MASTER THESIS

TITLE: Implementation of a Tx/Rx OFDM System in a FPGA

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Francisco Martín Gutiérrez

DIRECTOR: Pere Lluis Gilabert

DATE: April, 30th 2009
Overview

The aim of this project consists in the FPGA design and implementation of a transmitter and receiver (Tx/Rx) multicarrier system such as Orthogonal Frequency Division Multiplexing (OFDM). This Tx/Rx OFDM subsystem is capable to deal with different M-QAM modulations and is implemented in a digital signal processor (DSP-FPGA). The implementation of the Tx/Rx subsystem has been carried out in a FPGA using both System Generator visual programming running over Matlab/Simulink, and the Xilinx ISE program which uses VHDL language.

This project is divided into four chapters, each one with a concrete objective.

The first chapter is a brief introduction to the digital signal processor used, a field-programmable gate array (FPGA), and to the VHDL programming language. The second chapter is an overview on OFDM, its main advantages and disadvantages in front of previous systems, and a brief description of the different blocks composing the OFDM system. Chapter three provides the implementation details for each of these blocks, and also there is a brief explanation on the theory behind each of the OFDM blocks to provide a better comprehension on its implementation. The fourth chapter is focused, on the one hand, in showing the results of the Matlab/Simulink simulations for the different simulation schemes used and, on the other hand, to show the experimental results obtained using the FPGA to generate the OFDM signal at baseband and then upconverted at the frequency of 3,5 GHz. Finally the conclusions regarding the whole Tx/Rx design and implementation of the OFDM subsystem are given.
El objetivo de este proyecto consiste en el desarrollo de un sistema de transmisión y recepción de señales multiportadora como la Orthogonal Frequency Division Multiplexing (OFDM). Este sistema OFDM es capaz de soportar diversas modulaciones M-QAM y está implementado en un procesador digital (DSP-FPGA). Para la implementación del sistema y llevar a cabo la programación de la FPGA se ha utilizado el System Generator corriendo sobre Matlab/Simulink o bien el programa Xilinx ISE que utiliza el lenguaje de programación VHDL.

Este proyecto está dividido en cuatro capítulos, cada uno de ellos con un objetivo concreto.

El primer capítulo es una breve introducción al tipo de procesador digital utilizado (field-programmable gate array -FPGA) y el lenguaje de programación VHDL. El segundo capítulo es una introducción de qué es OFDM, sus ventajas y desventajas frente a sistemas empleados anteriormente, y finalmente una introducción a los diferentes bloques que componen un sistema básico de transmisión y recepción OFDM. El capítulo tres presenta cómo se han implementado cada uno de los diferentes bloques del sistema y además incluye una breve introducción a la teoría que hay detrás de cada uno de estos bloques para comprender mejor de qué forma se han implementado. El cuarto capítulo se centra por un lado en mostrar los resultados de las simulaciones hechas en Matlab/Simulink, y por el otro lado también muestra los resultados de las pruebas hechas con la FPGA en banda base y a la frecuencia de 3,5 GHz utilizando el material disponible en el laboratorio. Finalmente se presentan las conclusiones a las que se han llegado tras el desarrollo del proyecto.
<table>
<thead>
<tr>
<th>OUTLINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION ................................................................. 1</td>
</tr>
<tr>
<td>CHAPTER 1 PROGRAMMABLE LOGIC DEVICES......................................... 2</td>
</tr>
<tr>
<td>1.1. Field Programmable Gate Arrays (FPGA) ......................................... 2</td>
</tr>
<tr>
<td>1.2. VHDL ................................................................................. 4</td>
</tr>
<tr>
<td>CHAPTER 2 ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (OFDM) .................. 6</td>
</tr>
<tr>
<td>2.1 Introduction to OFDM................................................................ 6</td>
</tr>
<tr>
<td>2.2 OFDM Advantages and Disadvantages........................................... 8</td>
</tr>
<tr>
<td>2.3 Main blocks in an OFDM system.................................................. 9</td>
</tr>
<tr>
<td>2.3.1 OFDM Transmitter ......................................................... 10</td>
</tr>
<tr>
<td>3.3.1 OFDM Receiver ............................................................. 11</td>
</tr>
<tr>
<td>CHAPTER 3 IMPLEMENTATION OF THE OFDM SYSTEM ......................... 13</td>
</tr>
<tr>
<td>3.1 Introduction........................................................................... 13</td>
</tr>
<tr>
<td>3.2 Random bit generator ............................................................ 13</td>
</tr>
<tr>
<td>3.3 S/P &amp; P/S converters ............................................................. 14</td>
</tr>
<tr>
<td>3.4 Constellation mapper (encoder).................................................. 15</td>
</tr>
<tr>
<td>3.4.1 Constellation encoder ...................................................... 15</td>
</tr>
<tr>
<td>3.4.2 Constellation decoder ...................................................... 17</td>
</tr>
<tr>
<td>3.5 (I)FFT ................................................................................. 18</td>
</tr>
<tr>
<td>3.5.1 The FFT algorithm ......................................................... 19</td>
</tr>
<tr>
<td>3.5.2 Implementation scheme (VHDL) ......................................... 22</td>
</tr>
<tr>
<td>3.5.3 Implementation scheme (System Generator) ........................... 23</td>
</tr>
<tr>
<td>3.6 Modulator/Demodulator .......................................................... 26</td>
</tr>
<tr>
<td>3.3.2 4-QAM mod/demod ......................................................... 26</td>
</tr>
<tr>
<td>3.3.3 M-QAM mod/demod ......................................................... 28</td>
</tr>
<tr>
<td>CHAPTER 4 OFDM IMPLEMENTATION RESULTS ................................. 31</td>
</tr>
<tr>
<td>4.1 Introduction........................................................................... 31</td>
</tr>
<tr>
<td>4.2 Simulation phase .................................................................... 31</td>
</tr>
<tr>
<td>4.2.1 4-QAM Simulation scheme ............................................... 31</td>
</tr>
<tr>
<td>4.2.2 M-QAM Simulation scheme ............................................... 37</td>
</tr>
<tr>
<td>4.3 Hardware phase .................................................................... 42</td>
</tr>
<tr>
<td>4.3.1 Baseband modulation ....................................................... 42</td>
</tr>
</tbody>
</table>
INTRODUCTION

Orthogonal Frequency Division Multiplexing (OFDM) is a multicarrier modulation technique. OFDM provides high bandwidth efficiency because the carriers are orthogonal to each other and multiple carriers share the data among themselves. The main advantage of this transmission technique is their robustness to channel fading in wireless communication environment. The main objective of this project is to implement and test an OFDM transmitter and receiver using a FPGA at baseband and later upconverting the OFDM signal up to 3.5 GHz.

This project aim is to give an idea of what is an OFDM system, its implementation and the analysis of the obtained results of the simulations and the hardware testing. This OFDM system is able to support different M-QAM modulation schemes.

All the different OFDM basic system blocks are introduced in this project. However, special attention is paid to the processing block of the OFDM system, composed by the Fast Fourier Transform (FFT) block and the Inverse Fast Fourier Transform (IFFT). These blocks are introduced theoretically, to know what is behind them, and also is shown the implementation and configuration of each block.

The M-QAM modulation schemes used for the multicarrier OFDM Tx/Rx subsystem are basically 4-QAM, 16-QAM and 64-QAM modulations.

Experimental and simulation results are provided within this projects. Simulation results have been obtained through the System Generator and Matlab/Simulink tools. Experimental results were obtained by transmitting and receiving a 4-QAM OFDM signal through a typical RF transmitter at 3.5 GHz.

The simulation results are the theoretically expected, showing a perfect match between the OFDM Tx and RX. However, experimental results have shown how critical is to adjust the delays in the Rx in order to recover the transmitted signal properly.
CHAPTER 1 PROGRAMMABLE LOGIC DEVICES

All the logic functions, combinational or sequential, can be reduced to simple representation in form of products addition. This way, any complex logic function can be reduced to discrete logics. This characteristic of logic functions makes possible to develop the programmable electronic components, called programmable logic devices, PLD.

The firsts programmable devices are from the 70’s, after the development of the ROM programmable memories (PROM’s). Then appeared the PLA (programmable logic array), devices which contained uniform arrays, with programmable connections in an efficient and easy way. The problem with these devices was that the time for transferring the data was too long.

