UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Daniel Garcia Urdaneta

Fast Fourier Transform Implementation and Integer Carrier Frequency Offset Estimation for the IEEE802.15.4g Standard

Implementação da transformada rápida de Fourier e estimador de erro de frequência para a norma IEEE802.15.4g

Campinas

2019

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Daniel Garcia Urdaneta

Fast Fourier Transform Implementation and Integer Carrier Frequency Offset Estimation for the IEEE802.15.4g Standard

Implementação da transformada rápida de Fourier e estimador de erro de frequência para a norma IEEE802.15.4g

Dissertation presented to the Faculty of Electric and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electric Engineering in the area of Telecommunications and Telematics

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 Mestre em Engenharia Elétrica, na Área de Telecomunicações e Telemática.

Supervisor: Prof. Dr. Luis Geraldo Pedroso Meloni

Co-orientador Dr. Eduardo Rodrigues de Lima

Este exemplar corresponde à versão final da dissertaçã defendida pelo aluno Daniel Garcia Urdaneta, e orientada pelo Prof. Dr. Luis Geraldo Pedroso Meloni

Campinas

2019

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

G164f

Garcia Urdaneta, Daniel, 1982-
Fast Fourier transform implementation and integer carrier frequency offset estimation for the IEEE802.15.4g Standard / Daniel Garcia Urdaneta. – Campinas, SP : [s.n.], 2019.

Orientador: Luis Geraldo Pedroso Meloni.
Coorientador: Eduardo Rodrigues de Lima.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. FPGA (Field Programmable Gate Array). 2. Fourier, Transformadas de. I. Meloni, Luis Geraldo Pedroso, 1958-. 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: Implementação da transformada rápida de Fourier e estimador de error de frequência para a norma IEEE802.15.4g
Palavras-chave em inglês:
FPGA (Field Programmable Gate Array)
Fourier transform
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Lusi Geraldo Pedroso Meloni
Cristiano Akamine
Gustavo Fraindenraich
Data de defesa: 13-03-2019
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: 0000-0002-8693-3586
- Currículo Lattes do autor: http://lattes.cnpq.br/2221979852984408

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato : Daniel Garcia Urdaneta RA: 161520

Data da Defesa : 13 de março de 2019

Título da Tese : "Implementação da transformada rápida de Fourier e estimador de erro de frequência para a norma IEEE802.15.4g".

Prof. Dr. Luis Gerardo Pedroso Meloni (Presidente)

Prof. Dr. Cristiano Akamine

Prof. Dr. Gustavo Fraindenraich

Dedico esta Dissertação à minha familia.

Agradecimentos

Agradeço principalmente à minha família, meus irmãos e os meus pais, pelo apoio e a força, ainda nos tempos difíceis que estamos vivendo no nosso pais.

Ao Instituto de pesquisas Eldorado, o líder de projetos Eduardo Rodrigues de Lima, ao Centro de Pesquisa e Desenvolvimento em Telecomunicações CPqD, os organizadores do programa CI-Brasil e o Professor Luís Geraldo P. Meloni.

Aos meus amigos.

Abstract

Fast Fourier Transforms (FFT) and Inverse Fast Fourier Transforms (IFFT) algorithms play an important role in the modulation and demodulation of orthogonal frequency-division multiplexing (OFDM) signals. However, the implementation of these algorithms in hardware consumes a considerable amount of resources, which makes optimizations necessary for optimum hardware implementation. This work presents the implementation of a radix-2 fast FFT/IFFT transform based on COordinate Rotation DIgital Computer (CORDIC) for the MR-OFDM (MR stands for Multi-Rate Multi-regional) mode and compliant to the IEEE802.15.4g standard. This standard addresses service networks (SUN) in the context of low-rate wireless personal area networks (LR-WPAN). In addition, the work presents an architecture for estimation of integer carrier frequency error (ICFO), which takes advantage of the structure of the Orthogonal frequency-division multiplexing (OFDM) demodulator and the cross-correlation technique, involving the calculation of the Discrete Fourier Transform (DFT). This architecture has low hardware consumption and low complexity to perform the correlation calculation when compared to usual estimation methods. It correlates the received preamble and the training sequence found in the synchronization field of the IEEE802.15.4g packet. The work shows results for a implementation in a Field-Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC), as well as comparisons with other architectures. This work is part of the development and implementation of a transceiver compatible with the IEEE802.15.4g standard at the Eldorado Research Institute.[1].

Keywords: IEEE802154g, Field Programmable Gate Array FPGA, Fast Fourier Transform, Discrete Fourier Transform, Integer Carrier Frequency Offset, Frequency Synchronization, Smart Metering Utility Networks, Orthogonal Frequency Division Multiplexing.

[1] This work was Hosted and Funded by Eldorado Research Institute and "*Conselho Nacional de Desenvolvimento Científico e Tecnológico* CNPq" Under the CI-Brasil Program

Resumo

Os algoritmos da transformada rápida de Fourier (em inglês Fast Fourier Transform FFT, e Inverse Fast Fourier Transform IFFT) desempenham um papel muito importante na modulação e demodulação de sinais de multiplexação por divisão de frequências ortogonais OFDM (Orthogonal Frequency-Division Multiplexing) . Porém, a implementação destes algoritmos em *hardware* consume uma quantidade considerável de recursos, o que faz necessárias implementações otimizadas da mesma. Este trabalho apresenta a implementação de uma transformada rápida FFT/IFFT radix-2 baseada no computador digital para rotação de coordenadas CORDIC (COordinate Rotation DIgital Computer) para o modo MR-OFDM (MR significa multi-região multi-taxa, do inglês Multi-rate Multi Regional) de um modem compatível com a norma IEEE802.15.4g. Esta norma está voltada para redes de serviços SUN (Smart Utility Networks) no contexto das redes de área pessoal sem fio com baixa taxa de transmissão (Low-Rate Wireles Personal Area Networks LR-WPAN). Em adição é apresentada uma arquitetura para um estimador de erro de frequência inteiro (Integer Carrier Frequency Offset ICFO) que aproveita a estrutura do demodulador OFDM e da técnica de correlação cruzada, o que envolve o cálculo da transformada discreta de Fourier DFT (Discrete Fourier Transform). Esta arquitetura possui um baixo consumo de hardware e baixa complexidade para realizar o cálculo da correlação quando comparado com métodos usuais de estimação. Ela correlaciona o preâmbulo recebido e a sequência de treinamento encontrado no campo de sincronização do pacote IEEE802.15.4g. Os resultados da implementação em FPGA (Field Programmable Gate Array, em português Arranjo de Portas Programáveis em Campo) e ASIC (Application Specific Integrated Circuit, em português circuitos integrados de aplicação específica), assim como as comparações com outras arquiteturas, são mostrados para ambas implementações. Este trabalho faz parte do desenvolvimento e implementação de um transceiver compatível com a norma IEEE802.15.4g no Instituto de Pesquisas Eldorado[2]

Keywords: IEEE802154g, Field Programmable Gate Array FPGA, transformada de Fourier, transformada discreta de Foutier, error inteiro de frequencia, sincronização de frequência, Redes de utilidades, Orthogonal Frequency Division Multiplexing (OFDM).

[2] Patrocinado e Hospedado pelo Instituto de Pesquisas Eldorado e o Conselho Nacional de Desenvolvimento Científico e Tecnológico CNPq como parte do Programa CI-Brasil.

List of Figures

Figure 2.1 – Spectrum of an OFDM communication scheme. 21
Figure 2.2 – OFDM basic structure. 22
Figure 2.3 – OFDM basic structure usign the IDFT and DFT Fourier transforms . . 24
Figure 2.4 – OFDM symbol with 64 sub-carriers in frequency domain. 25
Figure 2.5 – ISI effect over adjacent symbols. 26
Figure 2.6 – ISI effect over adjacent symbols with CP. 26
Figure 2.7 – ICFO of k_o subcarriers - without error(top), with error (bottom). . . . 28
Figure 2.8 – STO early and late estimation. 29
Figure 3.1 – Format of the MR-OFDM PPDU and LTF. 34
Figure 3.2 – MR-OFDM Modulator according to the IEEE802.15.4g Standard. . . . 34
Figure 3.3 – Implemented MR-OFDM receiver. 36
Figure 4.1 – First stage in the DIF computation process. 41
Figure 4.2 – Decimation in frequency butterfly. 42
Figure 4.3 – Decimation in frequency diagram. 42
Figure 4.4 – Simple In-place architecture. 43
Figure 4.5 – Address Generator. 44
Figure 4.6 – 8 point FFT DIF diagram. 45
Figure 4.7 – DIF Butterfly Architecture. 46
Figure 4.8 – Rotation of a vector. 47
Figure 4.9 – CORDIC Cell. 50
Figure 4.10–FFT/IFFT internal architecture. 52
Figure 4.11–IFFT Verification Diagram 53
Figure 4.12–FFT Verification Diagram 54
Figure 5.1 – ICFO Test High-Level Model Diagram 59
Figure 5.2 – LTF Cross-Correlation output. 59
Figure 5.3 – Probability of success of the FFT Based Correlation. 59
Figure 5.4 – Integer carrier frequency offset estimator/corrector. 60
Figure 5.5 – Architecture of the complex multiplier. 61
Figure 5.6 – Architecture of the peak searcher. 62
Figure 5.7 – ICFO Test High-Level Model Diagram 63
Figure 5.8 – ICFO Verification Results VHDL Simulation and MAtlab Model 63
Figure 5.9 – Integer carrier frequency offset estimator/corrector with correlator. . . 65
Figure 5.10–Altera's PPT Correlator Architecture. 66
Figure 5.11–Altera's PPT Correlator Adder Tree. 66

Figure 5.12–Altera's PPT Correlator LEs. 67
Figure 5.13–Power Delay Profile of the Multipath Channel. 68
Figure 5.14–LTF Cross-Correlation output with SFO. 69
Figure 5.15–Example of the computation of fine STO by Canet Method. 70
Figure 5.16–Probability of Success in Canet STO Estimation Algorithm (STO of 32 samples). 70
Figure 5.17–Probability of Success in Canet STO Estimation Algorithm (STO of 16 samples). 70
Figure 5.18–Example of the results of the square difference algorithm. 71
Figure 5.19–Example of the results of the auto correlation function algorithm. . . . 72
Figure 5.20–Probability of success of the CP based methods in a noisy channel for all OFDM Options. 73
Figure 5.21–Probability of success in estimating the STO for the Data-Aided frequency domain algorithm for all MR-OFDM Options. 74
Figure 5.22–Probability of success in estimating the STO for all MR-OFDM Options. 74
Figure 5.23–Probability of success in estimating the STO for CP based square difference under a multipath Channel. 75
Figure 5.24–Probability of success in estimating the STO for CP based ACF under a multipath channel. 75
Figure 5.25–Probability of success in estimating the STO for the phase difference in frequency domain algorithm under a multipath channel for all MR-OFDM Options. 76
Figure 5.26–Example in estimating a ICFO of 35 carriers for the BoAi Proposed Method. 77
Figure 5.27–Example of the maximum of correlations method for estimating the ICFO. 78
Figure 5.28–Probability of success in estimating the ICFO for the BoAi and maximum of correlations Proposed Methods for all OFDM Options. 79
Figure 5.29–Probability of success in estimating the ICFO for the BoAi and maximum of correlations Proposed Methods for all OFDM Options under a multipath channel. 79
Figure 5.30–Serial Architecture for the CP Based ACF algorithm 80
Figure 5.31–Serial Architecture for the CP Square Difference algorithm 81
Figure 5.32–Parallel multipliers in STO ACF architecture 81
Figure 5.33–Architecture for the Fine STO estimator in frequency domain 82
Figure 5.34–BoAi algorithm implementation 82
Figure 5.35–BoAi algorithm implementation in parallel 83
Figure 5.36–Peak searcher for the FFT based maximum of correlations algorithm . 84

Figure A.1 – ASIC design flow 97

List of Tables

Table 3.1 – Main Parameters of the MR-OFDM Mode 33
Table 4.1 – Angle values for a 16 size FFT 46
Table 4.2 – Bit representation of Counter B shifted for a 16 point FFT 46
Table 4.3 – CORDIC Modes of operation 50
Table 4.4 – Results of CORDIC Generalized equations 50
Table 4.5 – Some Functions Computed by the CORDIC 50
Table 4.6 – High-level IFFT Model/Matlab-IFFT Function Error 53
Table 4.7 – RTL-VHDL Model/High-Level IFFT Model error 53
Table 4.8 – FPGA IFFT/RTL-VHDL Model error 53
Table 4.9 – High-level FFT Model/Matlab-FFT Function Error 53
Table 4.10–RTL-VHDL Model/High-Level FFT Model error 54
Table 4.11–FPGA FFT/RTL-VHDL Model error 54
Table 4.12–IFFT RTL-VHDL Model/Matlab Model error 55
Table 4.13–IFFT FPGA Implementation/Matlab Model error 55
Table 4.14–FFT RTL-VHDL Model/Matlab Model error 55
Table 4.15–FFT FPGA Implementation/Matlab Model error 55
Table 4.16–128-point FFT Implementation Results for Altera's 5CGXFC5C6F27C7N 56
Table 4.17–Resource Utilization by Entity in the FFT/IFFT 56
Table 4.18–Resource Utilization by Entity in the OFDM Modulator 57
Table 5.1 – CFO Implementation Results for Altera's 5CGXFC5C6F27C7N 63
Table 5.2 – Resource Utilization by Entity in the FFT Based ICFO 64
Table 5.3 – CFO Implementation Resource utilization by entity in the MR-OFDM Demodulator 64
Table 5.4 – Resource Utilization by he ICFO without the FFT 64
Table 5.5 – Resource comparison between complex multiplier and correlator 67
Table 5.6 – Resource Utilization by Entity in the ICFO architectures 85
Table 5.7 – Resource Utilization by Entity in the STO architectures 86
Table 5.8 – Resource Utilization of the ICFO architectures 86
Table B.1 – Logical synthesis results for 65nm CMOS for the FFT Architecture . . . 100
Table B.2 – Logical synthesis results for 65nm CMOS for the ICFO Architecture . . 101
Table C.1 – LTF for MR-OFDM Option 1 102
Table C.2 – LTF for MR-OFDM Option 2 103
Table C.3 – LTF for MR-OFDM Option 3 103

Table C.4 – LTF for MR-OFDM Option 4 103
Table C.5 – Modulation Schemes for every MCS and OFDM Option 103

Acronyms

ACI Adjacent channel interference

ALM Adaptive Logic Module

ASIC Application Specific Integrated Circuit

BPSK Binary Phase Shift Keying

CFO Carrier Frequency Offset

CORDIC COordinate Rotation DIgital Computer

CP Cyclic Prefix

DC Direct Current

DFT Discrete Fourier Transform

DSSS direct sequence spread spectrum

FAN Field Area Network

FFO Fractional Frequency Offset

FFT Fast Fourier Transform

FPGA Field Programmable Gate Array

MR-FSK Multi-Rate and Multi-Regional Frequency Shift Keying

HAN Home Area Network

HDL Hardware Description Language

ICFO Integer Carrier Frequency Offset

ICI Inter Carrier Interference

IDFT Inverse Discrete Fourier Transform

IEEE Institute of Electrical and Electronics Engineers

IFFT Inverse Fast Fourier Transform

IFO Integer Frequency Offset

IoT Internet of Things

ISI Intersymbol Interference

LE Logic Element

LR-WPANs Low-Rate Wireless Personal Area Networks

LTF Long Training Field

MAC Media Access Control

MCS Modulation and Coding Scheme scheme

MDSSS multiplexed direct sequence spread spectrum

MR-OFDM Multi-rate and Multi-regional Orthogonal Frequency Division Multiplexing

OFDM Orthogonal Frequency Division Multiplexing

PFA Prime Factor Algorithm

PHR PHY Header

PHY Physical Layer

PPDU Physical Packcage Data Unit

PSDU Packet Service Data Unit scheme

QAM Quadrature Amplitude Modulation

MR-O-QPSK Multi-Rate and Multi-Regional Offset Quadrature Phase-Shift Keying

QPSK Quadrature Phase-Shift Keying

RTL Register Transfer Level

SCO Sampling Clock Offset

SHR Synchronization Header

STF Short Training Field

STO Symbol Time Offset

SUN Smart Ubiquitous Network

WFTA Winograd's Fourier Transform Algorithm

Wi-SUN Wireless Smart Ubiquitous Network

Contents

1 INTRODUCTION 18

2 ORTHOGONAL FREQUENCY-DIVISION MULTIPLEXING 21

2.1 Synchronizations Errors in OFDM 27

2.1.1 Carrier Frequency Offset 27

2.1.2 Symbol Time Offset 29

3 IEEE802.15.4G STANDARD 31

3.1 IEEE802.15.4g Standard 32

3.1.1 IEEE802.15.4g MR-OFDM 33

4 THE DISCRETE FOURIER TRANSFORM AND FAST FOURIER TRANSFORM 38

4.1 FFT Cooley-Tukey Algorithm 39

4.2 Architecture of the Fast Fourier Transform 42

4.2.1 Proposed Architecture 42

4.2.1.1 FFT Implementation 43

4.2.1.1.1 Hardware Considerations 43

4.2.1.2 CORDIC 47

4.2.2 IFFT Shifter 51

4.2.3 Variable Length FFT/IFFT 51

4.3 Implementation Results 51

4.3.1 FPGA Prototyping 56

5 INTEGER CARRIER FREQUENCY OFFSET ARCHITECTURE . . 58

5.1 ICFO FFT-based Architecture 60

5.1.1 Complex Multiplier 61

5.1.2 FFT 61

5.1.3 Peak Searcher 61

5.1.4 Symbol Shifter 61

5.2 Implementation Results 62

5.2.1 FPGA Prototyping 62

5.2.2 Synthesis Analysis 63

5.3 ICFO and STO 67

5.3.0.1 Canet Fine STO estimation 69

5.3.0.2 Non-Data-Aided STO Estimation . . . 71
5.3.0.3 Fine STO estimation in frequency domain . . . 72
5.3.0.4 ICFO Estimation in the frequency domain with immunity to STO . . . 76
5.3.0.5 FFT Based ICFO Estimation with immunity to STO . . . 77
5.4 STO and ICFO hardware implementations . . . 78
5.4.1 CP Based Architectures . . . 78
5.4.2 Fine STO estimation in frequency domain Architecture . . . 81
5.4.3 BoAi Algorithm Architecture . . . 82
5.4.4 Maximum of correlations Algorithm . . . 83

6 CONCLUSIONS . . . 88
6.1 Future Work . . . 89

BIBLIOGRAPHY . . . 90

ANNEX 95

ANNEX A – ASIC DESIGN FLOW . . . 96
A.1 Front-end . . . 97
A.2 Back-end . . . 98

ANNEX B – ASIC RESULTS . . . 100
B.1 Synthesys Results . . . 100

ANNEX C – IEEE802.15.4G PARAMETERS . . . 102
C.1 Long training Field . . . 102

1 Introduction

The network of interconnected objects also known as the Internet of Things (IoT) has been a subject of many studies in recent years. This new kind of networks is composed of sensors and actuators that gathers information and interact with the physical world. The needs of IoT impose several challenges and carry with them new and specialized communication protocols. Wi-SUN Alliance is among the many communication groups working to develop protocols and standards to overcome the challenges presented by the IoT. It promotes the adoption of open industry standards for Wireless Smart Ubiquitous Network (Wi-SUN).

The Institute of Electrical and Electronics Engineers (IEEE) released in 2012 the IEEE802.15.4g standard [1], IEEE802.15.4g standard was adopted by the Wi-SUN alliance as the physical layer of Field Area Network FAN) and Home Area Network (HAN) Profiles of Wi-SUN. The standard was initially intended to Smart Utility Networks, but recently, due to its characteristics, its use was extended to IoT applications. Thus, the acronym now stands for Smart Ubiquitous Networks (SUN). The IEEE802.15.4g standard is an amendment to IEEE802.15.4, it was designed to cope with current and future specific communications needs for SUN and has three operation modes, including the Multi-Rate Multi-Regional Orthogonal Frequency Division Multiplexing (MR-OFDM) mode. SUN devices are intended for wireless low power applications; therefore, implementations that maximizes performance while attaining low power consumption are of main concern. This is the main objective of this work, to find simple, low-cost hardware implementations that achieve low area and low power consumption and implement those solutions for an Application Specific Integrated Circuit (ASIC) in compliance to the IEEE802.15.4g standard.

Specifically, to find an optimal implementation of the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) that achieves the requirements of the MR-OFDM mode of the IEEE802.15.4g standard. The Fast Fourier Transform (FFT) is a well known algorithm that reduces the computing complexity of the DFT algorithm. Several methods have been proposed to implement the FFT efficiently in hardware, high-throughput high performance parallel pipelined FFTs are extensively used: [2][3] implement the FFT by means of radix-2 butterflies, higher radixes [4][5], mixed-radixes [6] and split-radix [7] implementations reduces the arithmetic complexity and improve performance at the cost of more hardware consumption.

On the other hand, architectures based on a single butterfly unit [8][9][10][11],

require less hardware at the expenses of more computation time. These architectures try to optimize memory accesses while at the same time reduce the amount of memory used. Another approaches try to minimize hardware resources and power consumption by replacing the complex multiplier with the COordinate Rotation DIgital Computer (CORDIC) [12][13][14]. The presented architecture will gather the advantages of the previous implementations methods, while achieving the hardware and technical requirements of the IEEE802.15.4g MR-OFDM mode.

Another important issue concerning the implementation of OFDM modems is that of the optimal and robust implementation methods for the synchronization and frequency error correction at the receiver side. For this reason, the second part of this work is to implement an Integer Carrier Frequency Offset(ICFO) estimator and corrector for the IEEE802.15.4g MR-OFDM mode. Many methods have been proposed to estimate the ICFO, in [15][16][17] correlations using known sequences at the receiver are employed to estimate the ICFO,[18][19] uses pilot assisted method and correlation of consecutive symbols to achieve the estimation, [20] employs maximum likelihood estimation also based on pilots to solve the problem. Generally, ICFO estimation is based on cross-correlation or auto-correlation, an operation that consumes a considerable number of resources in hardware. The methods found to estimate the ICFO show no hardware implementation; the research on the subject has been limited to only simulations. Moreover, ICFO performance evaluation is carried out under the assumption that previous estimations are perfect.

This work will show a simple ICFO estimator method based on the DFT usage, that is believed to reduce the hardware complexity. The simplification is accomplished by smart reuse of the available hardware, involving the FFT, that makes possible to implement the correlation operation with fewer resources than the typical implementation. The proposed estimation/corrector method can be adopted in other OFDM-based systems as well, since it exploits the OFDM structure to tackle the problem. Additionally, the study of the estimation in a more rigorous environment, where Symbol Time Offset (STO) is not always correctly estimated is also performed.

The methodology followed in the implementation of the architectures is as follows: first, a high-level model is developed, it simulates numerically the algorithm or method that is to be implemented. This high-level model serves as a reference for the Register Transfer Level (RTL) behavioral description. The RTL behavioral model, written in a Hardware Description Language (HDL), is a description of the architecture; it describes the behavior and structure of a digital circuit. This RTL behavioral model is then compared against the golden model for bugs in the description. Once the RTL behavioral model is bug-free, the physical verification process starts. The RTL is synthesized in a

logic synthesis tool that checks if the behavioral description can be turned into a digital hardware. Finally, once the RTL behavioral description is bug-free, a Field Programmable Gate Array (FPGA) is programmed with the design; then it is excited with the same data as the Golden and RTL behavioral model, and its outputs compared. After these steps, the (FPGA) prototype goes through the debugging process.

This thesis is organized as follows:

- In Chapter 1, a theoretical background of OFDM is presented, deriving the necessity of the IDFT/DFT and showing some of the OFDM synchronization issues.
- Chapter 2 is an overview of the IEEE802.15.4g standard, focusing on the technical aspects of MR-OFDM mode, the modem architecture, as well as the frame structure, are described.
- Chapter 3 looks into the FFT algorithm; the well known radix-2 Fast algorithm is derived from the original DFT equation. Additionally, the proposed FFT architecture is shown along with the methods implemented that simplify the hardware, as well as the results of its FPGA implementation.
- Chapter 4 presents an FFT-based ICFO estimation architecture. FPGA implementation results are also shown. This chapter also explores the problem of the ICFO estimation in conjunction with the STO.
- Chapter 5 Presents an overview of the ASIC design process. A brief description of the stages in the process are reviewed. Finally, the ASIC synthesis results are shown.

Finally, conclusions and future work are presented.

2 Orthogonal Frequency-Division Multiplexing

In order to overcome the limitations imposed by the complexity of the equalizer in the receiver, due to high data rate transmission in single carrier modulations, a multi-carrier transmission scheme approach can be adopted. In a multi-carrier transmission scheme, as the Frequency Division Multiplexing (FDM), a high data rate stream or wideband signal is divided into several low rate or narrowband parallel streams, each one called of sub-carriers. The division of a wideband signal into parallel low rate signals allows the approximation of a frequency selective wideband channel by multiple non-frequency selective narrowband channels, thus, reducing the complexity of the equalizer to a single tap-equalizer per sub-carrier [21].

Another type of multicarrier system that has some advantages over the multicarrier systems is called Orthogonal Frequency Division Multiplexing (OFDM). In OFDM, carriers are orthogonal among each other, which saves bandwidth since its spectrum overlap, as shown in Figure 2.1. Another advantage of OFDM resides in its implementations. OFDM modems can be realized through the Inverse Discrete Fourier Transform IDFT and Discrete Fourier Transforms DFT, which in turn can be efficiently implemented by the Fast Fourier Transform FFT, making OFDM systems implementation less-complex. The use of DFT and IDFT to implement the modulation and demodulation in multicarrier systems avoids the use of banks of subcarrier oscillators and coherent demodulators, which makes FDM systems excessively expensive and more complex as the number of subcarriers increases [22] [21]. Figure 2.2 shows the structure of an OFDM modem.

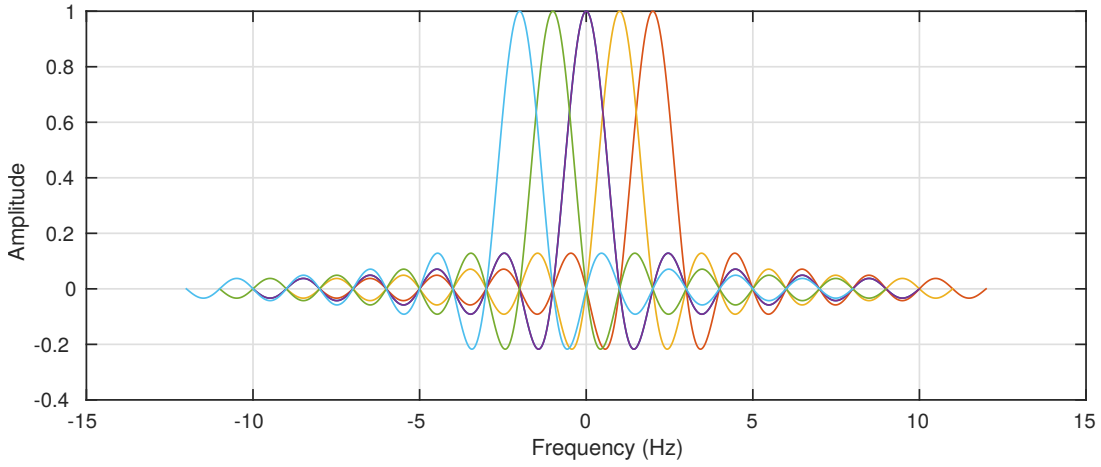

Figure 2.1 – Spectrum of an OFDM communication scheme.

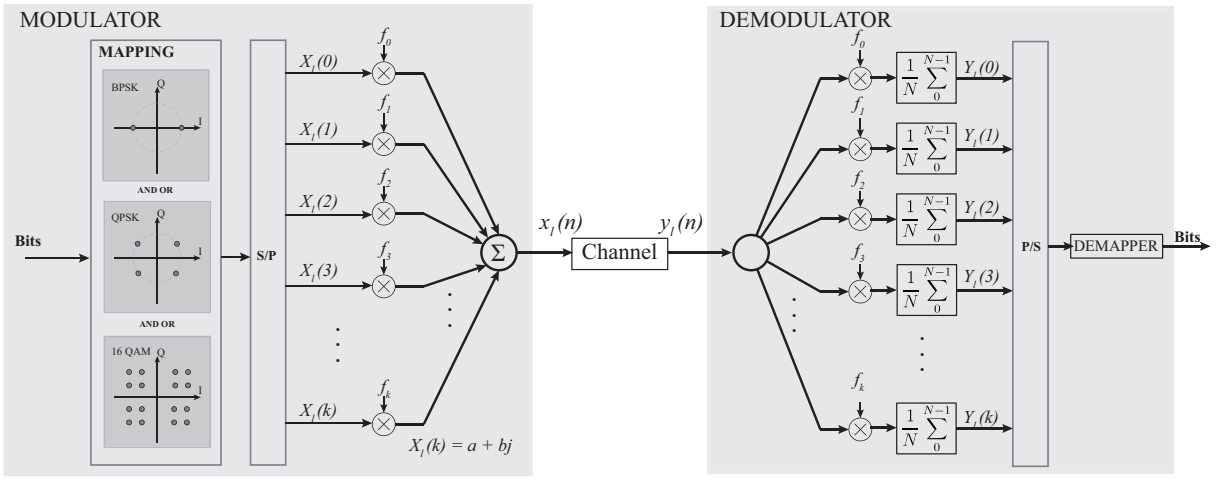

Figure 2.2 – OFDM basic structure.

An OFDM symbol can be described by:

$$x_l(t) = \sum_{k=0}^{N-1} X_l(k) e^{j2\pi f_k t}, \quad 0 < t < T_s \tag{2.1}$$

Where, $X_l(k)$ represents the *l-th* symbol at the *k-th* sub-carrier within an OFDM symbol of duration T_s for $k = 0, 1, \ldots, N-1$. As shown in Figure 2.2, in the OFDM transmission, the information data bits are first mapped into a single carrier symbol, e.g., BPSK, M-QAM or QPSK symbols, and then converted into N parallel streams. This parallel stream of symbols is then carried out by N orthogonal sub-carriers, each one with a frequency f_k. The orthogonal sub-carriers spectra overlap in frequency domain, achieving high bandwidth efficiency [21].

