received signal presents a combination of synchronization errors, including Sampling Clock Offset (**SCO**), Carrier Frequency Offset (**CFO**) and Symbol Timing Offset (**STO**), also called Symbol Boundary Error or Time Of Arrival (**TOA**) error. Due to the required orthogonality among subcarriers, OFDM systems

are more vulnerable to those errors than conventional single-carrier systems [4]. However, thanks to some advantages, such as the reduced complexity in channel equalization, this modulation technique is still a good solution for high data rate applications, and has been adopted in many standards, like IEEE 802.11 (for Wireless Local Area Network) and IEEE 802.16d (for Metropolitan Area Network) [5].

STO has three main causes: first, the random initialization of the OFDM receiver that makes the OFDM symbol be sampled at a position with a positive or negative offset with respect to the ideal OFDM symbol boundary; the second one is the wrong estimation of the boundary, due to algorithm issues or due to the spreading of the impulse response caused by multipath channel; finally, the third one is the drift from ideal sampling point due to clock errors, caused by the crystal frequency deviation. A direct consequence of the STO is the adoption of a wrong Fast Fourier Transform (FFT) window, when the border error is such that the receiver "sees" a sequence of samples as a single OFDM symbol, when this sequence actually consists of parts of two distinct transmitted symbols. In this case, Inter Symbol Interference (ISI) occurs, since part of the previous/next symbol will be demodulated as belonging to the current symbol [6].

The CFO, in turn, is caused by motion-induced Doppler shifts and/or mismatch between the transmitter and the receiver oscillators. It is commonly normalized in relation to the subcarrier spacing and split into two components: the Fractional CFO (**FCFO**), corresponding to a shift in the spectrum smaller than the subcarrier spacing; and the Integer CFO (**ICFO**), which is equal to a multiple of the subcarrier spacing. The FCFO reduces signal amplitude and introduces Inter Carrier Interference (ICI), while ICFO causes a cyclic shift of subcarrier indices [7]. Generally, the FCFO is first estimated in the time domain by means of autocorrelation and then the ICFO is estimated in the frequency domain, while the complete CFO is compensated by a feedback chain.

In order to properly recover the transmitted data, special signal processing must be realized by the OFDM receiver, which generally relies on some extra data added by the transmitter, such as the CPs, the sparse pilots, and the preamble, which can be composed of one or more OFDM symbols. Several works in the literature have explored OFDM synchronization issues and proposed solutions. Some of them combine more than one kind of error in a single work, while others focus on specific errors. A brief literature review on time and frequency recovery will be presented next.

One of the classical works in this area is by Van de Beek [8], which proposes a joint Maximum Likelihood (ML) algorithm for time and frequency offset estimation that takes into account only the CP present in OFDM symbols, without requiring additional pilots. However, the proposed metric is also based on the estimated Signal-to-Noise Ratio (SNR), and this dependence represents more implementation complexity and less reliability, since the required SNR estimation process is subject to error as well [4].

Speth et al. divide their work into two parts: the first one [6] analyzes the effect of non-ideal transmission conditions, including fading channel, frequency and timing errors; the second one [9] presents the design of a complete receiver consisting of symbol synchronization, carrier/sampling clock synchronization and channel estimation, targeting the European standard DVB-T for digital TV. Regarding time and frequency recovery, all the algorithms selected by the authors were based on maximizing correlations with the use of CP, as well as of continual and scattered pilot tones (in this case, just a few tones within the OFDM symbols are known to both transmitter and receiver, and they may appear in different positions). They designed a receiver for streamed data, which allowed the selection of non data-aided recovery algorithms for acquisition and tracking that could run for relatively long periods. The approach in [9] is different from ours, that targets packet-based data transmission, and so must focus on preamble-aided algorithms.

Schmidl and Cox [10] propose an algorithm for frame and CFO recovery based on two OFDM pilot symbols that maximizes a power-normalized metric to detect the beginning of the frame, and then use the same metric to estimate the frequency error. However, such metric based on autocorrelation in time domain presents a plateau with width proportional to the CP, which can lead to an offset between the estimated and the perfect frame start point. Besides, due to the structure of the proposed preamble, the CFO range is limited to ± 2 subcarriers.

Minn [11] generalizes the approach in [10] by forming the preamble with as many repetitions of a given subsequence as required, with different polarities, in order to increase the CFO range according to the number of repetitions and eliminate the plateau presented in [10]. Another scheme based on [10] is presented in [12], which considers a single-symbol preamble preceded by a conveniently-wide CP to derive an estimator for time and fractional frequency errors. The estimation process is split into two stages, the first one based on [10], and the second one based on a differential correlation computed on a shorter portion of the preamble, which represents a reduced complexity.

In [13], a frame synchronization method is proposed for the IEEE 802.11a standard applying the Mean Squared Error (MSE) and defining a new periodicity metric. Due to the similarity between the frame structures defined by IEEE 802.11a and IEEE 802.15.4g standards and to its reduced complexity at the acquisition phase, this algorithm was the first one proposed for the latter standard as part of our work [14]. However, it was verified afterwards that such scheme was not robust to CFO error, which led us to use the scheme proposed in [11].

In all the references mentioned so far, the FCFO is commonly estimated by means of time domain correlation during the frame detection. Regarding the ICFO, it is generally estimated after the FFT, in the frequency domain. Works as [15, 16] propose ICFO estimation algorithms based on ML, robust to both timing offset and multipath

channels, while other works as [17, 18] jointly estimate the ICFO and the Channel State Information (CSI) via ML. However, those are methods based on exhaustive grid-search, with performance efficiency proved only by means of computer simulation. Furthermore, no hardware implementation is proposed.

Other methods based on differential demodulation of active pilots in the frequency domain are presented in [19, 20]. A more complete work for frequency-domain ICFO estimation can be found in [21], which adapted the method presented in [19] to propose a low complexity hardware implementation for the estimator. In our work, we derive the complete mathematics that results in the same algorithm, then we simplify such algorithm for the specific case of IEEE 802.15.4g standard, and also propose a novel and simple scheme for ICFO compensation in the frequency domain, by taking into account not only the cyclic-shifted tone indices, but also a phase accumulation occurred due the previous presence of the CP, which is commonly neglected in the studied literature.

Our purpose in this work is proposing methods for symbol boundary detection and CFO estimation and compensation, targetting the MR-OFDM PHY of IEEE 802.15.4g standard. In the following sections, we will introduce the main characteristics of the standard; present the three issues (STO, FCFO and ICFO) and detail the algorithms proposed to solve them; present the architectures designed to implement those algorithms; and finally present the results of high-level simulations and tests in hardware.

Regarding our scientific contribution to the literature, we have presented in [14] the first version of the frame synchronizer, which was based on [13]. We also have the paper “A Low Complexity ICFO Estimator and Compensator for IEEE 802.15.4G MR-OFDM PHY: Algorithm Proposal and Hardware Implementation” recently accepted to be presented in “IEEE International Symposium on Personal, Indoor and Mobile Radio Communications 2018”, which will be held on September 2018.

2 IEEE 802.15.4g MR-OFDM PHY

In the previous chapter, we introduced the synchronization issues in OFDM systems that this work aims to solve targeting the MR-OFDM PHY of the IEEE 802.15.4g standard, and we also presented a literature review of some classical and state-of-the-art synchronization methods proposed so far. In this chapter, we introduce the IEEE 802.15.4g standard, detail the features relevant for frame and frequency recovery, and present the architecture of the implemented receiver in order to locate the modules proposed in this work.

2.1 The standard

The MR-OFDM PHY supports data rates from 50 kbps to 800 kbps and includes four operation modes, each one characterized by the number of active tones (subcarriers) along the header and the payload [1]. Table 2.1 summarizes the main parameters for options from 1 to 4, and Table 2.2 presents the available Modulation and Coding Schemes (MCS).

PARAMETER	OPTION 1	OPTION 2	OPTION 3	OPTION 4	UNIT
FFT SIZE	128	64	32	16	-
NOMINAL BANDWIDTH	1094	552	281	156	kHz
TONE SPACING	10416.667	10416.667	10416.667	10416.667	Hz
BASE SYMBOL DURATION	96	96	96	96	us
CP DURATION	24	24	24	24	us
SYMBOL DURATION	120	120	120	120	us
SYMBOL RATE	8.333	8.333	8.333	8.333	kSymbols/sec
SAMPLE RATE	1333.33	666.66	333.33	166.66	kHz
#ACTIVE TONES	104	52	26	14	-
#PILOT TONES	8	4	2	2	-
#DATA TONES	96	48	24	12	-
#DC NULL TONES	1	1	1	1	-

Table 2.1 – Main MR-OFDM Parameters

	MCS0	MCS1	MCS2	MCS3	MCS4	MCS5	MCS6
MODULATION	BPSK	BPSK	QPSK	QPSK	QPSK	16-QAM	16-QAM
CODE RATE	1/2	1/2	1/2	1/2	3/4	1/2	3/4
FREQ. SPREADING	4x	2x	2x	-	-	-	-

Table 2.2 – Modulation and Coding Schemes

Data is transmitted in frames called PHY Protocol Data Units (**PPDU**), which are composed of the fields shown in Figure 2.1. The Synchronization Header (SHR) is composed by a Short Training Field (STF) and a Long Training Field (LTF), and it is

used in the receiver for detecting the beginning of the frame and correcting frequency errors. The PHY Header (PHR) contains the configuration of the frame, including the MCS, the frame length and scrambling seed. The Packet Service Data Unit (PSDU) is the PHY payload, i.e. the field that carries useful data, and it is followed by the Tail and Pad fields, used respectively to flush the convolutional encoder and to fill the OFDM symbols when the number of mapped symbols (resulting from previous BPSK, QPSK or 16-QAM mapping) is not a multiple of the OFDM symbol size. For the purpose of this work, we will focus on the STF and the LTF, which are described in more details in the sequel.

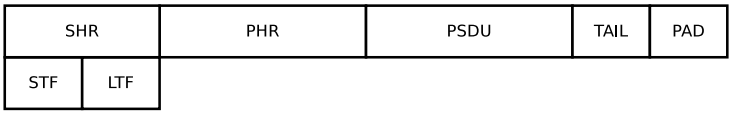

Figure 2.1 – Format of the MR-OFDM PPDU

2.1.1 Short Training Field (STF)

The STF is the first field of the MR-OFDM PPDU and it is used for detecting the beginning of the frame. In the IEEE 802.15.4g standard [1], it is described by four tables that set the frequency-domain representations for all operation modes, i.e. each table determines which tones must be active during the transmission of the STF OFDM symbols on the respective operation mode. For options 1 and 2, 12 tones have non-zero contribution, while for options 3 and 4, the number of active tones is 6. For details refer to [1] or to Annex A.

In order to build the complete STF field in the time domain, first a single time-domain OFDM symbol is generated via IDFT by using the tones pre-defined in frequency-domain representation, according to

$$f(i) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{j2\pi ik/N}, \tag{2.1}$$

where N is the amount of samples (or DFT/IDFT points), $i = 0 \ldots N-1$ is the time domain sample index, $k = 0 \ldots N-1$ is the tone index, and $F(k)$ is the contribution of the k-th tone (extracted from the frequency-domain tables defined in the standard). Then, four time-domain OFDM symbols are concatenated, and the last half of the fourth symbol is negated. Besides, each OFDM symbol employs a CP of length $N/4$ which means that the length of the complete STF field is $4 \times (1 + 1/4) \times N = 5N$ samples, as presented in Figure 2.2.

For a single OFDM symbol, the active tones in the STF are chosen so that the IDFT operation results in a periodic complex signal that is composed of 2 or more time-domain sub-sequences (or repetitions), depending on the operation mode. For instance, in option 1, a single time-domain OFDM symbol with N points is composed of 8 repetitions

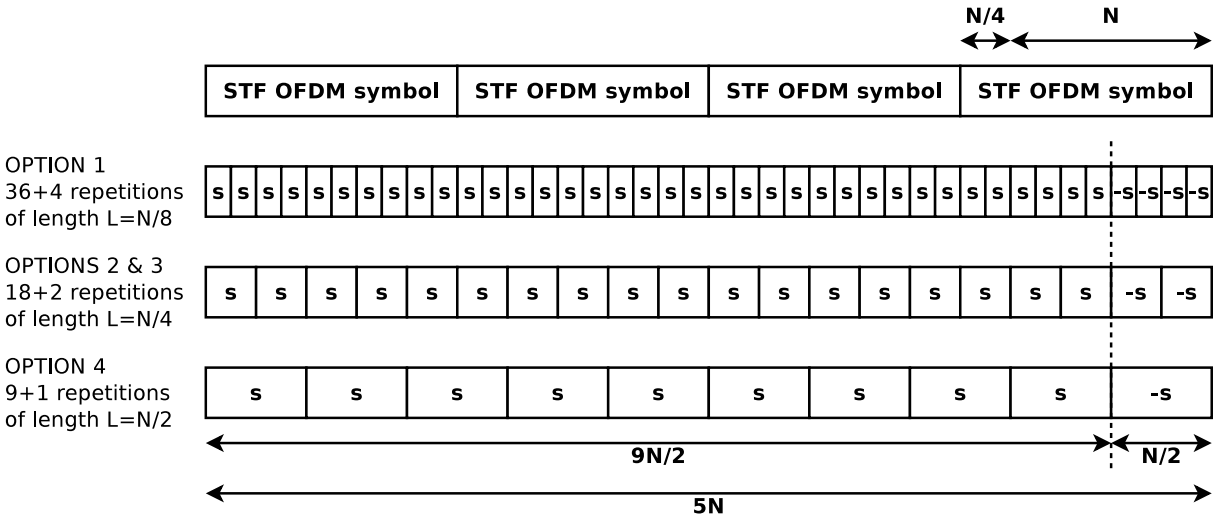

Figure 2.2 – Time-domain structure of STF for OFDM Options 1, 2, 3 and 4.

"s" with length $L = N/8$. Therefore, instead of considering the complete STF field as a concatenation of four independent OFDM symbols that must pass through the IDFT and have their respective CPs attached, we can see it just as a time-domain periodic signal that is composed by a given number of repetitions (with the last ones negated), and each repetition corresponds to a fraction of the result of a single IDFT operation. Table 2.3 summarizes the length and the amount of repetitions for all OFDM options.

Option	N**: DFT size**	**#normal repetitions**	**#negated repetitions**	L**: repetition length**
1	128	36	4	N/8
2	64	18	2	N/4
3	32	18	2	N/4
4	16	9	1	N/2

Table 2.3 – STF repetitions.

2.1.2 Long Training Field (LTF)

The LTF is the second field of the MR-OFDM PPDU and is used in the receiver for estimating the ICFO and the channel response. In the standard, it is described by four tables that set the frequency-domain representations for all operation modes, as it is done for the STF field. However, different from the STF, almost all the tones, except for the DC and guard tones, present non-zero contribution. For options from 1 to 4, the number of active tones are respectively 104, 52, 26 and 14. For details on the LTF frequency-domain representation, refer to [1] or to Annex A.

A single time-domain LTF OFDM symbol is generated via IDFT as in (2.1), by using the tones pre-defined in frequency-domain representation. The complete LTF field

is composed by a CP equal to half OFDM base symbol (result of IDFT) followed by two copies of the LTF base symbol, as presented in Figure 2.3.

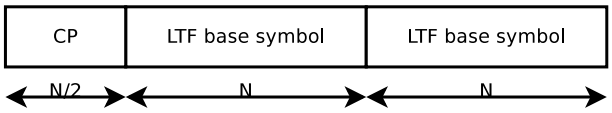

Figure 2.3 – Structure of LTF for OFDM Options 1, 2, 3 and 4.

2.2 The implemented receiver

In this section, we present the IEEE 802.15.4g MR-OFDM receiver architecture that is being implemented as part of a bigger project, and where the proposed frame and frequency synchronization modules are located. The complete transceiver (composed of both transmitter and receiver portions) has been already presented in [22], however, more blocks have been implemented in the receiver since then.

The baseband receiver architecture presented in Figure 2.4 was designed based on the reference modulator defined by the standard [1] and on well-known synchronization issues associated to OFDM systems. It can be split into **inner receiver**, responsible for signal detection, for estimation and correction of frequency and timing errors and for equalization, and **outer receiver**, responsible for reverting the data encoding process done by the transmitter.

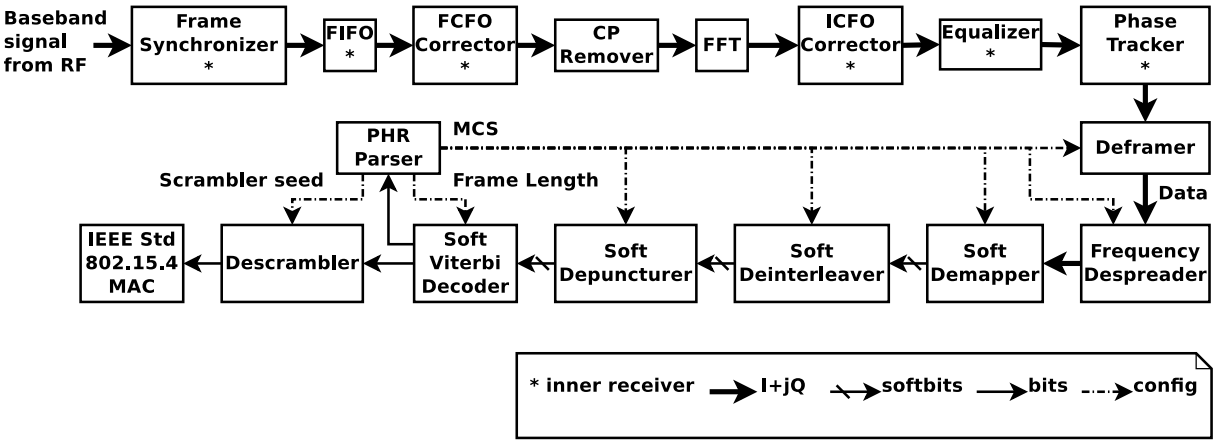

Figure 2.4 – Baseband OFDM Receiver

The **Frame Synchronizer** detects the beginning of the frame by looking for repetitive sequences in time domain corresponding to the STF portion, and also estimates the FCFO. The First-In First-Out (**FIFO**) block changes the dataflow from free running mode (only Frame Synchronizer) to a domain that uses a ready-valid protocol defined in this project. This protocol will not be detailed here, but it basically consists of a few signals that indicates whether the next block is ready to receive a new data and whether

the current block has a valid data to send. The **FCFO Corrector** compensates the error estimated by the Frame Synchronizer via derotation. The **CP Remover** removes the CP inserted at the beginning of each time domain OFDM symbol. The **FFT** block [23] converts the data from time to frequency domain. The **ICFO Corrector** estimates the ICFO by correlating the LTF samples in frequency domain and then compensates the error by cyclic shifting the tones. The **Equalizer** estimates the channel response based on the LTF field and compensates its influence over the received samples, as well as the phase caused by tolerable STO. The **Phase Tracker** estimates and compensates residual phase error caused by residual CFO and SCO.

In the receiver, at the beginning of the reception, the configuration (frame length, modulation, code rate etc.) of the received frame is unknown until the PHR is decoded. Therefore, from the Deframer forward, only the number of symbols corresponding to the PHR field are passed to the next blocks, since its size is fixed for each OFDM option. The PSDU field is passed afterwards, when the PHR has been decoded and its fields corresponding to the PSDU configuration have been recovered.

The **Deframer**, **Deinterleaver**, **Depuncturer** and **Descrambler** blocks are similar to their counterpart in the transmitter: they simply revert the encoding process detailed in [1]. When frequency spreading, which may be of 2 or 4 times [1], has been applied to transmitted symbols, the **Frequency Despreader** combines the replicated carriers in order to recover the spread mapped symbols. The **Soft-demapper** then receives those data tones, represented as I and Q components, and calculates the soft-bits, which are integer numbers that represent the probability of each bit within a given symbol being '0' or '1'. The **Viterbi Decoder** recovers the data encoded by the convolutional encoder, correcting errors generated during the transmission. The **PHR Parser** extracts the configuration inserted in the PHR field and notifies the decoded configuration to the other blocks, resuming the frame reception for processing the PSDU field.

The synchronization modules detailed in this work are: the **Frame Synchronizer**, the **FCFO** and the **ICFO Correctors**.

3 Proposal of Synchronization Methods

In the previous chapters, we stated the problem of this work, which is time and frequency synchronization for the MR-OFDM PHY of IEEE 802.15.4g standard, and we also presented additional information regarding the standard itself and the receiver architecture that includes the synchronization modules. In this chapter, we propose the algorithms for time and frequency synchronization. Based on the study of several works, which were reviewed in Chapter 1, we propose two algorithms:

1. Frame synchronization and FCFO estimation based on the proposal of [11] with a few modifications;
2. and a novel method for ICFO estimation and compensation in the frequency domain.

3.1 Frame Synchronization

In this section we address the problem of frame synchronization in OFDM systems, which is basically the process of discovering the beginning of the frame and detecting the borders of the OFDM symbols.

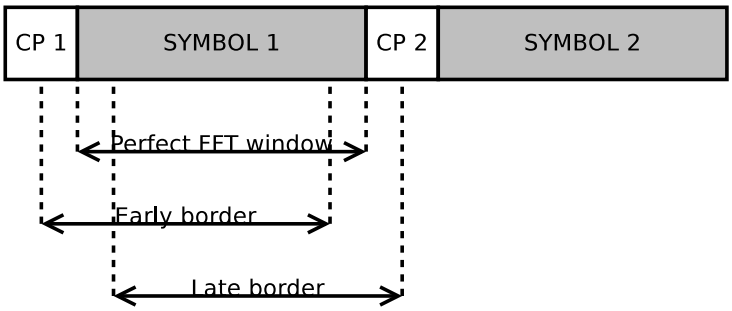

Figure 3.1 – FFT window

As already mentioned in Chapter 1, a consequence of the timing offset is the adoption of a wrong FFT window in the OFDM receiver. This offset can be either towards CP (early border) or towards next OFDM symbol (late border), as presented in Figure 3.1. When the OFDM symbol border is early but within the CP region, a phase rotation occurs in the frequency domain, which can be handled by the equalizer [6]. On the other hand, in case of late border or even too early border, i.e. outside the CP region, the orthogonality among subcarriers are destroyed by the resulting ISI and ICI. For more details on the effects of timing offset, refer to [11]. A common approach to aid in the frame synchronization process is the inclusion of a preamble in the beginning of the

frame, in addition to the OFDM data symbols. This is the case of the IEEE 802.15.4g MR-OFDM PHY and of the algorithm we show next.

In [11], Minn proposed a synchronization algorithm based on autocorrelation that generalizes the frame detection method presented in [10]. While [10] defines the frame preamble as a concatenation of two OFDM symbols preceded by a CP, [11] proposes as preamble a sequence of two or more time-domain repetitions which can have different polarities. It basically defines a basis sub-sequence and then replicates it by keeping or changing its sign as many times as required to build the preamble. The length of such sub-sequence, the number of repetitions and the sign pattern may vary according to the application.