To solve the time problem, was developed the PAL (programmable array logic), that is a PLD with a fix logic, letting only to program the logic connections.

With the goal of increasing the capacity of logic implementation in the programmable devices, new architectures with flip-flops in its structure to make easier the development of the state machines and circuits with synchronous logic like for example SPLD’s (Simple PLD) and CPLD’s (Complex PLD).

In general PLD’s are circuits that are configurable by the user to implement a logic function, these devices are able to interconnect two points of the circuit. Commonly these elements are the fuses, anti fuses, SRAM cells, transistors EPROM and EEPROM. This way, some of these devices can be reprogrammed and others are able only to be programmed once.

1.1. Field Programmable Gate Arrays (FPGA)

In order to implement large circuits, it is convenient to use a type of chip that has a large logic capacity. A field-programmable gate arrays (FPGA) is a programmable logic device that supports implementations of relatively large logic circuits. FPGA is different from other logic technologies like CPLD and SPLD because FPGA do not contain AND or OR planes. Instead, FPGA consists of logic blocks for implementing the required functions.

A FPGA contain three main types of resources: logic blocks, I/O blocks for connecting to the pins of the package, and interconnection wires and switches.

The logic blocks are arranged in a two-dimensional array, and the interconnection wires are organized as horizontal and vertical routing channels between rows and columns of logic blocks. The routing channels contain wires and programmable switches that allow the logic blocks to be interconnected in many ways. FPGA can be used to implement logic circuits of more than a few hundred thousand equivalent gates in size. Equivalent gates is a way to
quantify a circuit’s size by assuming that the circuit is to be built using only simple logic gates and then estimate how many of these gates are needed.

Each logic block in a FPGA typically has a small number of inputs and one output. The FPGA products on the market feature different types of logic blocks. The most commonly used logic block is a lookup table (LUT), which contains storage cells that are used to implement a small logic function. Each cell is capable of holding a single logic value, either 0 or 1. The stored value is produced as the output of the storage cell. LUT of various sizes may be created, where the size is defined by the number of inputs.

For a logic circuit to be realized in a FPGA, each logic function in the circuit must be small enough to fit within a single logic block. In practice, a user’s circuit is automatically translated into the required form by using CAD tools.

When a circuit is implemented in an FPGA, the logic blocks are programmed to realize the necessary functions and the routing channels are programmed to make the required interconnections between logic blocks.
The FPGA device is configured by using the in-system programming (ISP) method, which means that the FPGA can be programmed while the chip is still attached to its circuit board. The storage cells in the LUTs in an FPGA are volatile, which means that they lose their stored contents whenever the power supply for the chip is turned off.

Hence the FPGA has to be programmed every time power is applied. Of this, a small memory chip that holds its data permanently, called a programmable read-only memory (PROM) is included on the circuit board that houses the FPGA. The storage cells in the FPGA are loaded automatically from the PROM when power is applied to the chips.

1.2. VHDL

VHDL is an acronym for VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. It is intended for documenting and modeling digital systems ranging from a small chip to a large system. It can be used to model a digital system at any level of abstraction ranging from the architectural level down to the gate level.

Due to the increase of the amount of resources needed for the PLD’s and the FPGA’s, it became difficult to program this devices, to solve this issue, method to plan the desired logic functions was created:

VHDL language can be regarded as an integration of the following languages:

- Sequential language.
- Concurrent language.
- Net list language.
- Timing specifications.
- Waveform generation.

It allows the user to model the system as an interconnection of components. Test waveforms can be generated using the same constructs. All the above constructs may be combined to provide a comprehensive description of the system in a single model.

The models written in VHDL can be verified using a VHDL simulator. It inherits extensive range of modeling capabilities that are difficult to understand. Fortunately, it is possible to quickly assimilate a core subset of the language that is easy and simple to understand without learning the complex features. This subset is sufficient to model most applications. The complete language has sufficient power to capture the description of the most complex chips to a complete electronics system.

VHDL is a hardware description language that allows designers to model a circuit at different level of abstraction, ranging from the gate level, RTL level, and behavioral level to the algorithmic level.
Synthesis process is to construct a gate-level net list from a model of a circuit described in VHDL. The synthesis process is described in diagram below.

A synthesis program may alternately generate a RTL net list, which consists of register-transfer level blocks such as flip-flops, arithmetic-logic-units, and multiplexers interconnected by wires. All these are performed by RTL module builder. This builder is to build or acquire from a library predefined components, each of the required RTL blocks in the user-specified target technology.

The above synthesis process produced an unoptimized gate level net list. A logic optimizer can use the produced net list and the constraint specified to produce an optimized gate level net list. This net list can be programmed directly into a FPGA chip.
CHAPTER 2 ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (OFDM)

2.1 Introduction to OFDM

During the past 15 years, Orthogonal Frequency Division Multiplexing (OFDM) has been gaining year after year a well-deserved reputation, demonstrating its high data rate and robustness to wireless environments capabilities. In the multipath environment, broadband communications will suffer from frequency selective fading.

OFDM is an attractive modulation scheme used in broadband wireless systems that encounter large delay spreads. OFDM avoids temporal equalization altogether, using a cyclic prefix technique with a small penalty in channel capacity.

Where Line-of-Sight (LoS) cannot be achieved, there is likely to be significant multipath dispersion, which could limit the maximum data rate. Technologies like OFDM are probably best placed to overcome these, allowing nearly arbitrary data rates on dispersive channels.

An OFDM signal consists of \( N \) subcarriers spaced by the frequency distance \( \Delta f \) thus, the total system bandwidth \( B \) is divided into \( N \) equidistant subchannels. On each subcarrier, the symbol duration \( T_s = 1/\Delta f \) is \( N \) times as large as in the case of a single carrier transmission system covering the same bandwidth.

![Fig. 2.1 Bandwidth divided into N subchannels](image)

Additionally, each subcarrier signal is extended by a guard interval – called cyclic prefix – with the length \( T_g \). All subcarriers are mutually orthogonal within the symbol duration \( T_s \).

\[
\int_{-\infty}^{\infty} x_1(t) x_2(t) dt = 0
\]  \hspace{1cm} (3.1)
For each subcarrier a rectangular pulse shaping is applied. The guard interval or cyclic extension is added to the subcarrier signal in order to avoid Inter-Symbol Interference (ISI), which occurs in multipath channels. At each Receiver the cyclic prefix is removed and only the time interval $[0, T_s]$ is evaluated. The total OFDM block duration is $T = T_s + T_g$. Each subcarrier can be modulated independently.

![Fig. 2.2 Cyclic Extension](image)

The spectra of the subcarriers overlap, but the subcarrier signals are mutually orthogonal, and the modulation symbol can be recovered by a simple correlation.

![Fig. 2.3 OFDM subcarriers representation](image)

In a practical application, the OFDM signal is generated in a first step as a discrete-time signal in the digital signal processing part of the Transmitter. As the bandwidth of an OFDM system is $B = N\Delta f$, the signal must be sampled with
sampling time $t = 1/B = 1/N\Delta f$. The samples of the signal can be calculated by an Inverse DFT (IDFT) which is typically implemented as an Inverse FFT (IFFT).

The subcarrier orthogonality is not affected at the output of a frequency selective radio channel; therefore, the received signal can be separated into the orthogonal subcarrier signals by a correlation technique. Alternatively, the correlation at Receiver can be done as a Discrete Fourier Transform (DFT) or a Fast Fourier Transform (FFT) respectively.

If the subcarrier spacing $\Delta f$ is chosen to be much smaller than the coherence bandwidth, and the symbol duration $T$ much smaller than the coherence time of the channel, then the transfer function of the radio channel can be considered constant within the bandwidth of each subcarrier and the duration of each modulation symbol. In this case, the effect of the radio channel is only a multiplication of each subcarrier signal by a gain factor.

Also channel coding plays an important role in OFDM systems. Due to the narrowband subcarriers and the appropriate cyclic prefix, OFDM systems suffer from flat fading. In this situation, efficient channel coding leads to a very high coding gain, especially if soft decision decoding is applied. For this reason OFDM systems will always have to make use of channel coding.