After sampling, the discrete time OFDM symbol can be described as in (2.2), with $t = nT_s/N$ and $f_k = k/T_s$

$$x_l(n) = \sum_{k=0}^{N-1} X_l(k) e^{j2\pi kn/N}, \quad k = 0, 1, 2 \ldots N-1 \tag{2.2}$$

Since we assume that the OFDM carriers are orthogonal, two sub-carriers, ϕ_k and ϕ_i must meet the following condition in continuous time, the function $\delta[i-k]$ is defined as Kronecker delta:

$$\begin{aligned}\frac{1}{T_s}\int_0^{T_s}\phi_i(t)\phi_k^*(t)dt &= \frac{1}{T_s}\int_0^{T_s}e^{j2\pi f_i t}e^{-j2\pi f_k t}dt\\ &= \frac{1}{T_s}\int_0^{T_s}e^{j2\pi(f_i-f_k)t}dt\\ &= \frac{1}{T_s}\int_0^{T_s}e^{j2\pi\frac{(i-k)}{T_s}}dt\\ &= \delta[i-k]. \end{aligned} \tag{2.3}$$

The discrete form of the orthogonality property with sampling intervals of $t = nT_s/N$ and $f_k = K/T_s$ can be written as:

$$\begin{aligned}\frac{1}{N}\sum_{k=0}^{N-1}e^{j2\pi\frac{i}{Ts}\frac{nTs}{N}}e^{-j2\pi\frac{k}{Ts}\frac{nTs}{N}} &= \frac{1}{N}\sum_{k=0}^{N-1}e^{j2\pi\frac{n(i-k)}{N}}\\ &= \delta[i-k]\end{aligned} \tag{2.4}$$

At the receiver the signal can be recovered using this orthogonality property of (2.3). Without taking into account the channel effects, the received signal $y_l(t)$ can be written as in (2.5)

$$y_l(t) = \sum_{k=0}^{N-1}X_l(k)e^{j2\pi f_k t}, \quad 0 < t < T_s \tag{2.5}$$

By applying the orthogonality property on the received OFDM symbol, this yields:

$$\begin{aligned}Y_l(k) &= \frac{1}{T_s}\int_0^{T_s}y_l(t)e^{-j2\pi f_k t}dt \qquad (2.6)\\ &= \frac{1}{T_s}\int_0^{T_s}\sum_{i=0}^{N-1}X_l(i)e^{j2\pi f_i t}e^{-j2\pi f_k t}dt\\ &= \sum_{i=0}^{N-1}X_l(i)\frac{1}{T_s}\int_0^{T_s}e^{j2\pi(f_i-f_k)t}dt\\ &= \sum_{i=0}^{N-1}X_l(i)\delta[i-k]\\ &= X_l(k).\end{aligned} \tag{2.7}$$

The orthogonality property in its discrete form yields:

$$
\begin{aligned}
Y_l(k) &= \sum_{n=0}^{N-1} y_l(n) e^{-j2\pi \frac{kn}{N}} \qquad (2.8)\\
&= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{i=0}^{N-1} X_l(i) e^{j2\pi \frac{in}{N}} e^{-j2\pi \frac{kn}{N}} \\
&= \sum_{i=0}^{N-1} X_l(i) \frac{1}{N} \sum_{n=0}^{N-1} e^{j2\pi \frac{in}{N}} e^{-j2\pi \frac{kn}{N}} \\
&= \sum_{i=0}^{N-1} X_l(i) \frac{1}{N} \sum_{n=0}^{N-1} e^{j2\pi \frac{(i-k)n}{N}} \\
&= \sum_{i=0}^{N-1} X_l(i) \delta[i-k] \\
&= X_l(k) \qquad (2.9)
\end{aligned}
$$

Equations (2.2) and (2.8) are known as the IDFT of $X_l(k)$ and DFT of $y_l(n)$, which means that the OFDM modulation/demodulation can be implemented as a IDFT/DFT pair as shown in Figure 2.3. As mentioned before the IDFT/DFT pair can be efficiently computed by the FFT algorithm.

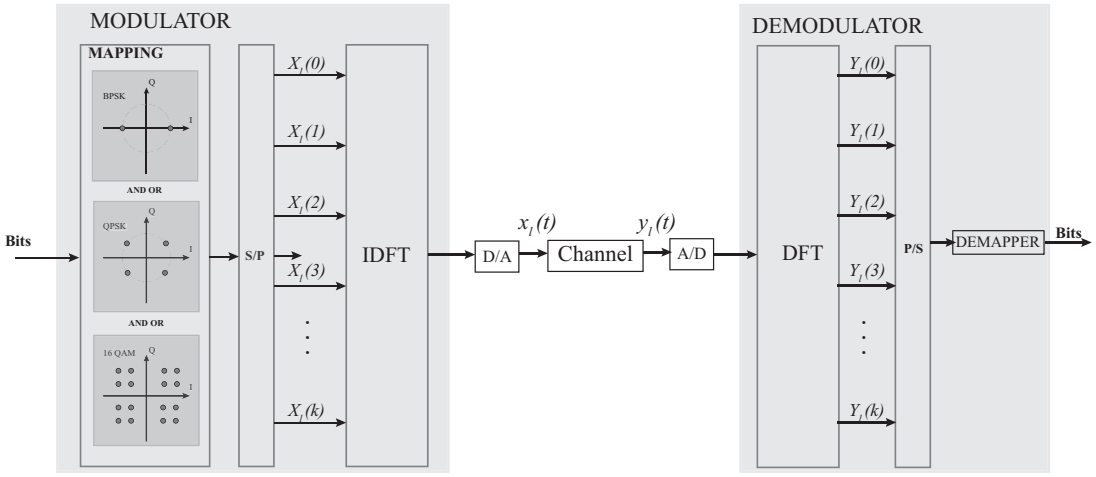

Figure 2.3 – OFDM basic structure usign the IDFT and DFT Fourier transforms

OFDM signals often show bandwidth spillage or leakage, which causes adjacent channel interference (ACI), since sub-carriers are time-limited and no band limited. Null carriers are added at the adjacencies of the OFDM symbol to avoid ACI, these null also protect the OFDM signal from leakage from other adjacent systems. In general, another null sub-carrier is the Direct Current (DC) sub-carrier, this null sub-carrier corresponds to the zero frequency sub-carrier. Figure 2.4 shows an OFDM symbol with 64 sub-carriers in frequency domain, DC, guard band and data carriers are also shown in the figure.

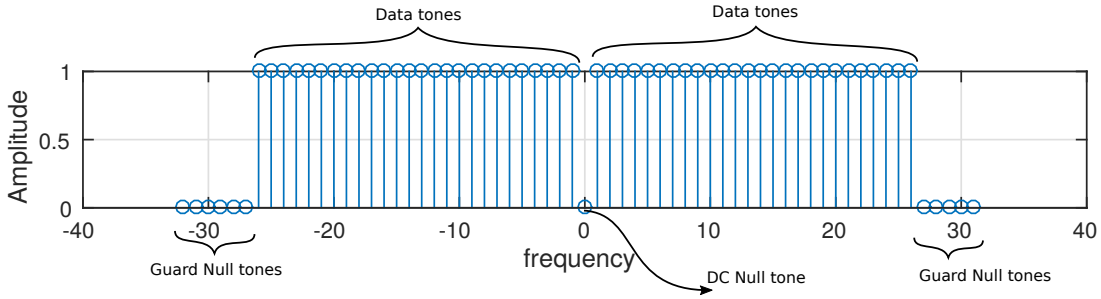

Figure 2.4 – OFDM symbol with 64 sub-carriers in frequency domain.

Another modification that is performed over the OFDM symbol in time domain, is that a copy of the last G samples of the OFDM symbol is prepended to itself, to prevent Inter Symbol Interference (ISI). The prepended part is known as the Cyclic Prefix (CP). ISI is originated by the multi-path fading channel, that causes that a delayed, attenuated and phase shifted copy of the previously sent signal to interfere with the arriving signal. The receiver signal can be modeled in time domain as:

$$y(n) = x(n) * h(n) + z(n) = \sum_{m=0}^{\infty} x(m)h(n-m) + z(n). \tag{2.10}$$

where, $y(n)$ is the received signal, $x(n)$ the sent signal, and $h(n)$ the impulse response of the multi-path fading channel $z(n)$. The effect of multi-path channel is illustrated in Figure 2.5, which shows that the delayed received signal overlaps with the current received symbol and interferes with the next symbol. The received signal in frequency domain is represented by:

$$\begin{aligned}
Y(k) &= \sum_{n=0}^{N-1} [x(n) * h(n) + z(n)] e^{\frac{-j2\pi kn}{N}} && (2.11)\\
&= \sum_{n=0}^{N-1} [\sum_{m=0}^{N-1} x(n)h(n-m) + z(n)] e^{\frac{-j2\pi kn}{N}} \\
&= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m)h(n-m) e^{\frac{-j2\pi kn}{N}} + Z(k) \\
&= \sum_{m=0}^{N-1} x(m) \underbrace{\sum_{n=0}^{N-1} h(n-m) e^{\frac{-j2\pi kn}{N}}}_{\text{DFT circular shift property}} + Z(k) && (2.12)\\
&= \sum_{m=0}^{N-1} x(m) H(k) e^{\frac{-j2\pi km}{N}} + Z(k) \\
&= H(k) \sum_{m=0}^{N-1} x(m) e^{\frac{-j2\pi km}{N}} + Z(k) \\
&= X(k)H(k) + Z(k) && (2.13)
\end{aligned}$$

According to (2.13), the channel compensation can be accomplished by simply dividing the received symbol in frequency domain by the channel response $X(k) = Y(k)/H(k)$, which represents a one tap equalizer for a particular index k. The channel response can be estimated using a data-aided method, in which known training sequences are transmitted in advance, the estimation process at the receiver then correlates the received and known sequences in the receiver, which yields the channel response. On the other hand, blind estimation or non-data aided estimation is also employed, in blind approaches the channel response is accomplished by statistics of the received signals [23]. Also $Y(k) = X(k)H(k)$ is true only when $y(n) = x(n) \circledast y(n)$, where $\circledast$ denotes circular convolution. This makes possible to apply the DFT circular shift property in (2.12), the circular convolution holds true only when the CP is prepended to the OFDM symbol [21].

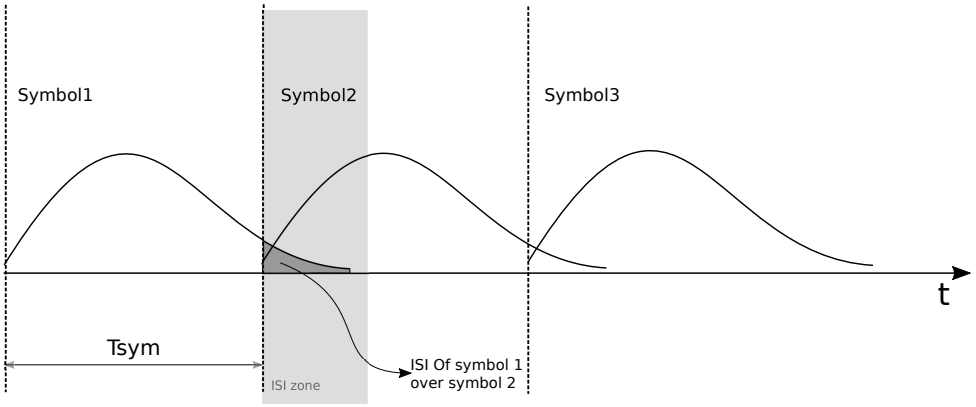

Figure 2.5 – ISI effect over adjacent symbols.

The CP appended to the beginning of the OFDM symbol in time domain safeguards the OFDM symbol from being affected by the multi-path fading channel effect over the previously transmitted symbol, as long as the channel delay is shorter than that of the CP duration. Figure 2.6 illustrates the effect of ISI over the OFDM symbol when the CP is added.

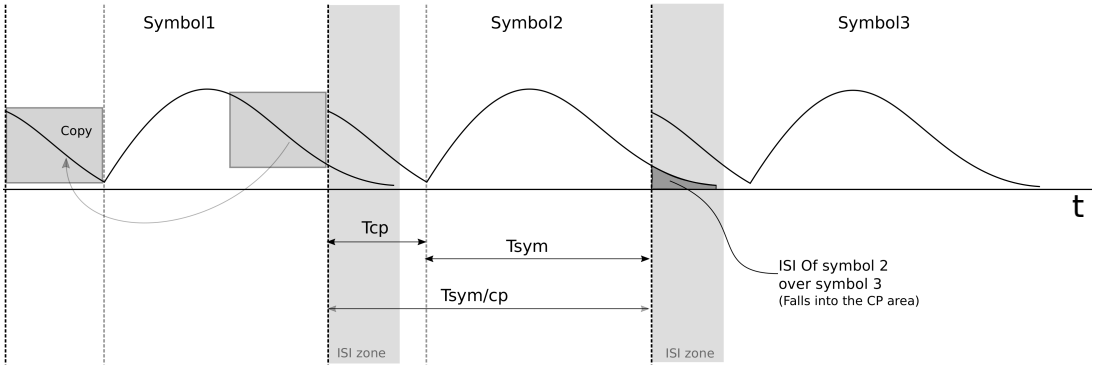

Figure 2.6 – ISI effect over adjacent symbols with CP.

2.1 Synchronizations Errors in OFDM

Although OFDM is widely used due to its robustness against multipath fading channel as well as the reduced complexity single-tap equalizer per subcarrier [23], it has large Peak to Average Ratio (PAPR) [24] and it is very sensitive to the receiver synchronization errors [25]. Errors such as Symbol Time Offset (STO), i.e., finding the start of the OFDM symbol unknown at the receiver, and Carrier Frequency Offset (CFO) caused mainly by the frequency mismatch between local oscillators at the receiver and transmitter. Synchronization errors as STO and ICFO can cause Inter-Symbol Interference (ISI) and Inter-Carrier Interference (ICI) that diffcults the proper recovery of the OFDM symbols and degrades system performance. Thus, the receiver must estimate and compensate those errors to properly recover the symbols. The effects of those errors over the OFDM symbols are analyzed as follows.

2.1.1 Carrier Frequency Offset

The tolerance of local and remote oscillators causes a difference between the transmitter and receiver carrier frequency. This difference in frequencies, known as carrier frequency offset (CFO), has some detrimental effect on sub-carriers, hindering the symbols recovery. It can be divided into two parts regarding the sub-carrier space, a fractional frequency offset, which is a fraction of the sub-carrier spacing, and an ICFO which occurs in multiples of the sub-carriers distance.

The fractional part causes Inter-Carrier Interference (ICI) due to spectral leakage sub-carriers. It is typically corrected in two steps, first, a coarse estimation/correction is performed followed by a fine frequency estimation/correction. On the other hand, the integer part causes a circular shift of the sub-carrier indexes in the frequency domain. It is usually estimated in frequency domain, and its correction can be performed either in time or frequency domain.

Assuming perfect symbol synchronization, i.e., zero STO (the frame start is perfectly estimated), and without considering the impairments introduced by the channel, the transmitted symbol in frequency domain is described by (2.14), while received symbol in time domain can be described by (2.15).

$$X_l(k) = \sum_{n=0}^{N-1} x_l(n) e^{\frac{-j2\pi kn}{N}}, \qquad k = 0, 1, ..., N-1 \tag{2.14}$$

$$y_l(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_l(k+\beta) e^{\frac{j2\pi kn}{N}}, \qquad n = 0, 1, ..., N-1 \tag{2.15}$$

where $\beta = \epsilon + k_o$ is the frequency offset, divided in ϵ, the fractional frequency offset (FFO) and k_o the integer frequency offset (IFO). $X_p(k)$ the transmitted symbol in frequency domain with k sub-carriers, $x_p(n)$ in time domain and $y_p(n)$ the received signal in time domain. Assuming a fractional frequency offset of zero ($\beta = k_o$), the term $x_l(k+\beta)$ can be written as in (2.16)

$$x_l(k+k_o) = \sum_{b=0}^{N-1} x_l(b) e^{\frac{-j2\pi(k+k_o)b}{N}}, \qquad k+k_o = 0,1,...,N-1. \tag{2.16}$$

Therefore, (2.15) can be written as:

$$y_l(n) = \frac{1}{N} \sum_{k=0}^{N-1} \Big\{ \sum_{b=0}^{N-1} x_l(b) e^{\frac{-j2\pi(k+k_o)b}{N}} \Big\} e^{\frac{j2\pi kn}{N}}, \qquad n = 0,1,...,N-1. \tag{2.17}$$

$$y_l(n) = \frac{1}{N} \sum_{b=0}^{N-1} x_l(b) e^{\frac{-j2\pi k_o b}{N}} \sum_{k=0}^{N-1} e^{\frac{j2\pi k(n-b)}{N}}, \qquad n = 0,1,...,N-1. \tag{2.18}$$

If the index $b = n$, then the $l-th$ symbol in time domain can be expressed as (2.19)

$$y_l(n) = x_l(n) e^{\frac{-j2\pi k_o n}{N}}, \qquad n = 0,1,...,N-1. \tag{2.19}$$

From (2.19) can be concluded that, the received symbol has the same value than that of the transmitted symbol with its phase rotated by $-j2\pi k_o n/N$. As the fractional part is equal to zero, the IFO effect over the sub-carriers in the frequency domain is a shift of carriers, as shown in the illustrative example of Figure 2.7.

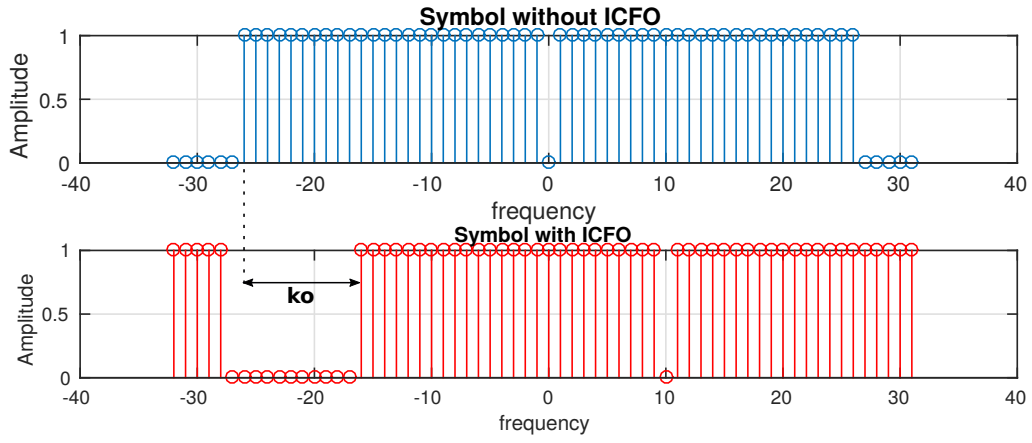

Figure 2.7 – ICFO of k_o subcarriers - without error(top), with error (bottom).

2.1.2 Symbol Time Offset

Another impairment regarding synchronization is the symbol time synchronization error. Timing synchronization algorithms attempt to find the start of the OFDM symbol and to correct what is called the STO, defined as the difference between the ideal symbol start and the estimated one. The STO must be zero so that the result of the FFT computation matches that of the transmitted symbol in the frequency domain.

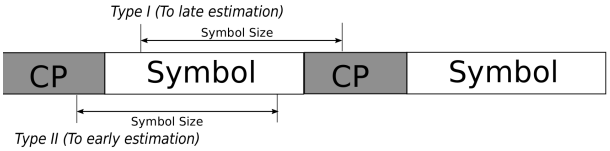

Figure 2.8 – STO early and late estimation.

A deviation from the ideal STO can be found after the estimation process as shown in Figure 2.8. STO can be estimated to early or too late, regarding the ideal case in which the exact point is expected to be. A late estimation (type I, as shown in Figure 2.8), which takes the starting point after the ideal case, generates a time window that is shifted onto the next symbol, that is, samples from the current symbol are lost, and samples from the next symbol are taken. In this case, ICI is found, and the orthogonality between sub-carriers is lost [21].

The second type of offset found is the one in which the starting point is estimated to early (type II, in Figure 2.8), that is before the ideal starting point. In this case, the first sample of the symbol falls inside the symbol CP, and the last samples are lost, no samples from the next symbols are taken. In this case, the orthogonality between sub-carriers is preserved, and only a phase offset proportional to the STO, and the carrier index is shown in the post-FFT signal [21].

If the STO falls inside the CP of the current symbol, and without considering the impairments introduced by the channel, the received signal can be expressed as:

$$S_l(k) = \sum_{n=0}^{N-1} x_l(n+\alpha) e^{\frac{-j2\pi nk}{N}}, \qquad k = 0, 1, ..., N-1 \tag{2.20}$$

where the shifted symbol then can be rewritten as:

$$x_l(n+\alpha) = \sum_{b=0}^{N-1} x_l(b) e^{\frac{j2\pi(n+\alpha)b}{N}}, \qquad b = 0, 1, ..., N-1 \tag{2.21}$$

Substituting (2.21) in 2.20 we obtain:

$$S_l(k) = \sum_{n=0}^{N-1} \Big\{ \sum_{b=0}^{N-1} X_l(b) e^{\frac{j2\pi(n+\alpha)b}{N}} \Big\} e^{\frac{-j2\pi nk}{N}} \tag{2.22}$$

Rearranging (2.22),

$$S_l(k) = \sum_{b=0}^{N-1} X_l(b) e^{\frac{j2\pi\alpha b}{N}} \sum_{n=0}^{N-1} e^{\frac{j2\pi n(b-k)}{N}} \tag{2.23}$$

If $b = k$ then, (2.23) becomes

$$S_l(k) = X_l(b) e^{\frac{j2\pi\alpha b}{N}} \tag{2.24}$$

That is, at the receiver, the same carrier in the frequency domain of the transmitter is modified in phase by a factor equal to $\frac{j2\pi\alpha b}{N}$, i.e. , STO only introduces a phase shift in frequency domain if the STO is smaller than the CP size and of type II.

STO of type I cannot be reversed because the start frame decision is already made and samples before the symbol start are discarded. On the contrary, type II STO errors can be reversed, as this kind of error only shows changes in signal phase. However, time offset greater than certain values degrade the signal significantly, if samples of the CP are affected by previous symbols due to the multipath fading channel [26], even if it falls inside the CP.

There are several methods for estimating and correcting timing errors and frequency errors based on repetitive structures inherent in the OFDM symbol (as the CP case). These are known as non-data-aided methods or blind methods, these algorithms save bandwidth, since no structure must be sent to estimate the errors. Frame-oriented systems benefit from this type of algorithms since the estimation is performed continuously. A second kind of estimation based on a known sequence sent in the Physical Package Data Unit (PPDU) can be employed. This kind of algorithms, known as data-aided, consumes more bandwidth but prove to be more robust than the non-data aided methods. Packet-based system benefit from this kind of estimation. These algorithms will be reviewed in the next chapters.

3 IEEE802.15.4g standard

IoT is a new paradigm in which everyday objects are connected to the Internet. In this evolution of the Internet, everyday objects are granted smart capabilities, allowing them to sense environmental changes, send and receive data and perform tasks. This new network of smart objects opens up a myriad of applications: smart cities, that employ adaptive lighting to optimize energy consumption; smart parking that notifies of available parking spaces; warnings of climate changes. Also in healthcare, for remote monitoring of patients and monitor and tracking of vital signs; in smart utility networks and smart metering for monitoring of water, oil and gas levels in storage tanks; smart grids for monitoring and management of the electrical system.

Since the range of applications is extensive, and smart devices are composed of several complex components, major vendors of specific areas are developing their solutions isolated. Thus, the success of this new technology relies to a large extent on standardization, that will guarantee interoperability, portability, and manageability between devices from different vendors.

An essential feature of the IoT device is its ability to communicate with other devices. The data exchange allows us to make decisions based on status or environmental changes, for example. Several communications protocols are employed for the IoT; its main features are low-data rates with low power and low fabrication costs. Some of the communications protocols being used in IoT devices are:

- *ZigBee*: It is based on the IEEE802.15.4 standard, a standard that defines the operation of Low-Rate Wireless Personal Area Networks (LR-WPANs). It specifies the requirements for a low-cost low-power mesh network; it is designed to carry small amounts of data over a short distance while consuming very little power. ZigBee operates on the 2.4 GHz band as well as in the 800-900 sub-GHz bands.

- *LoRaWan (Long Range Wide Area Network)*: Developed for long distance networks for national, regional or global areas. Composed mainly of wireless devices powered by batteries. It features bidirectional communications as well as low power consumption, some of the IoT requirements.

- *Sigfox*: The basic idea of this technology is to have base stations distributed across some small area and simple devices that can exchange messages with them. A Sigfox message has up to 12-bytes payload, and its frame will use 26 bytes in total. It relies

on the Ultra Narrow Band modulation technology to exchange messages over the air. Devices requirements are low-power and low-cost.

- *WI-SUN*: Supported by the Wi-SUN Alliance, this technology features low data rates, low power, short-range communication with very low implementation complexity. The Wi-SUN alliance defines the Wi-SUN as a technology based on the IEEE 802.15.4g standard, an amendment of the IEEE802.15.4 standard. Two profiles are defined by the Wi-SUN Alliance, FAN and HAN networks that support both star and mesh topologies. The communication range is usually from 10 to 75m with a low data rate of tens of kbps up to 250 kbps [27].

The IEEE 802.15 Working Group for Wireless Specialty Networks developed the 15.4g standard, featuring low-rate wireless connectivity with low power consumption and short-range communication. The IEEE 802.15.4g standard defines Physical (PHY) and Media Access Control (MAC) layers requirements for Low-Rate Wireless Personal Area Networks (LR-WPANs). The next section describes the IEEE802.15.4g standard and some of its features.

3.1 IEEE802.15.4g Standard

The IEEE802.15.4g standard is an amendment to the IEEE802.15.4. It defines alternate PHYs in addition to those in the IEEE802.15.4 as well as modulations, data rates, frequency bands, and other technical properties.

Three PHYs are defined in the IEEE802.15.4g standard: 1) the Multi-rate and Multi-Regional Frequency Shift Keying (MR-FSK) PHY, with data rates ranging from 50 kbps to 400 kbps provides good transmit power efficiency due to the constant envelope of the transmit signal [1], it shows low implementation complexity and supports various frequency bands including Europe, Canada, U.S, Korea and Worldwide; 2) The Multi-Rate and Multi-Regional Offset Quadrature Phase-Shift Keying (MR-O-QPSK) PHY shows similar features to those of the IEEE 802.15.4-2011 MR-O-QPSK PHY, thus, it guarantees interoperability between previously developed devices, it supports Direct Sequence Spread Spectrum (DSSS) and Multiplexed Direct Sequence Spread Spectrum (MDSSS). The Multi-Rate and Multi-Regional Orthogonal Frequency Division Multiplexing (OFDM) PHY, which provides the highest data rates of the three PHYs at the cost of a more complex structure and implementation, it supports data rates ranging from 50 kbps to 800 kbps [1]. Since the current focus of this work is the development of block concerning the MR-OFDM PHY, a more detailed description will be presented in the next section.

3.1.1 IEEE802.15.4g MR-OFDM

An OFDM modulator can be implemented as an N-point IDFT, which converts OFDM symbols from the frequency domain to time domain, while the demodulator can be performed by a DFT, where each block of N received samples are converted back to the frequency domain. It is well known that the implementation of IDFT/DFT consumes many resources, making mandatory the use of the Fast Fourier Transform (FFT) algorithms, which is one of the addressed topics in this work.

The MR-OFDM mode as specified in [1] supports BPSK, QPSK and 16-QAM modulations, depending on the Modulation and Coding Scheme (MCS) chosen. Channel encoding is mandatory with a convolutional encoder of coding rate $R = 1/2$, with octal generator polynomials $g_0 = 133_8$ and $g_1 = 171_8$. A rate of $R = 3/4$ can be achieved by puncturing. With DFT sizes of 128, 64, 32 and 16, the data rates for MR-OFDM ranges from 50 kbps to 800 kbps. Table 3.1 shows a summary of the main parameters for the OFDM mode.

Table 3.1 – Main Parameters of the MR-OFDM Mode

Parameter	**Option 1**	**Option 2**	**Option 3**	**Option 4**	**Unit**
Sampling Rate	$1.333\overline{3}$	$0.666\overline{6}$	$0.333\overline{3}$	$0.166\overline{6}$	MSamples/sec
DFT Size	128	64	32	16	–
Tone Spacing	$10.416\overline{6}$	$10.416\overline{6}$	$10.416\overline{6}$	$10.416\overline{6}$	kHz
FFT Duration	96	96	96	96	μs
GI Length	24	24	24	24	μs
Symbol Duration	120	120	120	120	μs
Symbol rate	8.333	8.333	8.333	8.333	kSymbols/sec
Active Tones	104	52	26	14	–
Pilot/Data/DC Tones	8/96/1	4/48/1	2/24/1	2/12/1	–
Bandwidth	1094	552	281	156	kHz

The MR-OFDM modulator has to encode the data and build the PPDU according to the structure defined in [1]. The PPDU is shown in Figure 3.1, it contains: Synchronization Header (SHR) used to estimate channel behavior and correct frequency and timing errors. The SHR is composed by the Short Training Field (STF) and Long Training Field (LTF); PHY Header (PHR), which carries frame length, scrambling seed, modulation and coding scheme (MCS); Packet Service Data Unit (PSDU), the actual payload; PPDU Tail Bits field (TAIL) and Pad Bits (PAD) field used to reset and fill buffers.

The modulator structure is depicted in Figure 3.2. This MR-OFDM PHY was designed at Eldorado Research Institute for the ASIC implementation of the IEEE802.15.4g standard, from the reference structure and specification found in [1]. The modulator structure in figure 3.2 shows the modulator divided into data and signal processing. Within the data processing part, conditioning of data bits is performed, it allows error correction

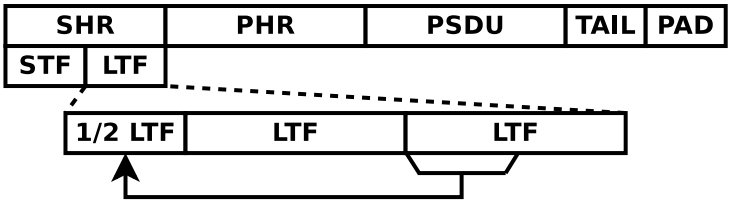

Figure 3.1 – Format of the MR-OFDM PPDU and LTF.

and protects the data from channel impairments. From the Mapper (point 9 in Figure 3.2) onward, digital signal processing is applied, where signal conditioning is applied to complex decimal data. Every numbered point referenced in Figure 3.2 represents conditioning or modification of the data. The complete data processing, and signal processing is as follows:

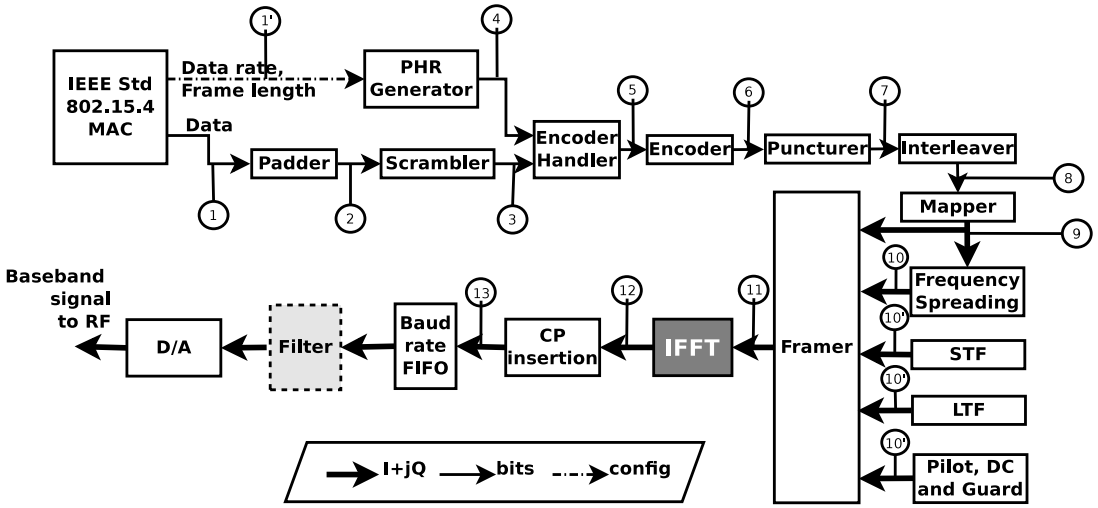

Figure 3.2 – MR-OFDM Modulator according to the IEEE802.15.4g Standard.