We will not model here the data transmission to present how the algorithm was developed, but only show its expression and how we adapted it to our work. For more details, refer to [11]. The Minn timing metric considers a training symbol containing R parts of L samples each and takes into account the signs (polarities) of the repetitions $p(u) : u = 0, 1, ..., R-1$. The metric to be maximized to find the beginning of the preamble is calculated as:

$$M_{minn}(t) = \left(\frac{R}{R-1} \frac{|P(t)|}{E(t)} \right)^2 \tag{3.1}$$

where

$$P(t) = \sum_{u=0}^{R-2} b(u) \sum_{i=0}^{L-1} r^*(t + uL + i) r(t + (u+1)L + i),$$

$$E(t) = \sum_{u=0}^{R-1} \sum_{i=0}^{L-1} |r(t + uR + i)|^2,$$

$$b(u) = p(u)p(u+1)$$

and $r(t)$ is the received signal. Based on Figure 2.2 and Table 2.3 of Chapter 2, we conclude that the number of repetitions and their length vary for each option of the MR-OFDM PHY. However, in order to make the hardware implementation easier, we can also consider that, for all options, the number of repetitions is $R = 10$, while their length is $L = 5N/10$, i.e., 64, 32, 16 and 8 for options 1, 2, 3 and 4 respectively.

Besides, we can also simplify (3.1) through an approximation: we compute the STF energy $E(t)$ based on $R-1$ repetitions, instead of all R repetitions, and then, suppress the scaling factor $R/(R-1)$. This way, the calculation of the simplified metric to be implemented is:

$$M_{sm}(t) = \left(\frac{|P(t)|}{E_{sm}(t)} \right)^2 \tag{3.2}$$

$$E_{sm}(t) = \sum_{u=1}^{R-1} \sum_{i=0}^{L-1} |r(t + uR + i)|^2.$$

Figure 3.2 illustrates how the metric is calculated in the implemented Frame Synchronizer, with the curved arrows representing the multiplications that are executed between samples spaced by L and then summed up to return the autocorrelation result. The correlation is executed using all samples, but the energy is calculated by taking into account the 9 repetitions in gray.

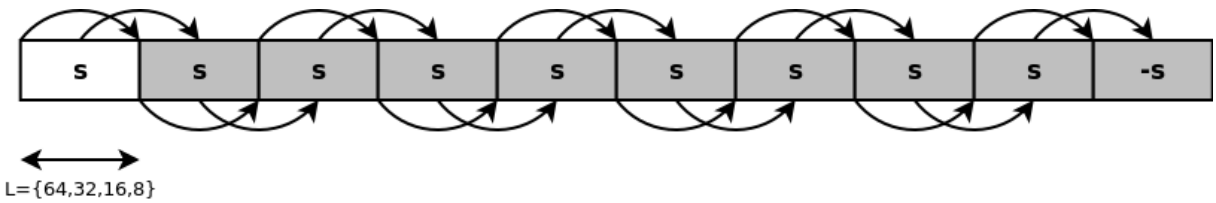

Figure 3.2 – Minn metric calculation in the proposed simplification

During the execution of the detection algorithm, it is unfeasible to store all the metrics to look for peaks among them. Therefore, a metric threshold T_{sh} is set so that, once a given metric achieves that value, the frame synchronizer can wait for a limited number of samples, and then it flags the sample with the largest associated metric as a Start Of Frame (SOF) .

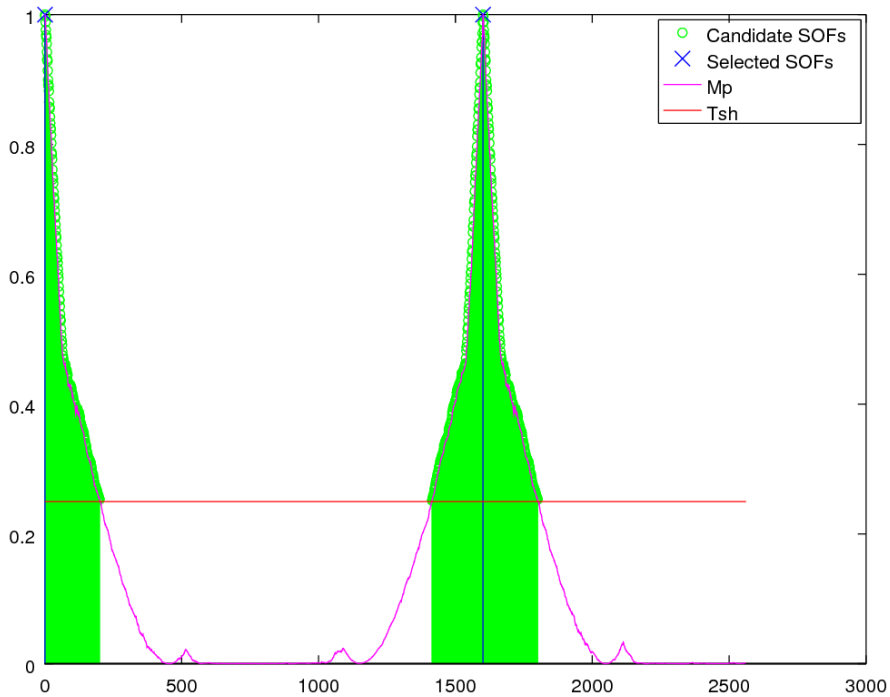

Figure 3.3 – Metrics calculated over two consecutive frames

Figure 3.3 presents an example of metrics calculated over samples of two consecutive PPDU frames without noise. The peaks correspond to the beginning of each frame,

the horizontal line corresponds to an arbitrary threshold $T_{sh} = 0.25$, the green circle markers (highlighted by the the green region) correspond to candidate SOFs whose metrics surpassed T_{sh}, and the "x" markers correspond to the selected SOFs that had the largest metrics among the candidate SOFs.

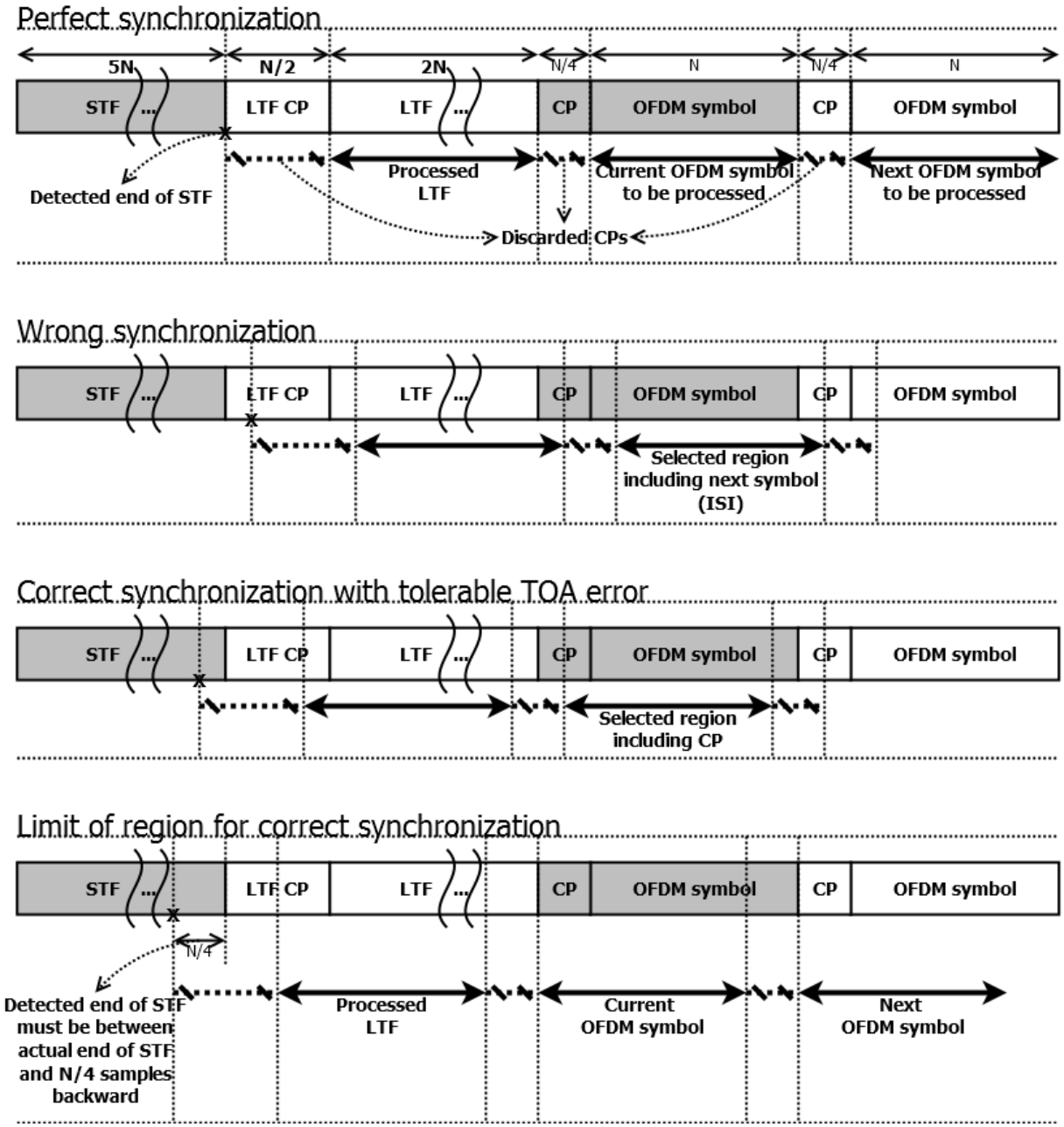

Figure 3.4 – STF detection

Another parameter for the frame synchronizer is the guard offset. Due to several impairments, such as multipath channel and noise, the acquisition algorithm is subject to errors, leading to detected borders that may be before or after the ideal border. Note that the beginning of the frame does not need to be perfectly detected, a TOA error is tolerable as long as the border is detected a slightly early. In this case, due to the presence of the CP, the FFT window may keep the samples from a single OFDM symbol. On the

other hand, if the detection occurs late or too early, it causes ISI, respectively from the next or the previous OFDM symbol. Figure 3.4 illustrates in which cases the detection is considered correct or wrong.

As explained above, the system does not tolerate any late frame border, but may be robust to an early border. Thus, the implemented frame synchronizer can prevent occasional ISI problems by indicating the end of STF a little earlier in relation to the point actually detected by the algorithm. In this work, this offset is generally set as $N/16$ or $N/8$, which are smaller than the CP length $N/4$. Note that the tolerable margin of $N/4$ samples for early detection is equal to the CP length if we consider channel with no fading. However, when the channel presents a delay spread that takes one or more sample periods, this margin reduces, as illustrated in Figure 3.5.

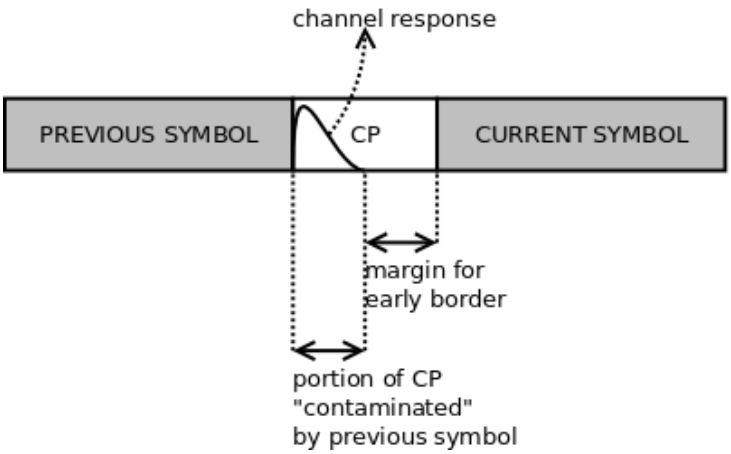

Figure 3.5 – Early border margin in the presence of spreading channel

3.2 FCFO Estimation and Compensation

The FCFO is estimated simultaneously when the symbol timing is acquired. The autocorrelation $P(t)$ used in (3.2) is also used for estimating the normalized FCFO via the closed-form presented in [4]:

$$\hat{\lambda} = \frac{\angle P(\hat{t})}{2\pi L/N}, \tag{3.3}$$

where $\hat{t}$ is the sample index that maximizes the Minn metric in (3.2). Note that the phase can only be resolved in $[-\pi, \pi]$, so (3.3) only estimates the part of the CFO that is within $-N/2L \leq \hat{\lambda} \leq N/2L$. When $L = N/2$, as proposed in previous section, this range is $-1 \leq \hat{\lambda} \leq 1$. The residual integer part is estimated afterwards. The FCFO is compensated by means of phase accumulation and rotation, i.e. by multiplying the received samples $r(t)$ by a complex sinusoid whose frequency is the negated estimated FCFO, as in $r'(t) = r(t) \times e^{-j2\pi\hat{\lambda}t/N}$.

3.3 ICFO Estimation and Compensation

For ICFO estimation, we adopt the same algorithm presented in [21], which is robust to both multipath channel and symbol boundary error. However, as we developed a simplification for the specific case of IEEE 802.15.4g standard and a novel and simple scheme for frequency-domain compensation, we will model here the complete data transmission. In order to make the understanding easier, some concepts or formulas already covered in previous sections may be repeated below.

Let us consider a sequence of complex symbols generated by an arbitrary constellation mapping (e.g. PSK, QAM). Those symbols are organized in blocks of N components that must be converted to the time domain via IDFT, according to

$$x_{l,i} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_{l,k} e^{j2\pi ik/N}, \tag{3.4}$$

where N is the OFDM symbol size, $k = 0, 1, ..., N-1$ is the tone index, $X_{l,k}$ is the k-th component of the l-th OFDM symbol, $i = 0, 1, ..., N-1$ is the sample index, and $x_{l,i}$ is the i-th sample of the l-th OFDM symbol in time domain. Generally, in order to avoid ISI in OFDM systems, a CP composed of the last N_g samples is prepended to each OFDM symbol after the IDFT. This way, the sample index can be extended to $i = -N_g, -N_g+1, ..., 0, 1, ...N-1$, and a single time-domain OFDM symbol l with CP can be described as $x_l = (x_{l,-N_g}, x_{l,-N_g+1}, ..., x_{l,0}, x_{l,1}, ..., x_{l,N-1})$.

In order to model the impairments of the received baseband signal and derive the ICFO recovery algorithm, we make the following assumptions: the detected start of frame is perfect or early as long as the residual timing offset δ_t in samples is toward the CP region; the channel represented by the impulse response $h = (h_0, h_1, ..., h_{L-1})$, with duration of L samples, is constant during the transmission of a single OFDM symbol; and the combination of timing offset and channel duration is $\delta_t + L \leq N_g$ so that no ISI occurs. As previously mentioned, the CFO can be decomposed into an integer and a fractional components, so the CFO normalized in relation to the subcarrier spacing is given by $\delta_f = \epsilon + \lambda$, where $-0.5 \leq \lambda < 0.5$ is the normalized FCFO and ϵ is the normalized ICFO. Those error components are generally estimated separately in the receiver [21]. In fact, we have already discussed how the FCFO can be estimated and compensated. Therefore, we now assume perfect FCFO compensation in the time domain (as it is also assumed in [17]), so that the normalized CFO will be defined in this model only by its integer component, as $\delta_f = \epsilon$.

Given those assumptions, each received OFDM symbol can be described as

$$y_{l,i} = x_{l,i-\delta_t} \circledast h_i e^{j2\pi\epsilon/N(d_l+N_g+i-\delta_t)} + w_{l,i}, \tag{3.5}$$

where d_l is the sample offset from the time origin to the CP of the l-th OFDM symbol (for instance, when all the OFDM symbols have the same CP length N_g, then $d_l = l(N+N_g)$),

$\circledast$ is the circular convolution operation, and $w_{l,i}$ is a complex-valued Additive White Gaussian Noise (AWGN). In order to recover the transmitted data, it is necessary to discard the CP and convert the received samples from time to frequency domain via Discrete Fourier Transform (DFT), according to

$$Y_{l,k} = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} y_{l,i} e^{-j2\pi ik/N}. \tag{3.6}$$

After a few manipulations of (3.6) with use of (3.4), (3.5), the time shift property of DFT and the convolution theorem, the received OFDM symbol in the frequency domain can be described as

$$Y_{l,k} = e^{j2\pi\epsilon/N(d_l+N_g)} X_{l,k-\epsilon} H_{k-\epsilon} e^{-j2\pi k\delta_t/N} + W_{l,k}, \tag{3.7}$$

where H_k and $W_{l,k}$ are the DFT of h_i and $w_{l,i}$ respectively.

Note the term $e^{j2\pi\epsilon/N(d_l+N_g)}$ is independent of the tone index k, while $e^{-j2\pi k\delta_t/N}$ is independent of the OFDM symbol index l, therefore we can define a common phase applied to all the tones of the l-th OFDM symbol as in

$$\theta_l = 2\pi\epsilon(d_l + N_g)/N, \tag{3.8}$$

as well as a common phase applied to all the OFDM symbols at a given subcarrier k as in

$$\phi_k = -2\pi k\delta_t/N. \tag{3.9}$$

Applying (3.8) and (3.9) into (3.7) results in

$$Y_{l,k} = e^{j\theta_l} X_{l,k-\epsilon} H_{k-\epsilon} e^{j\phi_k} + W_{l,k}. \tag{3.10}$$

At this point, it is possible to identify two effects of the ICFO on the OFDM symbols in frequency domain:

1. cyclic shift in the tone indices;
2. and phase accumulation in a symbol-by-symbol basis due to previous presence of the removed CP.

Note that if $N_g = 0$ in (3.8) for all OFDM symbols, the offset d_l to symbol l would be $d_l = lN$, and the phase would reduce to $\theta_l = 2\pi\epsilon l$, i.e. the term $e^{j\theta_l}$ of (3.10) would equal 1, and only the cyclic shift in subcarrier indices would remain as an ICFO effect.

The derivations presented in the sequel is based on the one presented in [19]. The estimation of the ICFO $\hat{\epsilon}$ is based on a pilot symbol known by both transmitter and

receiver. If we neglect the noise in (3.10) in order to derive the estimation algorithm via correlation, the multiplication of adjacent tones within a single pilot symbol results in

$$\begin{aligned} & Y^*_{REF,k} Y_{REF,k-1} \\ & \approx H^*_{k-\epsilon} X^*_{REF,k-\epsilon} H_{k-\epsilon-1} X_{REF,k-\epsilon-1} e^{j2\pi\delta_t/N} \\ & \approx |H_{k-\epsilon}|^2 X^*_{REF,k-\epsilon} X_{REF,k-\epsilon-1} e^{j2\pi\delta_t/N}, \end{aligned} \tag{3.11}$$

where $X_{REF,k}$ is the k-th tone of the reference pilot symbol and $Y_{REF,k}$ is the k-th tone of the received pilot symbol. Note that (3.11) assumes the channel contribution in adjacent carriers are similar, as it is assumed in [19], which is quite reasonable considering a channel with exponential power delay profile (the concept of power delay profile is detailed in Section 5.1.4 of Chapter 5, when the channel models used in this work are presented). By making $D_k = X^*_{REF,k} X_{REF,k-1}$ and $A_k = Y^*_{REF,k} Y_{REF,k-1}$, (3.11) can be rewritten as

$$A_k \approx e^{j2\pi\delta_t/N} D_{k-\epsilon} |H_{k-\epsilon}|^2. \tag{3.12}$$

Given that the effect of the timing offset δ_t is a phase rotation and all the terms $D_{k-\epsilon}$ rotate the same way, we can define a correlation function of a candidate $\tilde{\epsilon}$ as

$$R(\tilde{\epsilon}) = \left| \sum_{k=0}^{N-1} A_k D^*_{k-\tilde{\epsilon}} \right| \tag{3.13}$$

and estimate the ICFO by maximizing (3.13), as show in

$$\hat{\epsilon} = \max_{\tilde{\epsilon}} R(\tilde{\epsilon}). \tag{3.14}$$

From the inspection of (3.8), (3.9), and (3.10), it is clear that the ICFO (present in the term $e^{j\theta_l}$ and in the index shift of $X_{l,k-\epsilon}$) can be compensated in the frequency domain in an OFDM symbol-by-symbol basis, while the channel and the time offset (presented in the term $H_{k-\epsilon} e^{j\phi_k}$), can be easily compensated in a carrier-by-carrier basis further on by an equalizer in frequency domain. Therefore, we can reduce our problem to the term $e^{j\theta_l} X_{l,k-\epsilon}$.

In order to recover the frequency-domain mapped symbols within a given OFDM symbol l, the ICFO must be estimated according to (3.14) and then it can be compensated in frequency domain in two steps:

1. shift the carriers of the DFT result by $-\hat{\epsilon}$;
2. rotate each tone of the received OFDM symbol $Y_{l,k}$ by $-\theta_l|_{\epsilon=\hat{\epsilon}}$.

Let us now consider the common case where all the OFDM symbols have the same CP length, i.e. $d_l = l(N + N_g)$. Based on (3.8), the phase rotation can be rewritten as

$$e^{j\theta_l} = e^{j2\pi\epsilon(l(N+N_g)+N_g)/N} = e^{j(2\pi/r_g)\epsilon(l+1)}, \tag{3.15}$$

where $r_g = N/N_g$ is an integer. From (3.15), the phase is accumulated for each OFDM symbol with a step of $\theta_{step} = (2\pi/r_g)\epsilon$. Besides, as both r_g and ϵ are integers, the phase θ_l assumes a finite number of values, more precisely, $e^{j\theta_l}$ covers at most r_g points on the complex plane, which means that the derotation may be simplified, via Look Up Table (LUT) for example. Specially for $r_g = 4$, which is the case of IEEE 802.15.4g, the accumulated phase will assume values in $\theta_l \in \{0, \pi/2, \pi, 3\pi/2\}$, for which the rotation $e^{\theta_l} \in \{0, j, -1, -j\}$ can be compensated only by means of sign multiplication, as summarized in Tables 3.1 and 3.2. Table 3.1 presents how the phase is accumulated along a sequence of OFDM symbols depending on the ICFO ϵ, and Table 3.2 presents how the derotation is executed on the received tones $Y_{l,k}$ to recover the transmitted tones $\hat{X}_{l,k}$, based on the estimated ICFO $\hat{\epsilon}$. In Table 3.1, *mod* represents the remainder of an integer division, and in Table 3.2, $\Re(.)$ and $\Im(.)$ represent respectively the real and imaginary components of a complex number.