OFDM signals have a large peak-to-average power ratio (PAPR) due to the superposition of all subcarrier signals. Therefore, in each Transmitter the power amplifier will limit the OFDM signal by its maximal output power. This also disturbs the orthogonality between subcarriers, leading to both intercarrier and out-of-band interferences, which is unacceptable.

Although the principle of OFDM communication has been around for several decades, it was only in the last decade that it started to be used in commercial systems. The most important wireless applications that make use of OFDM are Digital Audio Broadcast (DAB), DVB, WLAN.

### 2.2 OFDM Advantages and Disadvantages

The major advantage of OFDM is its robustness against multi path propagation. Thus, it is suitable to be implemented in wireless environments. The introduction of cyclic prefix made OFDM system resistance to time dispersion. OFDM symbol rate is low since a data stream is divided into several parallel streams before transmission. This make the fading is slow enough for the channel to be considered as constant during one OFDM symbol interval.

Cyclic prefix is a crucial feature of OFDM used to combat the inter-symbol interference (ISI) and inter-channel-interference (ICI) introduced by the multi-path channel through which the signal is propagated. The basic idea is to replicate part of the OFDM time-domain waveform from the back to the front to
create a guard period. The duration of the guard period $T_g$ should be longer than the worst-case delay spread of the target multi-path environment. The use of a cyclic prefix instead of a plain guard interval, simplifies the channel equalization in the demodulator.

In wire system, OFDM system can offer an efficient bit loading technique. It enables a system to allocate different number of bits to different sub channels based on their individual SNR. Hence, an efficient transmission can be achieved.

One of the major disadvantages of OFDM is its requirement for high peak-to-average-power ratio. This put high demand on linearity in amplifiers.

Second, the synchronization error can destroy the orthogonality and cause interference. Phase noise error and Doppler shift can cause degradation to OFDM system. A lot of effort is required to design accurate frequency synchronizers for OFDM.

To sum up the main advantages and disadvantages of OFDM systems are as follows:

Advantages:
- $N$ narrowband transmissions are done, this way is easily to accomplish that $T_s$ (of each channel) is larger than $T$ (of the channel), this means the transmission is not affected by the channel.
- High transmission bitrates.
- Chance to cancel any channel if is affected by fading.

Disadvantages:
- High synchronism accuracy.
- Multipath propagation must be avoided in other orthogonality not be affected.
- Large peak-to-mean power ratio due to the superposition of all subcarrier signals, this can become a distortion problem.

### 2.3 Main blocks in an OFDM system

In this chapter will be presented the blocks of a basic OFDM system. Will be an overview of these blocks describing its functionality and what is expected at the input and at the output of each block.

First is going to be introduced all the blocks corresponding to the Transmitter of the OFDM system and then a briefly description of the Receiver blocks will also be commented, because the receiver blocks are basically the inverse of the Transmitter blocks.
2.3.1. OFDM Transmitter

The model considered for the implementation of the OFDM transmitter is the shown in the Fig. 2.4, and basically consist of the following blocks:

- Serial to parallel converter.
- Constellation modulator
- The IFFT block
- Parallel to serial converter
- Digital to Analog converter

The serial to parallel converter receive the M serial bits to be transmitted, and those bits are divided into N subblocks of $m_n$ bits each subblock. Those N subblocks will be mapped by the constellation modulator using Gray codification, this way $a_n + jb_n$ values are obtained in the constellation of the modulator.

The M-QAM encoder converts input data into complex valued constellation points, according to a given constellation, 4QAM, 16-QAM, 32-QAM and so on. The amount of data transmitted on each subcarrier depends on the constellation, 4QAM and 16QAM transmit two and four data bits per subcarrier,
respectively. Which constellation to use depends on the quality of the communications channel. In a channel with high interference a small modulation scheme like BPSK is favorable, since the required signal to noise ratio (SNR) in the receiver is low, whereas in a interference free channel a larger constellation is more beneficial due to the higher bit rate.

It is necessary to specify how the constellation will be mapped to implement that block. However, independently of the format of the constellation, the block encoder can be made by consulting a conversion table, implemented with a LUT that exists in LCs of FPGAs.

It is important to notice that in that mapping block, bits are converted into complex symbols (phasors) having the information of the constellation in its I, Q components.

The Inverse Fast Fourier Transform (IFFT) transforms the signals from the frequency domain to the time domain, an IFFT converts a number of complex data points, of length that is power of 2, into the same number of points but in the time domain. The number of subcarriers determines how many sub-bands the available spectrum is split into.

The CP is a copy of the last $N$ samples from the IFFT, which are placed at the beginning of the OFDM frame. There are two reasons to insert a CP. Assuming that the CP is longer than the channels impulse response, the convolution between the data and the channel impulse response will act like a circular convolution and interference from the previous symbol will only affect the CP. The CP is then discarded in the receiver and the circular convolution makes equalization in the receiver easier. However, if the number of samples in the CP is large, the data transmission rate will decrease significantly, since the CP does not carry any useful data. Thus, it is important to choose the minimum necessary CP to maximize the efficiency of the system.

### 3.3.1 OFDM Receiver

The blocks of the OFDM Receiver are shown in the Fig. 2.5, and those blocks are:

- Analog to digital converter.
- Serial to parallel converter.
- Cyclic prefix removal.
- The FFT block.
- M-QAM decoder.
- Parallel to serial converter.
After the cyclic prefix removal, the signals are passed through an $N$-point fast Fourier transform to convert the signal to frequency domain. The output of the FFT is formed from the first $M$ samples of the output.

The demodulation can be made by DFT, or better, by FFT, that is it efficient implementation that can be used reducing the time of processing and the used hardware. FFT calculates DFT with a great reduction in the amount of operations, leaving several existent redundancies in the direct calculation of DFT.

At the decoder, a mapped symbol (point) of the transmitted constellation may have changed due to the additive noise in the communications channel, a misadjustment in the sampling time at the receiver, or several other unwanted causes.

Therefore, it is necessary to define a threshold to facilitate the decision making in the receiver constellation. That is the function of the M-QAM decoder.

**Fig. 2.5 OFDM receiver**
CHAPTER 3 IMPLEMENTATION OF THE OFDM SYSTEM

3.1 Introduction

In this section a more detailed explanation of the basic blocks in the OFDM system (already seen in the previous chapter) will be provided. The theory behind all the blocks will be briefly explained and in particular, focusing in the FFT algorithm.

Then, the configuration of the main blocks of the modulator and demodulator involving the FFT/IFFT blocks will be presented, explaining not only the internal configuration of the blocks but also the input and output that must be achieved.

3.2 Random bit generator

The first block of the OFDM system is the random bit generator, this block generates the serial binary data that will arrive to the serial to parallel converter, in other words, this block generates the data with which all the system will work.

This block is based on the Linear Feedback Shift Registers (LFSR). An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops.

![LFSR example](image)

**Fig. 3.1** LFSR example

The **Fig. 3.1** shows the pseudo-random serial binary data stream generated with the VHDL code based on the previous theory:
This pseudo-random serial binary data stream can be configured setting its duration by the simulation time. This time will set how often the data will be repeated.

### 3.3 S/P & P/S converters

The aim of the serial to parallel converter is to receive the data that is going to be transmitted. The serial to parallel converter receive the $M$ serial bits to be transmitted, and those bits will be divided into $N$ subblocks of $m_n$ bits each subblock called symbols. The amount of bit of each channel can be different. Those $N$ subblocks will be mapped by the constellation modulator using Gray codification, this way $a_n + jb_n$ values are obtained in the constellation of the modulator.

The serial to parallel converter at the receiver has the function to receive the data that is going to be demodulated, with the same structure as it was at the transmitter.

To store the $M$ bits a buffer that will contain all the input data into different memory positions is needed. To obtain the $M$ data bits at the output we'll need the buffer to stop reading data, another option is that the amount of data stored at the buffer is $2M$, this way is not necessary to stop the reading, this way can read continuously.
The parallel to serial converter is only the opposite function of the serial to parallel converter, and it is placed just before sending the data through the channel by the digital to analog converter, at the transmitter, and just the last block before obtaining the final data at the receiver.

3.4 Constellation mapper (encoder)

The encoder of the constellation maps the M bits of the channel in a point \( a + jb \) in the constellation of the modulator. The decoding block receives that IQ symbol and demaps it obtaining the original \( m \) transmitted bits.