- *Modulator Data Processing*

  1. *PSDU Data*: The payload is received from the upper MAC layer, that is the information that is going to be modulated, i.e., a string of bits that represent some message.

  1′. *Configuration*: Generation of Frame length, MCS Level, Data rate, scrambler seed are configurations that come from the MAC layer.

  2. *Padder*: PAD bits and TAILS bits are added through the padder block to the PSDU.

  3. *Scrambler*: It randomizes the data bits, avoiding long sequences of zeros or ones.

  4. *PHR Generator*: It generates the PHR according to the configuration chosen in the MAC layer.

5. *Encoder Handler*: Since not all the data is encoded in the same way, this block exchanges between scrambled data and PHR data, in other words, it controls which data to be encoded.

6. *Encoder*: It adds redundancy to the incoming data to add error correction capabilities to the system.

7 . *Puncturer*: It changes the data rate according to the MCS.

8 . *Interleaver*: Changes the bits order in the data stream. It avoids burst errors.

9 . *Mapper*: Converts bits to symbols, it maps data bits to a BPSK, QPSK or 16 QAM symbol according to the MCS adopted.

- *Modulator Signal Processing*

10 . *Frequency Spreader*: Spread the signal in a frequency band greater than the original signal, replicas of the signal are combined to obtain a single symbol.

10′ . *STF*: Short training field, it adds a known sequence to the beginning of the frame, information that is used at the receiver to estimate and correct timing and frequency errors.

10′ . *LTF*: Long training field. Known sequence different from the STF. Although with the same function, used for synchronization, frequency error correction and channel equalization at the receiver.

10′ . *Pilot, DC and Guard*: Pilots tones for channel detection and channel gain estimation, DC tone, and guard interval to avoid channel interference.

11 . *Framer*: The framer rearranges the complex symbols and prepares the OFDM symbol.

12 . *Inverse Fourier Transform*: It converts the symbol from the frequency domain to the time domain.

13 . *CP Insertion*: Adds a Cyclic Prefix, to eliminate ISI.

Figure 3.3 depicts the demodulator structure, its main goal is to recover the transmitted data. Basically the receiver undoes the process done by the transmitter, however, before this can be accomplished, it has to deal with synchronization issues. Timing synchronization must be attained as well as frequency errors corrected to revert the process done in the transmitter. This is why almost all the signal processing part of the receiver is dedicated to synchronization and frequency error correction. Frequency synchronization and its implementation are one of the topics of this work, specific on the ICFO which is estimated and corrected by the block highlighted in Figure 3.3. A detailed

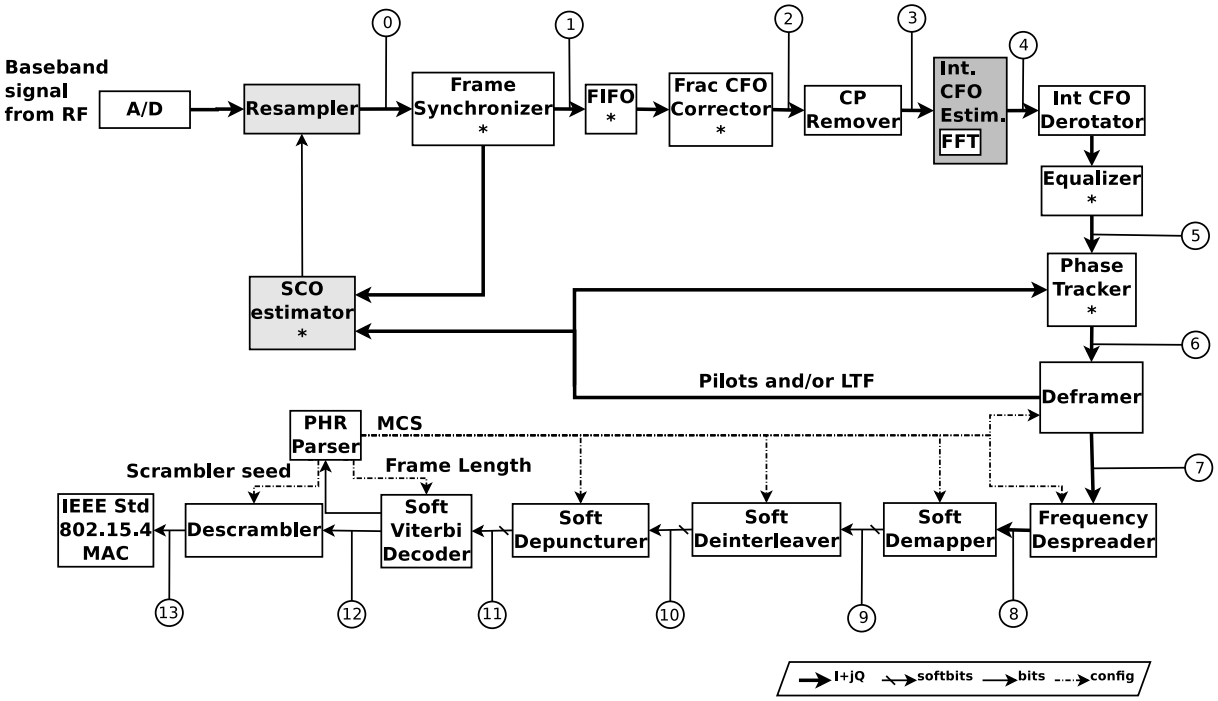

Figure 3.3 – Implemented MR-OFDM receiver.

description of the ICFO was described in section 2.1.1. The processing involved at the demodulator is as follows:

- *Demodulator Signal Processing*

  0. *Resamples*: Resamples the incoming signal to compensate for the Sampling Clock Offset (SCO) introduced by the frequency offset in the sampling clocks.

  1. *Frame Synchronizer*: The first step in the OFDM demodulation is the time synchronization. It determines the starting point of the OFDM frame.

  2. *Fractional CFO*: The frequency error is divided into two parts, a fractional part and an integer part regarding the subcarrier space. In the approach chosen in this work, the fractional correction is performed first.

  3. *CP Remover*: The prepended copy of the symbol end, to combat ISI, is removed.

  4. *FFT/ICFO*: ICFO estimation/correction is performed and time domain to frequency domain transformation is performed.

  5. *Equalizer*: The zero forcing equalizer [28] compensates for the channel distortion.

  6. *Phase Tracker*: Phase correction for residual errors not corrected in previous stages.

  7. *Defframer*: The frame is decomposed into its parts, namely, DC and pilots tones, Guard intervals, PSDU and PHR.

8 . *Frequency Despreader*: The frequency spreading is reverted, the replicas in the frequency domain are removed.

9 . *Soft demaper*: From symbols to bits, from this point on, only data bits are processed.

- *Demodulator Data Processing*

10 . *Soft deinterleaver*: Rearranges the data in order to revert the process done by the interleaver at the transmitter.

11 . *Soft Depuncturer*: Reverts the process of the puncturing at the transmitter.

12 . *Viterbi Decoder*: The decoding of the data to perform Forward Error Correction, depending on the noise levels in the transmission the output at this stage is free or with much fewer errors than before.

13 . *Descrambler*: Returns the original data sequence, the payload or PSDU data.

This chapter described briefly the MR-OFDM mode of the IEEE802.15.4g standard. The technical aspects covered, and the requirements of this standard will define the characteristics of the implementations of this work.

4 The Discrete Fourier Transform and Fast Fourier Transform

Frequency domain analysis is widely used in many areas, control systems engineering, electronics, signal and image processing are among the many fields which benefit from analysis of signals in the frequency domain. It allows analyzing system from a different point of view, simplifying its mathematical representation and allowing a more straightforward manipulation. One way of transforming a discrete signal from time domain to frequency domain is by means of the DFT. Since the direct implementation of the DFT is time and resource consuming and due to the wide range of applications of the DFT, several algorithms called Fast Fourier Transforms were conceived. Fast Fourier algorithms decrease the DFT complexity (fewer operations and/or simpler computations tasks) by means of a divide and conquer approach. Essentially it maps the problem into several sub-problems and applies the division recursively to the sub-problems as well, eventually leading to a reduction in computation complexity.

Fast algorithms can be classified into two major classes [29], those based on the Good's mapping, which basically factors the DFT or divides the DFT into smaller DFTs with its sizes being co-prime, this approach has the advantage of no producing any Twiddle Factor ($W_N = e^{\frac{-j2\pi}{N}}$), with this, a lower bound in complex multiplications is achieved. The Winograd Fourier Transform (WFTA) [30] and Prime Factor algorithm (PFA) [31] are two of the algorithms based on this method. The second class of algorithms known as Cooley-Tukey Fast Fourier Transform () or radix-r algorithms, suitable for r^n size sequences, where the use of twiddle factors is inevitable.

CT-FFT algorithms add regularity to the computation, considering that CT-FFT algorithms show a more repetitive use of fewer modules. Algorithms based on a CT-FFT approach are easier to implement in a parallel kind of way since the small repetitive modules are applied on contiguous sets of data. The repetitive structure is also advantageous since improvements in those small repetitive blocks allow for an overall improvement in the computation. Moreover, CT-FFT algorithms are more suitable for in-place computation (the computed result is written back to where it was read), a desired feature since no additional memory is required. On the other hand, algorithms like the WTFA require large memory and does not allow for in-place computation [29].

All the advantages of the CT-FFT algorithms over Good's based ones, lead to Cooley-Tukey algorithms begin more used in implementations. Complex multiplications alone are not the main concern in implementation and computation time, as well as

relevants are the number of additions, memory access and design budget. It is worth noticing that while a lot of research have been done in the subject due to the importance of the DFT and FFT algorithms, new algorithms continue to emerge, as is the case of the Sparse Fourier Transform presented in [32] and implemented in [33] for a million point DFT. The Sparce Fourier Transform algorithms take advantage of the sparsity shown in some signal where the DFT is applied, that is, only a few components of the signal are of importance since all other signal components in the frequency domain are zero, the algorithm then only computes the set of needed non-zero frequency components. This is not the case of the current intended application, therefore, the classic approach to the FFT implementation is adopted. A review of the Cooley-Tukey approach to the DFT computation is presented in the folowwing.

4.1 FFT Cooley-Tukey Algorithm

The DFT is a mathematical tool widely used in signal processing. Basically, the DFT of a N-point sequence converts discrete samples from time-domain to frequency-domain and vice versa through equation (4.1):

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \tag{4.1}$$

where $k = 0, 1, 2, \cdots, N-1$ and W_N (also known as twiddle factor) is given by:

$$W_N = e^{\frac{-j2\pi}{N}}. \tag{4.2}$$

An N-point inverse DFT (IDFT) is computed as:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk}. \tag{4.3}$$

The direct computation of the DFT is difficult to implement due to its high computation complexity, for instance, a N points computation of a sequence composed of complex samples requires N^2 complex multiplications (each complex multiplication is equal to 4 real multiplications and 2 real additions) and $N^2 - N$ complex additions (every complex addition is equal to 2 real additions), a complexity of $O(N^2)$ [34].

Based on a divide-and-conquer approach a new class of algorithms that optimizes the computation of the DFT were developed, this new class of algorithms are known as FFT. The Cooley and Turkey[35] algorithm is a well known FFT algorithm.

The divide and conquer approach divides the computation problem in smaller problems and solves every smaller problem recursively using the same algorithm, the solution to the original problem is then the combination of the smaller solutions. A common

approach is to divide the problem by half; then each half is then divided by half, and so on. The recursive division is known as radix-2 FFT; the partition goes until only size two DFTs remain. Following this approach, two types of FFTs are derived according to the domain where the partition is performed, the Decimation in Time (DIT), where the partition occurs on the input sequence (time domain), and the Decimation in Frequency (DIF) where the output sequence is partitioned (frequency domain).

The decimation in frequency algorithm is derived as follows, given a $x(n)$ sequence of length $N = 2^p$, where p is an integer, the DFT of a time sequence $x(n)$ computed as in (4.1) can be written as:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[n]W_N^{nk} + \sum_{n=N/2}^{N} x[n]W_N^{nk} \\ &= \sum_{n=0}^{N/2-1} x[n]W_N^{nk} + W_N^{Nk/2} \sum_{n=0}^{N/2-1} x[n+N/2]W_N^{nk}, \end{aligned} \tag{4.4}$$

Since $W_N^{Nk/2} = (-1)^k$ (4.4) can be written as:

$$X[k] = \sum_{n=0}^{N/2-1} [x(n) + (-1^k)x(n+N/2)]W_N^{nk} \tag{4.5}$$

Decimating in frequency (i.e. splitting in odds and even frequency components) and using the fact that $W_N^2 = W_{N/2}$,

$$X[2k] = \sum_{n=0}^{N/2-1} [x(n) + x(n+N/2)]W_{N/2}^{nk} \tag{4.6}$$

$$X[2k+1] = \sum_{n=0}^{N/2-1} [[x(n) - x(n+N/2)]W_N^n]W_{N/2}^{nk} \tag{4.7}$$

The entire coputation of the $N/2$ point sequences (4.5) and (4.7) can be rewritten as:

$$X[2k] = \sum_{n=0}^{N/2-1} g1(n)W_{N/2}^{nk} \tag{4.8}$$

$$X[2k+1] = \sum_{n=0}^{N/2-1} g2(n)W_{N/2}^{nk} \tag{4.9}$$

where

$$g1(n) = [x(n) + x(N/2 + n)] \tag{4.10}$$

$$g2(n) = [x(n) - x(N/2 + n)]W_{N/2}^{nk} \tag{4.11}$$

Figure 4.1 shows an example of the computation of (4.7) and (4.5) for a 8 point sequence. It can be seen as two DFTs of half the sequence size, with two input sequences given by (4.10) and (4.11).

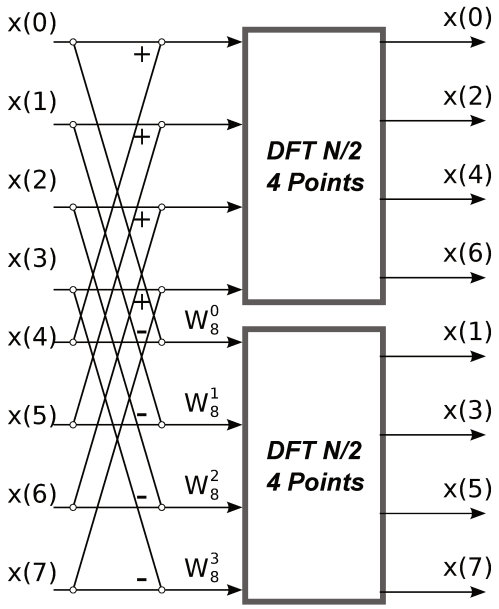

Figure 4.1 – First stage in the DIF computation process.

By performing the same computations on the $N/2$ point DFTs until only remains two points DFTs, an optimization in the algorithm can be made. Figure 4.3 shows a diagram of the entire computation process for a DFT of size 8 using the FFT. It can be seen highlighted the basic computation process performed in every stage, two complex samples are added, subtracted and multiplied by the twiddle factor. This operation is known as the butterfly (Figure 4.2), and it involves one complex multiplication and two complex additions.

There are $N/2$ butterflies per stage and $log2(N)$ stages. Thus, the total number of complex multiplications is $N/2log2(N)$, and complex additions is $Nlog(N)$ which is much less than N^2 complex multiplications and $N^2 - N$ complex additions of the direct implementation, this algorithm shows a complexity of $O(N/2log2(N))$. The optimization of the algorithms relies on the redundancy present on the twiddle factor which is exploited using fewer multiplications to perform the computation. Higher radices (4, 8, 16) can be

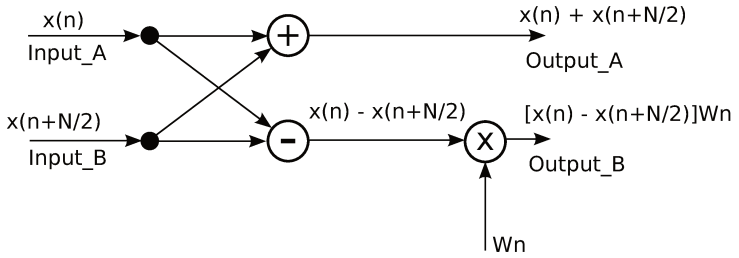

Figure 4.2 – Decimation in frequency butterfly.

derived when the division of the sequence done by 2^n $n = 1, 2, 3, 4, ..etc.$ Fewer multiplications at the cost of a more complex butterfly and control of the entire computations is achieved with higher radices. In the next section, the implications of the hardware implementation of the FFT algorithm is addressed.

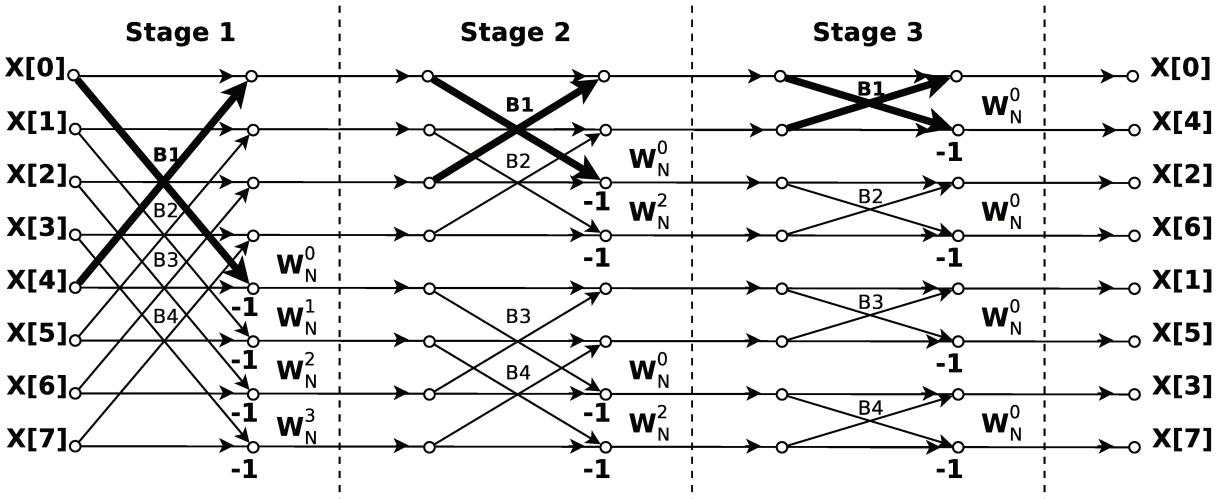

Figure 4.3 – Decimation in frequency diagram.

4.2 Architecture of the Fast Fourier Transform

4.2.1 Proposed Architecture

Typically two kinds of hardware implementations are adopted to compute the FFT, namely the "in-place" and pipeline or parallel approach [36]. The "in-place" approach performs a single computation per clock; thus a single computation unit (a butterfly unit) and a N size memory is used. Basically, the butterfly reads data from memory, performs the computation and writes the results back to the same memory space from where it read the data. This kind of implementation has low hardware cost since it reuses the computation unit and other hardware resources. However, with this approach, the throughput is relative low, since it can perform only one butterfly operation per cycle. In the pipeline approach [37] several butterfly units perform the computation

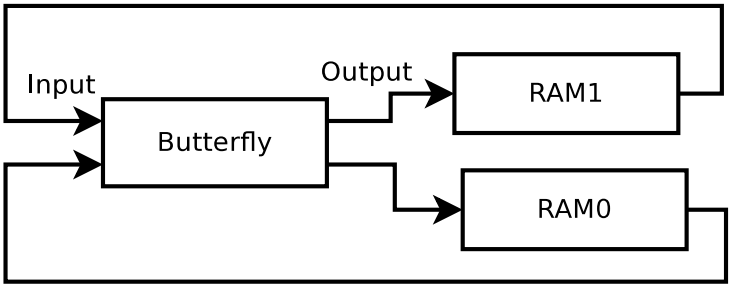

Figure 4.4 – Simple In-place architecture.

in parallel, typically one butterfly for every FFT stage. The parallel approach uses more resources, but the throughput is higher than that of "in-place" architectures.

4.2.1.1 FFT Implementation

4.2.1.1.1 Hardware Considerations

The architecture presented is a collection of methods and algorithms that are believed to simplify the complexity of the FFT hardware implementation. The architecture chosen to achieve the standard requirements for the DFT implementation is based on the "in-place" approach. As was mentioned before, this method has a low throughput, however, it consumes less hardware when compared to parallel architectures, as needed by the IEEE802.15.4g standard.

The "in-place" method also allows more hardware reuse if the architecture is intended for variable size FFTs. The same hardware is used to perform the computation, varying only the time spent to do so. This could not be accomplished by a pipeline architecture, since one butterfly and a memory module is needed per FFT stage, leaving resources unused for the lower size FFTs and wasting power. In the following sections, a description of the various parts that compose the FFT/IFFT architecture is presented.

Memories The "in-place" FFT computation consists in taking two data samples from memory, performing the computations and writing back the computed values to memory as shown in Figure 4.4, the process is repeated $N/2log_2(N)$ times. To take advantage of parallelism two banks of double-port memory are used, namely M0 and M1, allowing the butterfly unit to retrieve two data points simultaneously, as well as to write data to memory in the same manner.

Address Generator (ADG) In the "in-place" approach scheme the computed data is written back to the memory space from where it was read. In the ordinary flow of the algorithm this only occurs in the first stage of the FFT (See FFT Diagram in Figure 4.6), e.g taking samples $X[0]$ from M0 and $X[4]$ from M1, computing the butterfly (B1 in

the example of Figure 4.6) and write the result back in different memory banks. In [8], Xiao proposed an in-place addressing method that uses reduced logic to achieving an "in-place" approach where data is read/written from different memory banks in parallel Basically, Xiao's approach exchanges data at the input/output of the butterfly. Addresses are generated according to the data exchanged achieving an "in-place" memory read/write scheme. The address generation consists of two main counters, *CountB* and *CountS*. The former for the butterfly being computed and the latter for the stage that is currently in progress, according to Figure 4.3.

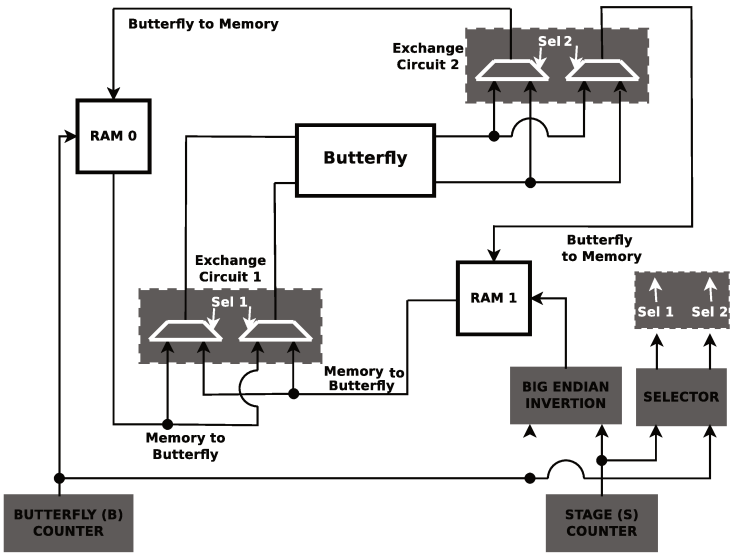

Figure 4.5 – Address Generator.

The number of butterflies/stages being computed depends on the MR-OFDM mode that is currently set. For an N-sized FFT, $log_2(N)$ stages and $N/2$ butterflies are required. Since the MR-OFDM FFT/IFFT supports different FFT sizes, the counters are chosen to support the maximum possible size (N=128). Once the butterfly counter has reached the appropriate value, a reset is applied, and the stage counter is incremented. In order to accomplish the variable FFT size, some constant values are set according to the chosen OFDM option. Those constants are used to compare the current butterfly and stage count, and to reset the counters. The "in-place" approach has some advantages over the pipeline implementations of the FFT, the hardware reuse of the "in-place" approach is almost total, the same hardware is used for the various FFT sizes need by the MR-OFDM.

The address generation logic is as follows, the addresses for the first memory bank (M0) is taken directly from the butterfly counter, as it increases so does the address. The address for the second memory bank M1 changes with the stage that is currently being computed, the value of *CountS* determines the number of bits from *CountB* that are inverted in a big-endian mode. This is accomplished by a series of 2:1 multiplexers, one for every address bit. The multiplexers are controlled by a B-sized shift register that shifts one

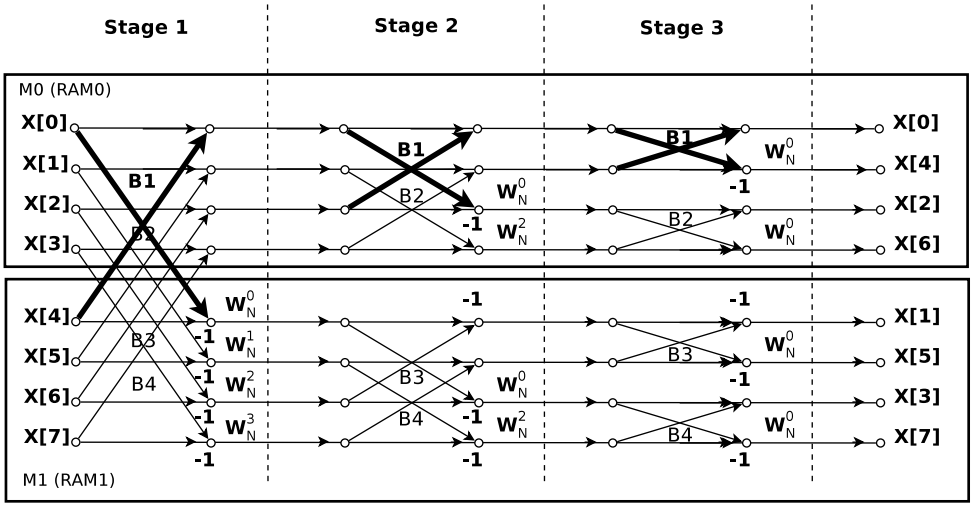

Figure 4.6 – 8 point FFT DIF diagram.

bit at each stage; its inputs are the value of *CountB* and inverted *CountB*. Additionally, exchange multiplexers are placed at the input/output of the butterfly. These exchange multiplexers alternate the butterfly input/output to the memories being read/written. The multiplexers control signals are generated according to the butterfly (*CountB*) and state counter (*CountS*). For computing a FFT of size $N = 2^n$, $log_2(N)$ stages and $N/2$ butterflies per stage are needed. Thus, the address generator needs *CountS* and *CountB* with $(n-1)$ bits and $\lceil log_2(n) \rceil$ ($\lceil . \rceil$ is defined as the ceil operator), bits, respectively. Figure 4.5 shows a simple FFT architecture with the blocks involved in the address generation highlighted.

Butterfly Unit The basic computation performed at every FFT stage is known as the butterfly computation. Figure 4.2 shows the butterfly diagram for a radix-2 decimation in frequency FFT. The operation involves taking two data samples, performing the addition and subtraction between each one, and multiplying the subtraction result by the twiddle factor(4.2).

The complex multiplication without any optimization [38] requires four real multiplications and two real additions/subtractions, an operation that is usually very large and time-consuming. The Coordinate Rotation Digital Computer (CORDIC) algorithm [39] is an alternative to perform the complex multiplication operation, it requires only add and shift operations, Figure 4.7 shows the detailed architecture of the DIF butterfly. The CORDIC algorithm is detailed in section 4.2.1.2.

Additionally, using the CORDIC to perform the complex multiplication, the ROM memory usually implemented to store the twiddle factor can be eliminated [14][13]. The substitution of the complex multiplier is as follows; the twiddle factor multiplication is equivalent to the rotation of the sequence $x(n)$ by an angle $-\frac{2\pi}{N}nk$, an operation that

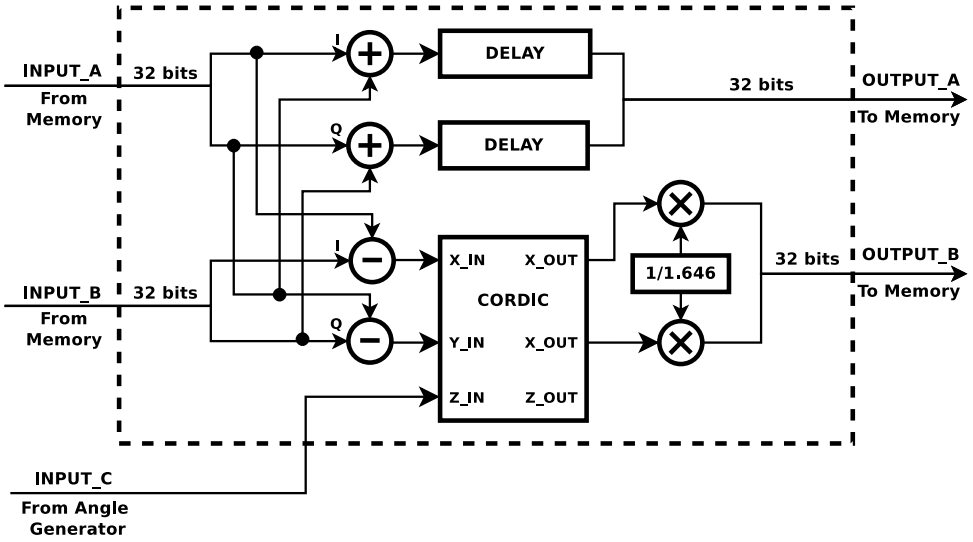

Figure 4.7 – DIF Butterfly Architecture.

can be performed by the CORDIC algorithm. Although the substitution of the multiplier by the CORDIC algorithm in the butterfly unit avoids the need of a memory to store twiddle factors values, a ROM memory would still be required to store the phase angles. However, if the order in which the data is read follows a pattern in the phase angles, they can be generated in a generic way, as is the case of the addressing scheme presented in the previous section.

