Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Prototyping FBMC/OQAM for 5G Mobile Communications proof of concept

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Xavier Arteaga Martinez

DIRECTOR: Mir Ghoraishi

DATE: May, 28th 2015

Declaration of originality

I confirm that the project dissertation I am submitting is entirely my own work and that any material used from other sources has been clearly identified and properly acknowledged and referenced. In submitting this final version of my report to the JISC anti-plagiarism software resource, I confirm that my work does not contravene the university regulations on plagiarism as described in the Student Handbook. In so doing I also acknowledge that I may be held to account for any particular instances of uncited work detected by the JISC anti-plagiarism software, or as may be found by the project examiner or project organiser. I also understand that if an allegation of plagiarism is upheld via an Academic Misconduct Hearing, then I may forfeit any credit for this module or a more sever penalty may be agreed.

Prototyping FBMC/OQAM for 5G Mobile Communications proof of concept

Xavier Arteaga

Author signature: Date: March 16, 2015

Abstract

It is predicted that the number of wireless devices will be substantially increased in the next years while the available spectrum will remain the same. Consequently, the communications systems for next generation of mobile communications (5G) require a better spectral efficiency. One of the candidate schemes is the Filter Bank Multi-Carrier (FBMC) with Offset Quadrature Amplitude Modulation (OQAM). This scheme is not only suitable for providing higher rates but also for having a better spectral efficiency. However, it requires a higher implementation complexity than current systems.

This project constitutes an experience of prototyping the FBMC/OQAM scheme for an FPGA based platform.

Contents

List of Figures

List of Tables

Abbreviations

BER Bit Error Rate

DC Direct Current

DSP Digital Signal Processing

EVM Error Vector Magnitude

FBMC Filter Bank Multi-Carrier

FIFO First In First Out

FPGA Field Programmable Gate Array

(I)FFT Inverse Fast Fourier Transform

LUT Look-Up Table

MIMO Multiple Input Multiple Output

QAM Quaternary Amplitude Modulation

RF Radio Frequency

ROM Read Only Memory

SISO Single Input Single Output

ZC Zadoff-Chu

ZFE Zero Forcing Equalizer

Chapter 1

Introduction

Nowadays, the number of wireless devices connected to the internet is growing quite fast. In fact, it is predicted that for the year 2017 there will be around $2,621,678$ thousands of wireless devices [1].

However, the available spectrum will remain the same. Consequently, the next generation of mobile communications (5G) is driven by more spectral efficient waveforms.

There are a number of candidate waveforms for 5G. One of them is the Filter Bank Multi-Carrier (FBMC) with Offset Quadrature Amplitude Modulation (OQAM) which is a variant of the current Orthogonal Frequency Division Multiplex (OFDM). The FBMC/OQAM is not only suitable for providing higher rates but also for having a better spectral efficiency than OFDM. In other words, in FBMC/OQAM the adjacent channel interference is reduced and consequently more users can be using the spectrum simultaneously.

The target hardware for this FBMC/OQAM prototype will the FPGA based platform miniBEE. It does not only provides the hardware platform but also a useful set of tools for Software Defined Radio (SDR) development.

Chapter 2

Aims and objectives

The aim of this project is proof that the Filter Bank Multi-Carrier (FBMC) with Offset Quadrature Amplitude Modulation (OQAM) can be ported onto hardware. Specifically, it has to run on the miniBEE platform.

The prototyping of the FBMC/OQAM scheme implies design the FBMC/OQAM specific modules such as reference signals pre-coder, filter banks and synchronization schemes.

The implementation must be modular. In fact, the design must be decomposed in several modules which can be tested and verified separately from the rest of the design. Therefore, the design modules can be easily reused or modified.

The fixed point, resource and latency optimizations are not targets in this project. However, the design may reuse area-expensive operations such as multiplications and divisions to make the design feasible in terms of computational resources.

The time to implement the said scheme is five months.

Chapter 3

Methodology

In order to achieve the objectives described above in the specified time a methodology has been defined.

First of all, the target modulation will be reviewed in order to understand its principles and benefits. Secondly, the provided hardware and software platform are reported. Next, a survey of prototyping communications systems will be done with the purpose of defining the proper development procedure and verification.

After that, the simulation software is implemented and the hardware code too. At same time, all the code will be tested and verified following the defined procedures.

Once the system is implemented, it will run on hardware where the results will be extracted. Next, these results are discussed and the future work is proposed. Finally, the conclusions are given.

Chapter 4

Literature Review

In this chapter, the FBMC/OQAM scheme is reviewed. First of all, the unified architecture of the FBMC transceiver is revised. Secondly, the target platform is considered. Also, it is explained the FPGA programming file generation flow. Finally, a survey of FPGA rapid prototyping design flow will be given.