3.4.1 Constellation encoder

It is important to notice that in that mapping we only perform a conversion of bits for the phasor that acts. However, nor modulation is performed, as in the case of QAM, because that as shown, it is done by IFFT.

It is necessary to specify how the constellation will be to be mapped, to implement this block. However, independently of the format of the constellation, the block encoder can be made through a consultation at a conversion table, implemented by LUT that exists in LCs of FPGAs.

As we have explained previously, a constellation mapper takes a serial bit stream as its input and segments the stream into \( N \)-bit symbols, which are mapped to coordinates in the signal constellation. The coordinates of each point in a two-dimensional signal constellation represents the baseband in-phase and quadrature (I-Q) components that modulate the orthogonal IF carrier signals. Because the constellation mapper defines the shape and dimension of the signal constellation, it defines the modulation scheme that is implemented.

Attempt that the entrance of the encoder a binary number of \( m \) bits, and that the exit generates two binary numbers, one in phase, the, and other in quadrature, \( b \), whose size is defined by IFFT.

The \( N \) bit symbol encoding method has significant effects on the transmission scheme performance. A common receiver error occurs when channel noise causes the receiver to deduce that an adjacent symbol was sent instead of the one actually transmitted. Symbols can be encoded such that adjacent symbols differ in one bit position. If the receiver incorrectly detects a given symbol, it is likely that only one of the \( k \) bits in the symbol is in error. This encoding scheme, known as Gray encoding, increases the robustness of the transmission scheme in the presence of a noisy channel.

The modulations taken into account for this project are BPSK, 4-QAM, 16-QAM, 32-QAM and 64-QAM. Depending on the modulation the amount of \( N \) bits
changes: 1 bit for BPSK, 2 bits for the 4-QAM, 4 bits for 16-QAM, 5 bits for the 32-QAM and 6 bits for the 64-QAM.

In Fig 3.4 is shown the constellation of a 16-QAM modulation. In this project the implementation of the constellation has been done, as commented previously, using LUT’s, tables with the hardcoded values of the output modulation depending on the values of the N input bits, these tables has been written into VHDL code.

The full block of the constellation encoder explained before will graphically be as follows:

The Fig. 3.6 shows the test bench simulations done for the VHDL code containing the blocks already seen (random bit generator, serial to parallel converter and constellation mapper):
This simulation shows serial input, marked by the blue arrow; this input is converted to N bits in parallel depending on the modulation, BPSK and 64-QAM modulation are shown.

For the BPSK modulation the I component can take the values \{1,-1\} and the Q component must be always 0. The 64-QAM modulation can take the values \{1,3,5,7,-1,-3,-5,-7\} for both components I and Q.

The next table shows the possible values the I,Q components can take at each modulation scheme:

**Table 3.1**IQ values depending on the modulation

<table>
<thead>
<tr>
<th>Modulation</th>
<th>I</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BPSK</strong></td>
<td>{1,-1}</td>
<td>0</td>
</tr>
<tr>
<td><strong>4-QAM</strong></td>
<td>{1,-1}</td>
<td>{1,-1}</td>
</tr>
<tr>
<td><strong>16-QAM</strong></td>
<td>{1,3,-1-3}</td>
<td>{1,3,-1-3}</td>
</tr>
<tr>
<td><strong>32-QAM</strong></td>
<td>{1,3,5,-1-3,5}</td>
<td>{1,3,5,-1-3,5}</td>
</tr>
<tr>
<td><strong>64QAM</strong></td>
<td>{1,3,5,7,-1-3,5,-7}</td>
<td>{1,3,5,7,-1-3,5,-7}</td>
</tr>
</tbody>
</table>

**3.4.2 Constellation decoder**

In the receiver, the point of the constellation transmitted it can have changed due to the noises of the transmission channel, mistake in the time of sampling of the receiver and several other causes.

**Fig. 3.7 Constellation at the receiver**
Therefore it is necessary to define a threshold so that it can be made the decision on which point in the constellation the received signal is acting. That is the function of the decoder. The demodulated I-Q coordinates may no longer correspond to the exact location of a signal point in the original constellation. The demapper must detect which symbol was most likely transmitted by finding the smallest distance between the received point and the location of a valid symbol in the signal constellation. Having detected the most likely coordinates, the constellation demapper then performs the decoding procedure to output the correct $N$ bit symbol represented by those coordinates.

![Fig. 3.8 Demapping example](image)

Once the demapper detects the transmitted I-Q coordinates, it can output the difference between the actual constellation point and the received coordinates as an error. This information indicates the level of confidence in the decision made by the demapper, and it can also be used by other parts of the demodulator to obtain a confidence estimate. Demodulator subsystems can use this output information to compensate for channel impairments such as phase noise, carrier noise, channel dispersion, or suboptimal A/D sampling.

### 3.5 (I)FFT

The OFDM modulation can be obtained through an IDFT. The fast implementation of IDFT, IFFT, can be used reducing the time of processing and the used hardware. The demodulation, in the same way, can be made by DFT, or better, by FFT, that is it efficient implementation.

FFT calculates DFT with a great reduction in the amount of operations, leaving several existent redundancies in the direct calculation of DFT. The only
limitation of this algorithm is that is only valid for sequences of length $2^N$ otherwise modified algorithms with less efficiency are needed.

The FFT algorithms can be classified depending on the domains in which are calculated, that can be made in both time domain and frequency domain, we have discussed ahead. If the length of the sequence is a $2^N$ value, the algorithm applied is called Radix-2. Other more efficient algorithms also exist, for instance the Radix-4, which can be applied when the input sequence length is a $4^N$ value.

### 3.5.1 The FFT algorithm

As we previously said, it is possible to divide the input of FFT successively generating small sequence in the domain of the time, that action is named time decimation, TD. It is also possible to separate the sequence at the exit of FFT successively, instead of dividing the entrance. That implementation is called frequency decimation, FD. There is no difference at all between both decimations, so the decision to take one or another depends only on the programmer. The decimation can be used so much for FFT as for IFFT.

The next Fig.3.9 shows the previous decomposition of the input time samples of the DFT into odds and evens samples (N/2 samples for each one), for the case of $N=8$. Thanks to this decomposition the input samples remain tidy since the first sample to the last sample at the frequency response.

As previously said, the N/2 DFT's can be decimated into N/4 DFT's and so on, this way is possible to repeat the process until reaching the possible maximum level of division. In that point the basic block decimation is generated so it will be used in the whole FFT (called butterfly). An example of the butterfly of the FFT radix-2 TD and FD is in the following illustration.
In the same way that the decimation in the time generates a butterfly, the decimation in the frequency generates a corresponding butterfly. However the alteration in the flow of data will have to be done in the exit, and no more at the input.

The order of the butterfly is defined by the decimation of the radix. If it goes TD, first it is made multiplication and later the sum. If it goes FD, first it is made the sum and then it is made the multiplication.

A better way to show how this algorithm works is watching an example of the FFT in the next figure. The figure shows the implementation of the FFT for an input of N=8 samples:

Once we have seen the implementation, is possible to say that the FFT algorithm consist of a number of stages, this number of stages will vary depending on the number of samples at the input (n=log₂N).
At each stage it will be necessary to use the calculations of the DFT’s done at the previous stages. This means that the last stage will consist of one group with a lot of calculations, so if the first stage is seen, there will be a lot of groups but with a very small size each group.

At the first stage the size of the group will be reduced to the minimum, this way will be only one complex operation at each calculation group. This low level operation (with only 2 samples) as we have previously mentioned, is called butterfly. This butterfly is the most basic operation that can be done at each stage of the FFT. At each stage the number of butterflies will be doubled (at the example shown, at the first stage there are 4 groups with 1 butterfly each stage, and the last stage consist of 1 group with 4 butterflies).

For programming the FFT algorithm the first stage will consist, as previously said, of a scramble of the input samples. The simplest way to do this is using the digit reverse (most of the DSP’s has this mode). This mode consist of assigning a memory path for each input sample, and do the inverse bit per bit, this way for instance, for the memory path 011 it is possible to obtain the 110 (in other words, the reading order is changed, the usual way from left to right is changed to right to left)

![Fig. 3.12 Scramble for N=8](image)

The result is the same as dividing the input samples into odds and evens samples.