Table 4.1 shows the angles values for a 16 point FFT in each stages. Here the angle values show a pattern, the non-zero values are always equal to $\pi/(N/2)$ multiplied by an integer. It can be noticed from the bit representation of this integer (Table 4.2) that the next stage value is equal to the previous one shifted in one bit, or the first value shifted S times, where S is the number of the n^{th} stage. Therefore, this integer value can be generated by taking the butterfly counter and shifting its value at every stage. Then, the angle can be obtained by multiplying this value with the constant $\pi/(N/2)$ value.

Table 4.1 – Angle values for a 16 size FFT

Stage 0	Stage 1	Stage 2	Stage 3
0	0	0	0
$1\pi/8$	$2\pi/8$	$4\pi/8$	0
$2\pi/8$	$4\pi/8$	0	0
$3\pi/8$	$6\pi/8$	$4\pi/8$	0
$4\pi/8$	0	0	0
$5\pi/8$	$2\pi/8$	$4\pi/8$	0
$6\pi/8$	$4\pi/8$	0	0
$7\pi/8$	$6\pi/8$	$4\pi/8$	0

Table 4.2 – Bit representation of Counter B shifted for a 16 point FFT

Stage 0	Stage 1	Stage 2	Stage 3
000	000	000	000
001	010	100	000
010	100	000	000
011	110	100	000
100	000	000	000
101	010	100	000
110	100	000	000
111	110	100	000

4.2.1.2 CORDIC

The CORDIC [39] is an algorithm that allows the implementation of many trigonometric functions using basic arithmetic, (subtraction, addition, and shifts), this is ideal for hardware implementation since trigonometric functions are resource consuming. CORDIC algorithms emerge from a simplification of the rotation of a vector, it is as shown in Figure 4.8:

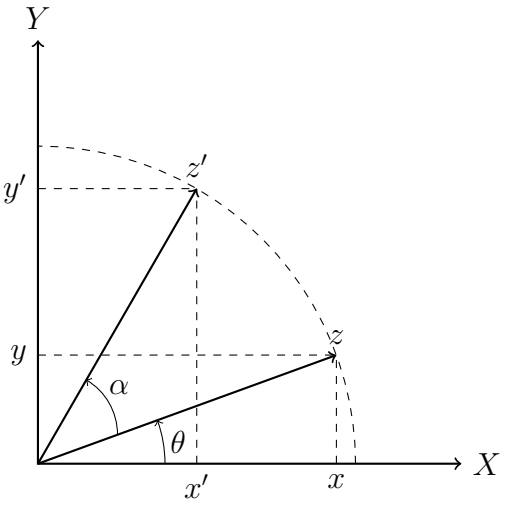

Figure 4.8 – Rotation of a vector.

The rotation of a vector of magnitude $||z||$ from position z to position z' is given by (4.12) and (4.13):

$$\begin{aligned} x^{'} &= \|z\|(\cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\theta)) \\ x^{'} &= x\cos(\theta) - y\sin(\theta) \end{aligned} \tag{4.12}$$

$$\begin{aligned} y^{'} &= \|z\|(\sin(\alpha)\cos(\theta) + \cos(\alpha)\sin(\theta)) \\ y^{'} &= y\cos(\theta) + x\sin(\theta) \end{aligned} \tag{4.13}$$

That can be expressed as

$$x^{'} = \cos(\theta)(x - y\tan(\theta)) \tag{4.14}$$

$$y^{'} = \cos(\theta)(y + x\tan(\theta)) \tag{4.15}$$

Removing the $\cos(\theta)$ term to simplify the operation, the rotation becomes a pseudorotation

$$x^{'} = x - y\tan(\theta) \tag{4.16}$$

$$y^{'} = y + x\tan(\theta) \tag{4.17}$$

The pseudorotation by an angle θ can be achieved by successive smaller rotations. Restricting the angles so that every smaller rotation becomes $\tan(\theta)^i = 2^{-i}$, the pseudo-rotation or the multiplication by $\tan(\theta)$ becomes a multiplication by 2^{-i}, which can be implemented in hardware by merely shifting a binary word. Thus, rotating an input vector by an angle θ is now an iterative process made-up of successive shifts and adds operations.

$$x^{i+1} = x(i) - d_i(2^{-i}y(i)) \tag{4.18}$$

$$y^{i+1} = y(i) + d_i(2^{-i}x(i)) \tag{4.19}$$

$$z^{i+1} = z^i - d_i \arctan 2^{-i} \tag{4.20}$$

$$d_i = \begin{Bmatrix} -1 & if & z_i < 0 \\ +1 & if & z_i > 0 \end{Bmatrix} \tag{4.21}$$

Equations (4.18), (4.19) and (4.20) describe the CORDIC algorithm, the third equation (4.20) is called the angle accumulator and in conjunction with d_i (4.21) (known as the decision operator) determines the direction of the rotation. The operation mode shown is known as the Rotation mode [39], and it rotates the input vector by a specified angle, the angle accumulator is initialized with the desired rotation angle, and for every iteration, the decision operator is chosen such that the magnitude of the residual angle tends to zero. The decision operator is then determined by the sign of the angle accumulated.

Since the $\cos(\theta)$ term is removed to simplify the algorithm the outputs x and y are scaled by a factor k_n, the scaling factor is given by:

$$\begin{gathered} k_n = \prod_{i=1}^{n} \frac{1}{\cos(\theta)^i} = \prod_{i=1}^{n} \sqrt{1 + \tan^2\theta^i} = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \\ k_n \to 1.6476 \text{ as n} \to \infty \end{gathered} \tag{4.22}$$

After n iterations the CORDIC output is given by:

$$x_n = k_n[x_0\cos(z_0) - y_0\sin(z_0)] \tag{4.23}$$

$$y_n = k_n[y_0 \cos(z_0) + x_0 \sin(z_0)] \tag{4.24}$$

$$z_n = 0 \tag{4.25}$$

A second operation mode called Vectoring mode can be accomplished if instead of $d_i = sign(z_i)$, is chosen such that $d_i = sign(y_i)$. In this mode the vector is rotated by the necessary angle so that the x component of the result vector is maximized and its y component tends to zero. After n iterations the CORDIC output in Vectoring mode is given by:

$$x_n = k_n\sqrt{x_0^2 + y_0^2} \tag{4.26}$$

$$y_n = 0 \tag{4.27}$$

$$z_n = z_0 + \tan^{-1}(y_0/x_0) \tag{4.28}$$

Besides the circular coordinate system described by the CORDIC equations (4.18), (4.19), (4.20) can be adapted to perform operations in other coordinate systems, namely hyperbolic and linear [40]. A new variable m, defines the coordinate system used. Equations (4.29), (4.30), (4.31) define the new generalized CORDIC equations, Table 4.3 shows the operation modes. In Table 4.4 the CORDIC outputs for the three modes of operation are shown and finally Table 4.5 shows some of the functions that can be computed with the CORDIC algorithm.

$$x_{i+1} = x(i) - md_i[2^{-i}y(i)] \tag{4.29}$$

$$y_{i+1} = y(i) + d_i[2^{-i}x(i)] \tag{4.30}$$

$$z_{i+1} = z^i - d_i\sigma_i \tag{4.31}$$

The implementation of the CORDIC is an iterative hardware, which is structured as a cell array integrated into one block. A CORDIC iteration cell is made of 3 add-subs, 1 mux, 3 registers, and 2 wired shift blocks, as detailed in Figure 4.9. The number of cells, n, is defined by the iteration parameter, which is, usually, equal to or less than the width of the input signals.

Table 4.3 – CORDIC Modes of operation

Mode	d_i	Coordinate System	σ_i	m
Vectoring	$sign(y_i)$	Circular	$\tan(2^{-i})$	1
Rotation	$sign(z_i)$	Hiperbolic	$\tanh(2^{-}i)$	-1
-		Linear	$2^{-}i$	0

Table 4.4 – Results of CORDIC Generalized equations

m	Rotation Mode	Vectoring Mode
1	$x_n = k_n[x_0\cos(z_0) - y_0\sin(z_0)]$ $y_n = k_n[y_0\cos(z_0) + y_0\sin(z_0)]$ $z_n = 0$	$x_n = k_n\sqrt{x_0^2 + y_0^2}$ $y_n = 0$ $z_n = z_0 + \tan^{-1}(y_0/x_0)$
0	$x_n = x_0$ $y_n = y_0 + x_0 z_0$ $z_0 = 0$	$x_n = x_0$ $y_n = 0$ $z_n = z_0 + (y_0/x_0)$
-1	$x_n = k_h[x_0\cosh(z_0) - y_0\sinh(z_0)]$ $y_n = k_h[y_0\cosh(z_0) + y_0\sinh(z_0)]$ $z_n = 0$	$x_n = k_h\sqrt{x_0^2 - y_0^2}$ $y_n = 0$

Table 4.5 – Some Functions Computed by the CORDIC

Mode	m	x_0	y_0	z_0	x_n	y_n or z_n
Rotation	1	k_n	0	θ	$\cos(\theta)$	$y_n = \sin(\theta)$
Vectoring	1	1	a	0	$k_n\sqrt{a^2+1}$	$y_n = \tan^{-1}(a)$
Rotation	-1	k_h	0	θ	$\cosh\theta$	$y_n = \sinh(\theta)$
Rotation	-1	a	a	θ	$k_n a e^{\theta}$	$y_n = Kae^{\theta}$
Vectoring	-1	a	1	0	$k_n\sqrt{a^2-1}$	$z_n = \cot^{-1}(a)$
Vectoring	-1	$a+1$	$a-1$	0	$2k_h\sqrt{a}$	$z_n = 0.5\ln(a)$
Vectoring	-1	$a+b$	$a-b$	0	$2k_h\sqrt{ab}$	$z_n = 0.5\ln(a/b)$
Vectoring	0	$\sin(\theta)$	$\cos(\theta)$	0	$\sin a$	$z_n = \tan(\theta)$
Vectoring	0	$\sinh(\theta)$	$\cosh(\theta)$	0	$\sinh a$	$z_n = \tanh(\theta)$
Vectoring	0	$\ln(\theta)$	$\ln(a)$	0	$\ln b$	$z_n = \log_b(a)$

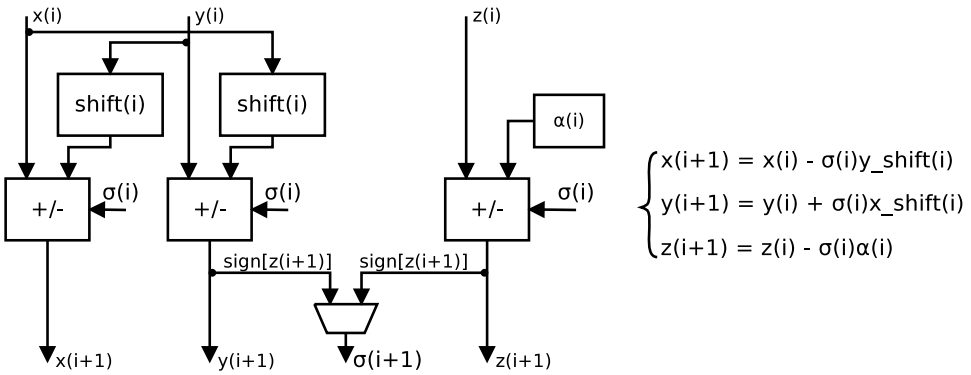

Figure 4.9 – CORDIC Cell.

4.2.2 IFFT Shifter

In order to accomplish the scaling need by the inverse Fourier Transform the IFFT process (4.3), a shift right is performed at the end of every stage of the computation process. Having $log_2(N)$ stages shifting one bit after every stage is equivalent to dividing the output by $1/N$ (Shifting right the output $log_2(N)$ times). As this scaling is necessary only for the IFFT process, one-bit control signal from the Control Unit selects whether to bypass or shift the data according to the selected operation (FFT or IFFT). This bit also controls the sign generated for the angle in the angle generator unit.

This approach allows decreasing the bit width of the IFFT. If the shift is not performed after every other stage, there is a growth in the samples magnitude per stage, therefore more bits are need to allocate that bit growth. Still, a shift would be needed after the complete computation given the expected results. With this approach, there is no significant bit growth, which allows to maintain the same bit width for the FFT and IFFT.

The complete FFT/IFFT architecture is shown in Figure 4.10. Counters, CORDIC based butterfly, IFFT shifter and angle generator are shown. A control unit controls the behavior of the IFFT/FFT block for the different MR-OFDM modes and inverse or forward transforms.

4.2.3 Variable Length FFT/IFFT

Since one of the requirements of the IEEE802.15.4g standard is the variable length of the DFT/IDFT this architecture supports variable FFTs sizes. The architecture was designed to support the maximum FFT size (128 points). Using comparators and configuration signals resets area applied to counters, variables in the angle generator are settled to accomplish the smaller FFT sizes. The hardware utilization is almost the same for every FFT size; however, the time spent to compute every FFTs significantly changes.

4.3 Implementation Results

After the definition of the architecture detailed in Section 4.2.1, a high-level model that of the FFT/IFFT was implemented in Matlab high-level scripting language. Finally, the FFT/IFFT block was implemented using the VHDL hardware description language and the results of its simulations compared to those of the high-level model. Afterward, the design was prototyped on a *Cyclone* 5 FPGA development kit from Altera. All the blocks described in the previous section were implemented, except the CORDIC block which was an IP block taken from the project developed in our group at Eldorado

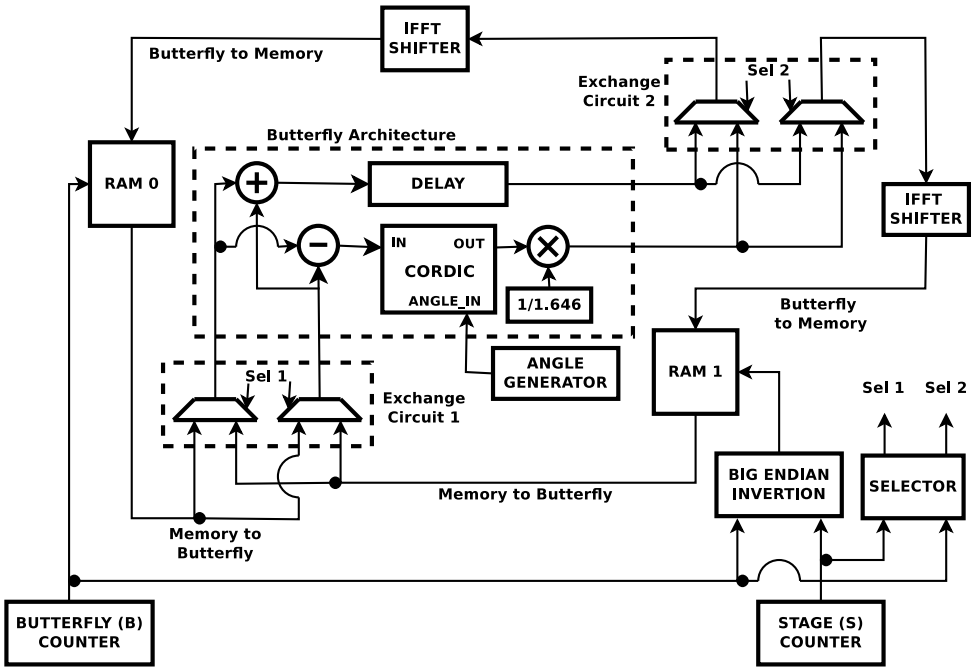

Figure 4.10 – FFT/IFFT internal architecture.

Institute.

Figure 4.11 depicts the IFFT verification process. First, a high-level model of the entire MR-OFDM modulator was used to generate the test data for the IFFT implementation. As can be seen in the verification diagram in Figure 4.11, random bits go through the entire MR-OFDM data processing (refer to Figure 3.2 in Chapter3 for a detailed description of the data processing process), then the processed bits are mapped into symbols, and finally the framer rearranges the OFDM symbol according to the MCS and OFDM option chosen. The SHR and PHR are also generated in this high-level model. The OFDM symbols at the output of the framer are then fed into four IFFT models: a high-level floating point model, generated using the Matlab IFFT function, a high-Level fixed-point model also written in Matlab that emulates the behavior of the IFFT architecture, an RTL behavioral model written in VHDL, and finally the FPGA implementation on the FPGA development board. Every output is then compared against its upper-level model, and the error is computed.

For the FFT case, the Matlab IFFT out is fed into the four FFT models, the FFT verification diagram is shown in Figure 4.12. Tables 4.6 4.7 and 4.8 show the IFFT verification results for the high-level model, RTL-VHDL model and the FPGA implementation, respectively. Tables 4.9 4.10 4.11 for the FFT case. A sequence of 10000 bits were generated for different MR-OFDM MCSs levels and Options, (refer to annex B for the different MCS levels specified in the standard). A quantization of 16 bits, with 11 fractional bits were used for the fixed point models. The error is measured as the ratio between the signal power and the error power (the error is the difference between the

expected and measured value), as shown in equation (4.32):

$$z_{err(dB)} = 10 \log_{10} \frac{P_{signal}}{P_{error}} \tag{4.32}$$

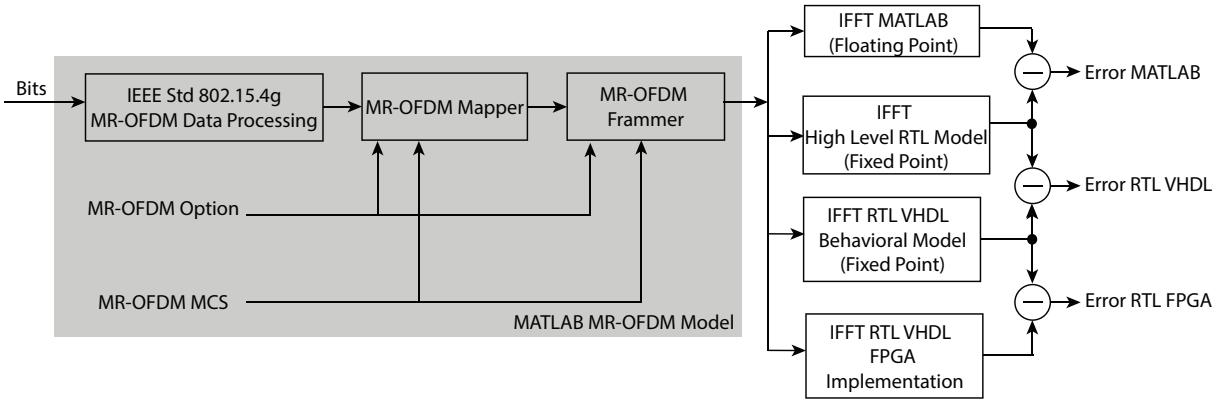

Figure 4.11 – IFFT Verification Diagram

Table 4.6 – High-level IFFT Model/Matlab-IFFT Function Error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	33.6001	35.5834	32.5428	-
MCS3 (QPSK)	27.3959	35.5102	41.7698	49.1669
MCS5 (16-QAM)	-	41.6800	34.8300	51.5497

Table 4.7 – RTL-VHDL Model/High-Level IFFT Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	26.0005	27.4329	24.6249	-
MCS3 (QPSK)	19.9895	27.3983	33.8124	41.5890
MCS5 (16-QAM)	-	33.6300	27.0218	43.3534

Table 4.8 – FPGA IFFT/RTL-VHDL Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	53.4725	Inf	Inf	-
MCS3 (QPSK)	45.3678	52.2401	54.5039	53.7774
MCS5 (16-QAM)	-	57.7434	50.2618	54.9637

Table 4.9 – High-level FFT Model/Matlab-FFT Function Error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	44.1185	45.2774	48.3403	-
MCS3 (QPSK)	45.6849	47.1415	49.9178	52.3192
MCS5 (16-QAM)	-	47.3495	50.8720	52.5051

As can be seen from Table 4.74.10 small differences were found between the VHDL RTL model and the high-level fixed-point model, for both, the IFFT and FFT implementation. This is due to the CORDIC implementation, since this block was not

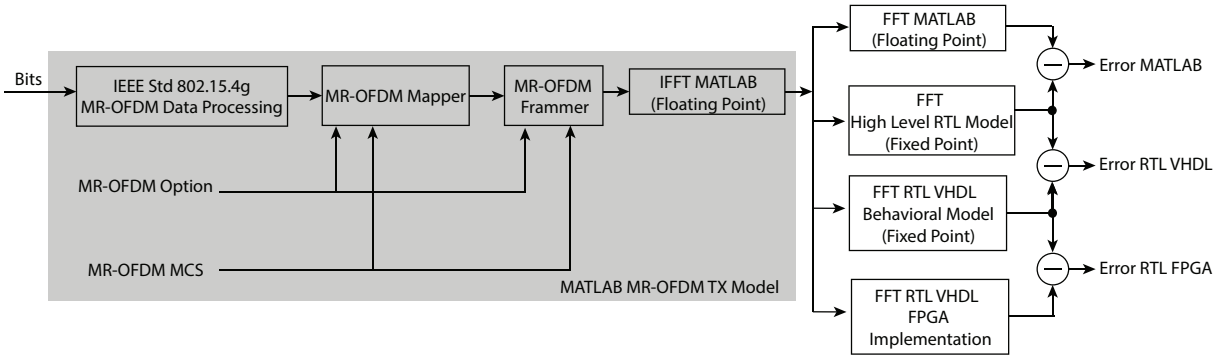

Figure 4.12 – FFT Verification Diagram

Table 4.10 – RTL-VHDL Model/High-Level FFT Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	37.0916	37.5182	41.3165	-
MCS3 (QPSK)	37.7206	38.3136	43.1312	47.7071
MCS5 (16-QAM)	-	39.0021	43.9751	47.5054

Table 4.11 – FPGA FFT/RTL-VHDL Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	36.0128	37.5355	40.2791	-
MCS3 (QPSK)	37.4271	38.8322	41.9002	45.7814
MCS5 (16-QAM)	-	39.1031	42.4393	46.0977

implemented, only instantiated in the architecture. Also, small differences between the RTL-VHDL simulation and the data taken from the FPGA implementation were found. Those differences mean that the FPGA synthesis tool generated netlist has some differences with the logic described in the RTL behavioral model. This may be due to: the FPGA synthesis process, this is an interpretation of an abstract RTL description, it is possible that the synthesis tool is performing some misinterpretation of this RTL Behavioral description; a second possible cause can be the FPGA synthesis tools settings and optimizations parameters, these parameters and settings try to optimize area, power or timing and sometimes show undesired results. An FPGA implementation debugging , and maybe changes to the RTL behavioral description must be performed to accomplish more accurate results. These differences do not compromise the integrity of the computation.

To show the accuracy of the RTL-VHDL simulation and the FPGA implementation Tables 4.12 ,4.13 ,4.14, 4.15 show the error between the RTL behavioral simulation, the FPGA implementation and the Matlab floating point model, for the both the IFFT and FFT computations. As can be seen from these results there is not much difference between the VHDL and FPGA implementation for the FFT case for all options for BASPK and QPSK symbols, for 16-QAM we see a loss in accuracy in the FPGA implementation. For the IFFT case, we also see a loss in precision, for the Option 1 and Option 2, for

options 3 and 4 the SNR due to the error decreases in a great amount. As was mentioned before, further debugging of the FPGA implementation and changes in the RTL desing are needed to get the desired results.

Table 4.12 – IFFT RTL-VHDL Model/Matlab Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	30.8096	31.7769	29.2071	-
MCS3 (QPSK)	24.7766	31.7623	38.2470	46.4801
MCS5 (16-QAM)	-	37.9841	31.3515	47.6666

Table 4.13 – IFFT FPGA Implementation/Matlab Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	30.8570	31.7769	29.2071	-
MCS3 (QPSK)	24.7766	31.7623	38.3970	46.0057
MCS5 (16-QAM)	-	38.0532	31.4776	47.3911

Table 4.14 – FFT RTL-VHDL Model/Matlab Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	43.0932	42.4841	49.4530	-
MCS3 (QPSK)	43.1822	42.5255	51.7607	49.2994
MCS5 (16-QAM)	-	43.9739	51.7811	48.9116

Table 4.15 – FFT FPGA Implementation/Matlab Model error

MCS	Opt 1 (SNR dB)	Opt 2(SNR dB)	Opt 3(SNR dB)	Opt 4(SNR dB)
MCS1 (BPSK)	42.7790	41.7801	46.5021	-
MCS3 (QPSK)	42.5880	42.2511	49.7080	49.2263
MCS5 (16-QAM)	-	43.3831	49.2263	48.1357

4.3.1 FPGA Prototyping

The proposed FFT/IFFT implementation along with the one provided by Altera was synthesized on a *Cyclone 5* FPGA development kit from Altera. Input/output of both FFTs have 32 bits (i.e., 16 bits In-phase, 16 bits Quadrature) and samples in natural order. The proposed design and Altera's FFT was synthesized in Quartus II version 14.1 software, maximum frequency and resource usage of the two designs are detailed in Table 4.16.

Table 4.16 – 128-point FFT Implementation Results for Altera's 5CGXFC5C6F27C7N

FFT	**Fmax**	**ALMs**	**DSP Blocks**	**Memory**	**Registers**	**Combinational**
Altera	204.62 MHz	2190 (5%)	6 (4%)	8508 (< 1%)	4032	1888
Our Design	118.44 MHz	1259 (5%)	0 (0%)	4096 (< 1%)	1796	1789

The architecture implemented in Altera's IP is a radix 2^2 single path delay feedback architecture, this approach has the computation complexity of a radix-4 butterfly but maintaining the structure of the radix-2 algorithm [7]. Unfortunately, Altera's FFT IP documentation does not show a detailed description of its architecture; therefore only results of the overall area consumption of the FFT radix 2^2 is shown (Table 4.16).

It is worth to notice that even though Altera's design can work with higher frequencies than our design, as shown in Table 3.1, the maximum FFT/IFFT Sample Rate for IEEE802.15.4g is 1.3333 MHz which is 98.87% lower than our maximum. Moreover, our design does not use DSP blocks and requires fewer registers and memory than the design provided by Altera (it is achieved a 51.85% reduction of memory, a 55.45% reduction of registers).

Table 4.17 – Resource Utilization by Entity in the FFT/IFFT

Entity	ALMs	Combinational ALUT	Registers	Memory
FFT Top	**1328.0**	**1872**	**1788**	**4096**
├─Control	27.7	46	25	-
└─FFT Module	1293	1814	1763	4096
├─Butterfly	1020	1515	1494	-
├─CORDIC	598	1168	950	-
├─Adders	4x8	4x16	0	-
└─Other Logic	390	283	544	-
├─Address Generator	228	217	253	-
├─Angle Generator	44	82	16	-
├─RAM M0	-	-	-	2048
└─RAM M1	-	-	-	2048

Table 4.17 shows the consumption by sub-entity of the entire FFT/IFFT design given by the Quartus 14.1 fitter (place and route). It can be seen that the butterfly

consumes the most resources of the FPGA, followed by the address generator. The control unit and angle generator are simple circuits consuming the least of hardware resources.

Table 4.18 – Resource Utilization by Entity in the OFDM Modulator

Entity	ALMs	Combinational ALUTs	Registers	Memory Bits
OFDM Modulator	2643.3	1821	3208	111336
IFFT	1206.5	1767	1756	4096
Framer	929.6	860	1135	0
Interleaver	123.8	211	71	192
Padder	83.3	142	50	0
CPI	65.1	112	31	2048
FIFO	49	80	29	105000
PHR Generator	44.7	58	47	0
Handler	33.9	52	18	0
Mapper	26.2	43	30	0
Scrambler	20.5	37	17	0
Puncturer	19.1	30	12	0
Encoder	12.5	20	13	0

The Table 4.18 shows the resource usage by entity in the OFDM transmitter. It can be seen how the FFT block compares with the whole design. Although the FFT is the most resource-consuming overall (1206 Adaptive Logic Module ALMs [1]), it falls after the Framer by 276.9 ALMs.

[1] The ALM is the basic building block of Altera's FPGAs. Each ALM can support up to eight inputs and eight outputs, contains two or four register logic cells and two combinational logic cells, two dedicated full adders, a carry chain, a register chain, and a 64-bit LUT mask.[41]

5 Integer Carrier Frequency Offset Architecture

In this Chapter, an architecture for the estimation and correction of the ICFO is presented. The architecture takes advantage of the FFT computation and the MR-OFDM demodulator structure to estimate the ICFO in a simple and low hardware complexity cost manner. Performance and implementation results are shown for the proposed method. Additionally, a more robust frequency error estimation that takes into account other impairments that affect the OFDM synchronization, is proposed.

The functioning of the employed method is as follows. Using a Data-Aided approach (a known sequence is employed to perform the estimation), a cross-correlation is performed between the received corrupted signal and the reference found in the Synchronization Header (SHR) of the PPDU of the MR-OFDM mode. The cross-correlation operation measures the similarity between two signals, e.g. $f(n)$ and $g(n)$, (5.1) shows the cross-correlation operation. The cross-correlation operation is a sum of products of signals samples, with one signal dislocated one sample at a time. Since the cross-correlation is a measure of the similarity of signals, and the ICFO is a shift of carriers in the OFDM symbol, the ICFO can be found employing the cross-correlation (and computing its maximum value).

$$f[n] \star g[n] = \sum_{n=0}^{N-1} f^*[n] \cdot g[n+l] \quad \text{for} \;\; l = 0 \text{ to } N-1 \tag{5.1}$$

Figure 5.2 shows the result of the cross-correlation operation between a signal of size 128 and its delayed version on 20 samples corrupted by additive white Gaussian noise. Clearly, the correlation result shows a peak value at the correlation sample number 21. The performance of the cross-correlation in estimating the displacement of a signal from its reference point for all MR-OFDM options, in this case between a clean LTF and the dislocated one corrupted with additive white noise and a multipath channel, is shown in Figure 5.3. Figure 5.1 shows the diagram of the high-level model used to estimate the performance of the ICFO, an LTF generated accordingly to the MR-OFDM option is corrupted with channel impairments, then the ICFO is estimated wit the FFT Based ICFO, the result is then compared with the ICFO added to the signal.

The performance is measured as the probability of success in finding the exact value of the displacement from the corrupted and reference signal. For that, 1000 iterations were performed for every SNR point, raging the SNR between $-25dB$ to $10dB$. The

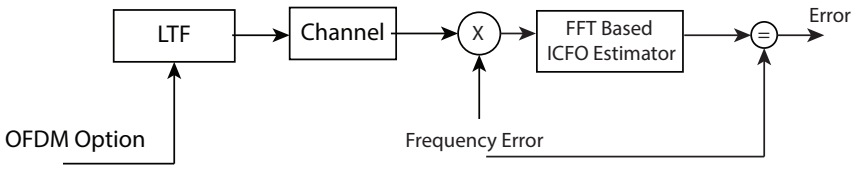

Figure 5.1 – ICFO Test High-Level Model Diagram

algorithm attains a perfect estimation (a probability of success of 1) for a noise level close to $-10dB$ for OFDM Options 1, for the MR-OFDM option 4 the probability of success is reached at approximately $-5dB$, the performance of the estimation diminished with the correlation size, that is the size of the symbol of the MR-OFDM option.