ϵ mod 4	θ_l, $l = (0, 1, 2, 3, 4...)$
0	$(0, 0, 0, 0, 0...)$
1	$(\pi/2, \pi, 3\pi/2, 0, \pi/2...)$
2	$(\pi, 0, \pi, 0, \pi...)$
3	$(3\pi/2, \pi, \pi/2, 0, 3\pi/2...)$

Table 3.1 – Phase accumulation for $r_g = 4$

$-\hat{\theta}_l$	$\Re\{\hat{X}_{l,k}\}$	$\Im\{\hat{X}_{l,k}\}$
0	$\Re\{Y_{l,k+\hat{\epsilon}}\}$	$\Im\{Y_{l,k+\epsilon}\}$
$\pi/2$	$-\Im\{Y_{l,k+\hat{\epsilon}}\}$	$\Re\{Y_{l,k+\epsilon}\}$
π	$-\Re\{Y_{l,k+\hat{\epsilon}}\}$	$-\Im\{Y_{l,k+\epsilon}\}$
$3\pi/2$	$\Im\{Y_{l,k+\hat{\epsilon}}\}$	$-\Re\{Y_{l,k+\epsilon}\}$

Table 3.2 – De-rotation for $r_g = 4$

3.3.1 ICFO estimation and compensation for MR-OFDM PHY

The proposed ICFO estimator uses only the LTF field of the MR-OFDM PPDU, which was already presented in Figure 2.3, since all its active tones carry non-zero training symbols (for details, see [1] or Annex A). Following the LTF field, all the header and data OFDM symbols are generated the same way: each frequency domain symbol is converted to time domain via IDFT and preceded by a CP of $N/4$ samples.

At this point, we make the following remarks:

1. as previously mentioned, the tolerable time offset lies within $0 \leq \delta_t \leq N_g = N/4$ (considering the CP length of the data symbols);
2. as the LTF training symbol is composed only by ± 1 and 0, the multiplication of adjacent tones will have the values $D_k \in \{0, -1, 1\}$.

Given that, based on (3.12) and (3.13), we can rewrite the correlation function

along a single LTF symbol as

$$R_{LTF}(\tilde{\epsilon}) \approx \left| \sum_{k=0}^{N-1} e^{j2\pi\delta_t/N} D_{k-\epsilon} |H_{k-\epsilon}|^2 D^*_{k-\tilde{\epsilon}} \right|$$
$$\approx \left| \sum_{k=0}^{N-1} (D_{k-\epsilon} D_{k-\tilde{\epsilon}}) e^{j2\pi\delta_t/N} |H_{k-\epsilon}|^2 \right|, \tag{3.16}$$

in which the accumulated term is 0 (when either $D_{k-\epsilon}$ or $D_{k-\tilde{\epsilon}}$ is 0) or $\pm e^{j2\pi\delta_t/N}|H_{k-\epsilon}|^2$ (sign + when $D_{k-\epsilon} = D_{k-\tilde{\epsilon}}$, sign − otherwise). Besides, if $0 \leq \delta_t \leq N/4$, then the phase of the rotation factor $e^{j2\pi\delta_t/N}$ lies within $0 \leq 2\pi\delta_t/N \leq \pi/2$, which means that the non-zero accumulated terms, in the absence of noise, are either in the first quadrant of the complex plane (positive real and positive imaginary components) or in the third quadrant (negative real and negative imaginary components). Ideally, when the correlation function is analyzed for $\tilde{\epsilon} = \epsilon$, all the accumulated terms will be in the first quadrant. In this case, one possible simplification for the algorithm is removing the absolute operator $|.|$ from (3.13) and maximizing the sum of the real $\Re(.)$ and imaginary $\Im(.)$ components of the correlation result, as proposed in

$$R_{LTF}(\tilde{\epsilon}) = \sum_{k=0}^{N-1} A_k D_{k-\tilde{\epsilon}}$$
$$\hat{\epsilon} = \max_{\tilde{\epsilon}} [\Re(R_{LTF}(\tilde{\epsilon})) + \Im(R_{LTF}(\tilde{\epsilon}))]. \tag{3.17}$$

The operation of adding the two components of a complex number is simpler than taking its magnitude value.

Returning to the estimation/compensation flow, the LTF field of the PPDU is used in the receiver for estimating the ICFO. It may be done by using either a single LTF symbol or both symbols. By using both of them, the term A_k used in (3.13) or (3.17) is calculated as $A_k = (Y^*_{LTF1,k} Y_{LTF1,k-1} + Y^*_{LTF2,k} Y_{LTF2,k-1})/2$. The LTF field must also undergo ICFO compensation, since it will be used to estimate the channel. In order to make the compensation process easier, let us set the initial sample time $n = 0$ at the end of the LTF portion, so that the symbol indices for LTF1 and LTF2 are respectively $l = -2$ and $l = -1$, as presented in Figure 3.6.

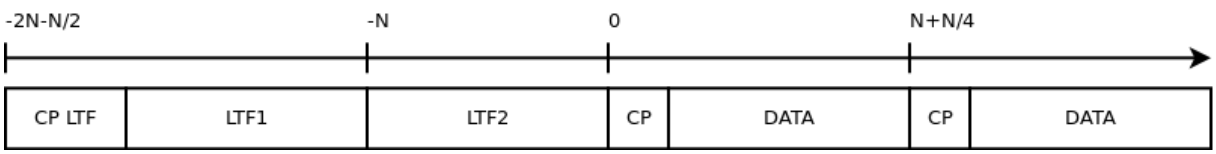

Figure 3.6 – OFDM symbol indices for ICFO compensator

Since a single CP is prepended to two consecutive LTF symbols, we can assume $N_g = N/2$ for LTF1 and $N_g = 0$ for LTF2. Note that, although no extra CP is pre-appended to LTF2, it will not suffer ISI since the previous symbol (LTF1) is a copy of the

LTF2 itself. This way, based on (3.8), the phase θ_l for LTF1 and LTF2 are respectively $\theta_{LTF1} = \theta_{-2} = 2\pi\epsilon(-2N - N/2 + N/2)/N = -4\pi\epsilon$ and $\theta_{LTF2} = 2\pi\epsilon(-N + 0)/N = -2\pi\epsilon$, which results in $e^{j\theta_{LTF1}} = e^{j\theta_{LTF2}} = 1$.

To compensate the ICFO of the header and data OFDM symbols, note that all of them have the same CP length $N_g = N/4$. Thus, from the first header symbol forward ($l = 0, 1, 2...$), we can use (3.15) to accumulate the phase to be compensated, as in

$$e^{j\theta_l} = e^{j(2\pi/r_g)\epsilon(l+1)} = e^{j(\pi/2)\epsilon(l+1)}. \tag{3.18}$$

In summary, the ICFO estimation and compensation are executed in the frequency domain as following:

1. estimate the integer CFO $\hat{\epsilon}$ based on the LTF symbols;
2. set the phase step $\theta_{step} = (\pi/2)\hat{\epsilon}$ and initialize the phase accumulator $\theta_0 = \theta_{step}$;
3. for the LTF OFDM symbols, only cyclically shift the FFT components by $\hat{\epsilon}$;
4. for each new header/data OFDM symbol, shift the FFT components by $\hat{\epsilon}$, derotate the tones by $-\theta_l$ and accumulate the phase $\theta_{l+1} = \theta_l + \theta_{step}$.

4 Implemented Architectures

In this chapter, we present the architectures designed to implement in hardware the synchronization algorithms proposed in Chapter 3.

4.1 Frame Synchronizer with FCFO estimation

The architecture proposed for the Frame Synchronizer is presented in Figure 4.1. The modules are organized into groups according to their functionality and the main signals are labeled and associated to the variables of Minn's algorithm. Note that the samples are indexed in the diagram in a way that the peak in the calculated metrics occurs at the end of the preamble, not at the start, as it was presented in Chapter 3, in order to make the hardware implementation easier.

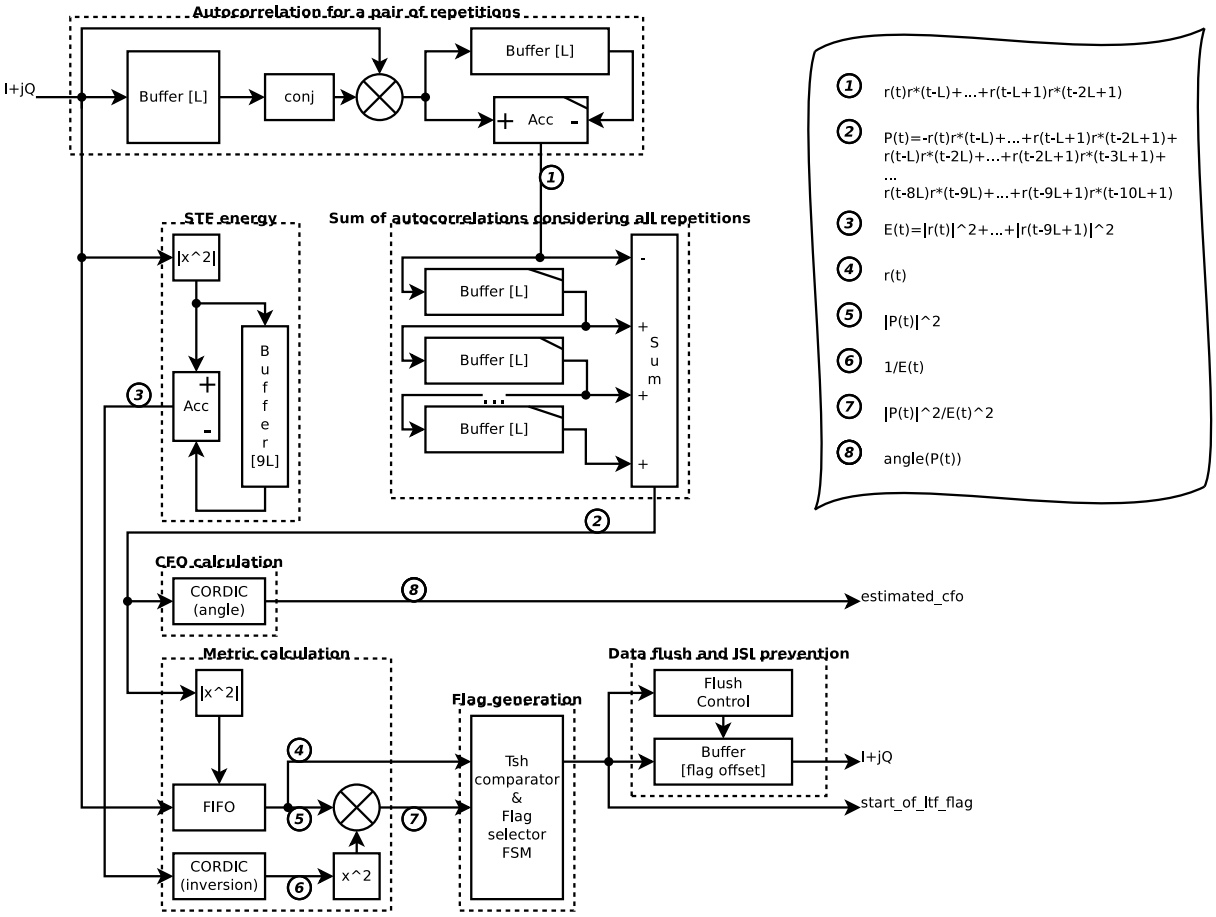

Figure 4.1 – Architecture of the Frame Synchronizer

The functional groups are:

1. **autocorrelation for a pair of repetitions**: at this first step, the autocorrelation for a given pair of repetitions is calculated, which corresponds to the term $\sum_{i=0}^{L-1} r^*(t+uL+i)r(t+(u+1)L+i)$ in Equation 3.2; but instead of taking into account the current and next repetitions indexed as $(u, u+1)$, the computation in hardware actually considers the current and previous repetitions, so that for $(u, u-1)|_{u=0}$, the sum is given by $\sum_{i=0}^{-L+1} r(t+uL+i)r^*(t+(u-1)L+i) = \sum_{i=0}^{-L+1} r(t+i)r^*(t-L+i)$;

2. **sum of autocorrelations**: at this point, the 9 autocorrelations considering the last 10 repetitions are summed up and their respective polarities are taken into account as well, which corresponds to the term $P(t)$ in Equation 3.2;

3. **STF energy**: the energy of the last 9 repetitions is calculated, corresponding to the term $E_{sm}(t)$ in Equation 3.2;

4. **metric calculation**: at this point, the STF energy $E_{sm}(t)$ is inverted by a Coordinate Rotation Digital Computer (CORDIC), whose latency is compensated by a First In First Out (FIFO) memory, and the simplified Minn metric is calculated; as the term $P(t)$ is complex, the numerator of the metric is calculated by squaring its real and imaginary components and then summing them up as $|P(t)|^2 = \Re\{P(t)\}^2 + \Im\{P(t)\}^2$;

5. **CFO calculation**: in this branch, a CORDIC calculates the phase of the autocorrelation accumulator $P(t)$ in order to output the phase to be used to compensate the FCFO;

6. **flag generation**: at this point, the flag indicating if the STF has been detected is generated, and if more than one sample satisfies the end of STF condition (with associated metric achieving the defined threshold), only one is chosen based on the highest metric;

7. **Data flush and ISI prevention**: finally, a short offset into the guard interval is applied to the detection flag, in order to avoid ISI in case the detection has occurred late; besides, this portion of the hardware architecture is also responsible for flushing the buffer when it still contains valid data and no other input data has been pushed into the buffer for a while.

4.2 FCFO compensator

As mentioned in Chapter 3, the estimated FCFO is given by $\hat{\lambda} = \frac{\angle P(\hat{t})}{2\pi L/N}$, so that the input to the FCFO compensator from the frame synchronizer is $\angle P(\hat{t}) = 2\pi\hat{\lambda}L/N$. The FCFO is compensated by multiplying the received samples by $e^{-j2\pi\hat{\lambda}n/N} = e^{-j\angle P(\hat{t})n/L}$,

therefore the received angle is first multiplied by the factor $-1/L$, the phase step $-2\pi\hat{\lambda}/N$ is registered and then accumulated for each new received data sample.

Figure 4.2 illustrates the architecture of the FCFO compensator. As presented in Chapter 3, the repetition length is $L = \{64, 32, 16, 8\}$, which is a power of 2, therefore the multiplication by $-1/L$ is simply implemented as a sign inversion followed by a shift-right operation. The phase is accumulated to lie in the range $[-\pi, \pi]$, and the rotation is executed by a CORDIC module.

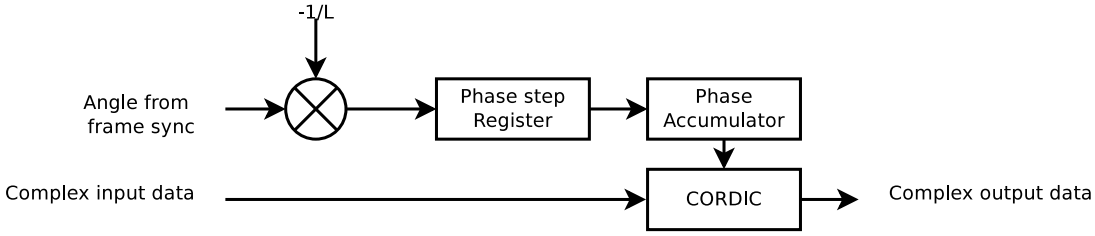

Figure 4.2 – Architecture of the FCFO compensator

4.3 ICFO estimator and compensator

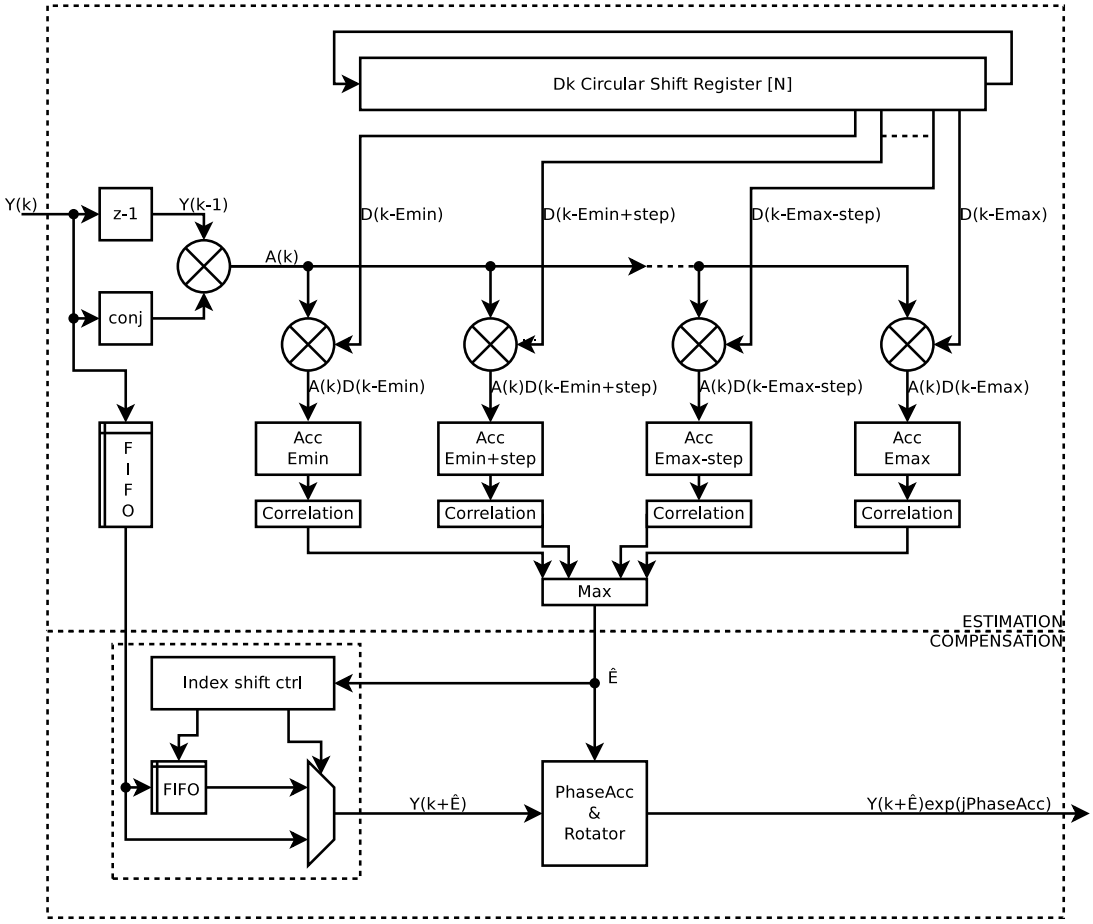

Figure 4.3 – Architecture of the ICFO Estimator/Compensator

Figure 4.3 illustrates the internal architecture of the ICFO Estimator/Compensator, which is composed of:

1. the `Dk Circular Shift Register`, responsible for providing the D_k terms of (3.17);
2. a set composed of a register (`z-1`), a conjugator (`conj`) and a complex multiplier, responsible for computing the term $A_k = Y^*_{LTF,k} Y_{LTF,k-1}$;
3. an array of sign multipliers, accumulators (`Acc`) and `Correlation` calculators (addition of complex components for simplified correlation), responsible for calculating the correlation $R(\tilde{\epsilon})$ (3.17) for each ICFO candidate $\tilde{\epsilon}$;
4. the `Max` block, responsible for finding the ICFO that maximizes $R(\tilde{\epsilon})$;
5. a `FIFO` memory, responsible for buffering the OFDM symbols due to the latency of the ICFO estimation process, since the LTF used for estimation must also have the ICFO compensated before being used by the equalizer;
6. a set composed of the `Index shift ctrl` block and a shorter `FIFO`, responsible for shifting the FFT components according to the estimated ICFO;
7. and finally the `PhaseAcc & Rotator` block, responsible for accumulating and compensating the phase in a symbol-by-symbol basis, following the operations presented in Tables 3.1 and 3.2.

The D_k terms are calculated offline and stored in the receiver for ICFO estimation. As the LTF reference is composed only by $X_{LTF,k} \in \{0, -1, 1\}$, only 2 bits are needed for each D_k element, and as the maximum OFDM symbol size is $N = 128$, a register bank can be used, so no RAM memory is needed. For each received LTF component $Y_{LTF,k}$, the term $A_k D_{k-\tilde{\epsilon}}$ is calculated in parallel for all the ICFO candidates $\tilde{\epsilon}$, and the shift register that stores the D_k terms executes a circular shifting, so that the array of correlators are always connected to the same positions of the shift registers.

The set of ICFO candidates varies according to the range of FCFO estimator and to the characteristics of the oscillators. For example, if the FCFO estimation range is ± 1, then the residual ICFO is a multiple of 2, and if the oscillator instability generates a CFO error up to ± 8 subcarrier spacings, then the set of possible ICFO is given by $\tilde{\epsilon} \in \{\pm 2, \pm 4, \pm 6, \pm 8\}$. For this example, in Figure 4.3, `Emin`$= -8$, `step`$= 2$ and `Emax`$= 8$.

4.3.1 Complexity comparison

As the ICFO corrector is based on a novel scheme and this specific error was fully addressed in [21] with a detailed proposal of hardware implementation, we will give a special attention to this module in order to prove its low complexity.

The proposed architecture presented in Figure 4.3 is area efficient in both estimation and compensation sides. Regarding the estimation process, the simplification in the correlation function (3.17) replaces complex multipliers (magnitude calculation in (3.13)) for simple adders. As for the compensation process, the frequency-domain compensation saves significant memory since there is no need to buffer the samples at the input of the time-domain CFO compensator.

In a conventional receiver architecture, although the FCFO and ICFO are estimated separately, the total CFO is compensated in the time domain via phase rotation [6, 9], i.e. a feedback chain shall deliver the estimated ICFO to the CFO compensator in order to combine it with the estimated FCFO, and for that, the samples coming after the preamble used for ICFO estimation must be buffered in order to be properly corrected considering both CFO estimations. The buffer size should equal the accumulated latency of the CFO compensator, FFT and ICFO estimator. Figure 4.4 presents the difference between a conventional receiver, based on a time-domain compensation of the combined CFO=FCFO+ICFO, and the architecture of a receiver that compensates the FCFO and ICFO separately, in time and frequency domains respectively.

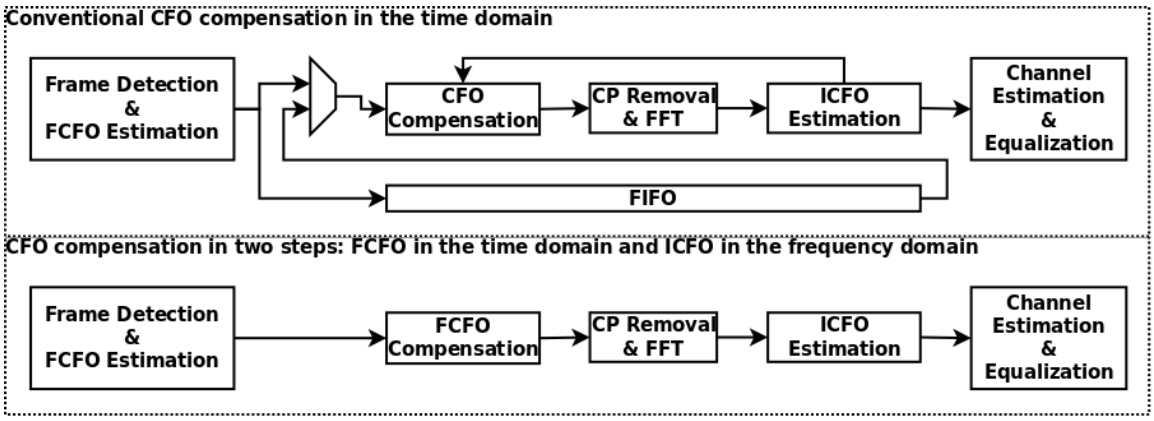

Figure 4.4 – Schemes for CFO compensation

In [4], two CFO recovery methods are presented: the first one following the conventional CFO compensation in the time domain, and the second one using a frequency-domain interpolator to compensate the CFO, different from what we propose here. In [21], the authors suggest that the ICFO compensation can be executed in the frequency-domain by cyclically shifting the indices of the OFDM symbols after the FFT, but they detail only the hardware architecture for the ICFO estimation process and do not mention the combined effect of cyclic-shift and phase accumulation, as we do in this work.