The only difference between decimation in time and decimation in frequency is that in DT this method is used with the input samples of the DFT, but using DF this method is used with the results obtained of the DFT, in other words, at the output of the DFT.

For the implementation of the FFT the \( W_N \) value, also known as twiddle factor, is needed. To calculate this factor it is necessary to use a CORDIC (Coordinate Rotation Digital Compute).

The aim of this project is not to explain in deep all the mathematics behind the FFT. So the way CORDIC works to calculate the sinus and cosinus values of this twiddle factor using the unit circle and the number of points N of the FFT, falls apart of the scope of this project.
3.5.2 Implementation scheme (VHDL)

The scheme followed to write the FFT algorithm is the one shown in Fig. 3.14, this scheme is able to be configured for a configurable input data width and N sample points in the FFT, and this scheme has the following blocks:

First of all when start is asserted and there is some input data, the address generator block assigns a memory position for each input sample. Both, data and memory path are saved into a dual port RAM.

The information contained in the dual port RAM is read by the 4 point FFT block. It will be read only when the address generator sends a start signal to this block. This will happen when the N paths have been assigned to the input data (N will change depending on the N point FFT). This block will contain the previously commented butterflies.
After the point FFT block, the twiddle factor will be calculated at the cordic and rotation factor generator blocks. These blocks will be activated by a start signal from the address generator block when N paths are assigned.

Finally this twiddle factors are applied to the output of the butterflies, and a bit reverse scramble is done, in the implementation of FFT, it is noticed that is necessary the remapping of the memory. In the implementation TD that remapping is made starting from the entrance, and in the implementation FD is starting from the exit of FFT. However that remapping can be made in a simple way. For instance, for the FFT radix-2 TD, the entrance has to be written in the addresses of memory 0, 2, 4, 6, 1, 3, 5 and 7. After having processed a scrambling phases, it has to write in 0, 4, 2, 6, 1, 5, 3 and 7. That scrambling follows a much defined order.

### 3.5.3 Implementation scheme (System Generator)

Due to the results obtained in the FFT programmed in VHDL, shown in detail in Annex A.2, and in comparison with the values of the FFT obtained in Matlab for the same data input as in the VHDL module, we decided to use the System Generator modules.

We used the FFT Xilinx blockset available in the Xilinx blockset library. During this section the configuration of the FFT block for the modulation and demodulation of the OFDM system will be explained.

The Xilinx FFT is a computationally efficient algorithm for computing a Discrete Fourier Transform (DFT) of sample sizes that are a positive integer power of 2. Fig. 3.15 shows the block interface used for the modulation and demodulation of the OFDM signal.

![Fig. 3.15 FFT/IFFT Xilinx block interface](image)
In Fig. 3.16 is shown the configuration window which gives the chance to configure the FFT block parameters the way better fits to the system.

![Configuration window](image-url)

**Fig. 3.16** Configuration of the FFT/IFFT block parameters

The values which are able to be configurated for the internal behavioral of the block are:

- **Implementation**: three architecture options to offer a trade-off between core sizes and transform time.
  - *Pipelined Streaming I/O*: Allows continuous data processing.
  - *Radix-4 I/O*: Loads and processes data separately, using an iterative approach. It is smaller in size than the pipelined solution but has a longer transform time.
  - *Radix-2 I/O*: Uses the same iterative approach as Radix-4, but the butterfly is smaller. This means it is smaller in size than the Radix-4 solution, but the transform time is longer.

The chosen configuration for our system is *Pipelined Streaming I/O*, otherwise after processing the N points the FFT block stops working.

- **Number of sample points**: Select the desired point size. All powers of two from 8 to 65536 are available. The number of sample points chosen is 1024.

- **Output ordering**: Each architecture offers the option of natural or reversed ordering of output data, with data being input in natural order. Natural order is the option selected because this way the samples corresponding to all the positions are correctly viewed.
scaling: The number of bits at the output can be scaled so the number of bits is the same as at the input, otherwise number of bits at the output will increase.

- **Scaled:** same number of bits at the output and at the input.
- **Unscaled:** the number of bits at the output is increased also depending in the size of the phase factor bit width and the number of points of the FFT.

The **unscaled** option is the chosen for the system because with the scaled option selected too much data precision is lost which is critical when connecting IFFT and FFT.

- **Rounding mode:** option to choose between **Truncation** and **Convergent Rounding** to be applied at the output of each rank. At the output of the butterfly, the LSBs in the data path need to be trimmed. These bits can be truncated or rounded using convergent rounding, an unbiased rounding scheme. When the fractional part of a number is equal to exactly one-half, convergent rounding rounds up if the number is odd, and rounds down if the number is even. Convergent rounding can be used to avoid the DC bias that would be introduced by truncation. **Convergent rounding** is the option chosen for the system.

- **Phase factor bit width:** option to choose a value between 8 and 24, inclusive to be used as bit widths for phase factors. This value is set to the default value which is 8 bits.

In the **Annex A.1** is possible to locate the different internal configurations of the FFT block depending on the selected block parameters.

Other important thing to know about this FFT/IFFT is the value and purpose of the input and output signals of the block (**block interface**):

- **Input signals:**
  - **Xn_re, Xn_im:** real and imaginary components of input data stream. The signal driving xn_re and xn_im can be a signed data type of width S with binary point at S-1, where S is a value between 8 and 24,bits inclusive.
  - **Start:** marks the beginning of each data frame. The start signal can be asserted as a pulse to start processing an input data frame or it can be tied to high
  - **Fwd_inv:** this signal configures the mode of the block, 0 for inverse transform (IFFT), and 1 for forward transform (FFT). By default, the FFT is configured for forward transform.
  - **Fwd_inv_we:** when this signal is asserted, loads the transform type from the input port forward for the next input data frame.
Output signals:

- **Xk_re, Xk_im**: real and imaginary components of output data stream. xk_re is the same as the input xn_re for Scaled mode, the same for imaginary component. The width of xk_re and xk_im signals grows left from the xn_re and xn_im binary point in the Unscaled mode by \((1+\log_2N)\).
- **Xn_index, xk_index**: marks the index of the input data (xn_index) and the output data (xk_index).
- **Busy**: this signal is high when the block is processing the current input data frame.
- **Vout**: marks when the output data is valid or marks it as invalid.
- **Done**: active high when the block is ready to output the processed data frame.

### 3.6 Modulator/Demodulator

This section will show the implementation of the OFDM modulation and demodulation schemes done with System Generator simulation tool. Two schemes have been taken into account:

- Modulation and demodulation of a 4-QAM input signal.
- Modulation and demodulation for different input signals.

The difference between both schemes is that in the scheme which has different input signals an M-QAM signal generator has been included. The insertion of the M-QAM signal generator means that a more complex QAM demapper must be placed at the output of the FFT block.

Basically the modulation and demodulation schemes of the OFDM system, as previously commented, consist of the concatenation of the IFFT and FFT blocks (the same block with different configurations), but this of course is not as easy as it sounds.

Next sections will show how the implementation of these schemes has been done.

#### 3.3.2 4-QAM mod/demod

The 4-QAM scheme for the modulation and demodulation of the OFDM signal has been implemented only using blocks from the Xilinx blockset library. The reason for using only Xilinx components is that the FPGA used in this project is from this brand, and using another kind of components the compilation will crash.

First of all a random signal generator is included with a message length of 1024 samples, this samples are mapped into a 4-QAM constellation symbols. The
output of the mapper can be converted to a certain number of bits if needed because depending on the number of bits at the input of the IFFT block the output will also have different output sizes. **Fig 3.17** shows the blocks for the random signal generator, 4-QAM mapper, and the adaptation of the output to be a correct input for the block IFF:

![Fig. 3.17 4-QAM generator](image)

In addition, two amplification blocks have been added because when the signal arrives to the FPGA there is attenuation, so the signal has to be amplified to observe the desired signal at the oscilloscope.

This way the IQ components that will go directly to the IFFT block for the modulation, are obtained.

The output of the IFFT block must be converted to a lower number of bits because the input of FFT block to which will be connected this output has a maximum bit size which otherwise will be exceeded.

Previously to the bit conversion, the output signal from the IFFT must be decimated; otherwise the FFT block will overflow.

The start signal of the FFT block must be high at least two time cycles before any input data arrives to the block, otherwise the FFT block will crash. To do this, a delay of two cycles has been assigned to the IQ input components of the block.