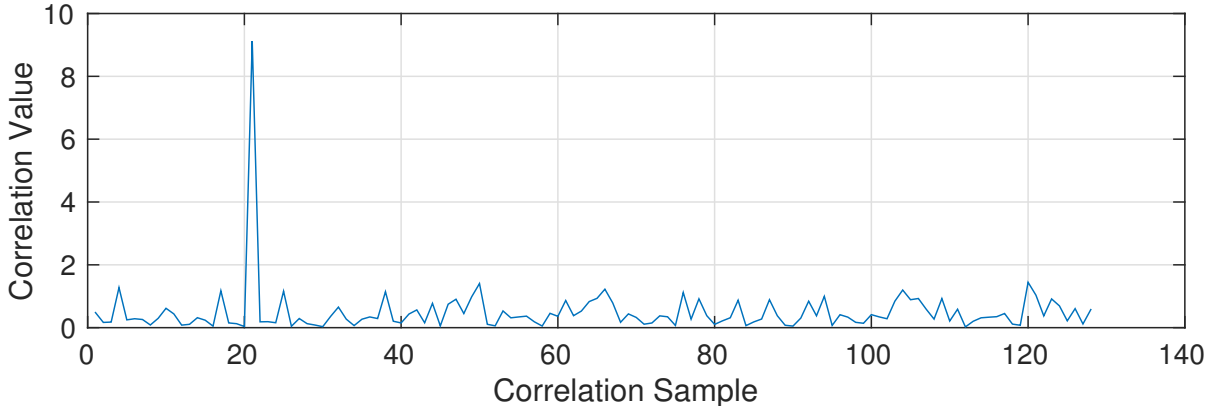

Figure 5.2 – LTF Cross-Correlation output.

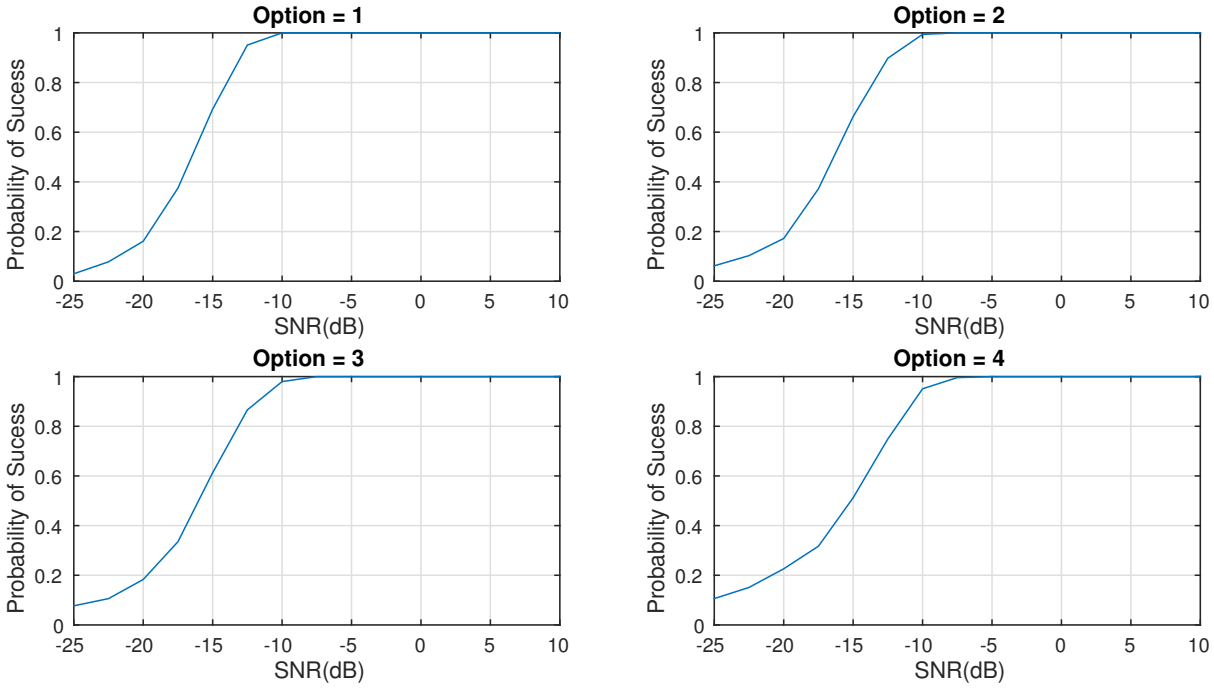

Figure 5.3 – Probability of success of the FFT Based Correlation.

The cross-correlation computation, based on FFT/IFFT, is equivalent to the operation in (5.2), where $f(n)$ and $g(n)$ are time discrete signals, $\star$ denotes circular corre-

lation, $\cdot$ pointwise product, F and F^{-1} direct and inverse Fourier transforms. According to (5.2) the cross-correlation in one domain (either frequency or time) can be computed by taking the product, in a pointwise manner, of the two signals in the opposite domain and then performing the conversion back to the desired domain in which the result is required. This equivalence operation is known as the cross-correlation theorem [42]. For the intended application, this can be advantageous, since it is possible to use the FFT not only to convert the incoming signal from time domain to frequency domain in the MR-OFDM demodulator but also, to calculate the cross-correlation between the two signals, saving a considerable amount of resources needed to compute the cross-correlation.

$$f^*[n] \cdot g[n] \overset{F}{\underset{F^{-1}}{\Longleftrightarrow}} F\{f[n]\} \star F\{g[n]\} \tag{5.2}$$

$$f[n] \star g[n] \overset{F}{\underset{F^{-1}}{\Longleftrightarrow}} F^*\{f[n]\} \cdot F\{g[n]\} \tag{5.3}$$

5.1 ICFO FFT-based Architecture

Taking advantage of the reuse technique to implement the cross-correlation, the architecture depicted in Figure 5.4 is proposed. It performs the estimation/correction as follows: first, during the estimation, the received corrupted LTF is stored in the FIFO and simultaneously sent to the complex multiplier, where it is multiplied by the conjugate of the reference LTF in time domain. Next, the result is processed by the FFT, being converted to the frequency domain.

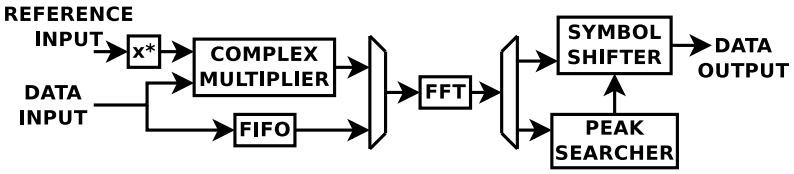

Figure 5.4 – Integer carrier frequency offset estimator/corrector.

The resulting data, which is the cross-correlated LTF in the frequency domain, is sent to the Peak Searcher, where the index of the maximum value of the correlation is drawn. After the initial estimation, data at the FFT input and output are exchanged. At the input, a multiplexor exchanges the result of the multiplier and the stored data in the FIFO. At the output a demultiplexor exchange the symbols in the frequency domain to be sent to the symbol shifter, which corrects the ICFO. This approach saves a significant amount of hardware resources and reduces the block latency, since typical correlation involves extra clock cycles (unless it is done in a parallel approach) and units, multipliers and adds, in this work the process is performed by the FFT and a complex multiplier.

5.1.1 Complex Multiplier

The Complex Multiplier is composed of two adders and four multipliers. For every clock, one received sample is fed into this block and multiplied by the reference's complex conjugate. The conjugated is obtained by a simple two's complement binary inversion of the imaginary component of the data. The FIFO and the multiplexor/demultiplexor ensure the correct flow of the input symbols according to the stage of the ICFO block, the estimation or the correction. Figure 5.5 shows the architecture of the multiplier.

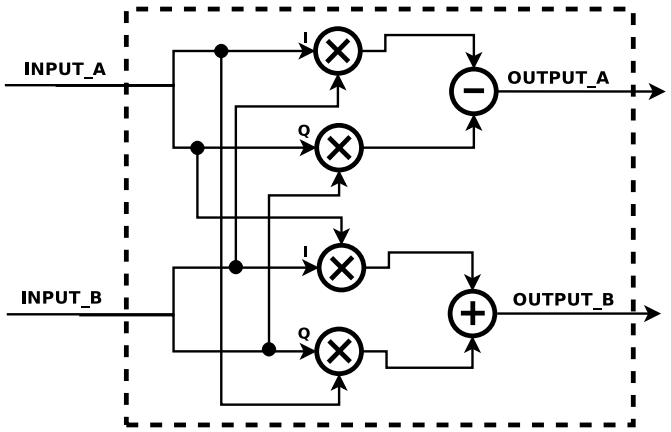

Figure 5.5 – Architecture of the complex multiplier.

5.1.2 FFT

The FFT engine is the IFFT/FFT core that uses Radix-2, based on CORDIC [39], presented in section 4.2.1.

5.1.3 Peak Searcher

The Peak Searcher looks for the index of the max value in magnitude on the FFT output, Figure 5.6 shows the architecture of the peak searcher. It is composed of a CORDIC working in vectoring mode and a comparator. For every sample received the magnitude of the complex data is computed and compared with its previous value, registering the higher one and updating a register with the current sample count (reg idx). So, when the last sample is received, the register result is equal to the index of the maximum peak among the FFT samples.

5.1.4 Symbol Shifter

The Symbol Shifter corrects the ICFO by adjusting the sub-carriers within an OFDM symbol according to the Peak Searcher output. As mentioned, the kind of error compensated by this architecture is equivalent to a number of subcarrier shifts equal to

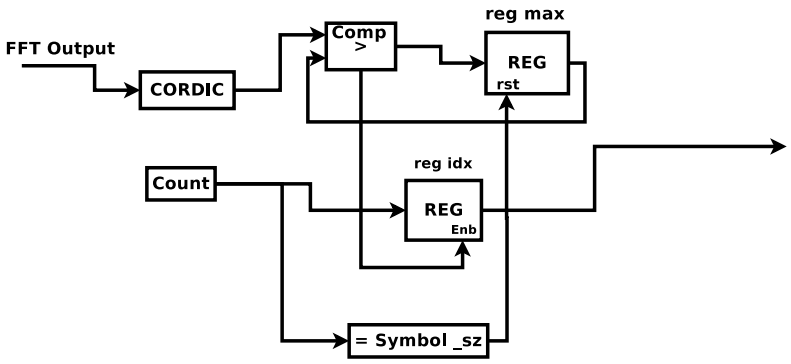

Figure 5.6 – Architecture of the peak searcher.

the ICFO estimated value. Null tones are appended before and after the actual significant data.

As can be seen to estimate/correct the ICFO simple blocks are added to the FFT, saving a considerable amount of hardware resources as a result, since the used method exploits the MR-OFDM structure re-using the available hardware. In the following, implementation results and quantitative results of the savings are shown.

5.2 Implementation Results

After the architecture detailed in Section 5.1 was defined, a golden model was implemented using Matlab high-level language. Subsequently, the integer CFO estimator block was implemented using the VHDL hardware description language and the results of its simulations compared to those of the golden model. In figure5.7 a diagram of the ICFO RTL VHDL verification is shown. The verification process is as follows, first an LTF with fequency offset is generated, then white noise is added. Afterwards, the ICFO is estimated using the VHDL RTL and the Matlab Golden Model. The computed ICFO of the two models (The RTL and Maltab model) is compared against the added ICFO, figure 5.8 shows the probability of success in estimating the ICFO for the Matlab Model and the RTL-VHDL simulation for every MR-OFDM option. The results show a decrease in performance with the sequence size, for Option 1 the performance of the ICFO architecture decreases in almost 30% for -10dB of channel noise, for Option 4 the performance for both models is almost the same. The FPGA verification and validation process is currently under development.

5.2.1 FPGA Prototyping

The proposed design was implemented on a *Cyclone 5* FPGA development kit from Altera. The data samples have 32 bits (i.e., 16 bits In-phase, 16 bits Quadrature).

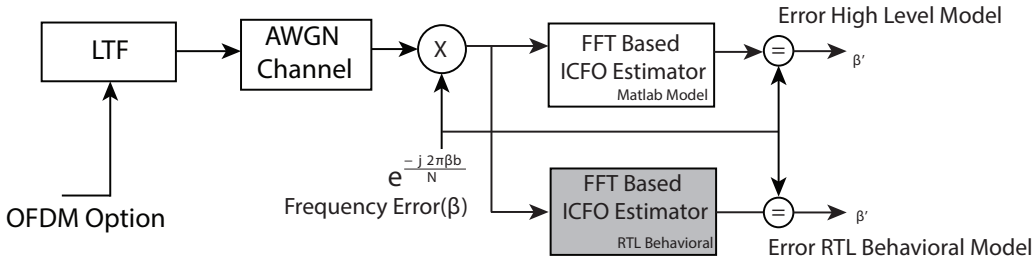

Figure 5.7 – ICFO Test High-Level Model Diagram

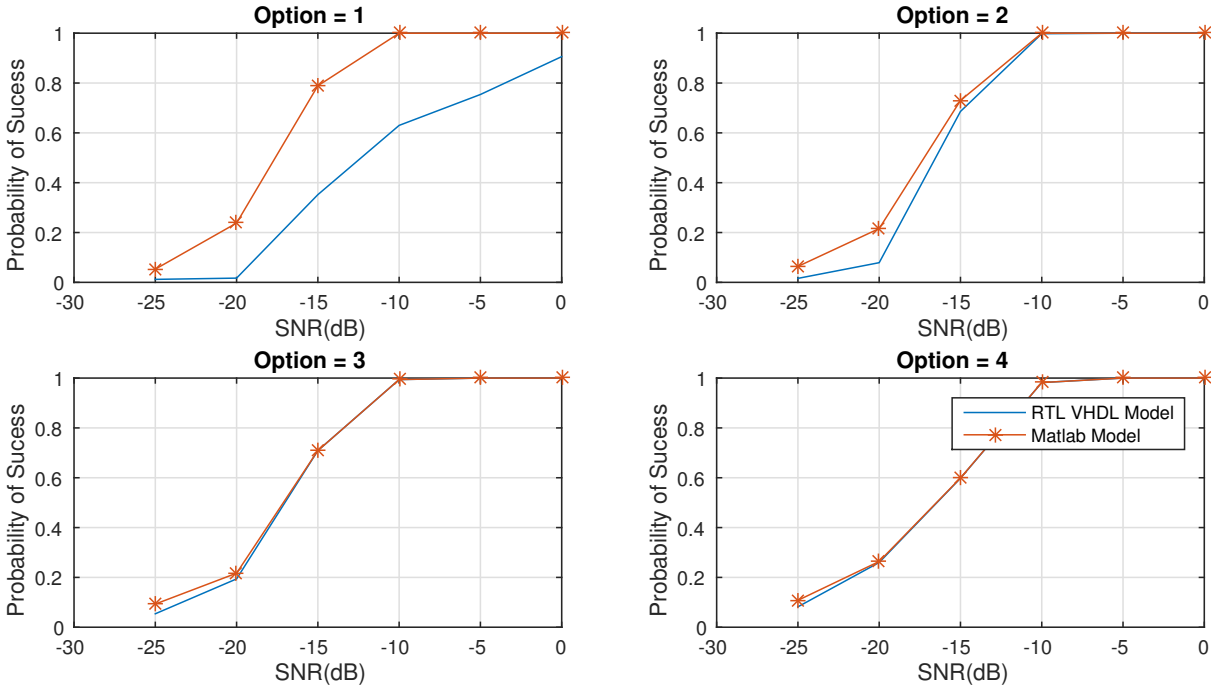

Figure 5.8 – ICFO Verification Results VHDL Simulation and MAtlab Model

Altera's Quartus II version 14.1 was used to synthesize the architecture, maximum frequency, and resource usage are detailed in Table 5.1. Fitter (place and route) detailed resource consumption by entity is shown in Table 5.2, combinational logic, registers used and ALMs occupied are presented.

Table 5.1 – CFO Implementation Results for Altera's 5CGXFC5C6F27C7N

	Fmax	**ALMs**	**DSP Blocks**	**Memory**	**Registers**
ICFO	59.32 MHz	2568 (8%)	0 (0%)	20244 (< 1%)	2737

5.2.2 Synthesis Analysis

As expected the FFT is the most resource consuming block in the entire ICFO as shown in Table 5.2, followed by the peak searcher and this in turn by the multiplier conjugate. The FIFO memory that stores the LTFs while the estimation is being performed (the LTFs are needed by the equalizer for channel estimation) is the least resource

Table 5.2 – Resource Utilization by Entity in the FFT Based ICFO

Entity	ALMs	Combinational ALUT	Registers	Memory
ICFO Top	**2568**	**4208**	**2737**	**20244**
├FFT	1261	1789	1781	4096
├Peak Search	473	909	665	-
├Multiplier/Conjugate	465	798	86	7680
└FIFO Memories	36	62	21	8448

Table 5.3 – CFO Implementation Resource utilization by entity in the MR-OFDM Demodulator

Enity	ALMs	Combinational ALUTs	Registers	Memory (Bits)
OFDM Demodulator	22077.1	31833	23615	1181025
Frame Synchronizer	7057.5	11018	8332	36169
Viterbi decoder	3852.4	3563	3942	0
Fractional CFO Corrector	2650.2	5151	2946	0
Integer CFO Corrector	***2479.7***	***4011***	***2719***	***20224***
Equalizer	2121.7	2451	2915	3160
Despreader	1106.5	790	935	0
Deinterleaver	125.2	217	63	960
Deframer	109.6	163	73	0
Demapper	91.9	174	59	0
CP Remover	51.3	71	41	0
Depuncturer	16.7	31	23	0
PHR Parser	14.3	9	33	0
Descrambler	10.3	18	14	0

consuming block of the entire architecture. On the other hand, the FIFO is the most memory consuming block, 8448 bits are needed to store the two OFDM LTF symbols. The multiplier/conjugate block makes use of 7680 bits to store the LTFs for every MR-OFDM option in time domain, and finally 4096 bits used for the computation of the IFFT/FFT block. Although the ICFO is one of the most resource consuming blocks of the entire MR-OFDM Demodulator (see Table 5.3 for a summary of the resource utilization by entity of the MR-OFDM receiver) its real resource consumption does not include the resources utilization of the FFT block, since this block is needed in the receiver to perform the demodulation, in spite of the ICFO. The resources needed by the ICFO without the FFT/IFFT block are shown in Table 5.4.

Table 5.4 – Resource Utilization by he ICFO without the FFT

Entity	ALMs	Combinational ALUT	Registers	Memory
ICFO	**973.8**	**1769**	**772**	**16128**
├Peak Search	473	909	665	-
├Multiplier/Conjugate	465	798	86	7680
└FIFO Memories	35.8	62	21	8448

To exhibit the advantages of the presented approach, the presented architecture

is compared with an alternative approach. Figure 5.9 shows an alternative architecture for the implementation of the integer CFO block.

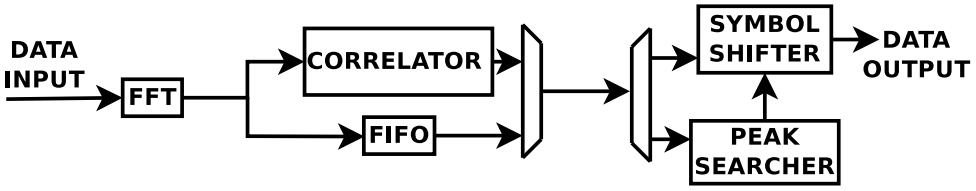

Figure 5.9 – Integer carrier frequency offset estimator/corrector with correlator.

In this approach the LTFs (a stored reference LTF and the received LTF) are correlated in the frequency domain. Therefore, the corrupted LTF is first passed through the FFT, then the cross-correlation is performed between the two LTFs and subsequently the peak value index is taken from the cross-correlation output. The correction is performed in exactly the same manner than it was previously explained, data is shifted according to the index given by the peak searcher. The FIFO memory stores the data while the frequency error estimation is performed. The operations performed by the two architectures are the same, correlate signals, find the peak value index and finally perform the sub-carriers shift to correct the data. The only difference between the two approaches is the correlation operation computing methods, one is performed by means of a correlator circuit, the other takes advantage of the cross-correlation theorem that uses a complex multiplier and reuses the FFT engine. From a hardware point of view the only difference between the two architectures is the multiplier and conjugate block, used in the proposed design, while the other architecture uses a correlator.

In [43], Altera proposes an high performance low-cost Parallel Samples, Parallel Coefficients and Time division multiplexing correlator (PPT Correlator) for wireless applications. The reference sequence used in Altera's correlator exhibits the same format than that of the IEEE802.15.4g in the frequency domain (the LTFs is composed of 0, 1, -1 valued samples), thus, is possible to employ Altera's approach to implement the correlation for our purposes. Figure 5.10 shows the architecture for altera's correlator.

This architecture calculates $n * d$ correlation points together, where n is the number of samples processed together, and d is the number of correlation points computed in parallel. After L/n clock cycles, where L is the length of the correlation sequence, the correlation computation for d correlation points is achieved. n reference samples are stored on n flip-flops, R in Figure 5.10. The reference is then multiplied by $n + d - 1$ samples. Since the reference sequence is composed of only $+1/-1$ samples in the frequency domain, the multipliers can be implemented as an exclusive or gate (XOR) of b bits, being b the bitwidth of the samples. The XOR multipliers output is then fed to an adder tree, as shown in Figure 5.11, the last adder of the adder tree sum all the intermediate results.

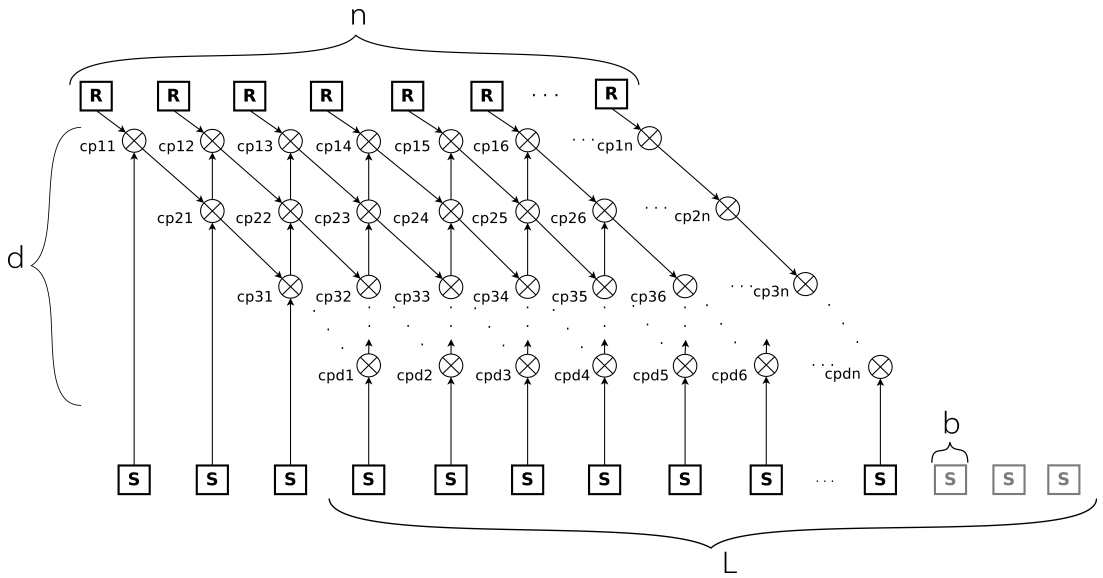

Figure 5.10 – Altera's PPT Correlator Architecture.

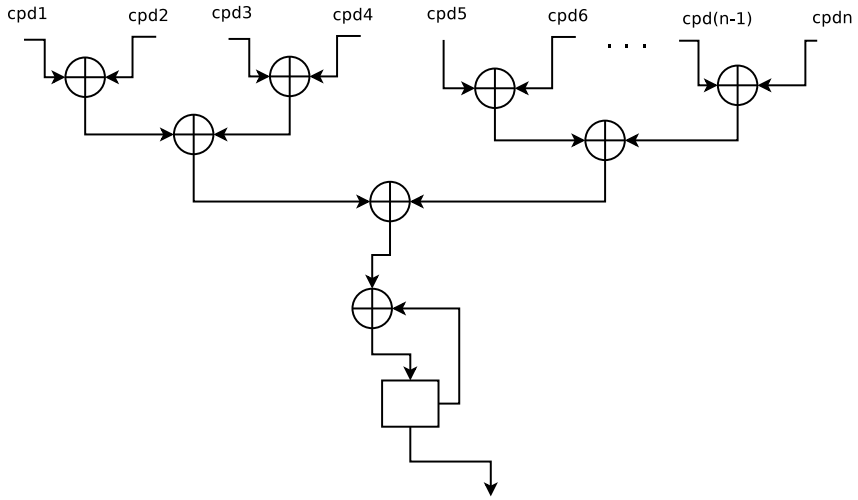

Figure 5.11 – Altera's PPT Correlator Adder Tree.

According to [43] the size of the correlator in logic elements (LE)[1] is given by (5.4):

$$N_{LEs} = n + b * (n + d - 1) + d * [2 * (log_2(L) + b) + \sum_{i=1}^{log_2(n)} \frac{n}{2^i} * (b + i)], \text{ where} \quad (5.4)$$

n = *Number of samples processed together*

d = *Number of correlation points calculated together*

b = *Number of bits per sample*

L = *Length of each shifted correlator sequence*

[1] LEs are the smallest units of logic in an Altera FPGA device architecture. (1 ALM = 2.65 LEs)

Altera's PPT correlator size in LEs according to (5.4) is of 8613 for the maximum symbol size of the MR-OFDM symbol and a single correlation point calculated per clock cycle. For five correlation points per clock cycle, the number of logic elements of the PPT correlator is of 26297. The ICFO's multiplier conjugate block size in ALMs is of 465 (Table 5.2). Each ALM is equal to 2.65 LEs [44]; thus the LE consumed by the multiplier conjugate block of the proposed architecture is of 1232, that means the correlator consumes 7.42 times more than the multiplier conjugate block in FPGA's LE with one correlation point per clock cycle. Changing the number of samples that the correlator process together, n to 16, with only a single correlation point calculated together the number of logic elements that the PPT correlator consumes is of 1112 LEs, which is close to the number of LEs consumed by the complex multiplier. The variation of LEs with the number of samples computed together and the number of correlation points calculated in parallel with a sequence length L = 128 and bitwidth of $16b$ per sample (a complex number of $32b$) is shown in Figure 5.12.

Table 5.5 – Resource comparison between complex multiplier and correlator

Architectures	Blocks	**ALMs**
Proposed ICFO	***Multiplier Conjugate***	**465**
Altera's Correlator [3]	**Min Parallelism (1 point)**	**3250**
	Parallelism of 5 Points	**13310**

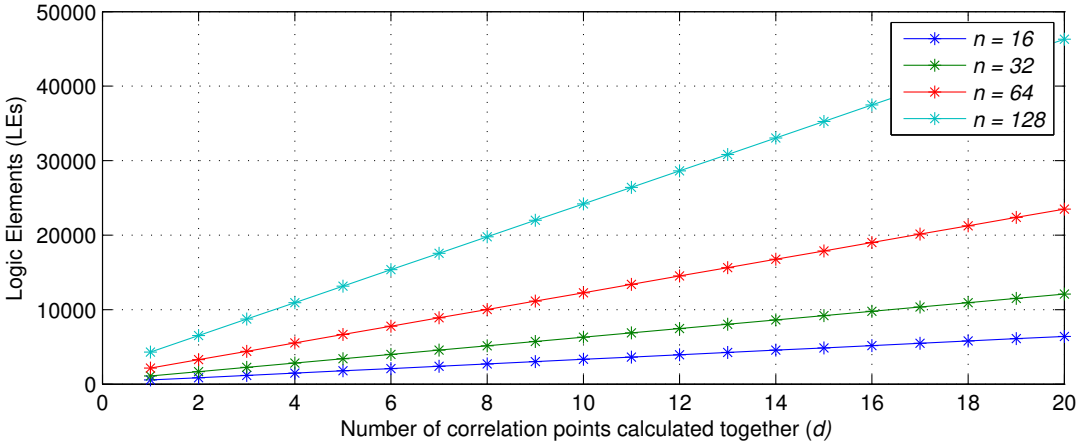

Figure 5.12 – Altera's PPT Correlator LEs.

5.3 ICFO and STO

Since the first step in the OFDM synchronization process is the timing synchronization and at this stage channel impairments, noise and CFO is still present in the signal, perfect timing estimation is always difficult to achieve. Algorithms that estimate the timing offset are not robust and residual time offset is often not entirely corrected.

A more realistic approach to the ICFO estimation is to take into account this residual offset. Figure 5.14 shows an example of the computation of the ICFO estimation for a 128 size cross-correlation with an ICFO of 20 carriers and in the presence of type II STO of 10 samples and white Gaussian noise. As can be seen, the correlation result maximum value is different from the actual ICFO in the signal; this is due to the shifting of the sequence in time. Since the data is shifted regarding the LTF reference, the pointwise multiplication performed in the time domain does not yield the same result as without STO. In the following, a study of the ICFO estimation in the presence of STO is presented, the performance of several methods as well as its hardware implementation costs are presented. All simulations were repeated 1000 times for a fixed SNR point in the performance curve. The power delay profile PDP (a measure of the signal power as a function of the propagation delay, computed as the spatial average of the power complex baseband channel impulse response) of the multipath channel used the simulations is shown in Figure 5.13.

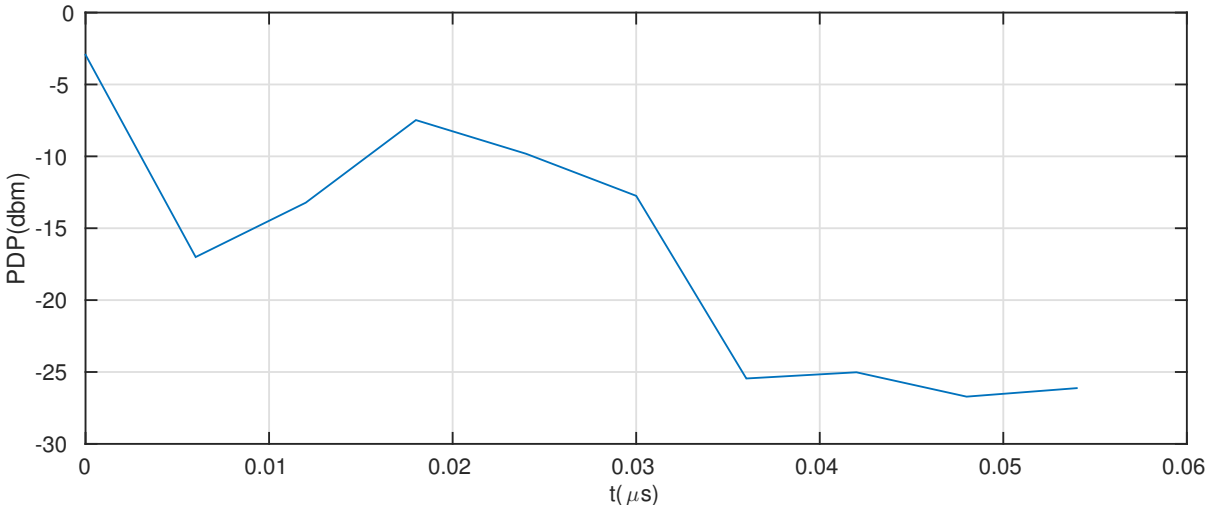

Figure 5.13 – Power Delay Profile of the Multipath Channel.