Now, returning to the area-efficiency aspect of our method, it is important to do a few computations. In the implemented receiver the proposed modules are part of, the time-domain CFO compensation is executed by a CORDIC with a latency of approximately 30 clock cycles, the FFT latency is $\log_2 N \times N/2$, and the latency of the proposed

ICFO estimator is $N + 2$, where N is the FFT size. If we followed the conventional CFO compensation approach in the time domain, we would need a buffer as large as that latency. Our frequency-domain ICFO compensator, in turn, uses only one FIFO with N positions and a simple sign operation to rotate the accumulated phase presented in (3.8).

For a matter of complexity comparison, we resume what is proposed in [21]. The work [21] optimizes the original estimation algorithm presented in [19], which is also based on the correlation (3.13), by reducing the pilots used for estimation and proposing an architecture that shares multiply-accumulate (MA) blocks. However, this area saving is irrelevant for the case of IEEE 802.15.4g, since the MAs in [21] are used for the operation $A_k D_{k-\tilde{\epsilon}}$, which in our case is simply a sign manipulation. Besides, in [21], the correlation of each candidate ICFO follows the original algorithm (3.13) [19] with a magnitude calculation, different from our proposal of simply adding the complex components.

In terms of Fixed-point (FP) and Sign-bit (SB) Complex Multiplications (Complex Mult.), FP Complex Accumulators (Acc.), FP Real Sum, and FP Memory Locations, Table 4.1 compares the complexity of the following algorithms/architectures:

- a conventional receiver that estimates the ICFO based on the original correlation (3.13) and on a complete CFO compensation in the time domain;
- a receiver based on the MA usage optimization proposed in [21];
- the ICFO compensator proposed in this work, with the simplified version of the ICFO estimation (3.17) and the ICFO compensation in the frequency domain.

	Conventional	[21]	Proposed
FP Complex Mult.	$1 + \lvert S_{\tilde{\epsilon}} \rvert$	$1 + \lvert S_{\tilde{\epsilon}} \rvert$	1
SB Complex Mult.	$\lvert S_{\tilde{\epsilon}} \rvert$	$\lvert S_{\tilde{\epsilon}} \rvert / 4$	$\lvert S_{\tilde{\epsilon}} \rvert + 1$
FP Complex Acc.	$\lvert S_{\tilde{\epsilon}} \rvert$	$\lvert S_{\tilde{\epsilon}} \rvert / 4$	$\lvert S_{\tilde{\epsilon}} \rvert$
FP Real Sum	0	0	$\lvert S_{\tilde{\epsilon}} \rvert$
FP Memory locations	L_{buffer}	N	N

Table 4.1 – Complexity comparison of ICFO compensation schemes

In Table 4.1, $|S_{\tilde{\epsilon}}|$ represents the number of candidate ICFOs, and L_{buffer} is the combined latency of the CFO compensator, FFT and ICFO estimator. Note that although the MA-optimization in [21] saves complex accumulators, our method uses only one complex multiplier, while [21] uses $|S_{\tilde{\epsilon}}|$ of them. Compared to the conventional architecture, besides saving complex multipliers, our method significantly reduces the memory locations needed to buffer the received samples for proper compensation.

5 Evaluation Methodology

The evaluation of proposed methods was executed in two ways:

1. via high-level simulations, by modeling in Octave [24] the selected algorithms and the data transmission/reception;
2. and via test in hardware, after the synthesis in Field Programmable Gate Array (FPGA).

5.1 High-level Simulations

For high-level simulations, we set up three environments:

1. one that includes the generation of a complete frame, in order to analyze the performance of the frame synchronizer in terms of failure rate and Mean Absolute Error (MAE) of FCFO estimation;
2. one that considers only the generation of the LTF field to analyze the performance of the ICFO estimator in terms of failure rate;
3. and the last one to analyze the false alarm probability of the frame synchronizer.

The following subsections detail each of those simulation environments and present their respective parameters and performance metrics. At the end of this section, the channel models used in all environments will be presented as well.

5.1.1 Simulation environment for Frame Synchronizer

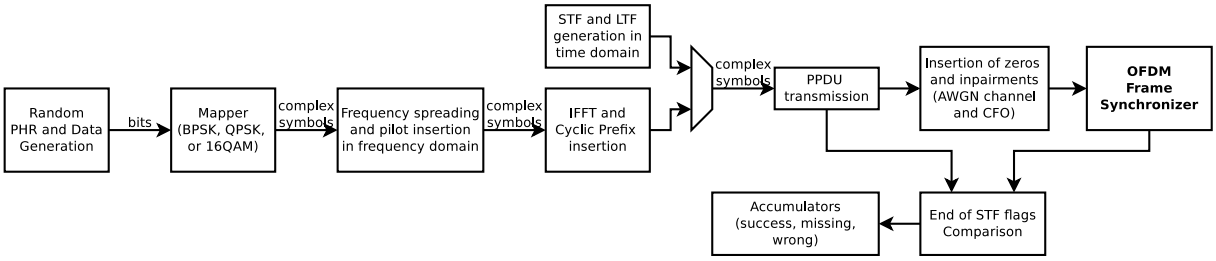

Figure 5.1 – High-level simulation environment for frame synchronizer

The environment built for simulating the frame synchronization process in Octave is presented in Figure 5.1. The objective of this simulation is verifying the efficiency of the OFDM frame synchronizer in detecting the STF within a sequence of symbols that

may present different power distribution according to the selected OFDM option and MCS. There is no need for filling in the PHR field with the correct configuration bits, neither for executing channel encoding. What is really important is guaranteeing that the symbols that follow the STF during the simulation represent the physical characteristics of a realistic signal. Therefore, the bits corresponding to PHR and payload are randomly and uniformly generated, just for composing the frame structure, while the steps related to symbol processing are properly executed by the models of the respective blocks, which follow the specifications presented in [1].

The PHR and payload bits are mapped by using the modulation specified by the MCS code (BPSK, QPSK or 16QAM). The resulting symbols pass through frequency spreading and pilot insertion, and then they are converted from frequency to time domain via IDFT. After CP insertion, the PHR and payload time-domain symbols are combined with the preamble composed by the STF and LTF sequences to build each PPDU. During the simulation, several PPDUs are generated and concatenated, and at the beginning of the concatenated PPDUs, a trash composed of zeros is inserted only to evaluate the synchronizer performance at a noisy environment that carries no modulated signal. Finally, the impairments (channel and CFO) are added to the transmitted PPDUs, and the received signal (trash and valid PPDUs) is passed to the Frame Synchronizer. The index of the time domain sequence corresponding to the end of each STF is well known. Therefore, based on the SOF flags signalized by the Frame Synchronizer, the number of successful detections, missing STFs and wrong detections are computed, as well as the error of the corresponding estimated FCFOs.

The parameters for the Frame Synchronizer simulation are:

1. SNR;
2. normalized CFO;
3. MCS;
4. STF guard offset from the point detected by the algorithm into the CP region;
5. algorithm threshold to consider a sample as a SOF candidate.

The metrics used to analyze the performance of the Frame Synchronizer are:

1. failure rate (missing STFs);
2. wrong detection rate (too early or late STF detections);
3. MAE of FCFO estimation.

We now clarify how some of the listed parameters and metrics are set and computed during the simulations. During AWGN noise generation, the computation of the SNR, defined as the relation between signal and noise power, takes into account the energy of the payload field in the time domain $P_{payload}$. In other words, during the simulation, the required noise power is calculated as $P_{noise} = P_{payload}/SNR$ and the complex samples randomly generated with unitary power are scaled by a factor of $\sqrt{P_{noise}}$. The normalized CFO, defined as the relation between the real CFO (in Hz) and the frequency spacing (also in Hz), is varied in the range $-1 \leq \lambda \leq 1$, which can be resolved by the time-domain FCFO estimator. The STF guard is the one responsible to anticipate the detected border in order to avoid ISI due to late detection. However, making this guard so large would reduce the margin for the channel length and could cause ISI anyway. Therefore, we will only consider STF guard up to half CP length, i.e. from 0 to $N/8$. As for the MAE of FCFO estimation, it is computed as follows: for each successfully detected frame, the FCFO $\hat{\lambda}$ is estimated based on the received STF, the absolute estimation error is calculated as $|\hat{\lambda}-\lambda|$, where λ is the known normalized CFO parameter, and the absolute error is accumulated and divided by the number of FCFO estimations.

5.1.2 Simulation environment for ICFO estimator

The environment built for the ICFO estimator simulations is presented in Figure 5.2. The process is simple: only the LTF field is generated in the time domain, since it is assumed the frame beginning has been properly detected, and then the impairments (channel, noise and CFO) are inserted; the CP is removed considering a random TOA error, as long as this offset is toward CP; finally, the ICFO is estimated and compared to the actual value.

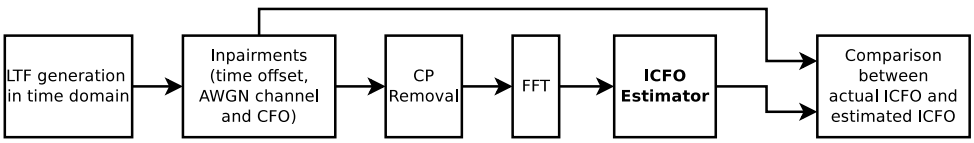

Figure 5.2 – High-level simulation environment for ICFO estimator

The parameters for the ICFO simulations are:

1. SNR;
2. ICFO $-8 \leq \epsilon \leq 7$ (8 was not included, since it is equivalent to -8 for OFDM option 4 with $N = 16$);
3. quantization of the internal data bus.

   As this environment is much simpler than the one built for the Frame Synchronizer simulations, it was possible to quantize the model in order to optimize hardware implementation. The quantization parameter is then represented as $\#bits/fractional\_width$,

e.g., when the internal bus is 8-bits wide with 5 bits for the fractional portion, the quantization scheme is defined as 8/5.

The performance metric for the ICFO estimator evaluation is the failure rate computed as following: for each combination of OFDM option, SNR and quantization scheme, all the possible ICFOs are generated, and each occurrence of wrong ICFO estimation $\hat{\epsilon} \neq \epsilon$ increments the error accumulator by 1; this way, the mean of wrong estimations considering all the ICFO values is calculated as the number of wrong estimates divided by the total number of runs executed for a given scenario of OFDM option, SNR and quantization scheme.

5.1.3 Simulation environment for analyzing False Alarm Probability

The last environment is simple, as presented in Figure 5.3: for each OFDM option, it generates several vectors of AWGN noise with size equal to the STF length ($5N$) and computes the Minn metric given by (3.2) for each vector; then the metrics are stored for analysis. Note that the focus here is computing false alarm rate, i.e. the probability of the algorithm signaling a frame when no frame exists. That is why only noise is generated and passed to the frame synchronizer submodule responsible for calculating the metrics, with no need to simulate all the frame synchronizer behavior. This way, once the metrics are stored, we can draw their histogram and check for which thresholds the algorithm would detect a frame when only noise exists.

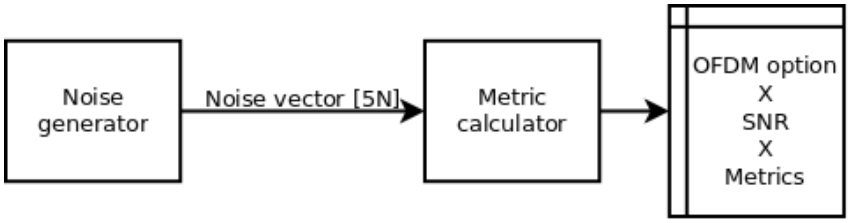

Figure 5.3 – High-level simulation environment for analyzing false alarm probability

The parameter for this simulation is just the SNR and the analysis is done by observing the histogram of the calculated Minn metrics.

5.1.4 Channel models

During the simulations, the following channel models were considered:

1. AWGN;
2. Rayleigh fading channel.

In the AWGN channel model, complex noise samples are added to the transmitted signal. A complex noise sample is generated as

$$w = \frac{\sqrt{P_{noise}}}{\sqrt{2}}(w_I + jw_Q),$$

where w_I and w_Q are independent Gaussian random variables with zero mean and unit variance, and P_{noise} is the required noise power. The generation of w_I and w_Q can be done by the Octave function `randn`.

In order to understand how the Rayleigh fading channel is modeled, it is necessary to review a few concepts. All the contents presented in this subsection, from this point forward, are based on [25, 26, 27].

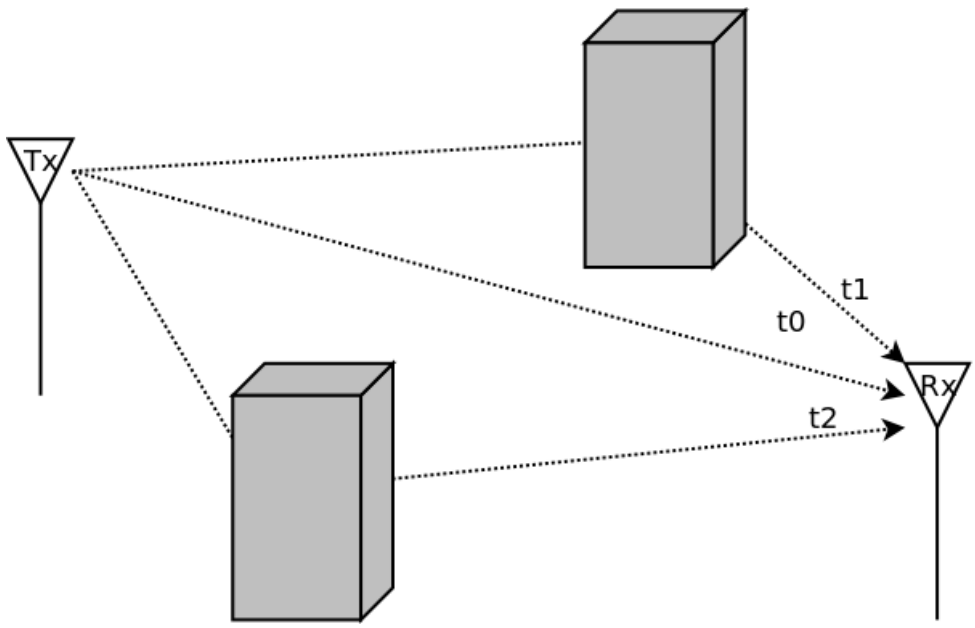

Figure 5.4 – Multipath channel

On the way toward the receiver, the transmitted signal can be reflected by several objects, which results on multiple reflections of the same signal arriving with different power levels at different time instants, as presented in Figure 5.4. This kind of channel is called "multipath channel" and is characterized by its Power Delay Profile (PDP), which gives a list of different time delays and their respective average power levels. In Figure 5.5, the PDP associated to the transmission illustrated in Figure 5.4 is defined by the vectors (t_0, t_1, t_2) and (p_0, p_1, p_2), where p_m is the average power of the m-th path, and the two graphics represents respectively the signal power in the transmitter (Tx) and Receiver (Rx). However, it is unfeasible to describe the attenuation and the exact time delay for each reflected path of a real environment in different time instants, therefore, those variables are generally modeled as random processes. In order to provide a guidance to system design, different environments are characterized by different PDPs obtained from empirical data.

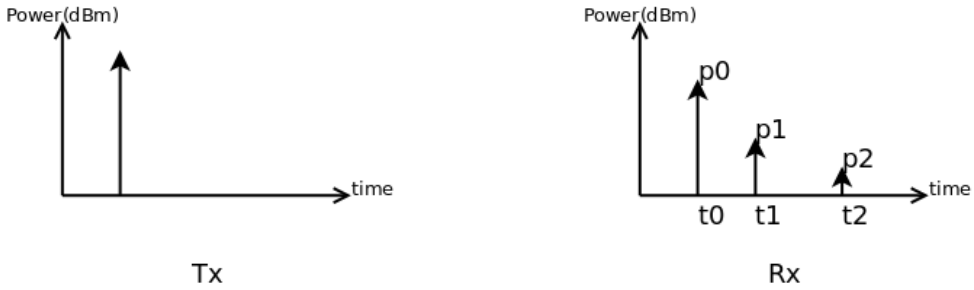

Figure 5.5 – Power Delay Profile

One important characteristic of the PDP is the Maximum Delay Spread, which is the difference between the arrival times of the first and the last signal replications, whose power levels are above a given threshold. Combined with the symbol time period, the delay spread can be used to classify a channel into frequency selective or non-selective (flat) category. The channel is frequency selective when the delay spread is greater than the symbol period, which introduces ISI, since the signal components extend beyond the symbol time. On the other hand, if the delay spread is less than the symbol period, the channel is considered flat, since all the scattered signal components arrive at the receiver within the symbol time, not interfering in the next symbol period.

Given those basic concepts, let us first model the multipath channel in continuous time as

$$h(t,\tau) = \sum_{m=0}^{M-1} a_m(t)\delta(t - \tau_m(t)), \tag{5.1}$$

where $a_m(t)$ and $\tau_m(t)$ are respectively the time-variant amplitude gain and delay of the m-th path. Note that (5.1) models a time-variant complex channel, in which the gain and the delay vary with time, due to the movement of objects in the surrounding or to any other change in the transmission medium. The time variance aspect of the channel is generally modeled by representing the path gains as complex Gaussian random variables, in a way that, at any given time t, the absolute value (or gain) of the impulse response $|h(t,\tau)|$ follows a Rayleigh or Rician distribution. Respectively, these two scenarios model the presence or absence of a Line of Sight (LOS) path between the transmitter and the receiver. If both real and imaginary components of the random complex gain have zero mean, the envelope of the complex variable follows a Rayleigh distribution, otherwise, if they have non-zero mean(s), the envelope follows a Rician distribution. In this work, we will use only Rayleigh fading channel models.

For simulation purpose, such continuous-time models are usually converted to discrete-time Tapped Delay Line (TDL) filters, which are basically Finite Impulse Response (FIR) filters, with a finite number of taps L with variable gains. In this case, the number of taps L is strictly related to the spread delay τ_{max} of the continuous-time PDP

and the symbol time period T_s, as in

$$L = \lceil \tau_{max}/T_s \rceil,$$

where $\lceil . \rceil$ is the ceiling operator. We will not study here how the characteristics of a continuous-time PDP are converted to coefficients of a TDL filter. We will instead assume that a discrete-time PDP, with the list of delays in samples $(d_0, d_1, ...d_{L-1})$ and their respective average power levels $(p_0, p_1, ...p_{L-1})$, is given, and then model the corresponding discrete-time channel as a TDL filter, as presented in Figure 5.6, where the filter taps a_l are complex random variables with average power p_l.

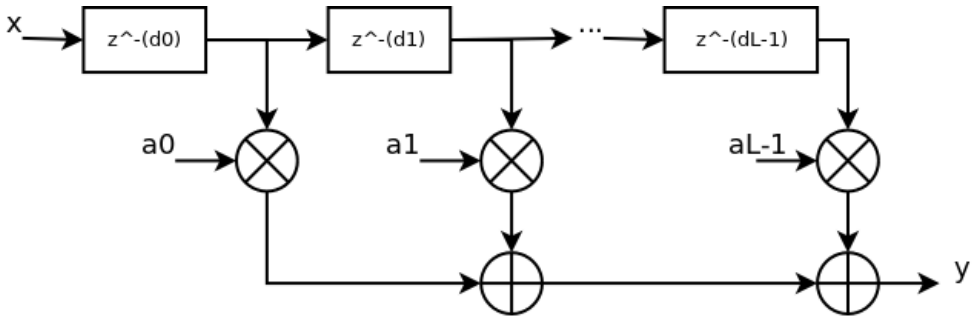

Figure 5.6 – Tapped Delay Line

Note that the number of taps L of a discrete TDL filter does not need to equal the number of paths M, present in (5.1), of its corresponding continuous-time PDP, since more than one path can lie within the same symbol time period, as well as there may be a time slot between the first and the last path, larger than the symbol period, that has no path arriving.

Given that, we can finally model the discrete-time multipath channel as

$$\begin{aligned} h_n &= \textstyle\sum_{l=0}^{L-1} a_l \delta(n - d_l) \\ a_l &= \tilde{a}_l \times g_l = \tfrac{\sqrt{p_l}}{\sqrt{2}} \times g_l, \end{aligned} \tag{5.2}$$

where $g_l = g_{l_I} + j g_{l_Q}$ is a complex random variable whose real and imaginary components are independent Gaussian random variables with unity variance and zero mean for Rayleigh fading model. Note the difference between the notations a_l, p_l and $\tilde{a}_l$: a_l represents a realization of the random l-th tap captured at a given sampling time, p_l is the average power of the l-th tap, and $\tilde{a}_l = \frac{\sqrt{p_l}}{\sqrt{2}}$ is a fixed scaling factor that multiplies all the variables randomly generated for the l-th tap.

In this work, the received signal is modeled as

$$y_n = x_n * h_n + w_n, \tag{5.3}$$

where $*$ represents the linear convolution operation, h_n corresponds to the taps of a Rayleigh fading channel and w_n corresponds to the AWGN noise. To simulate only AWGN channel, we simply make $h_n = \delta(n)$, which means that the channel does not change with time neither distorts the transmitted signal. To simulate the Rayleigh fading channel, it is necessary to set a discrete PDP. This PDP can be extracted from an empirical continuous-time PDP or defined in an arbitrary way. In this work we set arbitrary discrete-time PDPs whose average power gains follow an exponential function based on the following parameters:

1. L: number of channel taps;
2. r: ratio between average power of first and last taps in dB.

For example, if $L = 5$ and $r = 20dB$, the power gains in logarithmic scale are linearly spaced as $p_{dB} = (0, -5, -10, -15, -20)$, and then the linear power gains p_l are calculated as

$$p_l = 10^{p_{l_{dB}}/10}.$$

After that computation, the tap gains are normalized so that the impulse channel response has unity energy.