**Fig 3.18** shows the connections between both blocks:
The output of the FFT must be also amplified, it is the same case as the input of the IFFT block, because it also will be shown at the oscilloscope after been computed by the FPGA.

Other significant thing to take into account is the use of Shared Memories in the design. The Xilinx Shared Memory block implements a random access memory (RAM) that can be shared among multiple designs or sections of a design, are also very useful for controlling through the Matlab interface all the data involved in the design architecture.

Finally when the system is implemented and the simulation is working fine, an XtremeDSP Co-simulation block is added to the model and sent to the FPGA, this block will work during the simulation. The block behaves exactly as the subsystem from which it is originated, except that the simulation data is processed in hardware instead of software.

### 3.3.3 M-QAM mod/demod

The M-QAM modulation and demodulation scheme is only a modification of the previous architecture already shown. The part of the signal modulation and demodulation is basically the same, but the signal generation and the demapping blocks are different from the model seen before.

The signal generator for this scheme is able to generate different constellations, 4-QAM, 16-QAM and 64-QAM. A demapper block will be also needed at the output of the FFT block to demap each of the generated constellations.
The control of the generated constellations will be through Matlab and as we said in the previous scheme, the use of Shared Memories will be mandatory.

Fig. 3.19 shows the M-QAM signal generator and the adaptation its output to be a correct input for the block IFF:

The block diagram shown in the Figure is able to generate the following constellations: 4-QAM, 16-QAM and 64-QAM.

The M-QAM signal generator can be controlled through Matlab, which makes possible to select the desired constellation through this program’s interface.

This architecture has the following parts: the random symbol generator, the M-QAM mapper, two multiplexers, and the block to adapt the output signal.

The random symbols generated are mapped into the different types of modulations. The outputs of the mappers are the real and imaginary components, which go to different multiplexers. There is a selection signal which goes to both multiplexer, and depending on the value of the signal this output will vary. If the selection signal is 0, the output of the multiplexers are the IQ signals from the 4-QAM mapper, if 1 the outputs are IQ signals from the 16-QAM, and finally if is 2 the output will be from the 64-QAM. Finally the output of the multiplexer is converted to the demanded format needed by the IFFT block.

The IFFT/FFT blocks are exactly the same as in the 4-QAM mod/demod previously seen.

Finally the demapping blocks are placed after the FFT blocks. The main purpose of these blocks is to cancel the distortion introduced by some of the
blocks, as will be seen more in detail in the next chapter, applying the decision boundary already explained in this chapter.

Once we have explained the implementation of the main blocks of the OFDM system, next chapter will show the most relevant data obtained from its implementation already seen along this entire chapter.
CHAPTER 4 OFDM IMPLEMENTATION RESULTS

4.1 Introduction

During this chapter the results obtained by testing the complete implemented OFDM system will be shown. The testing can be divided into two parts, a simulation part where all testing is done on the PC and a hardware part where testing is done in the hardware.

First we will show the results obtained from the simulation part. All these simulations, as we said in the previous chapter, have been implemented using Matlab/Simulink. These results will correspond to the two implementation architectures already introduced.

Then, in the hardware part, the different equipment used for the testing of the OFDM system will be briefly introduced, and in addition, we will show the data obtained when using the FPGA device. The testing for the modulation and demodulation schemes have been carried out both at baseband and at the frequency of 3,5 GHz.

4.2 Simulation phase

For the simulation phase the testing is done on the PC using the System Generator tools. The implementation schemes used for the testing have already been introduced in the previous chapter so this section will be focused in the obtained results.

In the Annex A.3 the full schemes used are shown, although in the previous chapter all the parts of the scheme have already been described.

4.2.1 4-QAM Simulation scheme

As we previously said, the 4-QAM scheme for the modulation and demodulation of the OFDM signal has been implemented using only blocks from the Xilinx blockset library. The reason for using only Xilinx components is that the FPGA that will be used during the hardware phase uses only components from this brand, if any other kind of components is used the compilation will crash.

To have a better control above the data flow during the whole simulation is very interesting the use of Shared Memories in the design, the Xilinx Shared Memory block implements a random access memory (RAM) that can be shared among multiple designs or sections of a design, are also very useful for controlling through Matlab interface all the data involved in the scheme. This way the simulation can be controlled from Matlab without any problem.
Fig. 4.1 shows the main points of the OFDM system for the 4-QAM input signal. These points are before and after the IFFT and FFT blocks. Before the IFFT the results corresponding to the 4-QAM signal generated will be seen. After the IFFT the pictures corresponding to the modulation of the input signal will be shown. Then, the results of the adaptation of the output from the IFFT block to an acceptable input to the FFT block will be introduced. And finally is shown the FFT output, the results of the demodulation.

![Fig. 4.1 4-QAM Mod/Demod main points](image1)

Fig. 4.2 shows the IQ components of the output of the 4-QAM signal generator (point 1) introduced in the previous chapter, as already said, consist of a random bit generator and a 4-QAM mapper:

![Fig. 4.2 IQ components 4-QAM](image2)
The result of the generated signal as expected is a signal with the IQ values equal to 1 and -1.

**Fig. 4.3** also shows the 4-QAM signal that arrives to the IFFT block, but in this case is shown the symbol constellation, as already seen, the values are as expected, 1 and -1 for the real and also for the imaginary component.

![4-QAM generated constellation](image)

**Fig. 4.3** 4-QAM generated constellation

**Fig. 4.4** is a zoom over the point \{1,-1\}, all the point are exactly at the point \{1,-1\}, this picture is interesting when is compared with the demodulated constellation obtained at the output of the FFT block **Fig. 4.10**.
Next figure corresponds to the output of the IFFT block (point 2). This picture corresponds to the modulated signal, the modulation in time of the 4-QAM signal. Both components are shown, real and imaginary components.

Fig. 4.5 4-QAM signal modulated (IFFT output)

Fig. 4.6 shows the previous signal (modulated signal) but converted to a lower range of bits (point 3), the binary point of the signal has been shifted to the right; this means that the input signal has been divided by $2^N$, this way the overflow of the FFT block is avoided.
Next figure (Fig. 4.7) shows the constellation of the modulated signal after its adaptation to the FFT input range (point 3). This constellation shows the IQ plot corresponding to the different carriers of the OFDM signal.

**Fig. 4.7** OFDM carriers

**Fig. 4.8** shows the results of the demodulation, in other words, the output of the FFT block (point 4):
The output goes from 1 to -1 at each component as expected, like the 4-QAM signal was, but at this point it has been introduced some distortion (marked in red in the Fig. 4.10). This distortion has been introduced by the binary point shifter (division by $2^N$). Next figures will show more clearly how this distortion affects the signal.

**Fig. 4.9** shows the constellation after the demodulation point (**point 4**). The values of the symbols are the same that were generated by the 4-QAM signal generator at the transmitter, but now at the demodulation point there is some distortion which means that the demodulated symbols are not exactly at the $\{1,-1\}$ points for the real and imaginary components.
Next figure is a zoom to the previous 4-QAM demodulated constellation already seen. Is shown more in detail the distortion generated by the decimation block:

Fig. 4.10 Zoom to the demodulated 4-QAM signal (distortion)

The symbol that appears far away from the cloud of symbols corresponds to the peak that appears at the IQ components of the demodulated signal (marked in green). And the cloud of symbols as we already said, appear due to the little distortion that appears at the IQ components (marked in red).

4.2.2 M-QAM Simulation scheme

The M-QAM modulation and demodulation scheme is only a modification of the 4-QAM scheme already seen. The part of the signal modulation and demodulation is basically the same, but the signal generation and the demapping blocks are different from the model seen before.

The signal generator for this scheme is able to generate different constellations, 4-QAM, 16-QAM and 64-QAM.

A demapper block will be needed at the output of the FFT block to demap each of the generated constellations.

The control to choose one of the generated constellations will be through Matlab; therefore, the use of Shared Memories will be mandatory.

The points to obtain the results are exactly the same as in Fig. 4.1, so it is not necessary to put again the scheme.
The generated signals (point 1) that arrives to the IFFT block are the 16-QAM signal (Fig.4.11) and the 64-QAM signal (Fig.4.12), the 4-QAM signal is also generated, but since the results obtained for this modulation are exactly the same as already shown in the previous section, we don’t see the necessity to show them again.