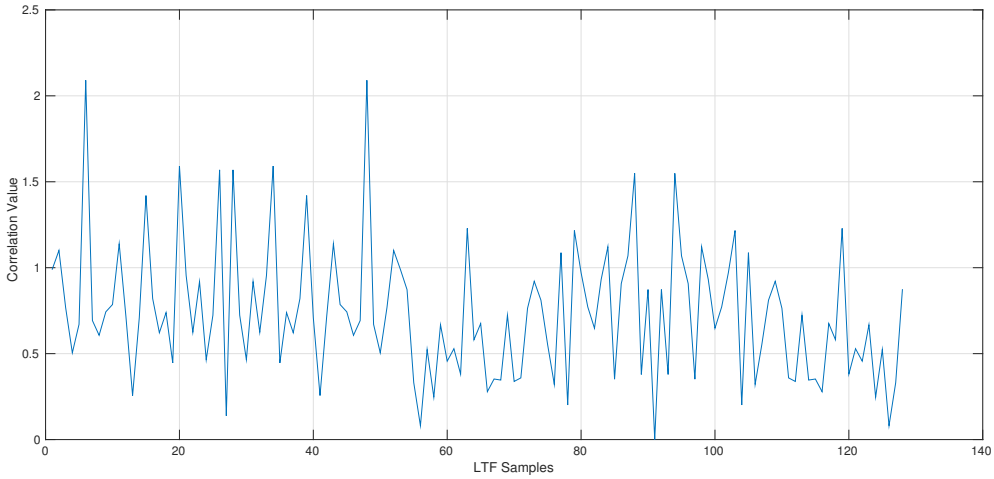

Figure 5.14 – LTF Cross-Correlation output with SFO.

5.3.0.1 Canet Fine STO estimation

An algorithm that tries to estimate this residual time offset is presented in [45]. Canet shows a Data-Aided fine time synchronization method based on the cross-correlation for the IEEE802.11a/g standard, a standard that has a similar PPDU structure than that of the IEEE802.15.4g. Here, fine time synchronization is performed using cross-correlation between the first 32 samples of the LTF sequence in the time domain of the IEEE802.11.ag standard, as in (5.5).

$$C_n = g_{32}^H r_n \tag{5.5}$$

$$r_n = \begin{bmatrix} r_n & r_{n+1} & \cdots & r_{n+31} \end{bmatrix}^T \tag{5.6}$$

$$g_{32} = \begin{bmatrix} LS_0 & LS_1 & \cdots & LS_{31} \end{bmatrix}^T \tag{5.7}$$

Where r_n is the received signal, and g_{32} are the first 32 samples of the LTF sequence. The result of the correlation for a fine STO of 10 samples, a signal without ICFO and 20dB of noise is shown in Figure 5.15. Clearly, the correlation output shows a peak value at sample *correlation size - STO*, sample 22 in this case. Probability of success for a noisy channel and type II STO (section 2.1.2) for a 128 size symbol and CP 1/4, with no ICFO is shown in Figure 5.16 and 5.17. For this simulation, a random STO value was generated for every iteration; the ICFO value is maintained in 0. Two scenarios were tested, in the first (Figure 5.16), the range of the random generated STO is constrained to the CP size (32 samples), that is, the STO varies between 1 sample and 32 samples, on the second(Figure 5.17) the size of the STO is constrained to only 16 samples. It is worth

noticing that this algorithm does not works well for STOs greater than half the CP size, as can be seen from the results, a perfect estimation is only achievable when the STO is between 1 and 16 samples. Although good performance under a noisy and multipath channel is achieved with this algorithm, it does not work with ICFO as mentioned in [26], so it must be compensated before the fine STO estimation.

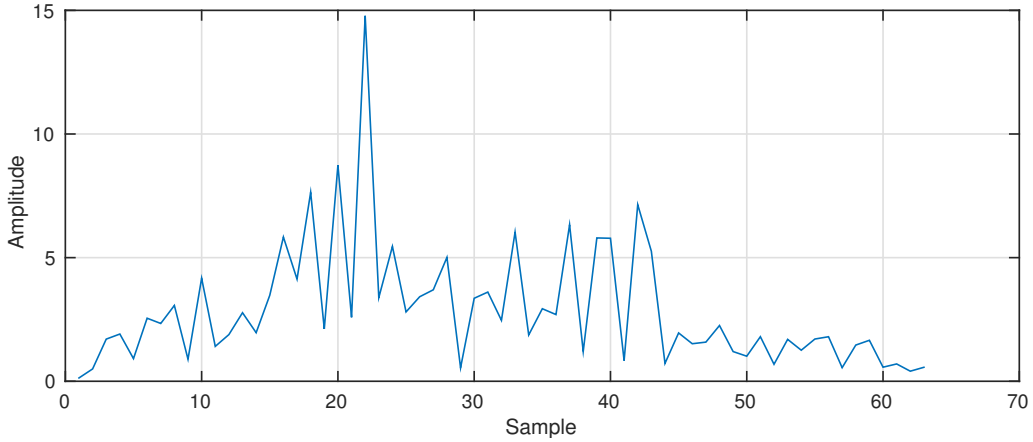

Figure 5.15 – Example of the computation of fine STO by Canet Method.

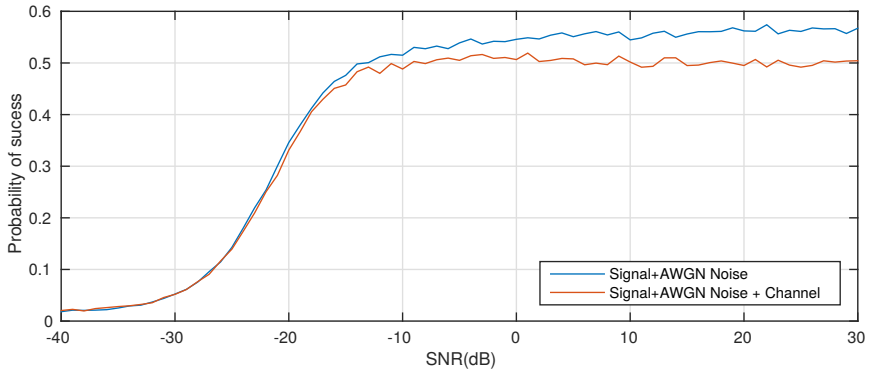

Figure 5.16 – Probability of Success in Canet STO Estimation Algorithm (STO of 32 samples).

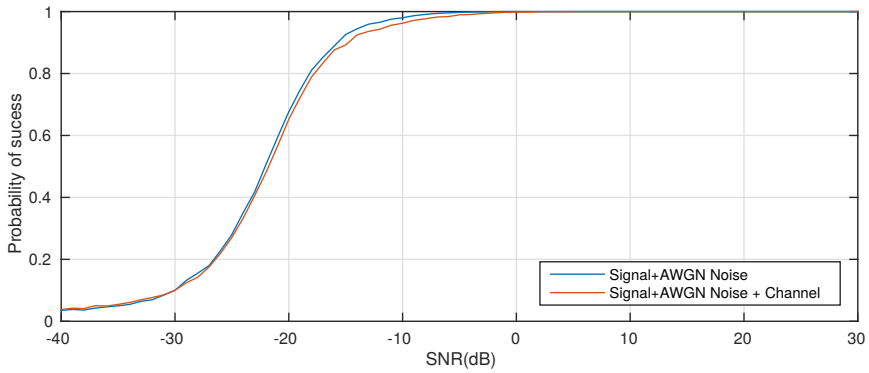

Figure 5.17 – Probability of Success in Canet STO Estimation Algorithm (STO of 16 samples).

5.3.0.2 Non-Data-Aided STO Estimation

Repetitive structures in the received signals can be used to decide the symbol start, as is the case of the CP, that repeats itself since, a copy of the OFDM symbol is pre-pended to itself to avoid ISI. Algorithms that behave this way are called Non-Data-Aided or blind synchronization algorithms since no known sequence is used to estimate the symbol start. Although this kind of algorithms are suitable for continuous transmissions, like the ones based on frames, and the IEEE802.15.4g standard defines a packet by packet oriented protocol, its analysis determines the difficulty of the estimation of the fine symbol start problem.

In Non-Data-Aided or blind STO CP based algorithms, usually, two sliding windows spaced by the symbol length are correlated in time domain. In [21], an STO estimation technique using the symbol CP is presented, it estimates the STO by taking the difference between the two halves of the size of the CP separated by the symbol size. (5.8) shows the mathematical operation, where, y_l is the received noisy signal, N_{cp} is the CP size, N the symbol size and $\hat{\sigma}$ is the estimated time offset from a set of Σ possible values. Therefore, when the difference between the two halves is minimum the similarity between them is maximum yielding the starting point. This algorithm shows immunity to CFO. Figure 5.18 shows an example of the square difference algorithm for a time offset of 10 samples for noisy symbol, with an SNR of 20dB. Here, symbol size is of 128 samples and CP size 32 samples.

$$\hat{\sigma} = \arg\min_{\sigma \epsilon \Sigma} \{ \sum_{n=0}^{N_{cp}-1} (|y_l[n+\sigma]| - |y_l^*[n+\sigma+N]|)^2 \} \tag{5.8}$$

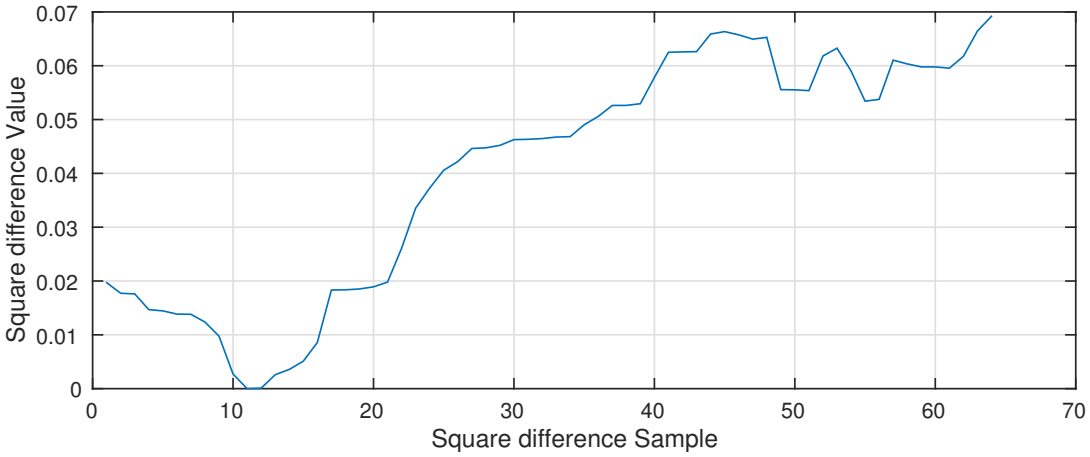

Figure 5.18 – Example of the results of the square difference algorithm.

Another blind algorithm that shows immunity to CFO is presented in [46], it estimates the starting point by using the auto-correlation function. The mathematical operation is shown in (5.9). In this approach, when the correlation is maximum so is the

similarity between the halves. Figure 5.19 shows the results of the ACF algorithms for a time offset of 10 samples and a noisy signal with SNR of 20dB. A symbol size of 128 and 32 samples of CP are used.

$$\hat{\sigma} = \arg\max_{\sigma \epsilon \Sigma} \{ | \sum_{n=0}^{N_{cp}-1} (y_l[n+\sigma] y_l^*[n+\sigma+N]) | \} \tag{5.9}$$

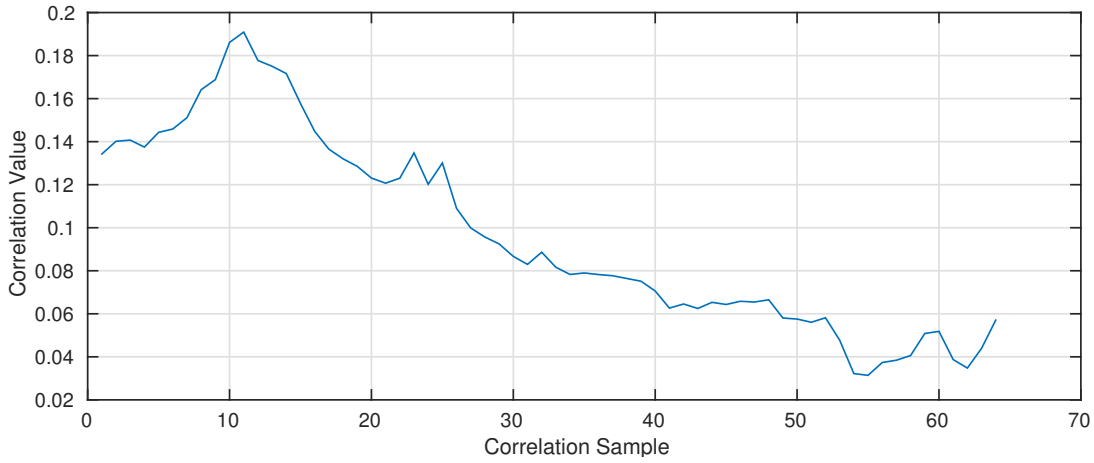

Figure 5.19 – Example of the results of the auto correlation function algorithm.

Figure 5.20, shows the probability of success for the CP based methods of (5.8) and (5.9) for a noisy channel in presence of ICFO for all four MR-OFDM Options. In this scenario a symbol taken after the SHR was used, that is all the estimations where performed adding noise toa random generated symbol, then equation (5.8) and (5.9) where used to estimate the STO. The correlated sequences sizes is of 1/4 symbol size, the size of the CP. Again, a random value of STO between 0 and the CP size is generated for every iteration, as well as a random value of ICFO between 0 and the symbol size. It can be seen from Figure 5.20 that the method based on the square difference shows better performance than that of the ACF method, in fact the ACF fails at achieving an optimal estimation even for small noise level or high SNR values. As can be seen from Figure 5.18 and Figure 5.19 a plateau is found at the minimum value of the curve for the square difference algorithm, which means that the differences between estimated values is small between adjacent σ values, thus, the minimum value is affected greatly by noise, similar behavior can be seen for the ACF algorithm, adjacent values in magnitude of the autocorrelation show smalls differences, thus hindering the estimation.

5.3.0.3 Fine STO estimation in frequency domain

A fine synchronization algorithm also presented in [21], estimates the STO in frequency domain using training sequences. According to (2.23, refer to Chapter 1), the time error introduces a phase shift that is constant, thus, if $Y[k] = Y[k-1]$, the

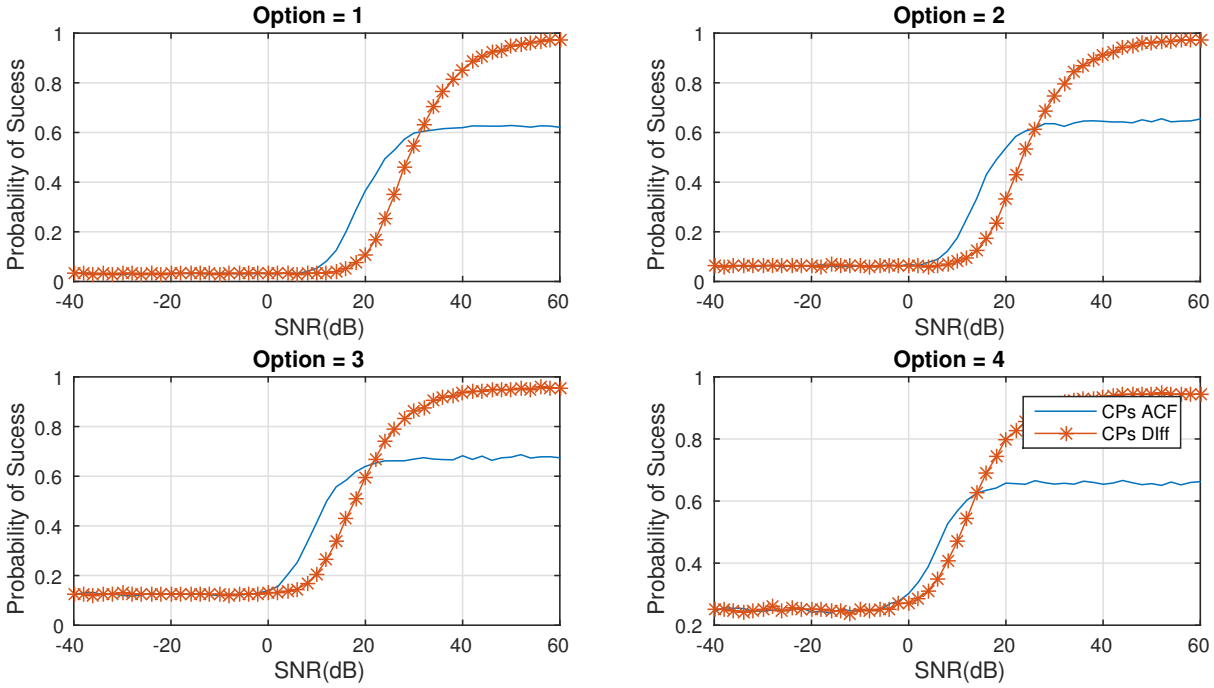

Figure 5.20 – Probability of success of the CP based methods in a noisy channel for all OFDM Options.

phase difference between adjacent carriers must be equal. (5.10) shows the mathematical operation.

$$\hat{\sigma} = \frac{N}{2\pi} \arg\{\sum_{k=1}^{N-1} (Y[k]Y^*[k-1])\} \tag{5.10}$$

Since the training sequence samples are not completely equal (as required by (5.10)) in the LTF field of the synchronization header for the IEEE802.15.4g standard, a slightly modification to (5.10) must be made. Taking the absolute value of every component of the complex samples yields (5.11)

$$\hat{\sigma} = \frac{N}{2\pi} \arg\{\sum_{k=1}^{N-1} |\Re(Y[k]Y^*[k-1])| + |\Im(Y[k]Y^*[k-1])|\} \tag{5.11}$$

The probability of correct STO estimation for the Data-Aided phase difference in frequency domain for all MR-OFDM Options is shown in Figure 5.21. In this scenario, an STO generated randomly in time domain is added to the LTF training sequence, then the FFT is performed over this sequence, and finaly the STO estimated according to (5.11). This process is repetead for every iteration in SNR point in the performance curve. For this algorithm the probability of success decreases with the symbol size, the smaller the symbol size (the LTF size) the better the estimation rate.

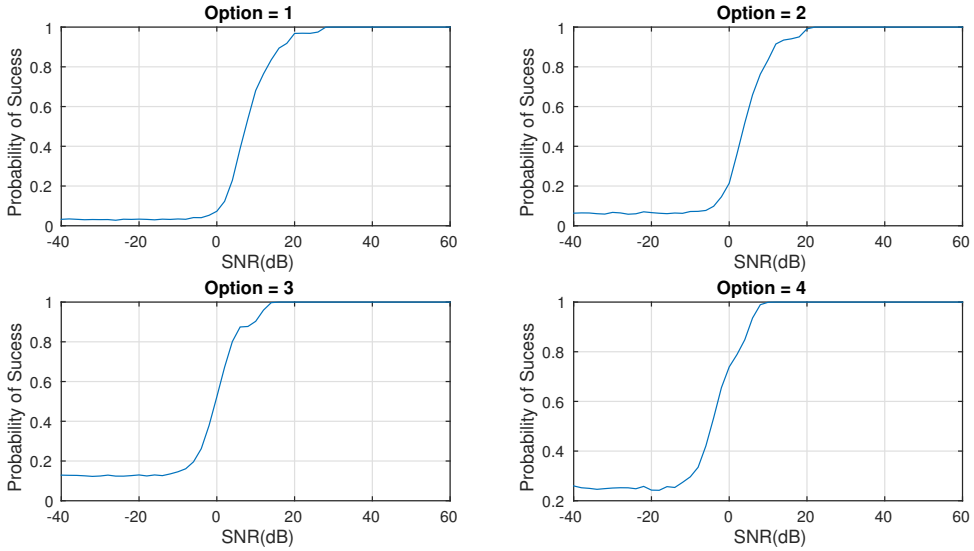

Figure 5.21 – Probability of success in estimating the STO for the Data-Aided frequency domain algorithm for all MR-OFDM Options.

The probability of success of the three STO estimation methods for a noisy channel is shown in Figure 5.22 for all MR-OFDM Options. The phase difference in frequency domain (the Data-Aided algorithm) shows better results than the CP based ones (the Non-Data-Aided algortihms). The phase difference in frequency domain attains a perfect residual timing estimation for a SNR greater than 10dB for the MR-OFDM Option 4 and SNR greater than 20dB for MR-Option 1, while only the square difference algorithm achieves close to optimal optimization after only 40dB of SNR.

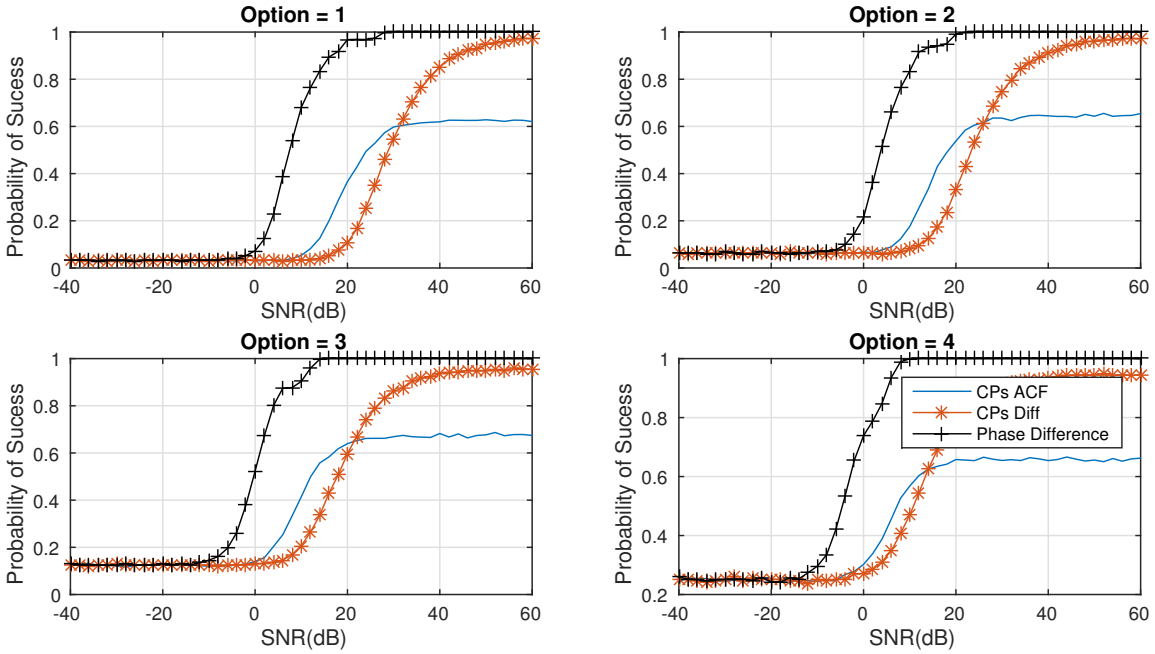

Figure 5.22 – Probability of success in estimating the STO for all MR-OFDM Options.

Another factor that hinders the proper demodulation of the incoming signals

that has not been taken into account is the signal degradation caused by the multipath channel. Simulation results for a multipath channel are shown in Figure 5.23 and Figure 5.24 for the square difference and the ACF based algorithm respectively. Both CP based algorithms show degradation in performance for a noisy and multipath channel, since both algorithms depend on the CP to estimate the STO.

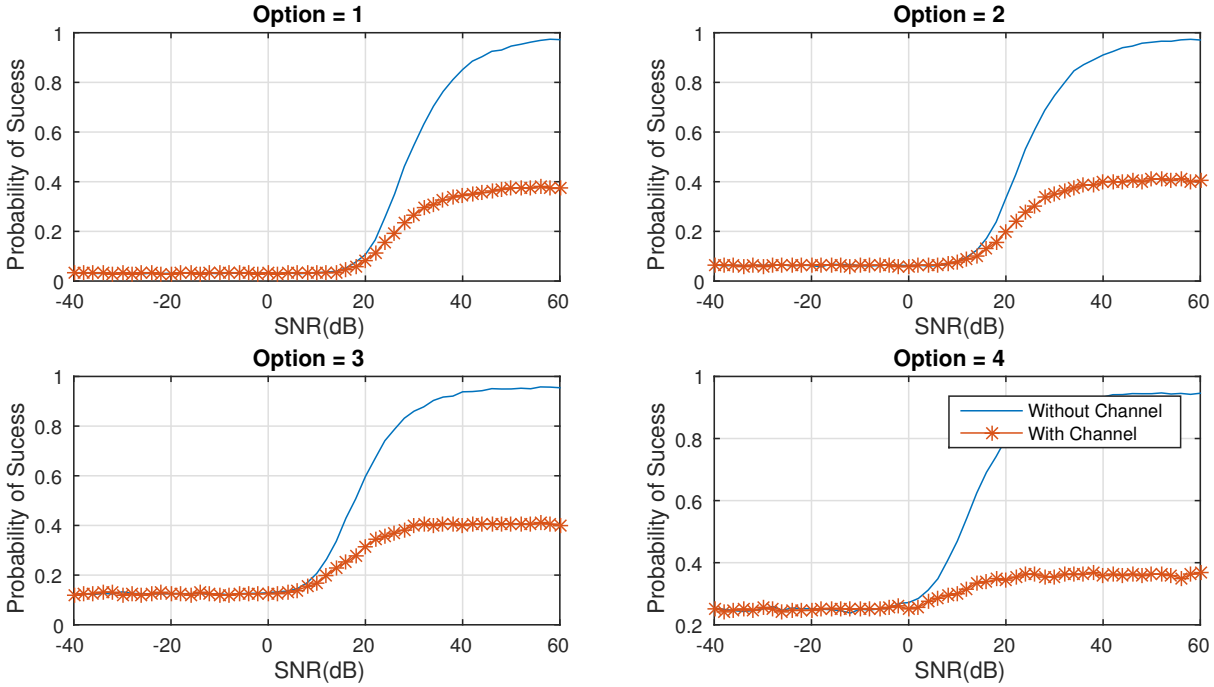

Figure 5.23 – Probability of success in estimating the STO for CP based square difference under a multipath Channel.

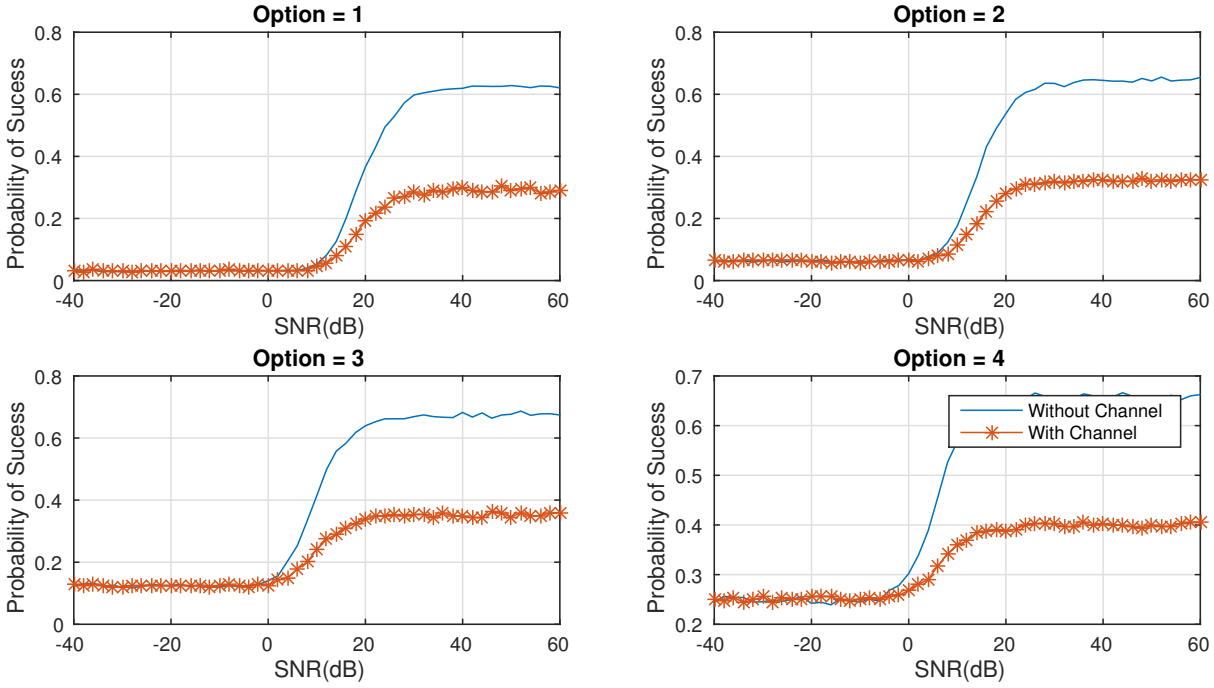

Figure 5.24 – Probability of success in estimating the STO for CP based ACF under a multipath channel.

Results for the phase difference in the frequency domain are shown in Figure 5.25. This algorithm fails at estimating the residual STO. Not only a degradation in the estimation is shown as in the CP based algorithms, but a completely inexact estimation.