The channel realizations can be expressed as a matrix $\mathbf{H}$ of size $V \times L$, where V is the number of channel realizations, which can be interpreted as how many times the channel variations were captured, and the elements $h_{v,l}$ of $\mathbf{H}$ are complex Gaussian random variables $g_{v,l}$ (both real and imaginary components with zero mean and unity variance) multiplied by their respective gain $\tilde{a}_l = \sqrt{p_l/2}$. We can also see it as a matrix multiplication

$$\mathbf{H}_{(V\times L)} = \mathbf{G}_{(V\times L)} \times \mathbf{A}_{(L\times L)},$$

where $\mathbf{G}_{(V\times L)}$ is composed by the complex random variables $g_{v,l}$ and $\mathbf{A}_{(L\times L)}$ is a diagonal matrix scaled by the gains $\tilde{a}_l$.

For simulation purposes, one can generate the matrix $\mathbf{G}$ with the Octave/Matlab function `randn` and, based on the vector $\tilde{a} = (\tilde{a}_0, ..., \tilde{a}_{L-1})$, generate the channel matrix $\mathbf{H}$. If we compute the power of each element of the randomly generated $\mathbf{H}$, as in $|h_{v,l}|^2$, and calculate their mean in a column-by-column basis, the result shall approximate to $(p_0, p_1, ...p_{L-1})$ if we have a reasonable number of channel realizations V.

To conclude, it is important to define how the different channel realizations are considered in the simulations, since the convolution presented in (5.3) will have its channel coefficients varying. For a matter of simplicity, as several frames are generated for both frame synchronizer and ICFO estimator, one channel realization is executed for each frame, and at the end of the simulation, the average power of the randomly generated channel taps is computed in order to check it the channel realizations follow the required

discrete-time PDP. The purpose of computing that average power is only to check if a sufficient amount of channel realizations were considered during simulations, i.e. to make sure that the simulations were not based on either too optimist (perfectly decaying channel taps) or too pessimist (first taps weaker than last ones) scenarios.

5.2 Tests in Hardware

As the IEEE 802.15.4g project progressed, the complete OFDM transmitter and receiver were implemented in Register Transfer Level (RTL) as well, i.e., using the Hardware Description Language (HDL) VHDL. This way, we were able to analyze the frame synchronizer performance in hardware. A platform for measuring the performance of the complete receiver was built, but in this section, only the elements relevant to the frame synchronizer analysis are presented in Figure 5.7.

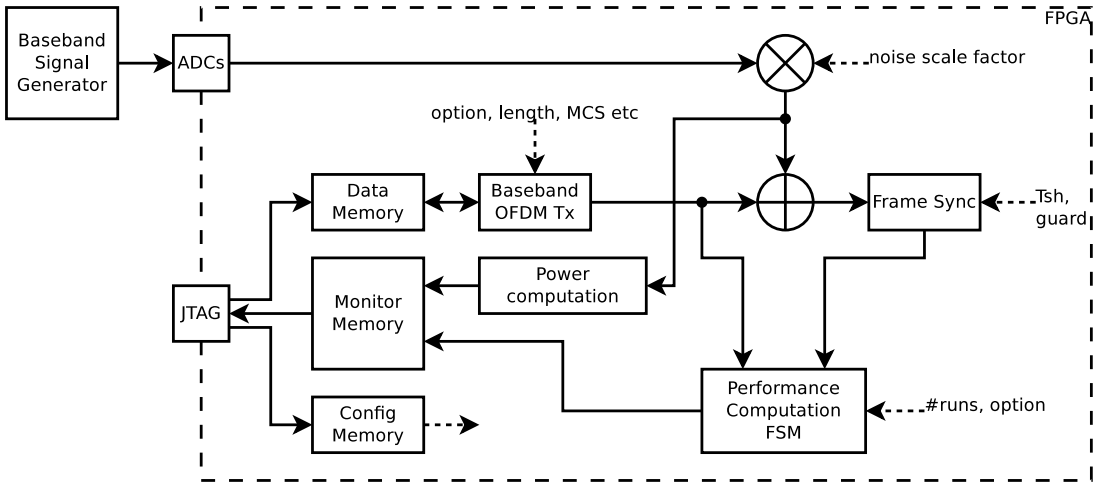

Figure 5.7 – Environment for test in hardware

The setup is composed of: the Altera Stratix IV GX FPGA Development Kit [28], the Terasic AD/DA Data Conversion Card [29] and the Agilent E4433B signal generator. For generating the data, the complete transmitter (`Baseband OFDM Tx`) is instantiated in FPGA, together with the frame synchronizer (`Frame Sync`) and the other auxiliary modules. As for the controlled noise, an external `Baseband Signal Generator` is used to generate baseband samples, which are sampled by two Analog to Digital Converters (`ADCs`) , one for each complex component (in-phase and quadrature). The sampled data is then scaled by a factor defined according to the required SNR and added to the signal outputted by the baseband OFDM transmitter. In order to calibrate the noise generation, the `noise scale factor` is first set to 1, the energy of the samples P_{adc} is calculated internally and monitored, and then the factor k can be redefined to generate the required noise power $P_{noise} = k \times P_{adc}$. The implemented frame synchronizer receives the noisy signal and signalizes when a new frame is detected. Several frames can be generated since the transmitter uses a ready-valid protocol to continuously ask data from the `Data`

`Memory`, and this can be done as long as the test setup enables the transmitter operation. The `Config Memory` is responsible for passing the test configuration for all the modules, as the threshold `Tsh` and the `guard` offset to the frame synchronizer, the number of frames (`#runs`) to be considered for failure rate calculation, and the PHR data (OFDM `option`, `MCS`, frame `length` etc) that must be used in the transmitter. The Finite-State Machine (`FSM`) responsible for computing the performance monitors the flag outputted by the frame synchronizer and the data outputted by the transmitter, so it knows whether the frame is properly detected or not. The FSM computations are stored in the `Monitor Memory`. The three memories are read and written using the Joint Test Action Group (`JTAG`) standard.

6 Results

In this chapter, we present the results of the high-level simulations and tests in hardware run in the environments presented in previous chapter.

6.1 High-level simulations for frame synchronizer

As presented in Subsection 5.1.1, the variable parameters for the Frame Synchronizer simulations are the SNR, normalized CFO, MCS, STF guard and Minn metric threshold, and the environment for high-level simulations was illustrated in Figure 5.1.

First, we ran several AWGN simulations with the generation of a few PPDUs for all OFDM options, in order to check which of the above parameters influence the algorithm performance. It was verified that the CFO and the MCS are irrelevant for the frame synchronizer behavior. Thus we fixed these variables to arbitrary values of $normalized\ CFO = 0.0$ and $MCS = 3$ and ran more exhaustive AWGN simulations, with more PPDUs, in order to draw the frame synchronizer performance.

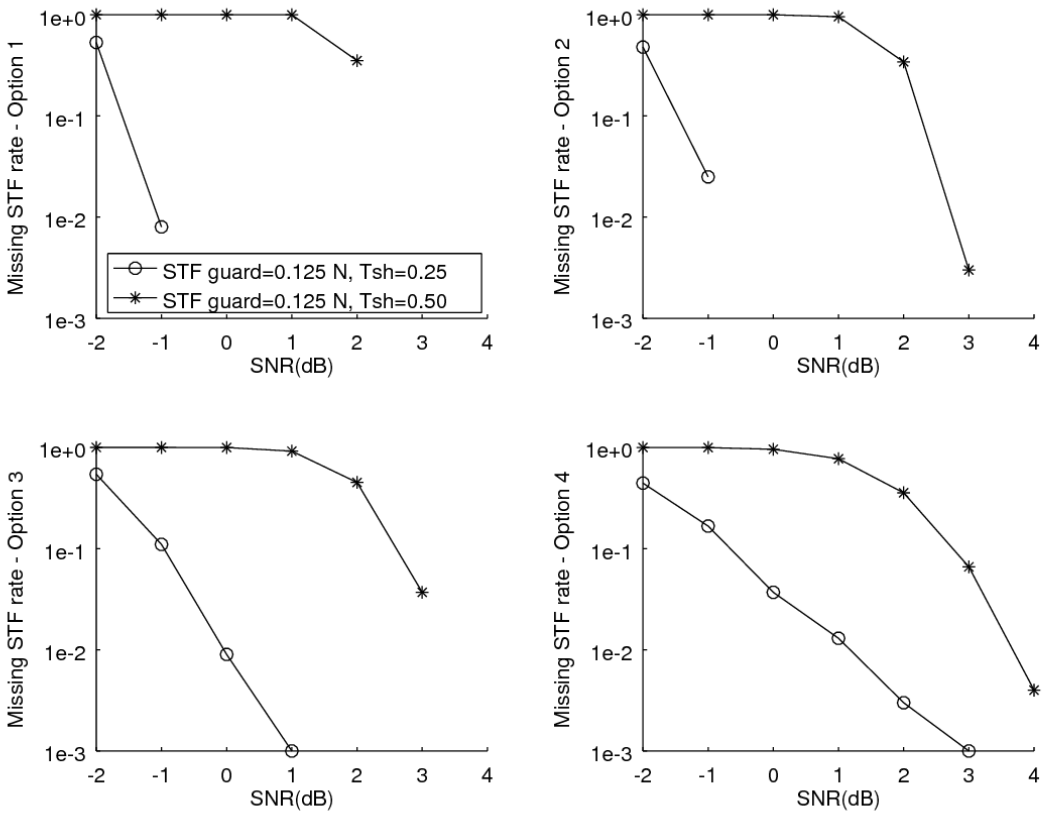

Figure 6.1 – Failure rate computed during AWGN simulations in Octave

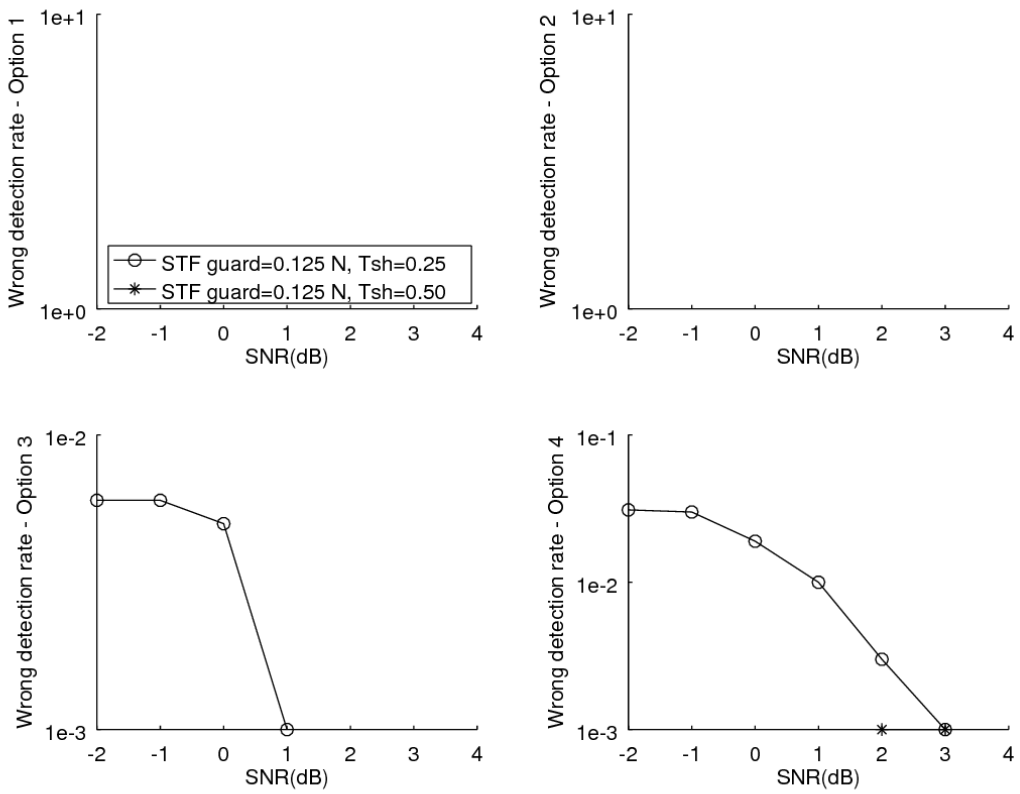

Figure 6.2 – Wrong detection rate computed during AWGN simulations in Octave

Figures 6.1 and 6.2 present, respectively, the failure rate (STFs that were not detected, i.e. missing STFs) and the wrong detection rate computed during the simulations that considered the transmission of 1000 frames over AWGN channel, $CFO = 0.0$, $MCS = 3$, $T_{sh} = \{0.25, 0.50\}$, and $STF\ guard = N/8$. Note that wrong detections mean late or too early detections, and they also account for the computation of the failure rate, i.e. the term "Missing STF" includes the STFs that were not detected at all and those detected outside the CP region. We note that the threshold 0.25 works better for all options, and that for options 3 and 4, it is more common to signalize SOF out of the tolerable margin of TOA error, which is justified by the shorter CP lengths ($N_g = 8$ for option 3, and $N_g = 4$ for option 4).

In addition to the failure rate and wrong detection rate, another important metric to analyze the frame synchronizer performance is the false alarm probability, which is the probability of detecting a frame when only noise exists. Before getting into the false alarm rate, it is important to review a few concepts on the probability theory, which can be found in [30]. Given a random variable X defined on a given sample space, the probability of X being less or equal a given value x is defined as the Cumulative Distribution Function (CDF), as in

$$F_X(x) = Pr[X \leq x].$$

For a continuous real-valued random variable, the Probability Density Function (PDF) is defined as the derivative of the CDF, as in

$$f_X(x) = \frac{d}{dx} F_X(x),$$

and can be also used to compute the probability of X being within a given interval I, as in

$$Pr[X \in I] = \int_I f_X(x)dx.$$

For a discrete random variable X in the space Ω_x, the PDF is defined as

$$f_X(x) = \sum_{y \in \Omega_x} p_X(y)\delta(x - y),$$

where $p_X(y) = Pr[X = y]$ and $\delta(x)$ is a Dirac delta function (single impulse in $x = 0$). In other words, for discrete random variables, the PDF simply describes the probability of such variable assuming a given value x.

Given those basic concepts, the false alarm probability can be computed based on the probability distribution of the Minn metric as following. For different values of SNR and preamble length, the Minn metric is computed over several sequences of noise and its histograms are used to estimate the PDF of the Minn metric for the different scenarios of SNR and OFDM option. For a given value $0 \leq m \leq 1$, $F_M(m)$ is the CDF of the Minn metric M and can be interpreted as how likely the calculated metrics lie in the range from 0 to m. In the same way, $1 - F_M(m)$ means how likely those metrics overpass the value m. Therefore, based on the CDF of the Minn metric $F_M(m)$ derived from its estimated PDF (empirical histograms), for a given threshold T_{sh}, the false alarm probability can be computed as

$$Pr_{fa} = 1 - F_M(T_{sh}).$$

Thus, given the close relation between false alarm probability and Minn metric probability distribution, we will focus, in this section, on analyzing the histograms generated during the simulations, instead of fixing the threshold and executing the detection algorithm to calculate the false alarm rate.

Based on the environment presented in Figure 5.3 of Chapter 5, 10000 noise vectors were generated for each OFDM option, considering $SNR = \{0, 5, 10\}$ dB (SNR, in this case, means the inverted noise power, which would be equivalent to consider unity signal energy). For each noise vector with the same length as the STF field, a metric was computed, and the histograms of the metrics calculated for each combination of OFDM option and SNR are presented in Figure 6.3. Note that the metric distribution is independent of the SNR; its variance, i.e. the spreading of the histogram, depends only on the OFDM option: the greater the OFDM symbol size (and consequently the STF length), the smaller the metric.

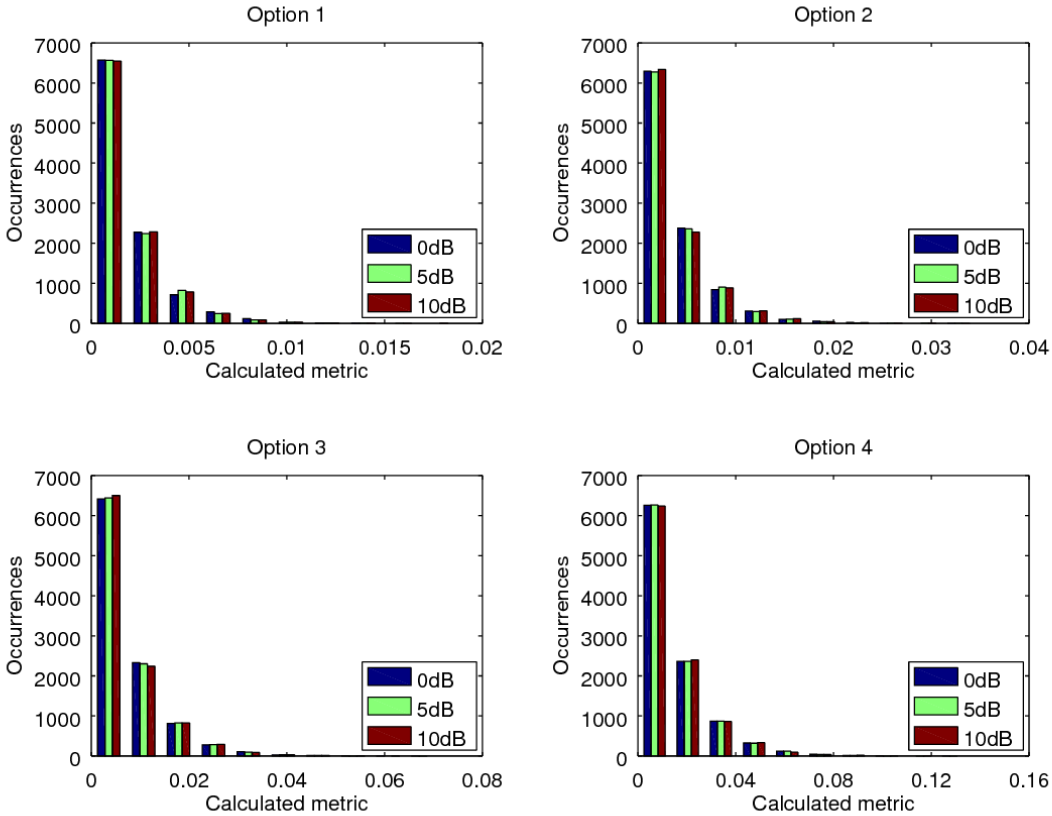

Figure 6.3 – Histograms of metrics calculated on noise vectors

In general, decreasing the false alarm probability by enlarging the detection threshold results also in decreasing the packet detection probability. This way, in the detection theory, there must be a compromise between the two conflicting goals [31]. In order to analyze the tradeoff between avoiding false alarm and increasing the probability of STF acquisition, we used the same environment illustrated in Figure 5.3 to generate several noisy STFs for each configuration of OFDM and SNR (only the preamble, not the complete frame) and compute the Minn metrics for them. In this case, we simply repeated the procedure presented before, but instead of generating a sequence composed only by noise, we generated the preambles and added noise to them. This way, we could draw the PDF of the Minn metrics for each scenario, considering both the presence and absence of valid STF, and check how far the PDFs are from each other in order to set the proper metric threshold for Frame Sychronizer algorithm. Figure 6.4 presents the results for options 1, 2, 3 and 4. Note that, different from the solid blue curves related to the metrics calculated when only noise is present, the dashed black curves associated to the metrics calculated over noisy STFs vary with the SNR, which is expected, since the autocorrelation of the STF repetitions must increase as the noise decreases.

Therefore, given that the PDF of the metrics in the absence of STF spreads up to around 0.2 for option 4, we can conclude that setting the metric threshold at $T_{sh} >$

0.2 would practically zero the false alarm rate. As previously observed during AWGN simulations, with $T_{sh} = 0.25$, the frame synchronization algorithm presents good results for all options, and in addition to that, this threshold is far enough from 0.2 to avoid false alarm. So, we can conclude that $T_{sh} = 0.25$ is a good choice for the Frame Synchronizer.

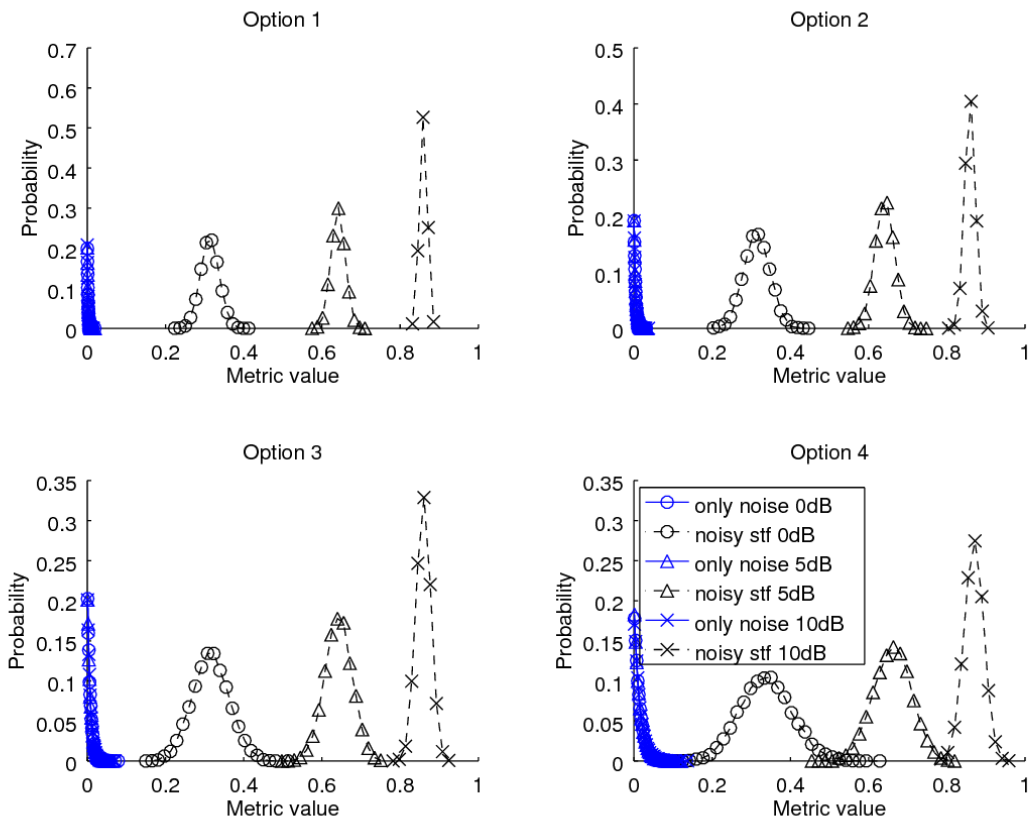

Figure 6.4 – PDFs of metrics calculated over pure noise and noisy STFs

Having defined the best metric threshold $T_{sh} = 0.25$, we ran new simulations in the environment of Figure 5.1 considering Rayleigh channel with arbitrary exponential PDP and varying the STF guard parameter in $STFguard = \{N/8, N/16\}$, in order to analyze the influence of channel spread. As presented in subsection 5.1.4, the arbitrary PDP is given based on the number of channel taps L and on the ratio r between average power of first and last taps in dB. We set $L = N_g/2 = N/8$, $r = 20dB$ and considered the generation of 1000 PPDUs. The average magnitude of the channel impulse response calculated over the 1000 channel realizations is presented in Figure 6.5.