As it can be seen in both figures, the transmitted values are the expected for both modulations, the different levels of the modulations are clearly seen in both constellations.

The values of the modulated signals (point 2 and also point 3 which is the same signal but adapted to a correct number of bit to be the input for the demodulation block) are shown in the Fig. 4.13, for both modulations, 16-QAM and 64-QAM:
OFDM Implementation results

Fig. 4.13 16-QAM and 64-QAM Tx modulated signals

At the output of the demodulation (FFT block), the results obtained (point 4) are shown in Fig. 4.14 for the 16-QAM signal and Fig 4.15 for the 64-QAM signal:

Fig. 4.14 16-QAM Rx demodulated signal

Fig. 4.15 64-QAM Rx demodulated signal
As happened with the previous scheme, the modulation and demodulation part introduces a little distortion to the demodulated signal. This distortion as is seen in the following pictures is not critical because the different levels of the constellation are clearly differentiated, this distortion don’t have any effect when the demapper block is added.

![Distortion at the 16-QAM demodulated constellation](image)

**Fig. 4.16** Distortion at the 16-QAM demodulated constellation

![64-QAM demodulated constellation](image)

**Fig. 4.17** 64-QAM demodulated constellation

At the output of the FFT (point 4), for this scheme is added an M-QAM demapper, which its main purpose is to avoid the little distortion introduced by the modulation and demodulation phases. This demapper applies different decision boundaries depending on the selected modulation.

**Fig. 4.18** shows the demapper included in the M-QAM simulation scheme. The output of the FFT is amplified; the value of the amplification depends on the
modulation generated at point 1. Then the corresponding demapper block demaps de signal correctly.

**Fig. 4.18** Demapped output of the FFT block

**Fig. 4.19** shows the results obtained for the demapping of the 16-QAM and 64-QAM signals. These demapped signals don’t show any distortion, the output values are the expected for each modulation, 16-QAM \{3,1,-1,-3\} for each component, and \{7,5,3,1,-1,-3,-5,-7\} for each component of the 64-QAM:

**Fig. 4.19** Demapped signals, 16-QAM and 64-QAM
The previous pictures confirm that the modulation and demodulation schemes in the simulation phase were correctly done, and the objectives were accomplished obtaining at the receiver the generated M-QAM signal.

4.3 Hardware phase

For the hardware phase the testing has been done using the laboratory equipment which will be introduced more in detail in Annex B.1, this equipment is:

- FPGA.
- IQ differential amplifier.
- Up/Down converter.
- Amplifier.
- Signal generators.
- Oscilloscope.
- Spectrum analyzer.

Once the system is implemented and the simulation is working fine, an XtremeDSP Co-simulation block is added to the model and sent to the FPGA, this block will work during the simulation. The block behaves exactly as the subsystem from which it is originated, except that the simulation data is processed in the FPGA instead of in the PC running the System Generator software.

So this way is possible to have a control of the data sent to the FPGA through the Matlab tool. The implementation schemes used for the hardware testing are already the same as in the simulation phase so during this section will be focused in the obtained results using the available hardware.

4.3.1 Baseband modulation

To obtain the results in baseband the devices which are needed for the assembly of the system are just the FPGA, IQ differential amplifier and the oscilloscope.

As already said the implemented and simulated scheme of the QAM schemes are sent to the FPGA via USB using the Xilinx XtremeDSP Co-simulation block tool.

Once the programmed core is stored in the FPGA the IQ components are sent through different output channels, in other words, the imaginary component is sent through one channel and the real component is sent by a different one.
Then the real and imaginary components go through the IQ differential amplifier, and finally the IQ components are shown independently by each channel of the oscilloscope.

**Fig. 4.20** shows the assembly commented above:

**Fig. 4.21** shows the screenshot of the OFDM modulation captured at the oscilloscope. The results obtained fit with the expected results obtained through the simulations of the scheme previously done. In these pictures is possible to see the modulated signal in time for each component, real and imaginary.
Fig. 4.21 Modulated signal shown at the oscilloscope

Fig. 4.22 shows a cloud of symbols. As previously seen at the Fig. 4.7 (simulation phase) this corresponds to the OFDM’s signal constellation, which means that all the carriers are shown:

Fig. 4.22 OFDM signal carriers

4.3.2 Modulation at 3.5 GHz

After the hardware testing done at baseband, a more complex assembly has been taken into account to test the signal at the RF-frequency band of 3.5 GHz.

To obtain the results at 3.5 GHz the connection between the devices is shown in the scheme in the Fig. 4.23 and in the Fig. 4.24:
Fig. 4.23 Scheme for the 3,5 GHz testing

Fig. 4.24 Assembly of the OFDM system at 3,5 GHz
The assembly at the beginning is exactly the same as in baseband modulation: the implemented and simulated scheme of the QAM schemes are sent to the FPGA via USB using the Xilinx XtremeDSP Co-simulation block tool. Once the core is loaded into FPGA, the IQ components are sent through different output channels, then the real and imaginary components go through the IQ differential amplifier which only has positive outputs.

The output goes connected to the module that first does the upconversion to desired frequency and then the downconversion.

The output which is connected to the upconverter is connected to the platform which contains several pass band filters. The other output goes directly connected to the oscilloscope to see the signal in the time domain.

Finally the output from the pass band filters will go to the spectrum analyzer to see the signal in the frequency domain.

A more detailed explanation of the assembly is shown in Annex B.2.

One last thing to say about the assembly already shown, is that the use of two signal generators, one at 1.95 GHz and the other one at 1.55 GHz, is not due to a limitation of the signal generators but it is because a limitation of the electronics used in the signal amplifiers, that is why two signal generators has been used and not only one at the single frequency of 3.5 GHz. The use of the signal generator must begin just when the output signal from the FPGA is stabilized; the reason of this is because before the stabilization of the signal some peaks are generated, and those peaks can damage the devices.

Fig. 4.25 shows the output signal from the FPGA sent to the oscilloscope:

![OFDM FPGA output signal](image-url)
Fig. 4.26 shows the OFDM spectrum at the spectrum analyzer, the OFDM signal at 3.5 GHz presents a bandwidth of 20 MHz:

![Spectrum OFDM signal at 3.5 GHz](image)

Fig. 4.26 Spectrum OFDM signal at 3,5 GHz

The demodulated constellation for the 4-QAM modulated signal has also the same distortion in the hardware phase as the already said in the simulation phase. Fig. 4.27 shows clearly this distortion in the demodulated constellation:

![Demodulated 4-QAM signal](image)

Fig. 4.27 Demodulated 4-QAM signal
The output values also correspond to the expected for a 4-QAM constellation, because the values shown are for a 25 bits output, so those values must be divided by $2^{25}$, this way the $\{1, -1\}$ values are obtained for the real and the imaginary component.

Finally, the HW testing has been done using the M-QAM scheme previously introduced.

The part of the OFDM system which adapts the output of the modulation block to be a correct input for the demodulation block has been modified. The gain and the delay have been modified, now these blocks don’t have a fixed value, they receive the value in real time through the Matlab interface. This process has been done in order not load the scheme in the FPGA every time these values change.

As the modulated signals are almost the same as previously seen in the figures Fig. 4.25 and Fig.4.26, is not needed to show them again. We are focused on the results of the demodulated signals obtained:

![Fig. 4.28 Demodulated 16-QAM and 64-QAM signals](image)

Like already happened with the 4-QAM results, the results obtained with the 16-QAM and 64-QAM demodulated signals fit with the results obtained in the simulation phase. Appears a little distortion at the output but it doesn’t affect significantly to the received signals, so the different levels of the M-QAM signals can be clearly identified.
CONCLUSIONS

The following conclusions have been asserted during the realization of this project:

- VHDL is a programming language that has some particularities, one of these particularities is that during the execution of the code most of the parts are not executed sequentially, are executed in a concurrent way.

- FFT/IFFT Xilinx blocks are very sensitive to delays and input data format. The input data to the FFT block should arrive two clock cycles after the start signal is assigned to the block.

- Some elements in the implementation schemes introduce a little bit of distortion but the obtained results from the simulation phase fit the theoretically expected results.

- An adjustable gain and delay are needed to configure the demodulation to obtain the right values. This characteristic is really interesting during the hardware phase because this way is not needed to download the scheme to the FPGA each time these values are changed.