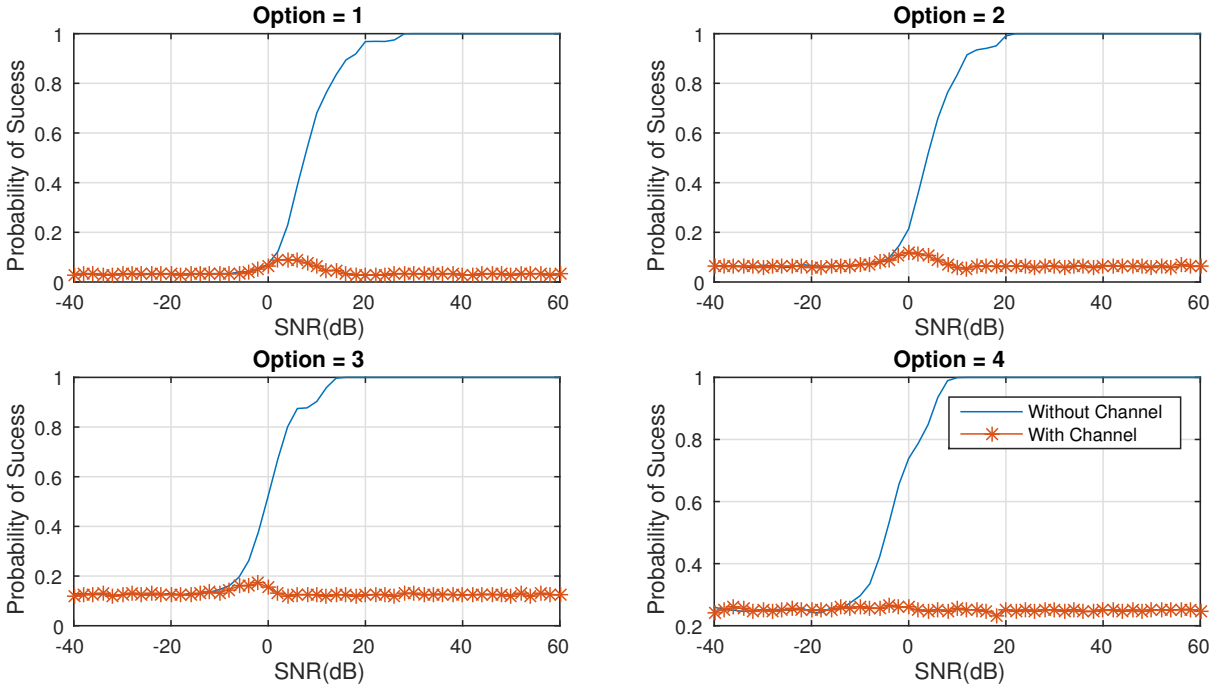

Figure 5.25 – Probability of success in estimating the STO for the phase difference in frequency domain algorithm under a multipath channel for all MR-OFDM Options.

Results from the estimation of STO show very poor performance for a noisy and multipath channel, neither the CP based approaches nor the phase difference in frequency domain approach achieves a perfect STO estimation, even for low level noise. The square difference algorithm shows the best performance among the three algorithms.

5.3.0.4 ICFO Estimation in the frequency domain with immunity to STO

Another approach to solving the problem is to find methods that estimate the ICFO even with the presence of the residual timing error, which is the case of the work presented by BoAi in [17]. Here, a Data-Aided algorithm in the frequency domain which is not affected by timing errors is presented. The operation is as described in (5.12)

$$\hat{M} = \max_{l \epsilon L} \{ \sum_{k=1}^{W} |Y_{m,k} X^{*}_{m,k+l}| \} \tag{5.12}$$

Here, W is the number of continuous samples, $Y_{m,k}$ is the k^{th} sample of the m^{th} received noise corrupted OFDM symbol after the FFT, $X_{m,k}$ is the uncorrupted reference in frequency domain, L is the estimation range, and l the sliding window.

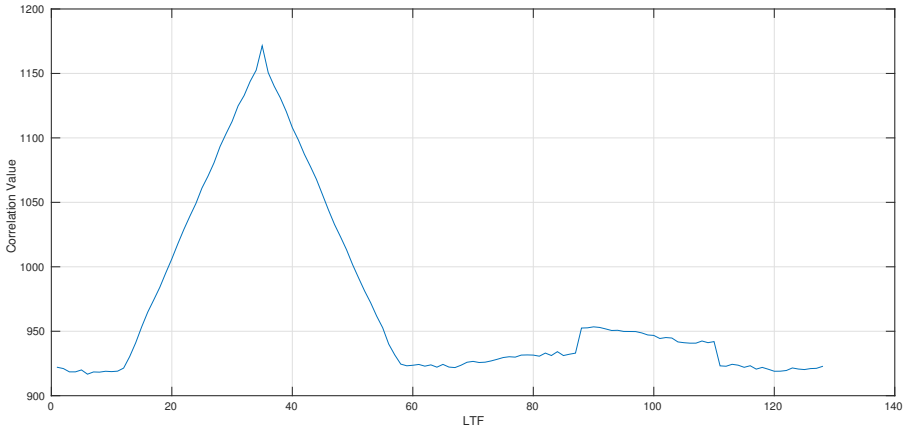

Figure 5.26 – Example in estimating a ICFO of 35 carriers for the BoAi Proposed Method.

Figure 5.26 shows the result of the BoAi algorithm for a residual STO no greater than the CP size, additive white Gaussian noise and an ICFO of 35 subcarriers for a symbol size of 128 samples. The result of the correlations shows that a maximum peak value is found at sample 35, the actual ICFO in the signal.

5.3.0.5 FFT Based ICFO Estimation with immunity to STO

Although computing the cross-correlation using (5.2) between the training sequences in the presence of a small STO does not yields the actual ICFO, the correlation must yield a maximum value when the similarity between two signals is maximum. In this sense, the ICFO must yield a maximum value from within a set of maximum of correlations values. Finding the correlations of the shifted signals with STO must yield a maximum when those signals similarities are maximized, that is when the STO is equal to zero. (5.13) shows the mathematical operation. Where, $y(n)$ is the received long training sequence in the time domain, $x(n)$ is the reference training sequence also in the time domain and $\odot$ implies pointwise multiplication between y and x. An example of the maximum of correlations is shown in Figure 5.27, where a 128 samples symbol is used, an STO of 7 samples and an ICFO of 23 subcarriers. Clearly, a peak value if found for the eighth correlation (when the STO becomes 0) in the 23rd sample of that correlation.

$$\hat{\theta} = \max_{\theta}[\max \mathcal{F}[y(n+\theta) \odot x^*(n)]] \tag{5.13}$$

Performance of the BoAi algorithm and the maximum of correlations method for all MR-OFDM options in the presence of only white noise is presented in Figure 5.28. As can be seen, the maximum of correlations outperforms the Boai algorithm by approximately 15dB. It reaches a perfect ICFO estimation at approximately -10dB, while the BoAi algorithm at approximately 5dB. Figure 5.29 shows the performance results of both

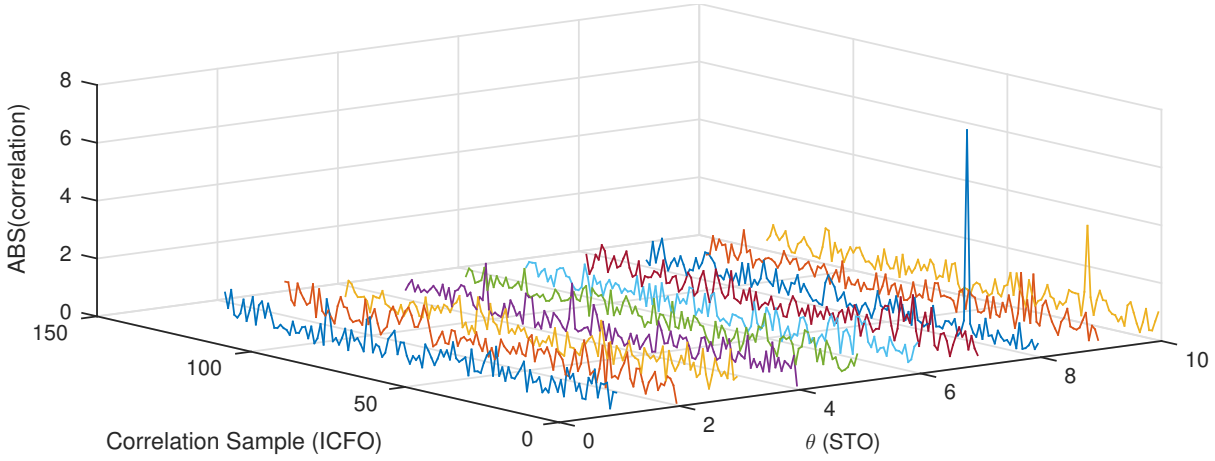

Figure 5.27 – Example of the maximum of correlations method for estimating the ICFO.

algorithms without a multipath channel (WoC), and with a multipath channel (WC), here the results are almost the same as with only white noise, the multipath channel does not affect the estimation in a considerable amount. It is worth noticing that for the option with the least number of carriers, MR-OFDM Option 4, the BoAi algorithm fails to estimate the ICFO correctly, this is not the case for the ICFO.

From the results obtained for the STO and ICFO estimation, we can conclude that the blind techniques tested show much worse performance than that of the Data-Aided techniques for a noisy channel. Also, a robust estimation of the STO is difficult to achieve at this point in the synchronization process, because channel impairments, as well as frequency errors, are still present in the signal making it difficult to estimate a perfect symbol time offset even for high SNR values. A small residual time offset that falls in the CP of the symbol can be corrected by the one tap equalizer [47], so perfect estimation is not entirely needed in the OFDM synchronization process.

5.4 STO and ICFO hardware implementations

To evaluate the tradeoff between performance and implementation, analysis of the cost of implementation of the previously presented methods for estimating the STO and ICFO are presented. Area and throughput are the main parameters to consider for the hardware architectures presented.

5.4.1 CP Based Architectures

As it was shown on Section 5.3.0.2, the CP of the OFDM symbol can be used to perform a not so robust fine time estimation. Two methods where presented employing similar operations to estimate the STO, one is based on the auto-correlation function

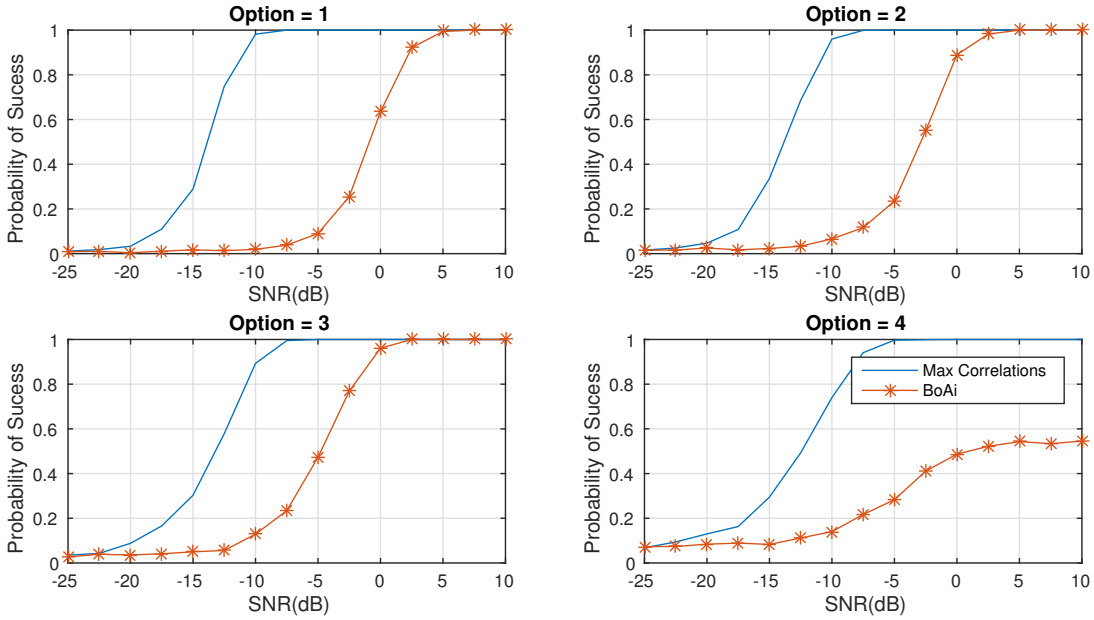

Figure 5.28 – Probability of success in estimating the ICFO for the BoAi and maximum of correlations Proposed Methods for all OFDM Options.

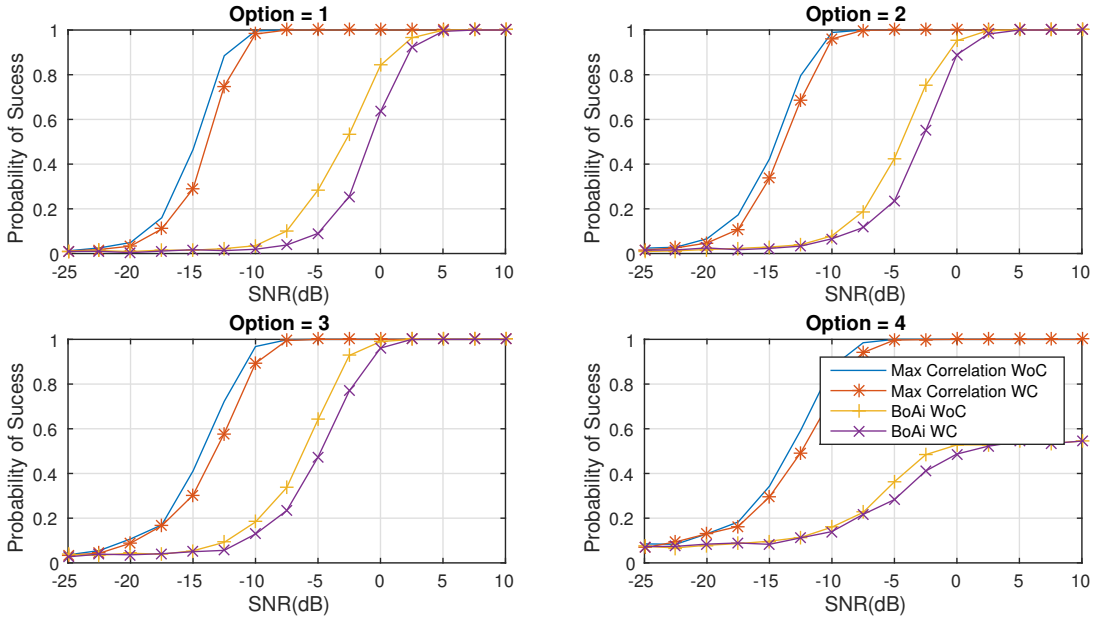

Figure 5.29 – Probability of success in estimating the ICFO for the BoAi and maximum of correlations Proposed Methods for all OFDM Options under a multipath channel.

(ACF) while the other takes the difference between two CP size spaced windows. The implementations of those methods are very similar in logic, as we will see.

Figure 5.30 shows the architecture for the ACF CP Based algorithm, its computation is as follows. For every clock cycle, two samples from two windows separated by the symbol size are multiplied, the result is sent to an adder that sums the current results with the previously computed one, the sum is saved in a register. CP clock cycles after,

the current sum stored value is sent to the CORDIC operating in circular coordinates in vectoring mode, its output is the module of the sum. Also every CP clocks cycles, a σ counter (Count σ in Figure 5.30) is incremented by one. The computed module is then compared with the previous greater module found, if the current value is greater than the previous one, its value is stored in the register *reg ACF* and the current count of the Count σ counter is saved. The process is repeated shifting the input sequence until the estimated range is reached; at that moment the current stored value of the sums counter is the estimated STO.

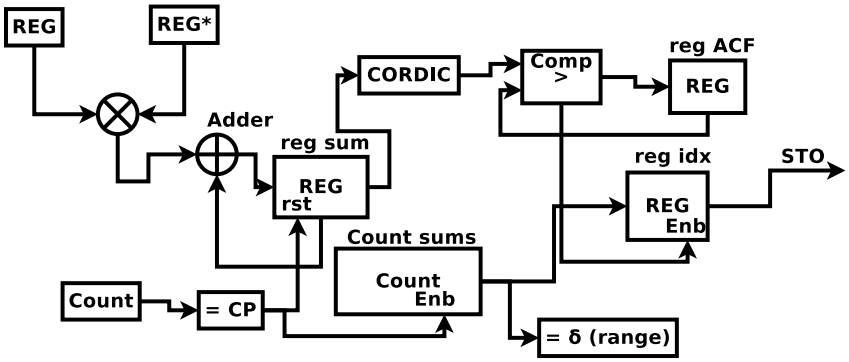

Figure 5.30 – Serial Architecture for the CP Based ACF algorithm

The architecture in Figure 5.30 is a serial architecture, therefore, it computes one operation per clock cycle. The total delay for the maximum CP size for the MR-OFDM option 1 and an estimation range of 10 (a maximum STO of 10 samples) is of 320 clock cycles.

Figure 5.31 shows the architecture of the square difference algorithm; the functioning of this architecture is similar to that of the ACF. For this algorithm accumulator followed by a comparator compute the minimum value among the many shifted squared difference values of the symbol size spaced sequences. In this case, the module is computed first. Two CORDIC blocks (one for each sample, with one of them conjugated) compute the modules input, modules that are then subtracted and its result finally multiplied by itself. This result then goes to the same structure as in the ACF architecture to compute the STO. The time spent in the computation is the same as in the ACF architecture, i.e., CP times the estimation range, for MR-OFDM option 1, 320 clock cycles.

The delay in the computation time of both architectures can be reduced by parallelizing the multiplication for the ACF and subtraction for the square difference algorithm, an example is shown in Figure 5.32 for the ACF architecture. There, the registers are replaced by a series of n registers connected that feed $n/2$ multipliers (4 in the example) and these registers, in turn, feed a tree of $n/2 - 1$ adders. The delay is reduced by $CP/(n/2)$. For the squared difference algorithm the multipliers are replaced by subtractors and multipliers, one pair for each sample.

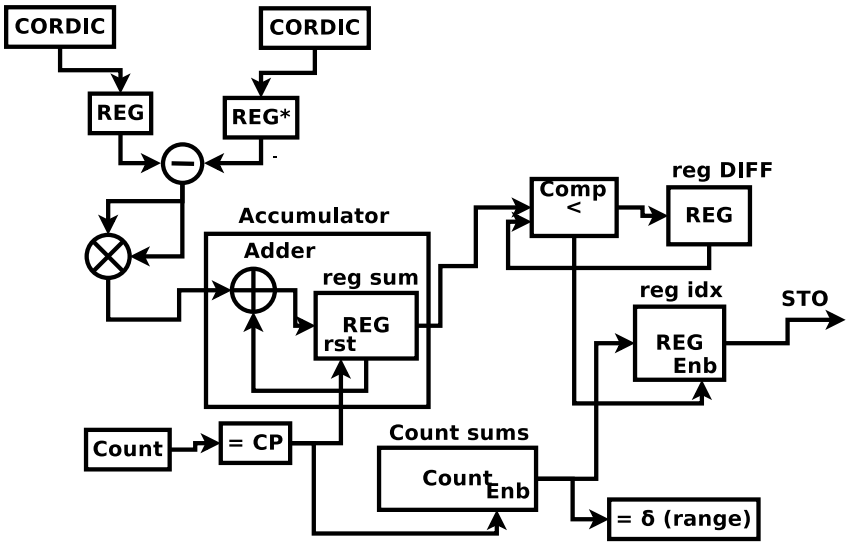

Figure 5.31 – Serial Architecture for the CP Square Difference algorithm

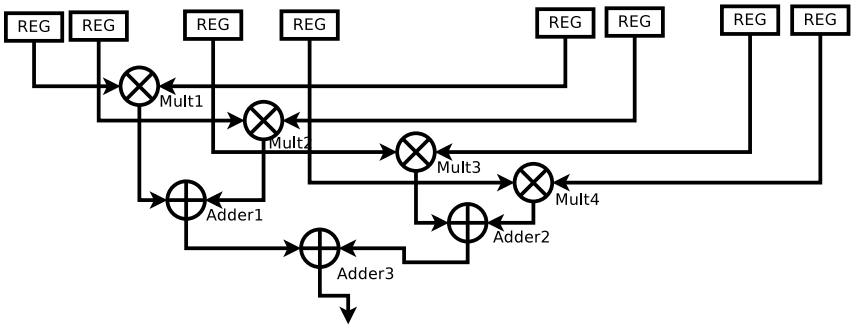

Figure 5.32 – Parallel multipliers in STO ACF architecture

5.4.2 Fine STO estimation in frequency domain Architecture

Figure 5.33 shows the architecture of the fine STO estimator in the frequency domain according. Its functioning is as follows, first each carrier of the OFDM symbol in the corrupted LTF is multiplied by its adjacent conjugate subcarrier, yielding its phase difference. The absolute value of each component of the complex signal is computed and the results accumulated for $symbol_size$ samples. Finally, the angle of the sum is computed using the CORDIC, and the result multiplied by $N/(2 * pi)$, this output is then the STO. The delay of this architecture is of one OFDM symbol, e.g., 128 clock cycles are needed to compute the STO for the MR-OFDM option 1. It is worth noticing that this architecture shows poor performance under a multipath channel; therefore additional hardware is needed to estimate the channel and compensate its contribution to the signal phase in the frequency domain in order to accurately estimate the STO.

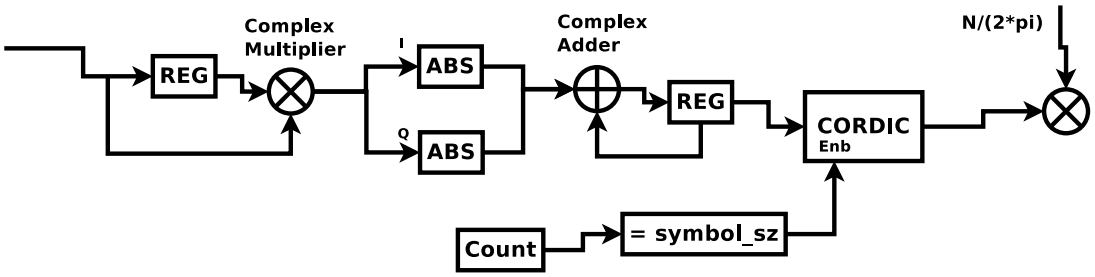

Figure 5.33 – Architecture for the Fine STO estimator in frequency domain

5.4.3 BoAi Algorithm Architecture

Figure 5.34 shows the BoAi implementation for the ICFO. It has some similarities with the CP based implementations. An accumulator and a register saves the current count and sends data to be compared to previously accumulated values. The computation ends when the estimation range is reached; at that moment the ICFO value is the value stored in the index registers, a registry that is updated every time a maximum value is found.

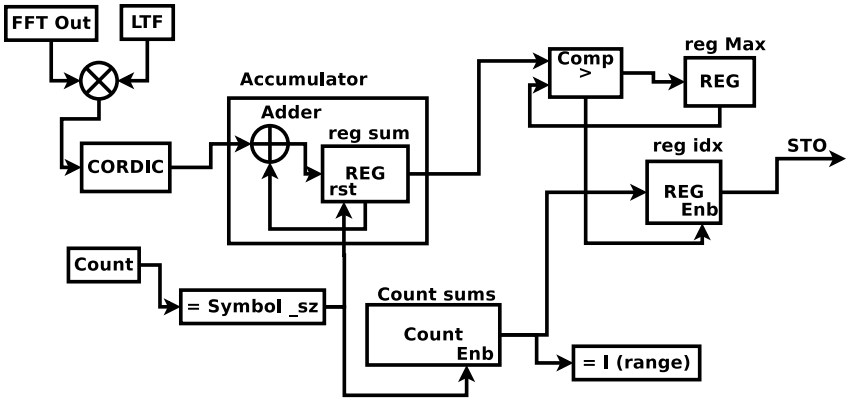

Figure 5.34 – BoAi algorithm implementation

Since the correlation of this algorithm is performed in the frequency domain, where the LTF is composed of +/-1s and 0s, a simpler implementation can be accomplished. The simplification resides in the multiplier at the input of the architecture, since the multiplication is performed in the frequency domain it becomes a 0 or $+1/-1$ multiplication, in hardware that can be accomplished by a series of XOR gates, one for every bit of the sample, followed by an adder.

The delay of this architecture if of 128x$range$ clock cycles. For example, for an estimation range of 10 subcarriers, 1280 clock cycles are required. As with the CP based architectures the process can be parallelized, XOR multipliers and CORDICs blocks in parallel (one for each sample) allow the reduction of the delay at the cost of more hardware. A parallelization of 4 samples is shown in Figure 5.35. The outputs of the CORDICs blocks are added and sent to the accumulator; the rest of the computation process remains the

same, its values are accumulated and compared to extract the maximum value. The downside of this parallelization is the use of the CORDIC block, which uses lots of resources. The number of clock cycles is reduced to $(SymbolSize)/(parallelinstances)$x$range$, for the example shown in Figure 5.35 for a estimation range of 10 carriers, $128/4 * 10 = 320$ clock cycles. Even though this implementation may use more sources than the STO implementations alone, the estimation of the ICFO by this method uses no additional hardware. That is, the costs in hardware to estimate the ICFO using the STO methods is greater than the costs in hardware needed by this method, since those architectures require the additional hardware of the ICFO shown in Figure 5.4.

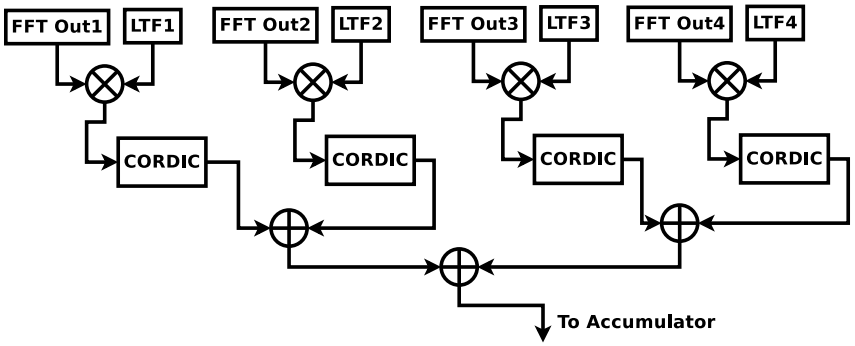

Figure 5.35 – BoAi algorithm implementation in parallel

5.4.4 Maximum of correlations Algorithm

Only a slightly modification to the implementation presented in Section 5.1 is performed for this architecture. At the input, a buffer of size $symbolsize + STOrange$ stores the symbol plus a small group of samples, corresponding to the STO ranges to estimate, for every correlation computed the buffer shifts one sample off the buffered sequence. The symbol size sequence is then point-wise multiplied in the time domain by the uncorrupted reference LTF and the result sent to the FFT block, the FFT output now goes to the peak searcher, that finds a local maximum, then the buffered sequence is shifted, and another correlation is computed. The local maximum of the computed correlation is then compared with the local maximum of all the correlations previously computed, the index of the maximum of local maximums (the global maximum) is the ICFO. The previously peak searcher shown previously in section 5.1 is changed to compute the maximum of maximum values. The new peak searcher block is shown in Figure 5.36.

To find the global maximum (the maximum of the maximum of all the computed correlations) the peak searcher compares the current FFT sample output with the previous sample. Every time a maximum value is found within the FFT samples, its value is stored in the register *reg local max*, in parallel, the local index of where this maximum value was found is stored in the register *reg idx max local*. When the samples count reaches

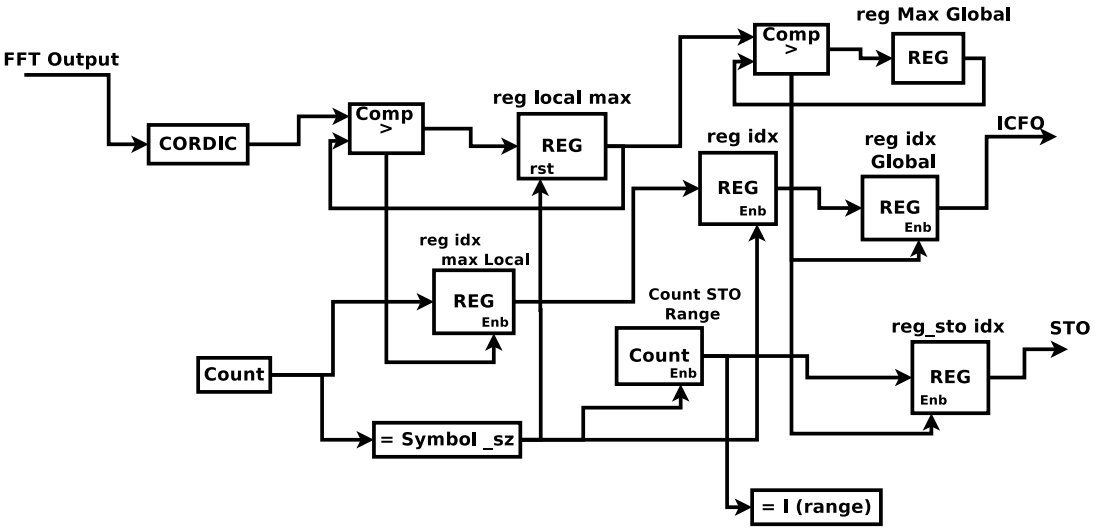

Figure 5.36 – Peak searcher for the FFT based maximum of correlations algorithm

the symbol size, the value of the maximum register along with the value of its index is passed to a global comparator that compares the current maximum local value with the previous partial global found. The index value is stored in an auxiliary register, if the current value is greater than the previous one, the current value is stored, and the global index register is updated with the auxiliary register value. The process ends when the STO range value is reached.

Since the computation of the ICFO using this method involves the FFT computation, the delay of the computation includes the delay of the FFT, which slows down the process. For this method the delay is given by (5.14).

$$delay_{maxcorr} = (delay_{FFT} + Symbol_{sz}) * STO_{range} \tag{5.14}$$

$$delay_{fft} = \frac{FFT_{sz}}{2} \log_2 FFT_{sz} \tag{5.15}$$

For an STO of 10 samples and MR-OFDM option 1, the delay of this architecture is of 5760 clock cycles. The delay of this architecture is the greatest of all the implementations presented. However, some modifications can be done to reduce this delay:

- *Pipeling the FFT* : with this method the delay of the computation is reduced to $Symbol_{sz} * STO_{range}$. Although it may seem that the advantages of the FFT implementation presented previously are lost with the use of a pipeline architecture, the use of a CORDIC based butterfly that eliminates the use of ROM memories to store twiddle factors can still be used in a pipeline structure. The only disadvantage of the

pipeline approach would be hardware reuse. For FFT sizes smaller than 128 much of the hardware occupied by the FFT would remain without utilization. It is worth noticing, that blocks with very large delays halt the system, requiring buffers to compensate those delay which is the case of the current FFT serial implementation. The pipeline version of the FFT could alleviate the need of those buffers, at the cost of more area and possibly more power. A pipelined FFT would allow implementing the presented method without much delay, and with advantages over other ICFO implementations, for example, the ICFO range that this implementation allows is of a symbol size (128 for OFDM option 1), whereas the BoAi implementation requires $128 * 128$ to estimate the same range of ICFO. The analysis of the requirements and the benefit of the pipeline architecture in the whole MR-OFDM system is still an ongoing task. Power is an important variable in determining the advantages and disadvantages of the pipeline FFT implementation.

- *Modules running at different clock frequencies* By running different modules at different clocks frequencies, a reduction in blocks delays can be achieved. If the FFT block runs at a higher frequency than that of the rest of the design, the delay of this block can be compensated, this, of course, has implications in power consumption.

Tables 5.6 and 5.7 show a summary of the resource consumption of every architecture presented in FPGA. The values shown are estimations of the area consumption of the architectures for comparative purposes taken from raw, simple implementations, no rigorous verification was performed on these blocks. It is worth noticing that the values are shown in Tables 5.7 are for only the STO estimation, the ICFO estimation with immunity to STO using these methods requires the use of the ICFO presented in section 5.1. Table 5.8 shows the area consumption required to compute the ICFO for all proposed methods. As can be seen from Table 5.7, blind STO algorithms in time domain consume much more hardware than the STO method presented in the frequency domain.