Figures 6.6 and 6.7 present respectively the failure rate and wrong detection rate for all options. Comparing the curves associated to the parameters $STFguard = N/8$ and $T_{sh} = 0.25$ (cicle markers) on Figures 6.1, for AWGN channel, and 6.6, for Rayleigh channel (they will be combined in the same graphic afterwards), it is clear that the channel spread significantly degrades the frame synchronizer performance. Besides, different from the AWGN simulations, the Rayleigh simulations presented wrong detections also for

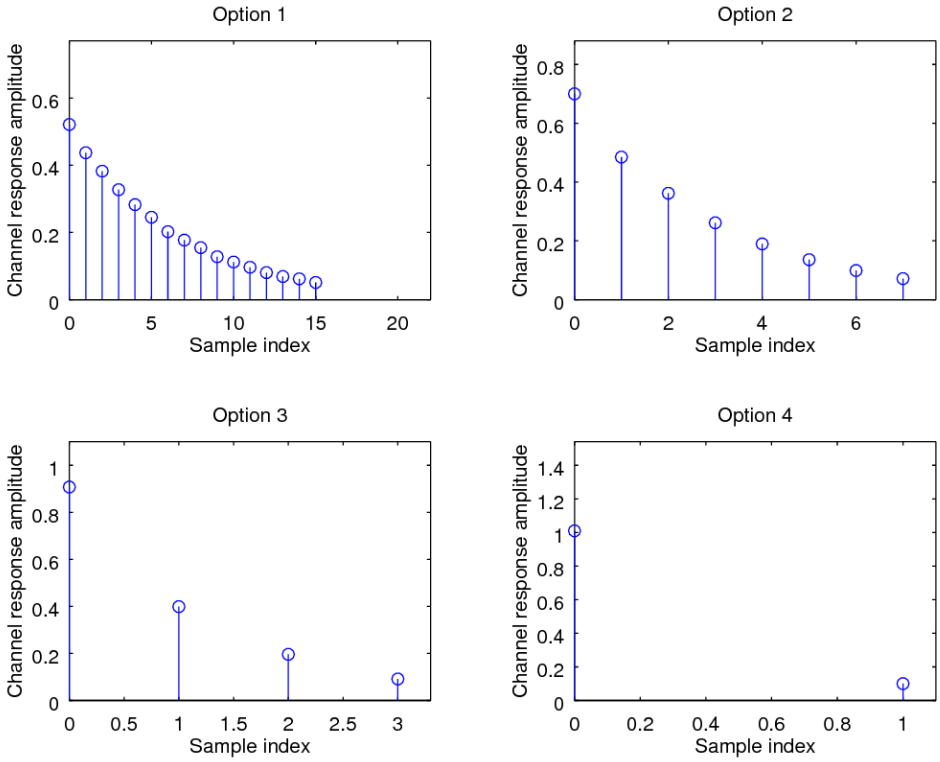

Figure 6.5 – Average channel impulse response with delay spread of $N_g/2$ samples

options 1 and 2 (compare Figures 6.2 and 6.7). We have not found an explanation for the behavior of the wrong detection rate curves in the Rayleigh channel simulations (with unexpected peak and valley instead of simple decaying curves), but we observe from these graphics that wrong detections occur more frequently on multipath channels than on AWGN channels (and consequently accounts more for the failure rate computation), which means that, even with the channel spread, the STFs are still detected, but with TOA errors out of the tolerable region.

Regarding the STF guard parameter, note that offsetting the detected border by a factor of $N/8$ delivers better result than offsetting by $N/16$. We conclude that the channel spread indeed makes the synchronization algorithm detect the border lately, so the purpose of the STF guard is compensating this problem. However, using an aggressive offset into the CP could also cause ISI if the channel length was large as well, i.e. the algorithm would avoid ISI with the next OFDM symbol but could induce ISI with the previous OFDM symbol. Therefore, it is difficult to select the best STF guard with no knowledge of the channel. Fine border detection or STF guard update after channel estimation could improve this scenario, however, this approach is out of the scope of this work.

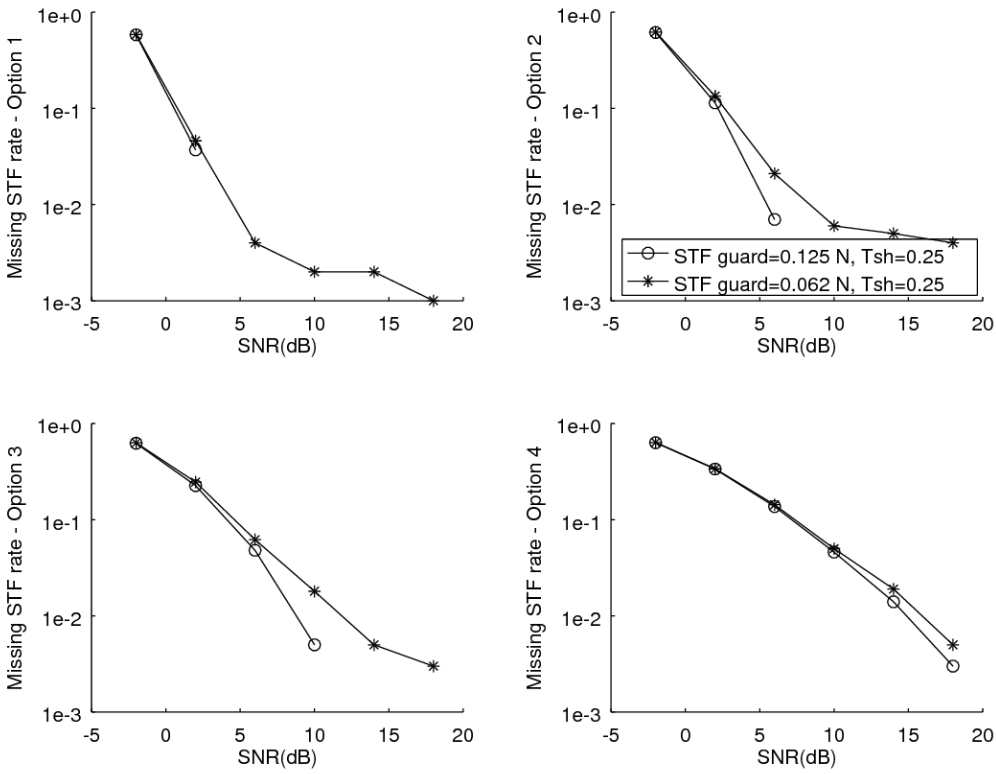

Figure 6.6 – Failure rate computed during Rayleigh channel simulation, $L = N_g/2$

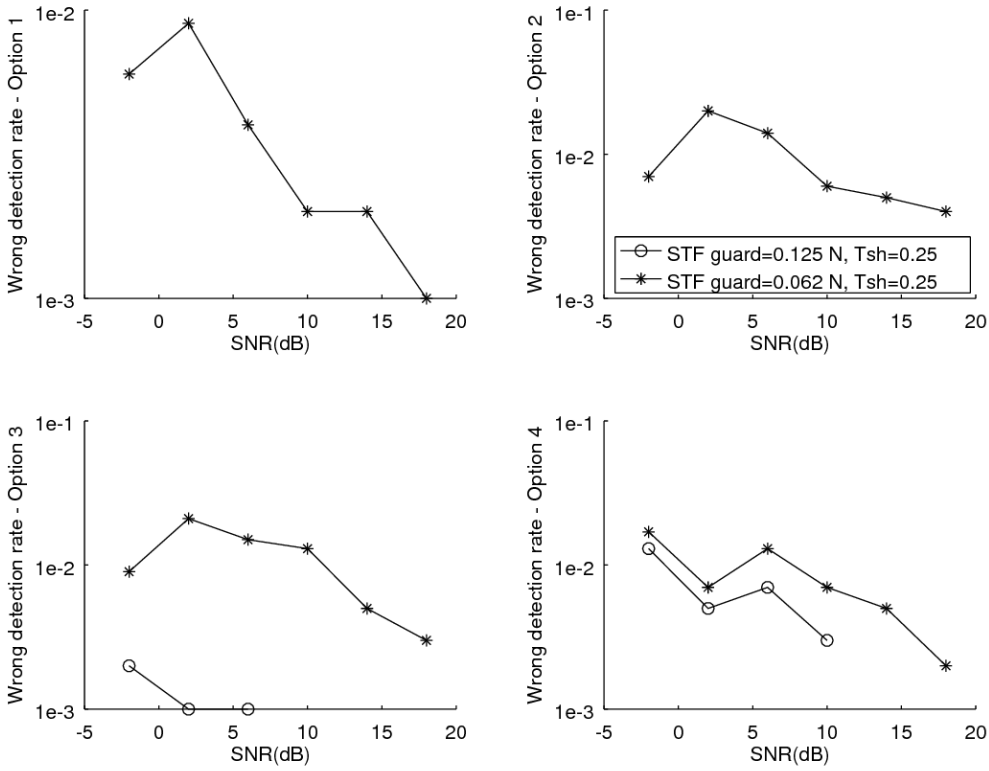

Figure 6.7 – Wrong detection rate computed during Rayleigh channel simulation, $L = N_g/2$

As previously mentioned, the proposed Frame Synchronizer not only estimates the OFDM symbol border, but also the FCFO based on the same autocorrelation computed for the Minn metric. During the AWGN and Rayleigh channel simulations, the FCFO estimations were monitored and their MAE was computed as well.

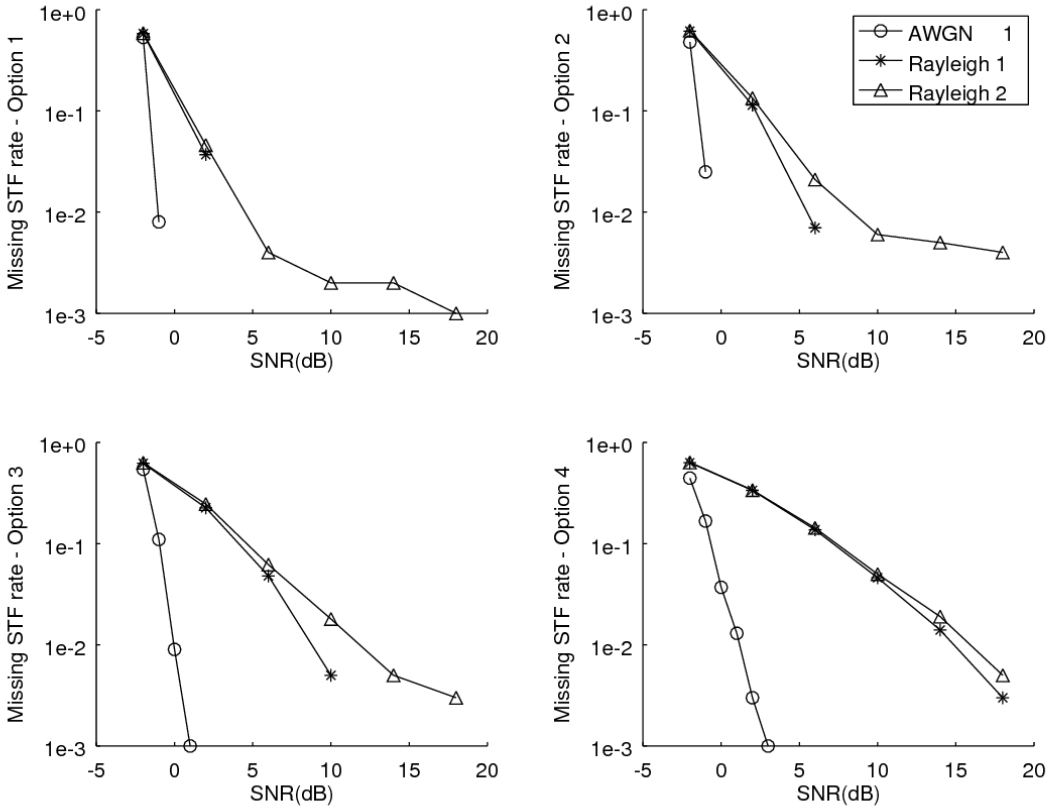

Figure 6.8 – Failure rate computed during AWGN and Rayleigh channel simulations
AWGN 1: $guard = N/8, Tsh = 0.25$
Rayleigh 1: $guard = N/8, Tsh = 0.25$
Rayleigh 2: $guard = N/16, Tsh = 0.25$

In order to analyze the Frame Synchronizer performance in terms of failure rate and MAE of FCFO estimation, we combine the results of AWGN and Rayleigh channel simulations in the same graphics, so that we can compare the effects of the channel and the algorithm parameters. Figures 6.8 and 6.9 present respectively the failure rate curves and the FCFO estimation MAE curves, combining the results for AWGN simulation with $T_{sh} = 0.25$ and $STFguard = N/8$, and Rayleigh channel simulation with $T_{sh} = 0.25$ and $STFguard = \{N/8, N/16\}$. Note that, for Rayleigh channel simulations, the FCFO estimation error curves practically do not vary with the STF guard parameter (see the triangle and "*" markers).

We conclude that the border detection algorithm is sensitive to both parameters (metric threshold and STF guard offset) and has its performance significantly degraded

by the channel. The FCFO estimation process, in turn, is not sensitive to the algorithm parameters. What contributes the most to its performance variation is the channel itself, but even that variation is not so drastic as the one observed in the failure rate curves.

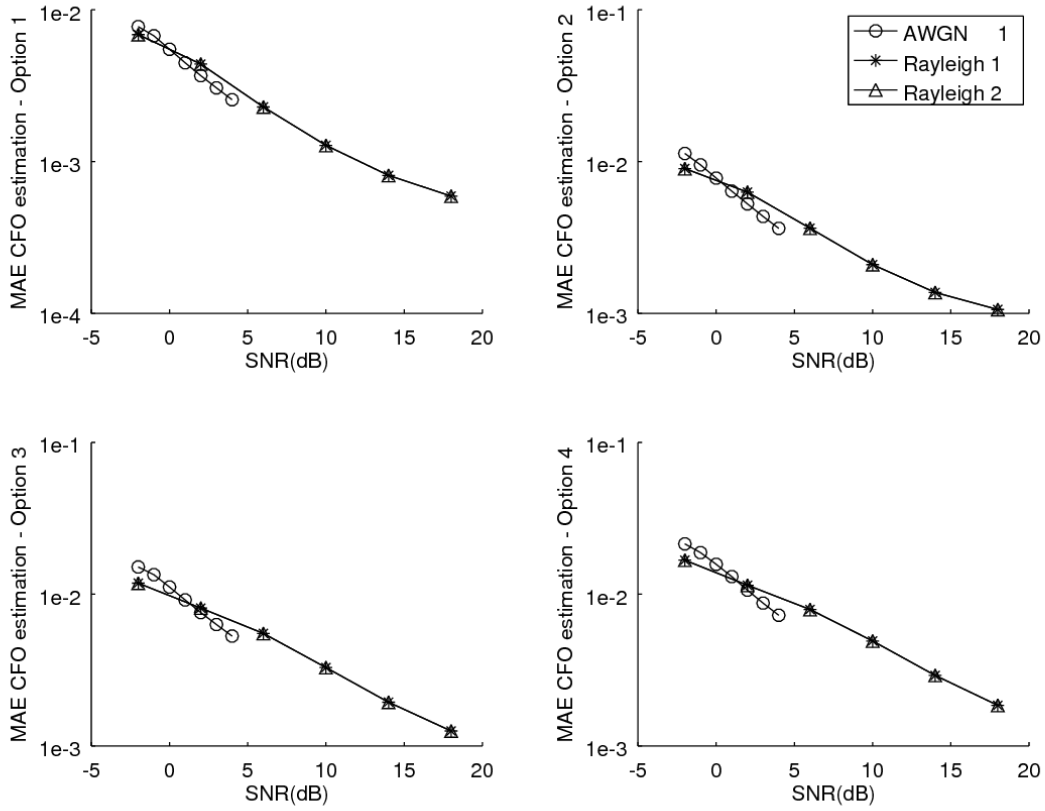

Figure 6.9 – MAE of FCFO estimation computed during AWGN and Rayleigh channel simulations
AWGN 1: $guard = N/8, Tsh = 0.25$
Rayleigh 1: $guard = N/8, Tsh = 0.25$
Rayleigh 2: $guard = N/16, Tsh = 0.25$

6.2 High-level simulations for ICFO estimator

As presented in Subsection 5.1.2 of Chapter 5, the variable parameters for ICFO estimator simulations were the SNR, the normalized ICFO (from -8 to 7) and the data quantization. The performance was analyzed in terms of failure probability: for each combination of OFDM option, SNR and quantization (represented as $\#bits/fractional\_width$), the mean of wrong estimates considering all the ICFO values was calculated.

First, we considered only AWGN channel. In this phase, for each configuration, 10000 estimations were executed considering ideal FCFO compensation, an arbitrary symbol boundary error of $\delta_t = 3$ samples and both correlation methods, the original (3.13) and the simplified (3.17). It had been already verified in preliminary simulations that the

time offset does not influence the ICFO estimation performance, as expected according to the mathematics derivation presented in Section 3.3, that is why we fixed δ_t for the exhaustive simulations. The results for AWGN simulation with different quantization and correlation schemes are presented in Figure 6.10.

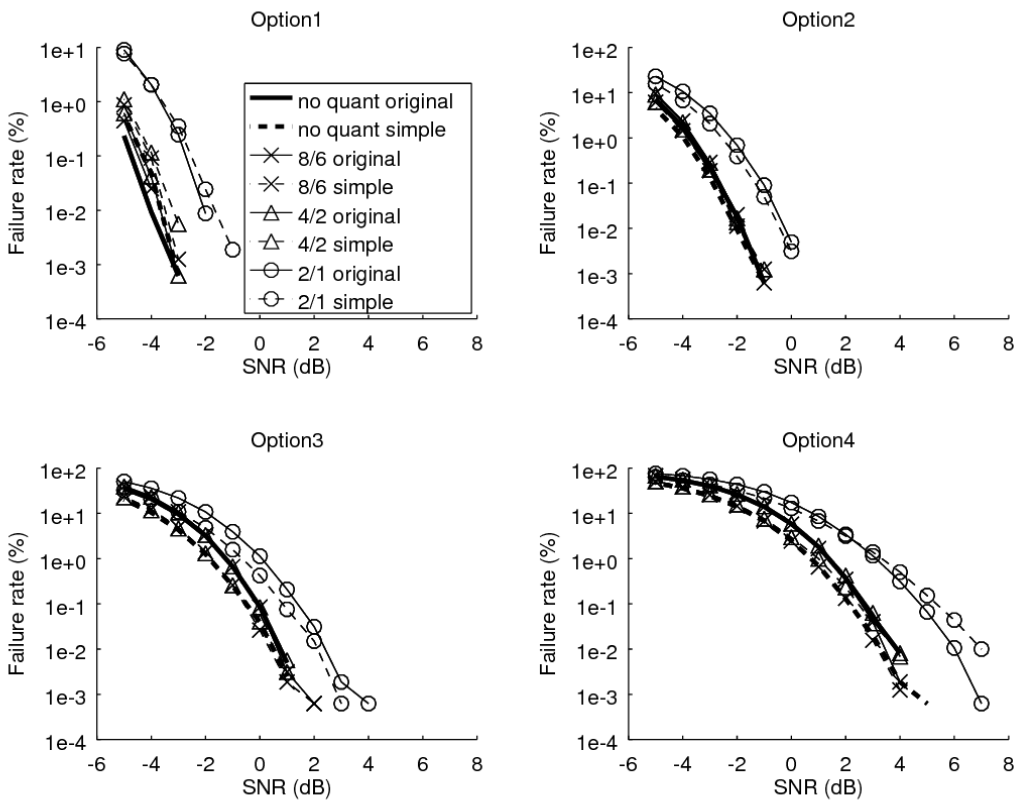

Figure 6.10 – Performance of ICFO estimator using original and simplified methods

The solid lines correspond to the original algorithm in (3.13), the dashed lines correspond to the simplified algorithm in (3.17), the thick lines are the reference curves with no quantization for both algorithms, and the different markers correspond to the quantization schemes `8/5`, `4/2` and `2/1`. As the graphics show, there is a little performance difference between the two algorithms: for option 1, the original algorithm performs better, while for options 2, 3 and 4, the correlation simplification improves the performance for all quantization schemes, except `2/1` in option 4. For both algorithms and for all OFDM options, the curves of the quantization schemes `8/5` and `4/2` are very close to the reference curves, even overlapping them in some cases (see the triangle and "x" markers in relation to the thick lines). On the other hand, there is a considerable degradation (from 1 to 3dB, depending on the OFDM option and the algorithm variation) when only 2 bits are used for internal quantization.

Once it has been verified the correlation simplification presented an overall better performance in AWGN channel compared to the original algorithm, simulation was run

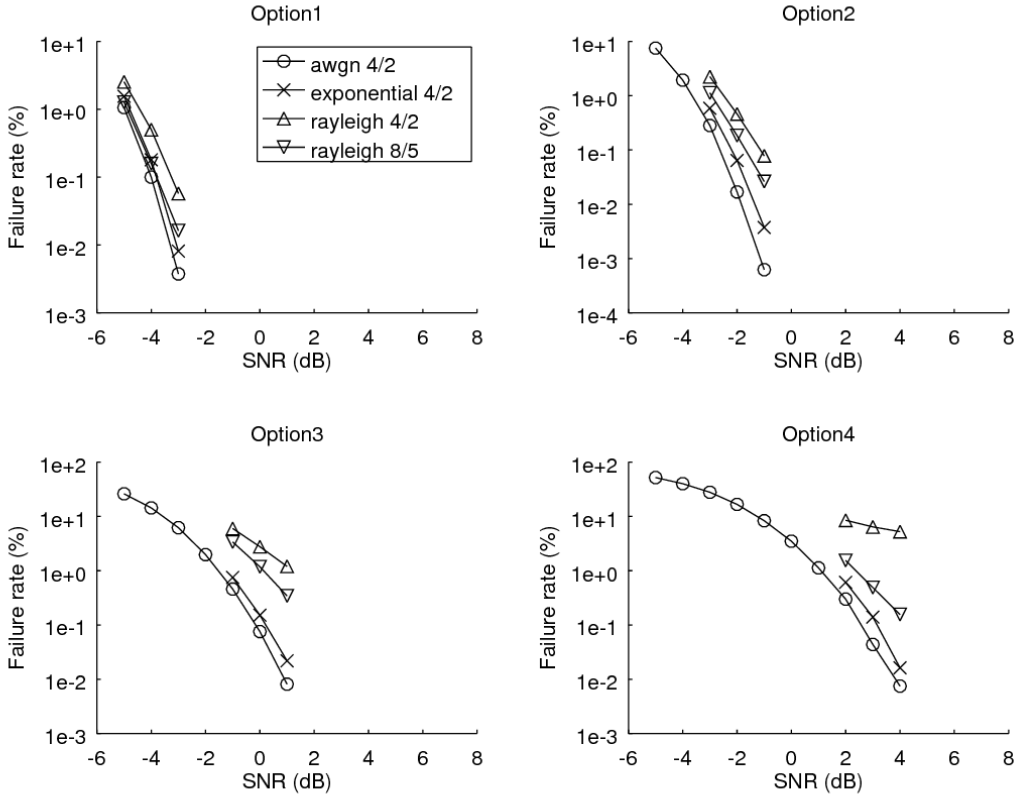

Figure 6.11 – Performance of simplified ICFO estimator in AWGN and multipath channels

again considering an arbitrary fading channel with fixed magnitude decaying, given by the impulse response $h_n = \delta_n + 0.5\delta_{n-1} + 0.1\delta_{n-2}$, where δ_n represents the unit impulse. At this time, in order to speed up the simulation, only the efficient quantization of `4/2` was used, and the estimations were simulated only for the SNR points where the simplified algorithm had presented a failure rate around 0.1% in AWGN simulation. The results are presented in Figure 6.11. Although there is a little difference in the curves (see the curves with circle and "x" markers), the simulation proved the robustness of the algorithm to fading channel with decaying magnitude, and so to exponential channels.