- The results obtained in the hardware phase correspond to the results obtained in the simulation phase.

Future work

As future work, there are the next points:

- VHDL implementation of the FFT/IFFT working properly. Solving the mismatch with the obtained results from the Matlab FFT/IFFT.

- Some extra functions could be added to the system like for example, configure a MIMO (multiple-input multiple-output) system, or adding other functions to make more robust the system, like for example adding a puncturer or an interleaver.

- Possibility to add some functions to calculate some signal parameters like EVM, BER and PAPR.

Environmental impact

Up to the simulation phase, it’s been used a laptop with Intel Core duo desktop processor which has a greater energy efficiency in comparison with previous processors. Then during the HW phase the increase of energy consumption has greatly increased due the amount of devices used, which supposes a higher impact.
BIBLIOGRAPHY


ANNEX
Annex A: FFT System Generator Block

A.1 Different configurations

In this section are introduced more in detail the different configurations that can be selected for the Fast Fourier Transform System Generator block used in this project.

A.1.1 Pipelined, Streaming I/O

The Pipelined, Streaming I/O solution pipelines several Radix-2 butterfly processing engines to offer continuous data processing. Each processing engine has its own memory banks to store the input and intermediate data.

The core has the ability to simultaneously perform transform calculations on the current frame of data, load input data for the next frame of data, and unload the results of the previous frame of data.

The user can continuously stream in input data and, after the calculation latency, can continuously unload the results. If preferred, this design can also calculate one frame by itself or frames with gaps in between. This architecture supports all three fixed-point arithmetic methods (unscaled, scaled, block floating-point) as well as floating-point input data. In the scaled fixed-point mode, the data is scaled after every pair of Radix-2 stages.

The block floating-point mode may use significantly more resources than the scaled mode as it must effectively perform an unscaled transform. Therefore, if the input data is well understood and is unlikely to exhibit large amplitude fluctuation, using scaled arithmetic (with a suit-able scaling schedule to avoid overflow in the known worst case) is sufficient and resources may be saved. The input data is presented in natural order.

The unloaded output data can either be in bit reversed order or in natural order. When natural order output data is selected, additional memory resource is utilized. This architecture covers point sizes from 8 to 65536.

The user has flexibility to select the number of stages to use block RAM for data and phase factor storage. The remaining stages use distributed memory.
A.1.2 Radix-4 configuration

With the Radix-4, Burst I/O solution, the FFT core uses one Radix-4 butterfly processing engine. It loads and/or unloads data separately from calculating the transform. Data I/O and processing are not simultaneous.

When the FFT is started, the data is loaded. After a full frame has been loaded, the core computes the transform.

When the computation has finished, the data can be unloaded, but cannot be loaded or unloaded during the calculation process.

The data loading and unloading processes can be overlapped if the data is unloaded in digit reversed order.

This architecture has lower resource usage than the Pipelined, Streaming I/O architecture but a longer transform time, and supports point sizes from 64 to 65536. Data and phase factors can be stored in block RAM or in distributed RAM (for point sizes less than or equal to 1024).
A.1.3 Radix-2 configuration

The Radix-2, Burst I/O architecture uses one Radix-2 butterfly processing engine. After a frame of data is loaded, the input data stream must halt until the transform calculation is completed.

Then, the data can be unloaded. As with the Radix-4, Burst I/O architecture, data can be simultaneously loaded and unloaded when the output samples are in bit reversed order.

This solution supports point sizes from 8 to 65536 and uses a minimum of block memories. Both the data memories and phase factor memories can be in either block memory or distributed memory (for point sizes less than or equal to 1024).
A.2 VHDL implementation

Next figures shows the comparison between the results obtained with the FFT done with VHDL language and the FFT obtained with Matlab. The testing was done with two modulations. Fig. A4 shows the results obtained for a 4-QAM input signal, and Fig. A5 shows the results for a 64-QAM signal:

Fig. A.4 4-QAM FFT comparison

Fig. A.5 64-QAM FFT comparison

As can be seen the results obtained with VHDL do not fit with the results obtained with the FFT in Matlab. The main problems found were related with the very small resolution needed by some components of the FFT (few picoseconds) which makes the results not to be match up with the obtained through Matlab, and usually makes crashing the program because of the part in charge of the RAM memory saturates.
A.3 Full schemes (System Generator)

A.1.4 4-QAM scheme
A.1.5 M-QAM scheme
Annex B: Hardware phase assembly

B.1 Equipment

During this section will be briefly introduced the different devices used during the hardware phase.

Fig. B.1 shows the Field Programmable Gate Array (FPGA) which downloads the data sent by the PC, this device will process the data, in our case the different OFDM schemes. This device contains D/A converters. These 14-bit converters are operating at 105 Ms/s:

![Fig. B.1 Field Programmable Gate Array](image1)

Then this signal goes an amplifier of the real and imaginary components received, which output only have positive values.

![Fig. B.2 IQ differential amplifier](image2)
Then the I/Q modulator (Maxim MAX2023, Fig. B.3). This mixing stage is used to shift the baseband signal to an IF at 1.95 GHz.

The MAX2023 low-noise, high-linearity, direct upconversion/ downconversion quadrature modulator and demodulator, is designed for single and multicarrier 1500MHz to 2300MHz DCS 1800/PCS 1900 EDGE, cdma2000, WCDMA, and PHS/PAS base-station applications. The MAX2023 also yields a high level of component integration. This device includes two matched passive mixers for modulating or demodulating in-phase and quadrature signals, two LO mixer amplifier drivers, and an LO quadrature splitter.

![Fig. B.3 Upconverter/Downconverter](image)

Fig. B.4 shows the platform which contains the passband filters which attenuates the harmonics of the signal.

![Fig. B.4 Platform with the Passband Filters](image)

Finally there is also an amplifier which is directly connected directly to the platform already introduced, its output and its input are connected to this platform, in the next section is explained the reason:
B.2 Assembly in detail

The implementation of the hardware phase has been done over a hardware platform which is based on a modular structure which supports several RF-frequency bands. It should permit measurements at 3.4 to 3.8 GHz (the frequency band used for this project is 3.5 GHz). The selected structure of the test-bench is depicted in Fig. B.6:

Fig. B.5 Amplifier

Fig. B.6 Scheme for the 3.5 GHz testing

The hardware setup is composed of three major parts:

- Digital signal processing at baseband.
- First mixing stage ($f_{c,1} = 1.95$ GHz).
- Second mixing stage and amplification ($f_{c,2} = 3.5$ GHz).

The input signal is generated in the host PC and then downloaded to the FPGA, where the predistortion operation is performed.
The baseband signals are sent to the D/A converters. These 14-bit converters are operating at 105 Ms/s. After the reconstruction filtering the D/A output signals are provided at the input of the I/Q modulator (Maxim MAX2023). This mixing stage is used to shift the baseband signal to an IF at 1.95 GHz, which is located within the US PCS-band. The selected I/Q-modulator supports IF-frequencies between 1.5 to 2.3 GHz.

The core devices of the second upconversion stage are the mixers (Mini-Circuits ZX05-42MH) and the preamplifiers (Hittite HMC409LP4). The frequency shifted 2nd-stage input signal is filtered to suppress all out-of-band signal components and scaled by a step-attenuator. In this way the DUT input power level can be varied over a range of 15 dB in 0.5 dB steps. By the use of the following preamplifier average output power levels of up to 25 dBm can be generated.

To generate low distortion local oscillator signals for the mixers, the outputs of the two LO generators were boosted by amplifiers and lowpass filtered to minimize the second and third order harmonic distortion.

After the amplifiers’ output signal is reduced to the desired magnitude it is applied to the input of the receiver branch. Here, the downconversion to the IF frequency is performed. The mixer output signal is then filtered to suppress all undesired signal components and amplified to the required input power level of the I/Q demodulator (also Maxim MAX2023).

The signals provided by the demodulator are filtered before they are fed into the A/D converters. These devices show the same characteristics as the corresponding D/A converter.

To assure a fixed phase relationship between the different frequencies used in the test-bench the local oscillators were locked to the same 10 MHz reference clock. A picture presenting the complete setup is shown in Fig. B7.

![Fig. B.7 Complete setup for the 3,5 GHz testing](image-url)