Table 5.6 – Resource Utilization by Entity in the ICFO architectures

Entity	Combinational ALUT	Registers
Maximum of Correlations	**1579**	**642**
├─Peak Search	700	642
└─Complex Multiplier	879	0
BOAI Top	**738**	**662**
├─CORDIC	653	574
└─XOR Multiplier	33	32

The results presented show that the choice of an optimal ICFO estimator is carried on plenty of variables to analyze, area consumption, estimation range, good performance under a noisy channel as well as immunity to multipath channel and STO

Table 5.7 – Resource Utilization by Entity in the STO architectures

Entity	Combinational ALUT	Registers
CP ACF Top	**2473**	**641**
├CORDIC	951	574
└Complex Multiplier	1423	0
CP Diff Top	**2267**	**1199**
├CORDIC	2*951	2*574
└Multiplier	298	0
STO Frequency	**1891**	**645**
├CORDIC	653	635
└Complex Multiplier	980	0

Table 5.8 – Resource Utilization of the ICFO architectures

Entity	Combinational ALUT	Registers
BOAI Top	738	662
Maximum of Correlations	1579	642
STO Frequency	3660	1417
STO CP ACF Top	4242	1413
STO CP Diff Top	3039	1840

are some of the variables that determine the choice of an optimal ICFO estimator. Other issues not yet analyzed could also impact the choice of an optimal ICFO estimator, e.g., power consumption, one of the requirements of the intended application.

The approach of estimation and correction of the STO before the ICFO computation shows poor performance for the Non-Data-Aided or blind algorithms. Also, the Data-Aided algorithm tested show poor performance with channel multipath in the frequency domain. Moreover, fine STO estimation algorithms show greater hardware consumption than ICFO estimation methods. In conclusion, robust STO estimation if difficult to achieve, it must deal with all the impairments in the system, yet to be corrected. However, estimation of small STOs does not have to be perfect or exact, since the phase difference introduced by it in the frequency domain can be corrected using the equalization process, as long as the residual error falls inside the symbol CP.

The other methods presented (maximum of correlations and the BoAi algorithm) are not affected by the STO, show immunity to the multipath channel and good performance for noisy signals. These methods are more suitable for the ICFO estimation since less area, and computation complexity is used. For the BoAi algorithm, the multiplications performed are simpler, since the multiplication is achieved through only XOR gates, it consumes much less hardware than a typical complex multiplier. In performance, the maximum correlation algorithm shows good results. If performance is a must, it can be achieved at much less costs with the maximum correlations at the cost of more computation time and more hardware consumption. However, this method can be improved

by pipelining the FFT or by a multi-frequency clock system, where the clock frequency is increased for certain blocks, in our case the FFT, reducing the overall time spent to perform the computation. The feasibility of a pipeline implementation of the FFT must be analyzed with the whole OFDM Modem, its consumption in area as well as power are two of the main parameters to consider. Chip costs are also another variable in which the algorithms can have an impact, a better ICFO estimator can mean a cheaper oscillator crystal which means a cheaper chip.

As was shown, the choice of the optimal algorithm for the ICFO estimator depends on many variables, some of them unknown at this point of the project, e.g., chip costs and chip area available. From the results obtained and the analysis performed until this point, it can be concluded that the proposed method, the maximum of correlations method, is an optimal choice only if performance is a must. The algorithm shows good performance for a noisy and multipath channel as well as immunity to residual STO. Also, the architecture can be very much improved, by different methods. It is believed that the impact not only on the ICFO but also on the overall MR-OFDM Modem could compensate for the modifications in the architecture, i.e., improving the frequency of the FFT computation could improve the entire PHY performance. This is a preliminary conclusion taken from the currently available results, other factors, as performance of other blocks in the system, could discard the advantages presented, i.e. if any of the other blocks in the receiver does not work with negative SNRs, the advantage of the current implementation with the cost of additional hardware would be lost. The overall performance would be limited by the lower performance block.

6 Conclusions

The main purpose of this work was to find optimal solutions to blocks concerning an ASIC intended system, thus, implementations that show low complexity, low area and good performance where pursued, as well as synthesizable architectures. This work also presented a methodology to ASIC design and hardware prototyping, the process from the conceptual problem definition until the hardware realization of the algorithms is showed.

An FFT/IFFT architecture that meets the system requirements were presented. It allows to process sequences of various sizes as well as forward and inverse Fourier Transforms while minimizing the resource consumption when compared with other FPGA implementations. The architecture presented is a collection of methods found in the literature that simplify the FFT implementation. It was prototyped in FPGA and integrated with the whole MR-OFDM PHY for a working demo that was presented in [48]. Also, in [49] details of the main components of the MR-OFDM PHY Modem implemented for preliminary hardware prototyping were presented, including the FFT engine proposed. A first version of the ICFO FFT based method and its ASIC implementation result was also published in [50]. The Logical and physical synthesis showed that the architecture is synthesizable and can be implemented in an ASIC.

Another issue addressed in this work is the synchronization in OFDM systems, more specifically the integer part of the frequency offset. An architecture that shows good performance and takes advantage of the OFDM system saving hardware resources was presented. This FFT based correlation architecture showed low area consumption when compared with similar approaches for the computation of the ICFO. The additional hardware required for the computation (besides the FFT) is minimal if compared with the direct implementation (that is the direct correlation performed in the frequency domain).

Although the frequency estimator when tested under more realistic conditions, such as in the presence of STO, showed poor results. a method to overcome this has been proposed. Two scenarios were analyzed: first, methods that try to estimate and remove the STO, to be used in conjunction with the ICFO estimator already implemented. A second scenario where methods to estimate the ICFO that show immunity to the STO where presented. Others variables that affect the estimation performance of both estimators (STO and ICFO) were also considered, e.g., the frequency selective channel. The maximum of correlations method, developed in this work showed good results for area consumption and the best performance for channel impairments with a trade-off in computation time, solutions to overcome the slowness of this architecture were also proposed.

6.1 Future Work

As was shown the STO and ICFO are closely related problems. Different methods for the computation of the STO and its performance were shown. Although the methods tested for the STO estimation did not yield optimal results, one that does would allow the use of the correlation using the FFT using the correlation theorem, which is an optimal solution in terms of hardware and performance. Hence, the finding of a robust STO estimator is an ongoing task.

Although the FFT shows good results in its implementation, some level of parallelism could be applied possibly reducing the power consumption in the process. In summary, the power consumption with some parallelism goes up, while the time spent performing the computation is reduced. A study of the power consumption, parallelism and computation time will give the optimal solution to the FFT computation. In any case, the architecture and methods proposed in this work can be of benefit for new architectures.

Since the chip is projected for mobile applications, power consumption is a must. Impact of the different implementation in power is yet to be performed. Also, a study of methods for power reduction applied to the FFT computation process could be performed. In [51] and [52], methods for reducing switching activity in the data bus of FFT and similar algorithms are shown.

This work was hosted and sponsored by Eldorado research Institute.

Bibliography

[1] "IEEE Std. 802.15.4g-2012: 'Part 15.4: Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks – Amendment 4: Physical Layer Specifications for Low Data Rate Wireless Smart Metering Utility Networks," March 2012.

[2] Wold and Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Transactions on Computers*, vol. C-33, no. 5, pp. 414–426, May 1984.

[3] H. L. Groginsky and G. A. Works, "A pipeline fast Fourier transform," *IEEE Transactions on Computers*, vol. C-19, no. 11, pp. 1015–1019, Nov 1970.

[4] W. Li and L. Wanhammar, "A pipeline FFT processor," vol. 19, 02 1999, pp. 654 – 662.

[5] R. M. Jiang, "An area-efficient FFT architecture for OFDM digital video broadcasting," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1322–1326, Nov 2007.

[6] N. Kirubanandasarathy, K. Karthikeyan, and K. Thirunadanasikamani, "VLSI design of mixed radix FFT processor for MIMO OFDM in wireless communications," in *2010 International Conference on Communication control and computing Technologies*, Oct 2010, pp. 98–102.

[7] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proceedings of International Conference on Parallel Processing*, Apr 1996, pp. 766–770.

[8] X. Xiao, E. Oruklu, and J. Saniie, "Efficient FFT Engine with Reduced Addressing Logic," in *Electro/Information Technology, 2007 IEEE International Conference on*, May 2007, pp. 390–395.

[9] E. Oruklu, X. Xiao, and J. Saniie, "Reduced Memory and Low Power Architectures for CORDIC-Based FFT Processors," *Journal of Signal Processing Systems*, vol. 66, no. 2, pp. 129–134, 2012.

[10] X. Xiao, E. Oruklu, and J. Saniie, "Fast memory addressing scheme for radix-4 FFT implementation," in *2009 IEEE International Conference on Electro/Information Technology*, June 2009, pp. 437–440.

[11] Y. Ma, "An effective memory addressing scheme for fft processors," *IEEE Transactions on Signal Processing*, vol. 47, no. 3, pp. 907–911, March 1999.

[12] S. Y. Park, N. I. Cho, S. U. Lee, K. Kim, and J. Oh, "Design of 2k/4k/8k-point fft processor based on cordic algorithm in ofdm receiver," in *2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (IEEE Cat. No.01CH37233)*, vol. 2, Aug 2001, pp. 457–460 vol.2.

[13] E. Oruklu, X. Xiao, and J. Saniie, "Reduced memory and low power architectures for cordic-based fft processors," *Journal of Signal Processing Systems*, vol. 66, no. 2, pp. 129–134, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11265-011-0586-x

[14] J.-C. Kuo, C.-H. Wen, C.-H. Lin, and A.-Y. A. Wu, "Vlsi design of a variable-length fft/ifft processor for ofdm-based communication systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 13, p. 439360, Dec 2003. [Online]. Available: https://doi.org/10.1155/S1110865703309060

[15] F. Classen and H. Meyr, "Frequency synchronization algorithms for ofdm systems suitable for communication over frequency selective fading channels," vol. 3, 07 1994, pp. 1655 – 1659 vol.3.

[16] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for ofdm," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, Dec 1997.

[17] b. ai, J.-h. GE, Y. Wang, S.-y. Yang, P. Liu, and G. Liu, "Frequency offset estimation for ofdm in wireless communications," vol. 50, pp. 73 – 77, 03 2004.

[18] M. Speth, S. Fechtel, G. Fock, and H. Meyr, "Optimum receiver design for ofdm-based broadband transmission .ii. a case study," *IEEE Transactions on Communications*, vol. 49, no. 4, pp. 571–578, April 2001.

[19] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Efficient integer frequency offset estimation architecture for enhanced ofdm synchronization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1412–1420, April 2016.

[20] D. Toumpakaris, J. Lee, and H.-L. Lou, "Estimation of integer carrier frequency offset in ofdm systems based on the maximum likelihood principle," *Broadcasting, IEEE Transactions on*, vol. 55, pp. 95 – 108, 04 2009.

[21] Y. Cho, J. Kim, W. Yang, and C. Kang, *MIMO-OFDM Wireless Communications with MATLAB*, ser. Wiley - IEEE. Wiley, 2010.

[22] S. Weinstein and P. Ebert, "Data transmission by frequency-division multiplexing using the discrete Fourier transform," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 628–634, October 1971.

[23] T. Chiueh, P. Tsai, and I. Lai, *Baseband Receiver Design for Wireless MIMO-OFDM Communications*, ser. Wiley - IEEE. Wiley, 2012.

[24] R. Prasad, *OFDM for wireless communications systems*. Artech House, 2004.

[25] T. Pollet, M. V. Bladel, and M. Moeneclaey, "BER sensitivity of OFDM systems to carrier frequency offset and Wiener phase noise," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 191–193, Feb 1995.

[26] M. J. Canet, V. Almenar, J. Marin-Roig, and J. Valls, "Time synchronization for the IEEE 802.11 a/g WLAN standard," in *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*. IEEE, 2007, pp. 1–5.

[27] T. Sato, D. Kammen, B. Duan, M. Macuha, Z. Zhou, J. Wu, M. Tariq, and S. Asfaw, *Smart Grid Standards: Specifications, Requirements, and Technologies*. Wiley, 2015.

[28] J. R. Barry, D. G. Messerschmitt, and E. A. Lee, *Digital Communication: Third Edition*. Norwell, MA, USA: Kluwer Academic Publishers, 2003.

[29] P. Duhamel and M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art," *Signal processing*, vol. 19, no. 4, pp. 259–299, 1990.

[30] S. Winograd, "On computing the discrete Fourier transform," *Mathematics of computation*, vol. 32, no. 141, pp. 175–199, 1978.

[31] D. Kolba and T. Parks, "A prime factor fft algorithm using high-speed convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 4, pp. 281–294, Aug 1977.

[32] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for Sparse Fourier Transform," in *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '12. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2012, pp. 1183–1194. [Online]. Available: http://dl.acm.org/citation.cfm?id=2095116.2095209

[33] A. Agarwal, H. Hassanieh, O. Abari, E. Hamed, D. Katabi, and Arvind, "High-throughput implementation of a million-point sparse Fourier transform," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2014, pp. 1–6.

[34] J. Proakis and D. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice Hall, 1996.

[35] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[36] E. Chu and A. George, *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*, ser. Computational Mathematics Series. CRC-Press, 2000.

[37] P. V. Stojanovic, "Lecture notes communication system design 6.973," Spring 2006.

[38] E. Chu and A. George, *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*, ser. Computational Mathematics Series. CRC-Press, 2000.

[39] J. E. Volder, "The cordic trigonometric computing technique," *Electronic Computers, IRE Transactions on*, no. 3, pp. 330–334, 1959.

[40] T. D. Perez, E. R. Lima, and L. G. Meloni, "Um Critério de Otimização para Implementação de um CORDIC Paralelo e sua Aplicação." in *XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT*, September 2016.

[41] Altera Corporation, "High Performance, Low Cost FPGA Correlator for Wideband CDMA and Other Wireless Applications," May 2016.

[42] E. Weisstein, *CRC Concise Encyclopedia of Mathematics, Second Edition*. CRC Press, 2002.

[43] Altera, "High performance, low cost fpga correlator for wideband cdma and other wireless applications," May 2003.

[44] Altera Corporation, "Cyclone V Device Overview," June 2016, https://www.altera.com/en_US/pdfs/literature/hb/cyclone-v/cv_51001.pdf [Accessed 14th Jun 2016].

[45] M. J. Canet, V. Almenar, J. Marin-Roig, and J. Valls, "Time synchronization for the ieee 802.11a/g wlan standard," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, Sept 2007, pp. 1–5.

[46] C. L. Nguyen, "Robust time and frequency synchronization in 802.11a communication wireless system," Theses, Université Paris-Nord - Paris XIII, May 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01235696

[47] D. c. Chang, "Analysis and compensation of channel correction in pilot-aided ofdm systems with symbol timing offset," in *2006 IEEE International Conference on Electro/Information Technology*, May 2006, pp. 324–329.

[48] A. F. Queiroz, G. S. da Silva, C. G. Chaves, T. D. Perez, D. G. Urdanetta, D. C. Alves, M. C. Garcia, and E. R. de Lima, "Demo: Fpga implementation of an ieee802.15.4g mr-ofdm baseband modem for smart utility networks," in *4th IEEE Global Conference on Consumer Electronics (GCCE 2015)*, Osaka, Japan, October 2015.

[49] D. C. Alves, G. S. da Silva, E. R. de Lima, C. G. Chaves, D. Urdaneta, T. Perez, and M. Garcia, "Architecture design and implementation of key components of an ofdm transceiver for ieee 802.15.4g," in *IEEE International Symposium on Circuits and Systems 2016 (ISCAS 2016)*, Montreal, Canada, May 2016.

[50] Daniel G. Urdaneta, Eduardo R. de Lima, Gabriel S. da Silva, Jacqueline G. Mertes, and Luis G.P. Meloni, "FFT-Based Integer Carrier Frequency Estimator and Corrector for IEEE802.15.4g MR-OFDM PHY," in *6th Workshop on Circuits and Systems Design (WCAS 2016)*, Belo Horizaonte, Brazil, September 2016.

[51] E. A. C. da Costa, J. C. Monteiro, and S. Bampi, *Gray Encoded Arithmetic Operators Applied to FFT and FIR Dedicated Datapaths*. Boston, MA: Springer US, 2006, pp. 281–297. [Online]. Available: https://doi.org/10.1007/0-387-33403-3_18

[52] Y. Shin, S.-I. Chae, and K. Choi, "Partial bus-invert coding for power optimization of application-specific systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, pp. 377–383, April 2001.

Annex

ANNEX A – ASIC Design Flow

Since the main purpose of this work is the design of blocks concerning an Application Specific Integrated Circuit ASIC, an overview of the design process is presented in this section.

ASICs are custom made integrated circuits intended to perform specific tasks. A Bitcoin Miner or a Radio Frequency Identification (RFID), transceivers or audio processing chips are all examples of ASICs. Design, test and evaluation of ASICs are very complex tasks. Three main variables dictates the constraints in the design process:

1. *Speed*: as in speed of operation, i.e. how fast the chip can perform its computations.
2. *Area*: in how much the design can occupy in terms of logic gates or transistors, since occupied area means manufacturing costs and portability.
3. *Power*: in how much power is consumed, since much of these devices are designed to operate in mobile applications that depends on battery, and due to heating problems caused by power dissipation.

ASIC design can be classified into two major categories: Digital Design, where circuits that use a great number of standard pre-designed standard cells that describe basic logic functions (as in *and*, *or* gates for example) are employed. Digital circuits can employ millions of those standard cells. On the other hand, analog circuits, the second major category of ASIC circuits, use a smaller number of cells. Analog circuits are custom made, use different voltage levels and much less transistors than digital circuits. Amplifiers, mixers and Voltage Controlled Oscillators are all examples of analog circuit implementations, whereas digital circuits implementations include, decoders, correlators, digital filters, communication protocols, routing algorithms. The review of the design flow presented in this chapter is only concerned with the digital design flow.

Figure A.1 shows the basic digital design flow. The first stage of the digital design process is the functional specification. It comprises a description of the system, application, requirements, as well as design constraints. The constraints are derived for the intended application and are based on the three key parameters mentioned before: power, area and speed. E.g. low power designs with low area consumption are the adopted approach for mobile applications, since this kind of application are often powered by batteries. After specifications analysis and constraints definitions, models are developed.

Fixed or floating point behavioral models, described in a high level language, validate the functional specification.

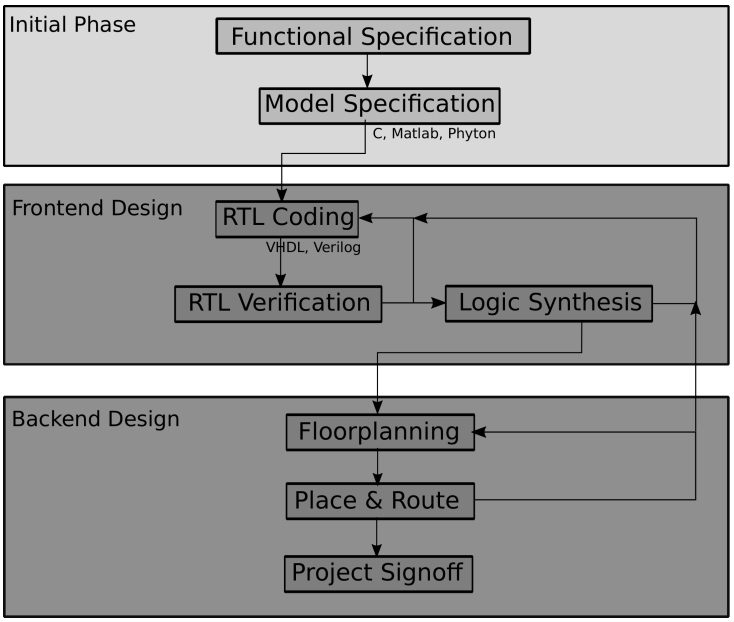

Figure A.1 – ASIC design flow

A.1 Front-end

Once the requirements are defined, the next step is to design the architecture of every block in the system. The architecture is a description of the digital circuit that must perform an intended tasks, also called of Register Transfer Level (RTL), the digital circuit performs complex computations, protocol management or reorders data. The tasks are described with logic gates, state machines, combinational and sequential logic, *and*, *or* gates, multiplexors and flip flops. After the architecture is completely defined, an RTL description in a Hardware Description Language (HDL) is performed. HDLs describe the structure and behavior of the digital logic circuit, similar to a programming language, it achieves the description by means of a textual description consisting of expressions and statements.

This RTL behavioral model then goes through a verification process, the behavioral RTL written in HDL language is simulated, and the results are compared with that of the high level language reference model. During the verification process, the circuit is stimulated with real data and timing and functional behavior is tested. The high level language model written in e.g. C, Matlab or Phyton, receives the same stimulus of the behavioral RTL model and generates a reference output. The two outputs are then compared and the RTL debugged if necessary, in case of differences with the reference. Time, behavioral and functional verification at this stage also allows for architectural explo-

ration, optimal data widths and quantization levels can be adjusted for blocks involving fixed point computations, also performance issues can be identified.

Once the RTL is verified and no errors are present, Gate Level Synthesis is performed. This synthesis verifies if the HDL description can in fact be mapped to a specific digital hardware. Every logic that is described in the HDL language is mapped onto basic logic gates, *and*, *or* gates and lookup tables, memories and registers are also inferred from the RTL HDL description. Those primitives components are also known as standard cells. The gate level synthesis output is then, a description on a more low level stage. The netlist (the output at this stage) shows an entire structural description of the RTL design. At this stage rough, area, timing and power estimations can be performed, therefore if the goals of the design are not meet e.g. if the maximum frequency of operation defined was 20MHz and the circuit only achieves 15MHz, the behavioral description or even the architecture must be changed in order to attain the time/power/area requirements.

FPGA prototyping is also performed at this stage, to extend the design verification. FPGA simulation time is faster than that of a PC allowing less simulation time. Moreover, FPGA prototyping allows for early integration in the embedded system and a more realistic verification, since real stimulus can be applied to the design. Another reason to expend time prototyping in FPGA is total development costs. FPGAs prototypes increase the probability of getting the design bug-free in fewer tape-outs.

A.2 Back-end

Once the frontend bug-free netlist is generated, a physical implementation is performed. Basically the process follows three key steps:

- *Floor Planing* In Floorplaning the subblocks that compose the design are strategically distributed on the chip area. Blocks are placed in such a way that noise between components is avoided, reduction in timing paths is also accomplished and power distribution of the chip is planed.
- *Placement* The placement process maps the logic with the standard cells and tries to adjust all the cells within the chip area to that of the restrictions defined in the floorplaning. The placement process also maps the standard cells to the actual technology used, the pre-designed cells are provided by the foundry (the semiconductor fabrication facility) chosen to manufacture the chip.
- *Routing*: Finally the routing process uses the information on the netlist to connect the standard cells one with another. The placement and routing process are iterative, they go back and forth until the process is completed.

After the routing process is complete and timing, function and power requirements are checked, a physical verification is performed. A series of tests are performed on the final circuit: the design rule check (DRC) test some design rules imposed by the foundry over the layout representation, so that the chip can perform as expected. Also, the layout versus schematic (LVS), where the functionality of the final layout is tested by comparing it with that of the schematic circuit. Finally an electric rule check (ERC) is performed on the physical verification, ERC checks for required values in the electrical components of the circuit.

There is a chance in almost all stages of the ASIC design flow (if the timing, area or power goals are not reached withing that stage) that it may be necessary to go back to previous stages and modify the design, architecture, RTL (even algorithms can change) if in the final stages the design does not reach the desired results, e.g. if the timing requirements are not met after the routing stage, it may be possible to change architectures defined in the front-end stage in order to met those timing requirements, and then repeat the whole process from that point on.

ANNEX B – ASIC Results

B.1 Synthesys Results

The proposed design was synthesized on the Cadence Encounter RTL Compiler version 14.20 for frequencies ranging from 1.3333 MHz (maximum FFT/IFFT Sample Rate for IEEE802.15.4g) up to 200 MHz. The Worst time Slack[1] (WS), amount of required cells [2], Area occupation and power consumption of the entire FFT/IFFT block, for each of the clock frequencies, are listed in Table B.1. Table B.2 show the results for the same parameters for the ICFO implementation.

Table B.1 – Logical synthesis results for $65nm$ CMOS for the FFT Architecture

Freq MHz	**Cells**	**CellsArea μm^2**	**WS ps**	**Leakage μW**	**Dynamic μW**	**Total μW**
1.333	13057	74864	523179	21.098	220.069	241.169
2.5	13047	74850	278166	21.086	261.741	282.828
5	13060	74856	138166	20.996	385.393	406.390
10	13046	74847	68166	20.973	1170.029	1191.003
20	13033	74836	33166	20.922	2116.889	2137.811
40	13030	74834	14537	20.963	3699.561	3720.524
80	13108	74902	3759	21.070	6767.198	6788.268
100	13880	75326	2603	21.082	8368.232	8389.314
166	16247	78345	166	22.655	14133.956	14156.611
200	16849	79965	-6	23.344	17647.131	17670.475

It is worth noticing that both designs can work until clock frequencies of 166 MHz without a negative worst slack, although the the maximum frequency required for the entire MR-OFDM transceiver is of 1.33 MHz, as shown in table 3.1, on chapter 2. The leakage power of the ICFO is greater than the leakage power as expected since the area of this block is greater than the area occupied by the FFT block.

[1] The difference between the required arrival time of a signal at the input of the block being analyzed and the actual arrival time.

[2] A cell is the minimum building block used by the synthesis tool. Unlike ALMs in FPGA, where there is only one type, the minimum building block in the Encounter RTL Compiler is of many types and can be composed of a few or several logic gates. Different kind of cells can be found in different technologies. Unfortunately, cells characteristics are proprietary and confidential so that no further information can be provided.

Table B.2 – Logical synthesis results for $65nm$ CMOS for the ICFO Architecture

Freq MHz	**Cells**	**CellsArea μm^2**	**WS ps**	**Leakage μW**	**Dynamic μW**	**Total μW**
1.333	31110	154046	515257	45.634	208.834	254.469
2.5	31100	154048	273744	45.638	246.130	291.769
5	31114	154052	135728	45.633	332.961	378.595
10	31106	154047	65728	45.882	1643.020	1688.902
20	31105	154050	30728	45.639	2789.674	2835.313
40	31134	154066	11544	45.435	5764.384	5809.820
80	31749	154568	2048	45.636	11227.396	11273.032
100	32980	155281	1221	45.900	13172.190	13218.099
166	37999	161416	34	48.768	22944.270	22993.039
200	39895	165869	-43	50.818	28016.074	28066.893

The successful logic synthesis and positive worst slack show that both designs are synthesizable and that its ASIC implementation is achievable. Integration of both blocks in the MR-OFDM modem was already done as can be seen in [49]; currently, the back-end process of the ASIC design flow is being performed. Also, an early power estimation given by the Cadence Encounter RTL compiler tool is shown in the results.

ANNEX C – IEEE802.15.4g Parameters

C.1 Long training Field

Table C.1 – LTF for MR-OFDM Option 1

Tone#	Value	Tone#	Value	Tone#	Value	Tone#	Value
–64	0	–32	–1	0	0	32	–1
–63	0	–31	–1	1	1	33	–1
–62	0	–30	–1	2	–1	34	–1
–61	0	–29	1	3	1	35	1
–60	0	–28	1	4	–1	36	1
–59	0	–27	–1	5	1	37	1
–58	0	–26	–1	6	1	38	1
–57	0	–25	–1	7	–1	39	1
–56	0	–24	–1	8	–1	40	1
–55	0	–23	–1	9	1	41	–1
–54	0	–22	1	10	–1	42	–1
–53	0	–21	1	11	1	43	–1
–52	–1	–20	–1	12	1	44	–1
–51	1	–19	1	13	1	45	–1
–50	1	–18	–1	14	1	46	–1
–49	–1	–17	–1	15	–1	47	1
–48	–1	–16	1	16	1	48	–1
–47	–1	–15	–1	17	1	49	1
–46	–1	–14	1	18	1	50	1
–45	1	–13	1	19	1	51	–1
–44	1	–12	1	20	1	52	1
–43	–1	–11	1	21	–1	53	0
–42	–1	–10	–1	22	1	54	0
–41	1	–9	–1	23	–1	55	0
–40	1	–8	1	24	1	56	0
–39	1	–7	1	25	–1	57	0
–38	–1	–6	–1	26	1	58	0
–37	–1	–5	1	27	–1	59	0
–36	1	–4	1	28	1	60	0
–35	1	–3	–1	29	1	61	0
–34	–1	–2	1	30	–1	62	0
–33	–1	–1	1	31	1	63	0

Table C.2 – LTF for MR-OFDM Option 2

Tone#	Value	Tone#	Value	Tone#	Value	Tone#	Value
–32	0	–16	1	0	0	16	1
–31	0	–15	–1	1	1	17	–1
–30	0	–14	1	2	–1	18	–1
–29	0	–13	1	3	1	19	–1
–28	0	–12	–1	4	1	20	–1
–27	0	–11	–1	5	–1	21	–1
–26	-1	–10	–1	6	1	22	1
–25	-1	–9	1	7	–1	23	–1
–24	–1	–8	1	8	–1	24	–1
–23	–1	–7	–1	9	1	25	–1
–22	1	–6	1	10	–1	26	1
–21	1	–5	1	11	1	27	0
–20	1	–4	1	12	1	28	0
–19	–1	–3	–1	13	–1	29	0
–18	1	–2	–1	14	–1	30	0
–17	–1	–1	–1	15	1	31	0

Table C.3 – LTF for MR-OFDM Option 3

Tone#	Value	Tone#	Value	Tone#	Value	Tone#	Value
–16	0	–8	1	0	0	8	–1
–15	0	–7	1	1	–1	9	1
–14	0	–6	1	2	–1	10	1
–13	1	–5	1	3	1	11	–1
–12	–1	–4	1	4	–1	12	–1
–11	1	–3	1	5	1	13	1
–10	–1	–2	1	6	1	14	0
–9	1	–1	–1	7	–1	15	0

Table C.4 – LTF for MR-OFDM Option 4

Tone#	Value	Tone#	Value	Tone#	Value	Tone#	Value
–8	0	–4	1	0	0	4	1
–7	1	–3	–1	1	–1	5	–1
–6	–1	–2	1	2	1	6	–1
–5	1	–1	1	3	1	7	–1

Table C.5 – Modulation Schemes for every MCS and OFDM Option

MCS	OFDM Option1	OFDM Option2	OFDM Option3	OFDM Option4
0	BPSK	BPSK	-	-
1	BPSK	BPSK	BPSK	-
2	QPSK	QPSK	QPSK	QPSK
3	QPSK	QPSK	QPSK	QPSK
4	-	QPSK	QPSK	QPSK
5	-	16-QAM	16-QAM	16-QAM
6	-	-	16-QAM	16-QAM