4.1 Introduction to Filter Bank Multi-Carrier

Currently, Orthogonal Frequency Division Multiplex (OFDM) is being used for the latest commercial mobile communications systems. In OFDM, large side lobes are generated due to the use of a rectangular shape window after performing the Inverse Fast Fourier Transform (IFFT). Those side lobes increase the interference in adjacent channels. This effect is called spectral leakage and its reduction would increase the spectral efficiency of the communication systems.

The main difference between OFDM and FBMC/OQAM is that the IFFT window is longer in time domain and it does not have abrupt transitions. As a result, the side lobes in frequency domain are reduced and also its spectral leakage. Moreover, it is said in a number of papers that FBMC modulation has better tolerance to carrier frequency offset (CFO) and sampling frequency offset (SFO) as well. However, the OFDM symbols are overlapped and consequently an Offset QAM (OQAM) modulation must be used.

The aim of using the OQAM modulation is make possible the signal regeneration. The OQAM consist in the creation of a half symbol time shift between the real

and the imaginary of the OFDM signal. In order to get this time shift, the OFDM transmitter processing chain is divided in two branches, the real and the imaginary. Each branch apply a phase shift to their samples before performing the IFFT per separate. Next, the imaginary branch time domain signal is delayed half symbol. After that, the window is applied in both branches before combining them.

In the receiver side it is performed the opposite process. The received frame is divided in two branches and shifted windows are applied before performing the FFT. After the FFT, the phase is recovered and the branches are combined again before recovering the data.

The figure 4.1 shows the comparison between OFDM and FBMC/OQAM main modules. They share the same resource generator and payload sink. Besides, the Digital Up and Down Converters can be reused.

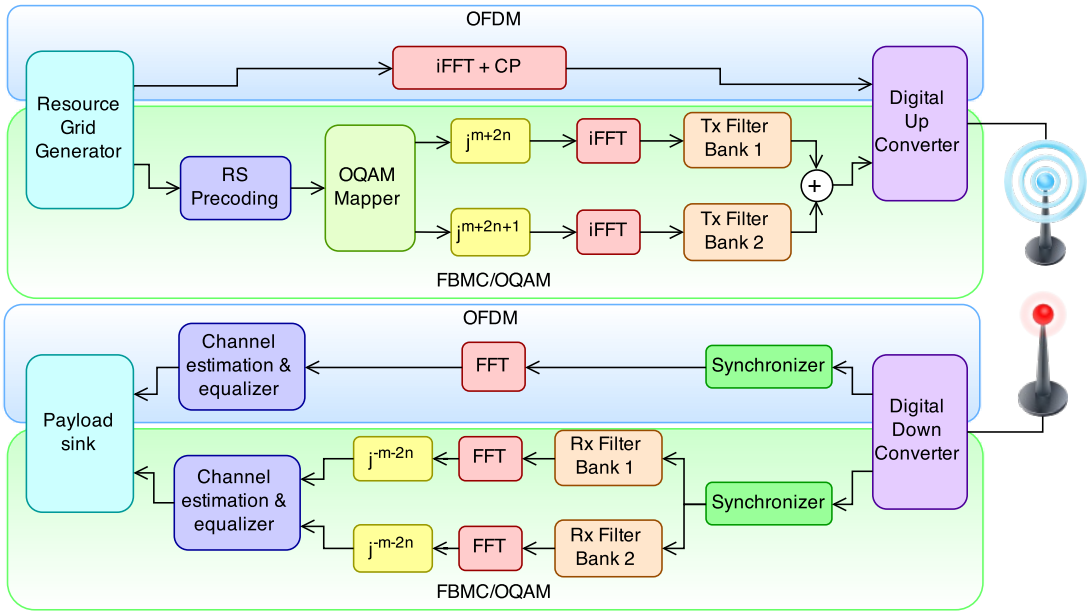


Figure 4.1: Comparison between OFDM and FBMC physical layer block diagram

Different filter shapes are proposed as IFFT windows in the literature. The most predominant are the Isotropic Orthogonal Transform Algorithm (IOTA) [2] and the reference PHYDYAS filter [3]. In fact, the reference paper [4] discusses the performance of both prototypes filters and the reference PHYDYAS filter gives better results than IOTA. The time domain shape of both filters are compared in the figure 4.2.