Afterwards, we ran other simulations with quantizations 4/2 and 8/5, but now considering a Rayleigh fading channel with exponential PDP and variable gains, by using the parameters $L = N_g/2$ and $r = 20dB$. During Rayleigh channel simulations, the results were not good for options 3 and 4 (see triangle markers) with 4 bits of quantization. This may be because several channel realizations did not present an exponential impulse response with a reasonable slope, and the estimation algorithm was derived in Section 3.3, Equation (3.11), with the assumption of exponential channel. The combined effect of variable channel behavior and the short data width of 4 bits degraded the overall performance of the ICFO estimator. However, for the quantization 8/5, the performance gets better, with the Rayleigh curves following the slope of the curves associated to the

AWGN and fixed exponential channels, having a degradation of around 2dB with respect to the AWGN curve (the one with circle marker).

We conclude from those several ICFO estimator simulations that the quantization 8/5 can be used in the implemented hardware to deliver a good performance for different channels.

6.3 Test in hardware for frame synchronizer

In this section we present the results obtained during tests in hardware. The objective here is analyzing how the data quantization, combined with the selection of algorithm parameters, influences the frame synchronizer performance. Due to the complexity of the models, the exhaustive high-level simulations for the border detection algorithm had considered only the variation of the algorithm parameters and the channel models, not the quantization, and now the effects of algorithm parameters and data quantization are analyzed on AWGN channel. In hardware, as we were not able to generate other random variables for channel taps, we considered only AWGN noise.

For the first test in hardware, the frame synchronizer was instantiated with a quantization of 16 bits, with 13 decimal bits. The test was run considering normalized $CFO = 0.0$, $MCS = 3$, $threshold = \{0.25, 0.40, 0.50\}$ (also quantized as 16/13), and $STF\ guard = \{N/16, N/8\}$. The failure rates computed for all OFDM options are presented in Figures 6.12, 6.13, 6.14 and 6.15. The threshold 0.25 worked better for all options, as it had been already observed in the high-level simulations. Besides, for options 2, 3 and 4, as the tests were executed with the $STF\ guard = N/16$ and $threshold = 0.25$, it was observed that the failure rate curves started to drop slowly (like a floor), specially for option 4, which could be associated to late detections. This floor was not observed during high-level simulations because quantization had not been considered, but now, in hardware, the quantization effects are evident. The tests were then run again considering $STF\ guard = N/8$, giving a larger margin into the CP. As expected, the curves significantly improved with $STF\ guard = N/8$ for options 2, 3 and 4. For option 1, as $N = 128$, the $STF\ guard = N/16 = 8$ showed to be large enough, with a little improvement with $STF\ guard = N/8 = 16$.

6.3.1 Analysis of quantization effects

For the previous test in hardware, the frame synchronizer was instantiated with a data quantization of 16 bits, with 13 bits for the fractional part, in order to validate the effects of the threshold and the STF guard already observed during high-level simulations. The next step was fixing those algorithm parameters and reducing the bus width, which influences directly into the area usage, specially due to the presence of several buffers

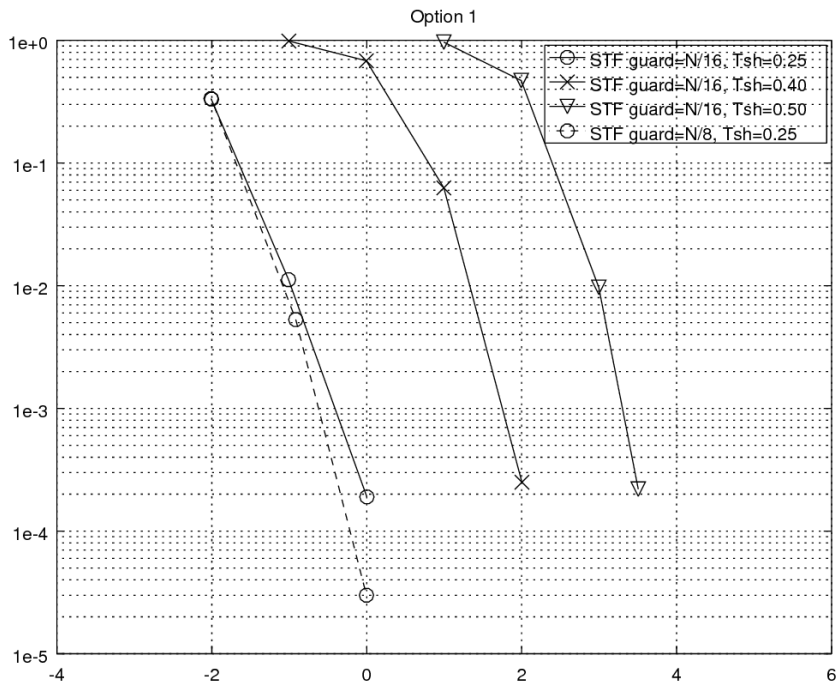

Figure 6.12 – Failure rate computed during tests in FPGA - Option 1

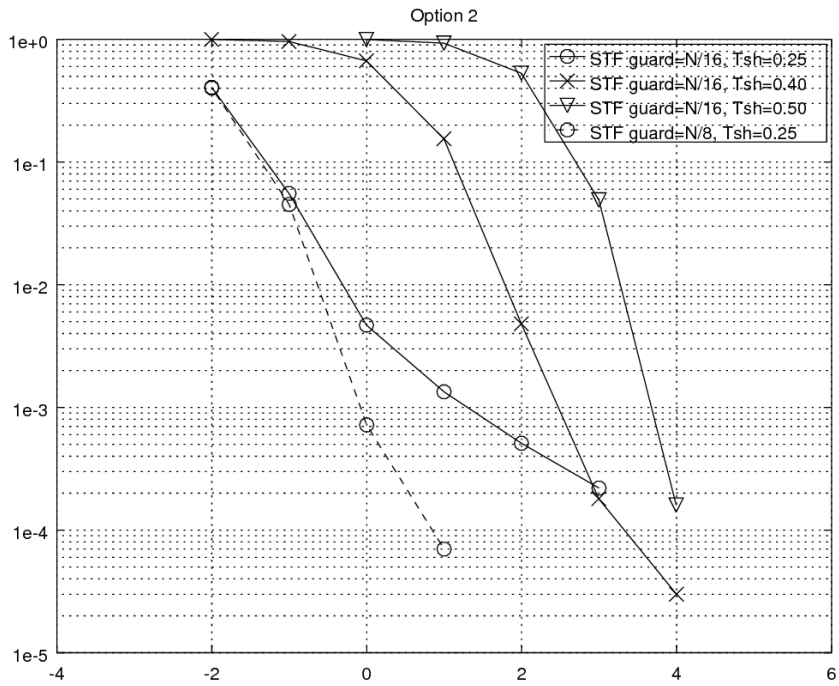

Figure 6.13 – Failure rate computed during tests in FPGA - Option 2

and a few multipliers. We can parameterize the data bus width (number of quantization bits) for the input/output and for the internal data (result of fixed-point multiplications) separately.

As the OFDM option 4 works with the shortest OFDM symbol size (16 FFT points) and is more sensitive to parameter variations due to its limited amount of signal data, we

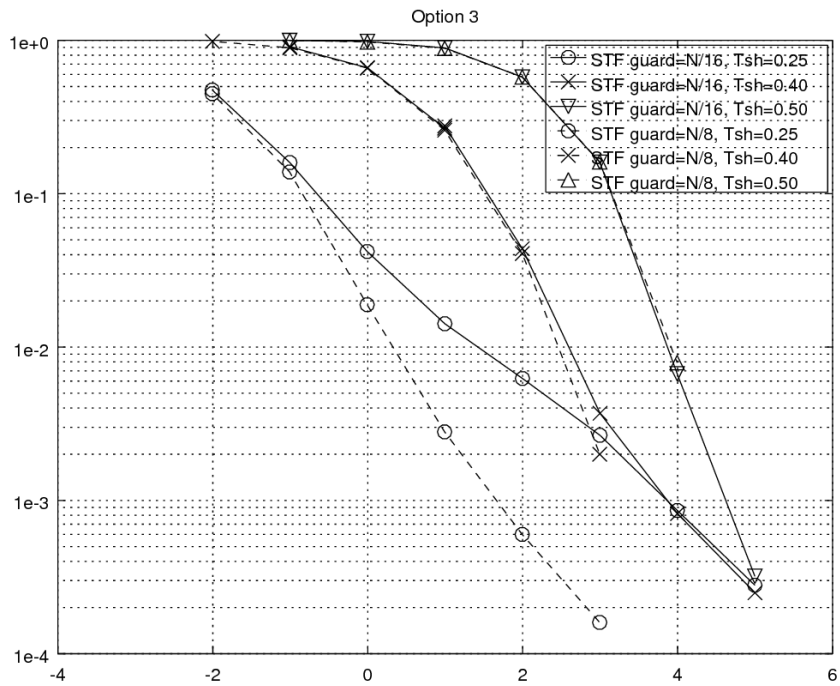

Figure 6.14 – Failure rate computed during tests in FPGA - Option 3

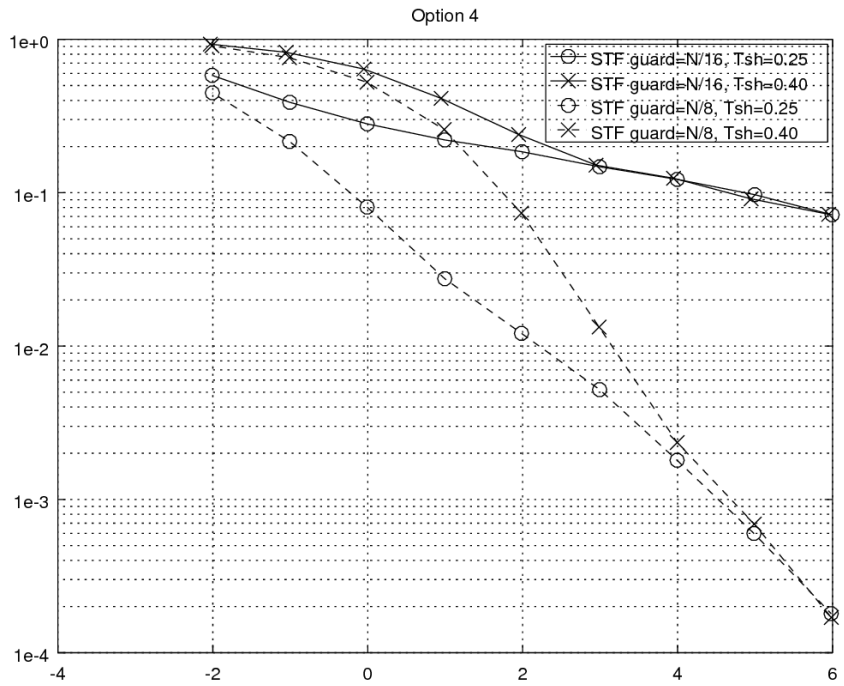

Figure 6.15 – Failure rate computed during tests in FPGA - Option 4

focused on that option to run new hardware tests by fixing the threshold $T_{sh} = 0.25$ and varying the data quantization.

Considering the notation `total_width / fractional_width`, Figure 6.16 presents the result of the tests for option 4, with quantization 16/13, 8/5 and 12/9 (this last one only for internal data bus). Note that the SNR in dB was calculated within the FPGA

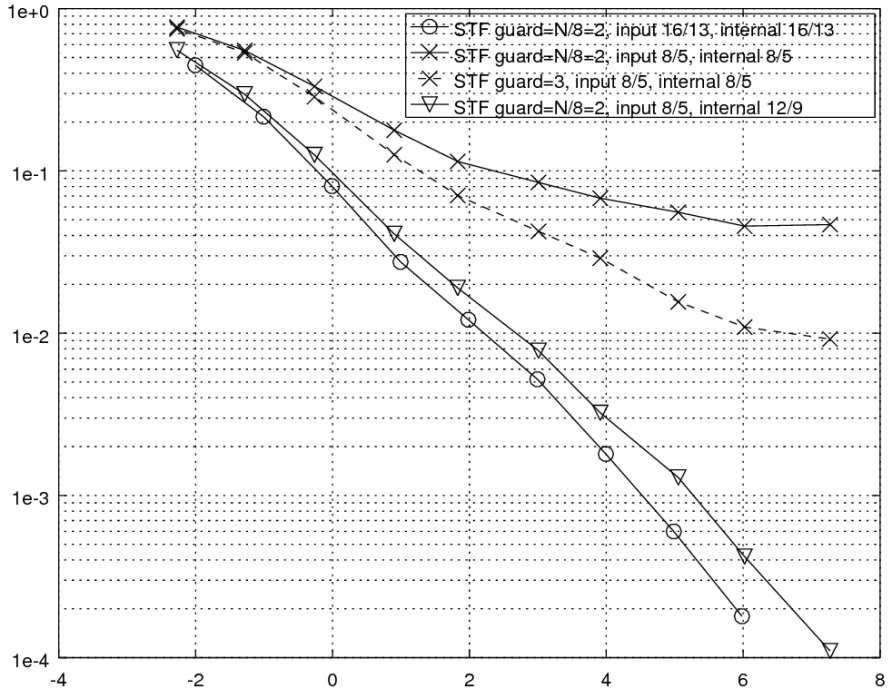

Figure 6.16 – Failure rate for option 4 with varying quantization

and already considers the quantization noise, since the sampled noise signal coming from the external generator is multiplied by the scaling factor, resized to the input bus width, and then passed to the power calculator auxiliary module.

If we quantize both input and internal data with 8/5 bits, the performance significantly degrades (see curves with "x" markers). One possible explanation for that is the presence of a plateau in the calculated metrics, which would lead the algorithm to detect the border late by selecting the sample associated to the last metric of the plateau. Note that this is just an effect of data quantization, we are not even considering multipath channels as it was done in high-level simulations. As already explained in the beginning of Section 6.3, the tests in hardware took into account only AWGN channel.

Now, returning to the influence of the STF guard offset parameter, if it is set to a value larger than N/8, the algorithm works better, as can be seen in the dashed line of Figure 6.16 ($STF\_guard = 3$), which corroborates our intuition. However, making this guard so large (longer than half CP length) would reduce the margin for the channel length and could cause ISI anyway. Therefore, we will only consider STF guard up to half CP length, i.e. N/8. If we keep the input bus width as 8/5 and extend the internal data bus to 12/9, the performance curve gets better (see the curve with triangle markers), having a degradation of less than 1dB with respect to the curve of quantization 16/13.

The failure rates computed for all OFDM options, considering $T_{sh} = 0.25$, STF guard $= \{N/8, N/16\}$ and input width $= \{8, 16\}$ are presented in Figures 6.17, 6.18, 6.19 and 6.20. The continuous lines are associated to $STFguard = N/16$, and the dashed lines

to $STFguard = N/8$. The circle markers are associated to the quantization 16/13, and the "x" markers, to input quantization 8/5 and internal bus width 12/9.

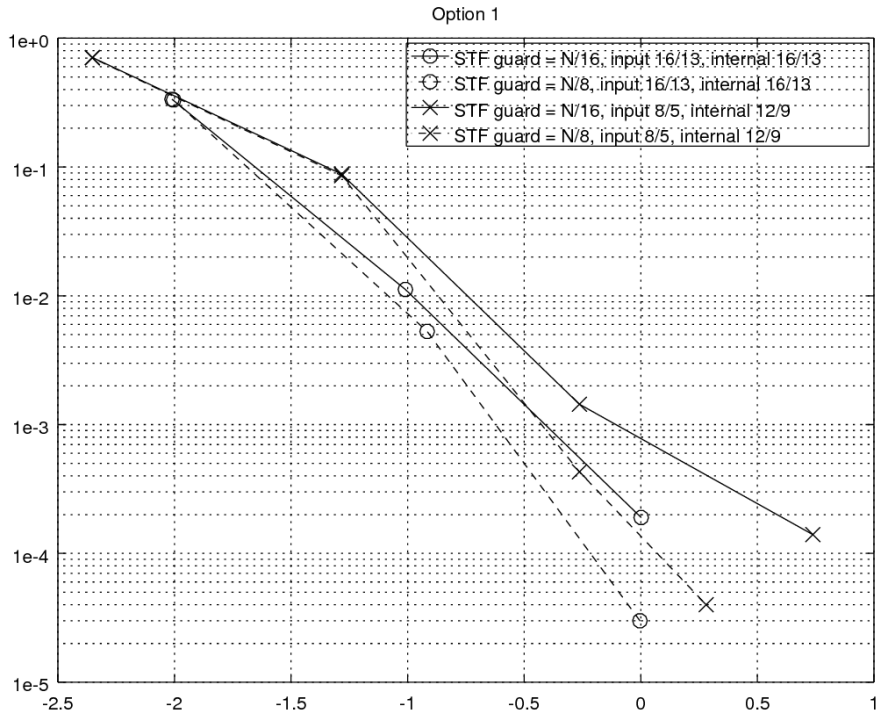

Figure 6.17 – Results of test in FPGA with varying quantization - Option 1

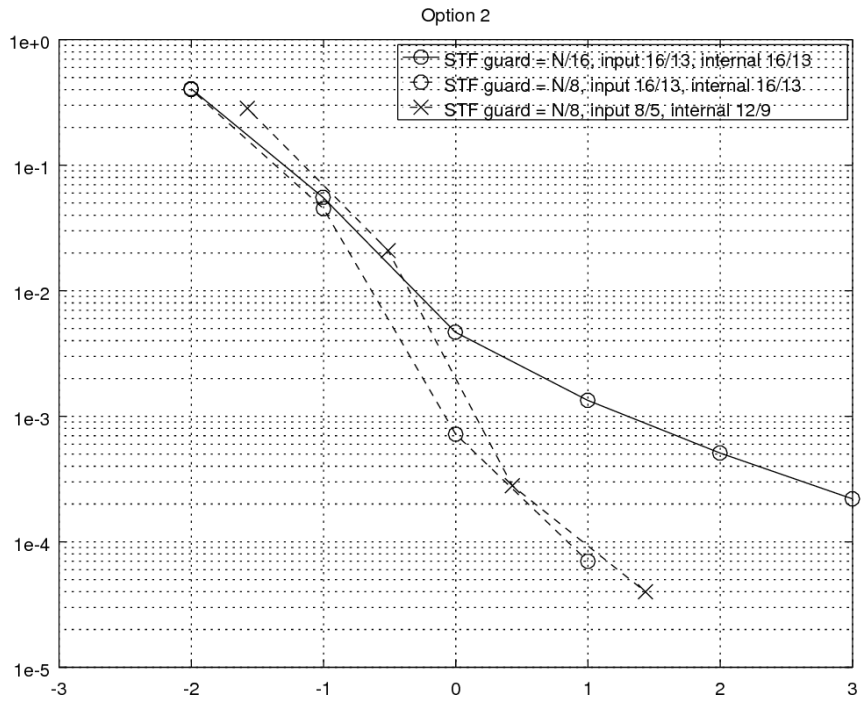

Figure 6.18 – Results of test in FPGA with varying quantization - Option 2

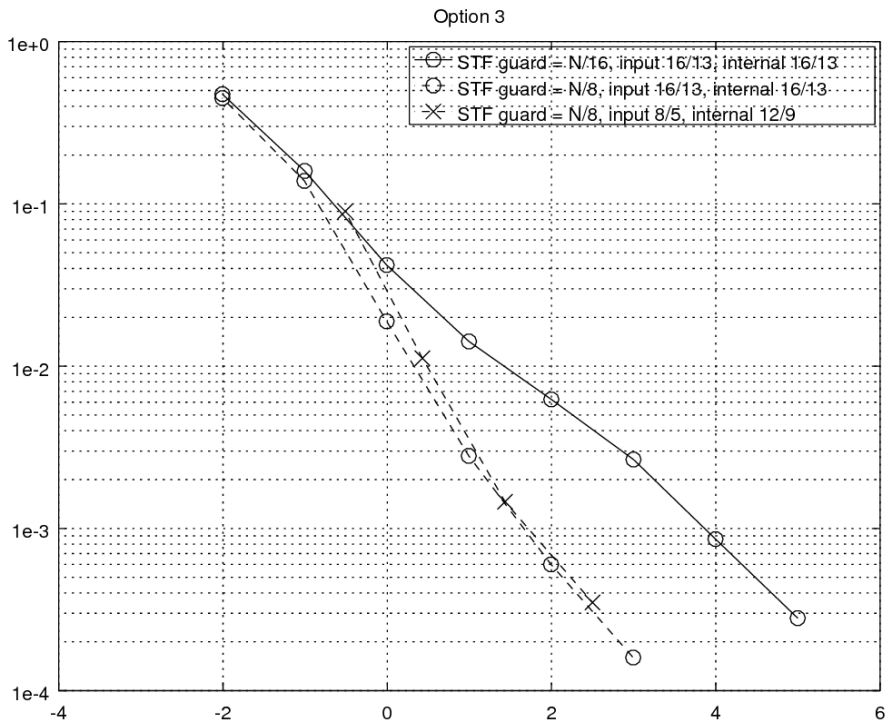

Figure 6.19 – Results of test in FPGA with varying quantization - Option 3

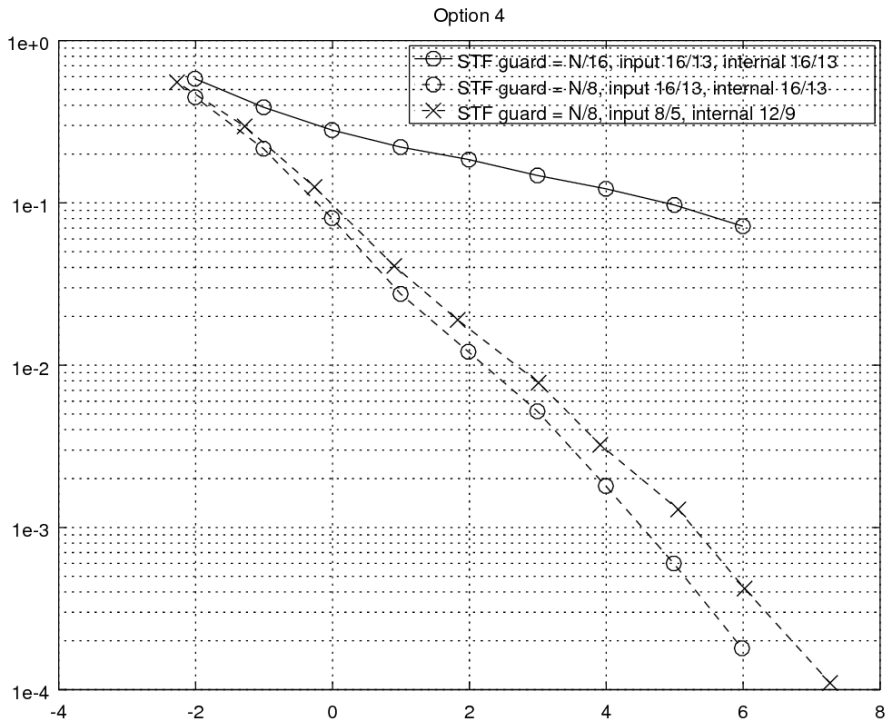

Figure 6.20 – Results of test in FPGA with varying quantization - Option 4

As a conclusion of the tests in hardware, we select the following parameters for the Frame Synchronizer:

- Minn metric threshold $T_{sh} = 0.25$;
- STF guard offset $= N/8$;
- Input quantization 8/5;
- Internal bus width 12/9.

6.4 Synthesis Results

In this section, we present the synthesis results obtained with the Altera Quartus software.

Table 6.1 presents the summary of resource usage while synthesizing the Frame Synchronizer with the following quantizations: input and internal data with 16/13; and input with 8/5 and internal data with 12/9. The compiler was set to target the Cyclone IV GX FPGA family and to run in area optimization mode. Note that reducing the data width from 16 to 8 bits for input data and from 16 to 12 for internal multipliers, saves approximately 20% of the Frame Synchronizer area.

	16/13	8/5 (input), 12/9 (internal)
Total logic elements	16007	12858
Combinational functions	15874	12714
Total registers	10208	7792
Memory bits	37808	29212
9-bit multipliers	40	24

Table 6.1 – Synthesis result for Frame Synchronizer

Table 6.2 presents the summary of resource usage while synthesizing the ICFO estimator and compensator with the following quantizations for the internal bus width: 4/2 and 8/5. The implemented module was parameterized to estimate the ICFO in the range $\tilde{\epsilon} = \{-8, -6, -4, 2, 0, 2, 4, 6, 8\}$. The step of 2 is due to the estimation range of the FCFO estimator, which can resolve $-1 \leq \lambda \leq 1$. The compiler was set to target the Cyclone IV GX FPGA family and to run in area optimization mode. Note that reducing the data width from 8 to 4 bits saves approximately 30% of the ICFO estimator/compensator area. As the data bus that connects the blocks within the receiver has its width fixed at 8/5, the varying quantization is related just to the internal metrics computed during the estimation algorithm, such as the complex additions and autocorrelation, while the FIFO used to buffer the input data for further compensation is fixed at 8/5. That is why the number of memory bits is the same in both synthesis results.

We have presented so far data quantizations applied to baseband complex signals, which are composed of real and imaginary components, and we have considered bus

	8/5	4/2
Total logic elements	2658	1874
Combinational functions	2633	1851
Total registers	847	751
Memory bits	6400	6400

Table 6.2 – Synthesis result for ICFO estimator and compensator

widths up to 16 bits in both simulations and tests in hardware. However, angle signals (from $-\pi$ to π) need more precision than baseband complex signals, so that the CORDIC properly rotates the signal samples. Therefore the estimated FCFO sent by the Frame Synchronizer to the FCFO compensator, which is the output of the internal CORDIC operating in angle calculation mode, is quantized with 30 bits, different from the data bus width of 8 bits, common to all the blocks. The CORDIC inside the FCFO compensator operates in rotation mode with 30 bits as well, but the output is resized back to 8 bits so that the complex baseband signal is passed to the next block. We have not studied the quantization effect of the angle signal since the CORDIC is an IP used in the implemented blocks, so we just used the recommended width.

Finally, Table 6.3 presents the result of the synthesis of the complete OFDM receiver targetting the Stratix IV FPGA family. The blocks were quantized as following: the bus width for input/output data was 8/5 for all the blocks; the internal bus width for the Frame Synchronizer was 12/9; the width of the signals associated to angle values and used by CORDIC IP was 30/27; and the internal data of ICFO estimator was quantized as 8/5. Note that, together with the Viterbi decoder, the implemented synchronization blocks account for most of the area of the receiver, due to their complexity. The other blocks, which operate soft- and hard-bits, are less complex than the signal processing blocks.

Block	Comb.	Reg.	Mem. Bits	DSP 18bit	DSP 9x9	DSP 12x12	DSP 18x18	DSP 36x36
Rx	23459	21996	1157853	44	14	6	3	4
FFT	1681	1772	4096	2	0	0	1	0
FIFO	135	39	1114112	0	0	0	0	0
CP remover	64	40	0	0	0	0	0	0
Deframer	157	73	0	0	0	0	0	0
Deinterleaver	164	63	960	2	2	0	0	0
Demapper	175	59	0	0	0	0	0	0
Depuncturer	28	23	0	0	0	0	0	0
Descrambler	17	14	0	0	0	0	0	0
Despreader	789	935	0	0	0	0	0	0
Equalizer	1680	2898	3160	10	2	6	0	0
FCFO comp.	4647	2940	0	8	0	0	0	2
Frame sync.	7608	7964	29125	18	6	0	2	2
ICFO comp.	2041	857	6400	4	4	0	0	0
PHR parser	7	33	0	0	0	0	0	0
Viterbi Decoder	3562	3910	0	0	0	0	0	0
Debug and glue logic	670	371	0	0	0	0	0	0

Table 6.3 – Synthesis result for Rx

7 Conclusion

In this work, we proposed methods for time and frequency synchronization applied to the MR-OFDM PHY of IEEE 802.15.4g standard. We have gone through the following steps to design a functional hardware implementation: we studied classical and state-of-the-art synchronization algorithms; proposed methods to detect frame border and recover carrier frequency offset; presented high-level simulation results for all the proposed algorithms; designed a hardware architecture for each of the three synchronization modules, by considering the tradeoff between resource usage and performance; implemented the algorithms in RTL; and finally built an FPGA-based verification platform to measure the performance of the frame synchronizer in hardware.

Based on high-level simulations considering AWGN and Rayleigh fading channels, the Minn's algorithm used to acquire the frame was shown to work properly even under the presence of CFO. For fractional CFO estimation, we adopted a well known method that uses the same autocorrelation computed during frame acquisition, while for integer CFO estimation and compensation, we developed a novel method that takes advantage of the frame structure proposed for the IEEE 802.15.4g MR-OFDM PHY and can be easily adapted to any other OFDM-based communications system.

The first novelty in our ICFO correction method was taking into account the phase accumulated due to the combined effect of the ICFO and the presence of the cyclic prefix in the transmitted OFDM symbols. Generally, only the cyclic shift in the tone indices is mentioned in the literature as an effect of the ICFO, but by developing the complete mathematics that models the baseband transmission and reception of the OFDM symbols, we could visualize the phase accumulation as a second effect of the ICFO and then propose a novel and simple compensation scheme in the frequency domain. The second contribution was the simplification of the estimation algorithm for the specific case of IEEE 802.15.4g: instead of taking the magnitude of the correlation to be maximized, we only add its complex components. The results of model simulation under AWGN and exponential channel presented good performance of the simplified algorithm for all MR-OFDM options.

The quantization effects on the ICFO estimator performance was analyzed during high-level simulations, while the frame synchronizer behavior was simulated only in floating-point precision. However, once the complete OFDM transmitter and receiver have been implemented in RTL, the frame synchronizer was tested in an FPGA-based verification platform in order to measure its performance considering not only the algorithm parameters, but also the quantization of the input and internal data.

Based on the several reported tests, we recommend the following parameters: for the Frame Synchronizer, the detection algorithm works better with a metric threshold of 0.25 and an STF guard offset of half the CP length ($N/8$), and the tradeoff between hardware resource usage and algorithm performance is achieved with a quantization of 8 bits for the input data and 12 bits for the internal data bus associated to multiplication results; for the ICFO estimator and compensator, the quantization of 8 bits is enough for a good performance result in AWGN and exponential channels.

For future work, some activities are proposed as following. The algorithms presented here focus on coarse border detection and frequency recovery, however, refinement of the detected border might be also required in the presence of SCO and channel delay spread for example, therefore more studies could be carried out in this area. Besides, although the simplification of Minn's algorithm for the Frame Synchronizer (with the suppression of the scaling factor and the exclusion of one repetition for the computation of the preamble energy) have showed to deliver a good performance in terms of BER, even under low SNR, this simplification was not properly compared with the original algorithm in exhaustive simulations. More simulations could be run in order to check if there is some degradation while simplifying the algorithm. As for the tests in hardware, the FPGA-based verification platform could be extended to support simulations of channels based on configurable TDL filters, not only AWGN channels. The randomization of the filter taps would be considerable complex, but even the use of fixed taps could help in generating more realist curves for channels with a given spread delay. Those are just suggestions to improve the overall performance of the complete system, but we can conclude that all the modules proposed and implemented in this work had their functionality proved in hardware and then can be used in a first demo of the OFDM transceiver.

References

1 SOCIETY, I. C. *IEEE Standard for local and metropolitan area networks; Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs); Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks.* IEEE 802.15.4g-2012 (Amendment to IEEE 802.15.4-2011). Citado 8 vezes nas páginas 16, 20, 21, 22, 23, 24, 33 e 43.

2 SOCIETY, I. C. *IEEE Standard for Information Technology – Telecommunications and Information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) Utility Networks.* Citado na página 16.

3 LI, Y. G.; STUBER, G. L. *Orthogonal frequency division multiplexing for wireless communications.* [S.l.]: Springer Science & Business Media, 2006. Citado na página 16.

4 CHIUEH, T.-D.; TSAI, P.-Y. *OFDM Baseband Receiver Design for Wireless Communications.* [S.l.]: Wiley Publishing, 2007. ISBN 0470822341, 9780470822340. Citado 3 vezes nas páginas 17, 29 e 40.

5 PHAM, T.; FAHMY, S.; MCLOUGHLIN, I. Low-power correlation for ieee 802.16 ofdm synchronization on fpga. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, v. 21, n. 8, p. 1549–1553, Aug 2013. ISSN 1063-8210. Citado na página 17.

6 SPETH, M. et al. Optimum receiver design for wireless broad-band systems using ofdm. i. *Communications, IEEE Transactions on*, v. 47, n. 11, p. 1668–1677, Nov 1999. ISSN 0090-6778. Citado 4 vezes nas páginas 17, 18, 25 e 40.

7 KIM, B.; CHOI, S. Integer frequency offset estimation for ieee 802.15.4g sun mr-ofdm phy. In: *2011 5th International Conference on Signal Processing and Communication Systems (ICSPCS).* [S.l.: s.n.], 2011. p. 1–4. Citado na página 17.

8 BEEK, J.-J. van de; SANDELL, M.; BORJESSON, P. Ml estimation of time and frequency offset in ofdm systems. *Signal Processing, IEEE Transactions on*, v. 45, n. 7, p. 1800–1805, Jul 1997. ISSN 1053-587X. Citado na página 17.

9 SPETH, M. et al. Optimum receiver design for ofdm-based broadband transmission .ii. a case study. *IEEE Transactions on Communications*, v. 49, n. 4, p. 571–578, Apr 2001. ISSN 0090-6778. Citado 2 vezes nas páginas 18 e 40.

10 SCHMIDL, T.; COX, D. Robust frequency and timing synchronization for ofdm. *Communications, IEEE Transactions on*, v. 45, n. 12, p. 1613–1621, Dec 1997. ISSN 0090-6778. Citado 2 vezes nas páginas 18 e 26.

11 MINN, H.; BHARGAVA, V.; LETAIEF, K. A robust timing and frequency synchronization for ofdm systems. *Wireless Communications, IEEE Transactions on*, v. 2, n. 4, p. 822–839, July 2003. ISSN 1536-1276. Citado 3 vezes nas páginas 18, 25 e 26.

12 NASRAOUI, L.; ATALLAH, L. N.; SIALA, M. An efficient reduced-complexity two-stage differential sliding correlation approach for ofdm synchronization in the awgn channel. In: *Vehicular Technology Conference (VTC Fall), 2011 IEEE*. [S.l.: s.n.], 2011. p. 1–5. ISSN 1090-3038. Citado na página 18.

13 LUI, C. *Method of Synchronization Using IEEE 802.11a OFDM Training Structure for Indoor Wireless Applications*. [S.l.]: Simon Fraser University, 2004. Citado 2 vezes nas páginas 18 e 19.

14 Denise C. Alves and Eduardo R. de Lima. A frame synchronizer for IEEE 802.15.4-g MR-OFDM PHY: Algorithm proposal and hardware implementation. In: *Communications (LATINCOM), 2015 IEEE Latin-America Conference on*. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 18 e 19.

15 LI, D. et al. Integer frequency offset estimation for ofdm systems with residual timing offset over frequency selective fading channels. *IEEE Transactions on Vehicular Technology*, v. 61, n. 6, p. 2848–2853, July 2012. ISSN 0018-9545. Citado na página 18.

16 ZHAN, Q.; MINN, H. New integer normalized carrier frequency offset estimators. *IEEE Transactions on Signal Processing*, v. 63, n. 14, p. 3657–3670, July 2015. ISSN 1053-587X. Citado na página 18.

17 MORELLI, M.; MORETTI, M. Integer frequency offset recovery in ofdm transmissions over selective channels. *IEEE Transactions on Wireless Communications*, v. 7, n. 12, p. 5220–5226, December 2008. ISSN 1536-1276. Citado 2 vezes nas páginas 19 e 30.

18 ABDZADEH-ZIABARI, H.; SHAYESTEH, M. G. Integer frequency offset recovery in ofdm systems. In: *2012 18th Asia-Pacific Conference on Communications (APCC)*. [S.l.: s.n.], 2012. p. 641–646. ISSN 2163-0771. Citado na página 19.

19 PARK, M. et al. Robust integer frequency offset estimator with ambiguity of symbol timing offset for ofdm systems. In: *Proceedings IEEE 56th Vehicular Technology Conference*. [S.l.: s.n.], 2002. v. 4, p. 2116–2120 vol.4. ISSN 1090-3038. Citado 4 vezes nas páginas 19, 31, 32 e 41.

20 GAO, Z.; ZHANG, C.; WANG, Z. Robust preamble design for synchronization, signaling transmission, and channel estimation. *IEEE Transactions on Broadcasting*, v. 61, n. 1, p. 98–104, March 2015. ISSN 0018-9316. Citado na página 19.

21 PHAM, T. H.; FAHMY, S. A.; MCLOUGHLIN, I. V. Efficient integer frequency offset estimation architecture for enhanced ofdm synchronization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 24, n. 4, p. 1412–1420, April 2016. ISSN 1063-8210. Citado 5 vezes nas páginas 19, 30, 39, 40 e 41.

22 ALVES, D. C. et al. Architecture design and implementation of key components of an ofdm transceiver for ieee 802.15.4g. In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.: s.n.], 2016. p. 550–553. Citado na página 23.

23 D. G. Urdaneta et al. FPGA Implementation and ASIC Resource Estimation of an FFT/IFFT for an MR-OFDM Transceiver Complaint with IEEE802.15.4g. In: *5th Workshop on Circuits and System Design (WCAS 2015)*. [S.l.: s.n.], 2015. Citado na página 24.

24 John W. Eaton. *About GNU Octave*. <http://www.gnu.org/software/octave/> [Accessed Dec 2017]. Citado na página 42.

25 VISWANATHAN, M. *Simulation of Digital Communication Systems Using Matlab*. [S.l.: s.n.], 2013. <https://www.gaussianwaves.com/simulation-of-digital-communication-systems-using-matlab-ebook/>. Citado na página 46.

26 VISWANATHAN, M. *Modeling a Frequency Selective Multipath Fading channel using TDL filters*. <http://www.gaussianwaves.com/2016/10/modeling-a-frequency-selective-multipath-fading-channel-using-tdl-filters/>. Online; accessed April 25th 2018. Citado na página 46.

27 VISWANATHAN, M. *Power Delay Profile*. <https://www.gaussianwaves.com/2014/07/power-delay-profile/>. Online; accessed April 25th 2018. Citado na página 46.

28 Altera Corporation. *Stratix IV GX FPGA Development Board Reference Manual*. 2012. <http://www.altera.com/literature/manual/rm_sivgx_fpga_dev_board.pdf> [Accessed Dec 2017]. Citado na página 50.

29 Altera Corporation. *Data Conversion HSMC Reference Manual*. 2008. <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=73&No=360&PartNo=3> [Accessed Dec 2017]. Citado na página 50.

30 BARRY, J.; LEE, E.; MESSERSCHMITT, D. *Digital Communication*. Springer US, 2003. ISBN 9780792375487. Disponível em: <https://books.google.com.br/books?id=ogUB2eYv9tgC>. Citado na página 53.

31 HEISKALA, J.; TERRY, P. J. Ofdm wireless lans: A theoretical and practical guide. 01 2001. Citado na página 55.

ANNEX A – Frequency-domain representation of STF and LTF

In this annex, the tables of STF and LTF frequency-domain representation will be presented in a shortened format, by omitting the tones that carry no energy.

A.1 STF in the frequency domain

Tables A.1, A.2, A.3 and A.4 present the active tones for OFDM options 1, 2, 3 and 4 respectively, where the OFDM symbol sizes are 128, 64, 32 and 16, and the number of active tones are 104, 52, 26 and 14. The others tones not presented in the tables have value equal to 0. Note that the negative tones correspond to the k range from $N/2$ to $N-1$ in the DFT operation, which indexes the tones from 0 to N-1.

Tones	**Value**
-48	-2.9439
-40	-2.9439
-32	-2.9439
-24	2.9439
-16	2.9439
-8	2.9439
8	2.9439
16	-2.9439
24	2.9439
32	2.9439
40	-2.9439
48	2.9439
other tones from -64 to 63	0

Table A.1 – STF active tones - Op. 1

Tones	**Value**
-24	-2.0817
-20	-2.0817
-16	-2.0817
-12	2.0817
-8	2.0817
-4	2.0817
4	2.0817
8	-2.0817
12	2.0817
16	2.0817
20	-2.0817
24	2.0817
other tones from -32 to 31	0

Table A.2 – STF active tones - Op. 2

Tones	Value
-12	2.0817
-8	2.0817
-4	2.0817
4	-2.0817
8	2.0817
12	-2.0817
other tones from -16 to 15	0

Table A.3 – STF active tones - Op. 3

Tones	Value
-6	1.5275
-4	1.5275
-2	1.5275
2	-1.5275
4	1.5275
6	-1.5275
other tones from -8 to 7	0

Table A.4 – STF active tones - Op. 4

The normalization operation in frequency domain is characterized by the scaling factor $\sqrt{N_{active}/N_{stf}}$, where N_{active} is the number of used tones in rest of the OFDM packet, and N_{stf} is the number of active tones within STF for a given operation mode. For options 1 and 2, $N_{stf} = 12$, and for options 3 and 4, $N_{stf} = 6$. In adittion to the normalization, power boosting shall be applied to the STF symbols in time domain in order do aid preamble detection, and this is done by multiplying the samples by 1.25.

A.2 LTF in the frequency domain

Tables A.5, A.6, A.7 and A.8 present the active tones for options 1, 2, 3 and 4 respectively, where the DFT sizes are 128, 64, 32 and 16, and the number of active tones are 104, 52, 26 and 14. The others tones not presented in the tables have value equal to 0.

Tones	Value	Tones	Value
-13	1	1	-1
-12	-1	2	-1
-11	1	3	1
-10	-1	4	-1
-9	1	5	1
-8	1	6	1
-7	1	7	-1
-6	1	8	-1
-5	1	9	1
-4	1	10	1
-3	1	11	-1
-2	1	12	-1
-1	-1	13	1

Table A.7 – LTF active tones - Op. 3

Tones	Value	Tones	Value
-7	1	1	-1
-6	-1	2	1
-5	1	3	1
-4	1	4	1
-3	-1	5	-1
-2	1	6	-1
-1	1	7	-1

Table A.8 – LTF active tones - Op. 4

Tone	**Value**	**Tone**	**Value**	**Tone**	**Value**	**Tone**	**Value**
-52	-1	-26	-1	1	1	27	-1
-51	1	-25	-1	2	-1	28	1
-50	1	-24	-1	3	1	29	1
-49	-1	-23	-1	4	-1	30	-1
-48	-1	-22	1	5	1	31	1
-47	-1	-21	1	6	1	32	-1
-46	-1	-20	-1	7	-1	33	-1
-45	1	-19	1	8	-1	34	-1
-44	1	-18	-1	9	1	35	1
-43	-1	-17	-1	10	-1	36	1
-42	-1	-16	1	11	1	37	1
-41	1	-15	-1	12	1	38	1
-40	1	-14	1	13	1	39	1
-39	1	-13	1	14	1	40	1
-38	-1	-12	1	15	-1	41	-1
-37	-1	-11	1	16	1	42	-1
-36	1	-10	-1	17	1	43	-1
-35	1	-9	-1	18	1	44	-1
-34	-1	-8	1	19	1	45	-1
-33	-1	-7	1	20	1	46	-1
-32	-1	-6	-1	21	-1	47	1
-31	-1	-5	1	22	1	48	-1
-30	-1	-4	1	23	-1	49	1
-29	1	-3	-1	24	1	50	1
-28	1	-2	1	25	-1	51	-1
-27	-1	-1	1	26	1	52	1

Table A.5 – LTF active tones - Op. 1

Tone	**Value**	**Tone**	**Value**	**Tone**	**Value**	**Tone**	**Value**
-26	-1	-13	1	1	1	14	-1
-25	-1	-12	-1	2	-1	15	1
-24	-1	-11	-1	3	1	16	1
-23	-1	-10	-1	4	1	17	-1
-22	1	-9	1	5	-1	18	-1
-21	1	-8	1	6	1	19	-1
-20	1	-7	-1	7	-1	20	-1
-19	-1	-6	1	8	-1	21	-1
-18	1	-5	1	9	1	22	1
-17	-1	-4	1	10	-1	23	-1
-16	1	-3	-1	11	1	24	-1
-15	-1	-2	-1	12	1	25	-1
-14	1	-1	-1	13	-1	26	1

Table A.6 – LTF active tones - Op. 2