Despite the fact that both filters time domain shape look similar, they have different frequency responses. The figure 4.3 compares the frequency domain transition band

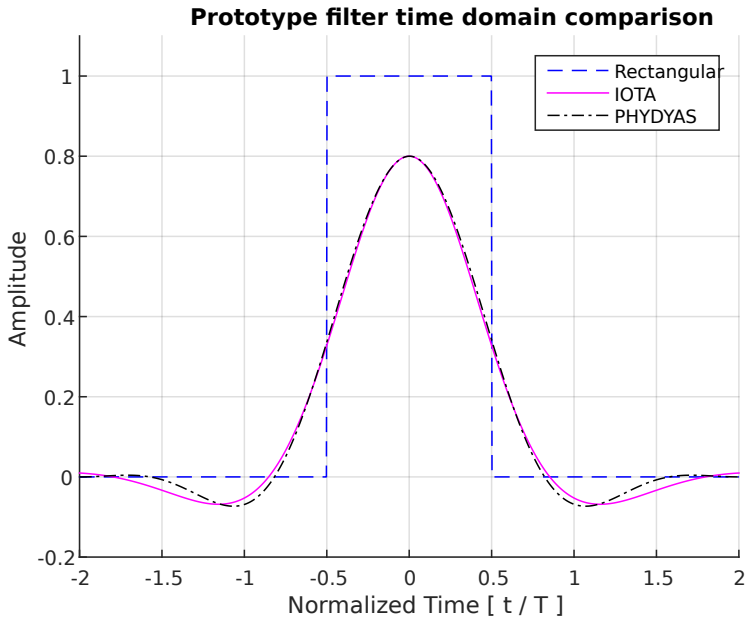


Figure 4.2: Time domain prototype filters

for a modulated signal in OFDM (rectangular window), IOTA and PHYDYAS. One can see how the side lobes are drastically reduced by changing the IFFT window.

In a number of publications, it is proposed to implement the efficiently the windowing process using a *poly-phase filter network* or *filter bank*. Where the coefficients are generated from the prototype filter. In fact, in the reference [5] its is proposed a unified parallel algorithm to perform the said windowing. Moreover, the reference paper [2] proposes a Isotropic Orthogonal Algorithm (IOTA) with blind channel estimation.

As a result of using a longer non-rectangular shape window, it does not only creates Inter Carrier Interference (ICI) but also Inter Symbol Interference (ISI). Because of that there are two different approximations for channel equalization depending on their reference signal pattern: preamble based or scattered.

On the one hand, in preamble based methods [6], a known reference sequence is transmitted occupying all the band. This known sequence is transmitted for a number of symbols in exchange of removing the intrinsic interference. Consequently, its capability to estimate variant channels is lower and the overhead is higher.

On the other hand, in the scattered reference signals scheme, distributed well known elements are set in the resource grid. In this case the intrinsic interference can be removed from the Reference Signals (RS) in the transmitter. It not only requires

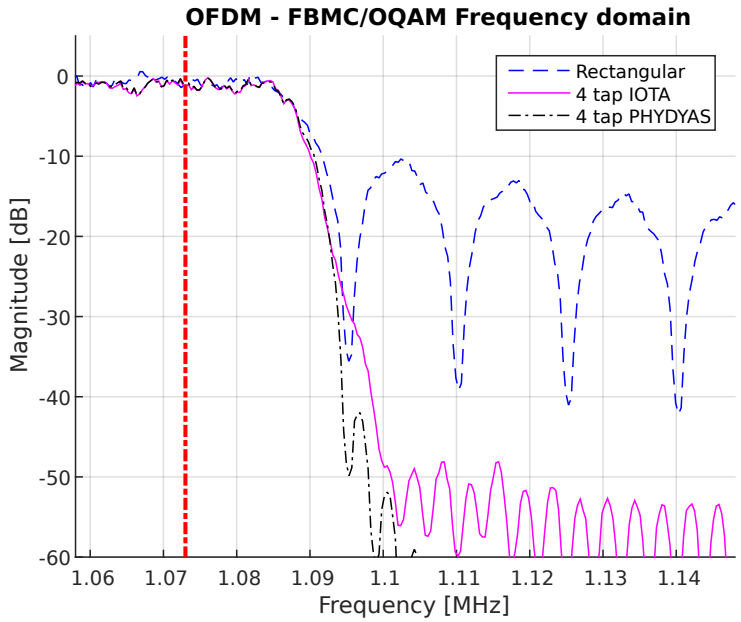


Figure 4.3: Frequency domain prototype filters

extra complexity and computational resources in the receiver channel estimation but also in the transmitter for pre-coding the RS.

In [7] a SDR based prototype is presented where the intrinsic interference at the reference signals is rejected using auxiliary pilots. In contrast, it is proposed use a pre-coding technique in [8] where the interference is removed in the transmitter.

4.2 Hardware platform

The provided hardware platform for prototyping is the miniBEE which has been supplied by BEEcube . This is a FPGA based platform where the FPGA has a direct PCI communication with an on-board Intel i7 computer and a FPGA Mezzanine Card (FMC) model 111.

The on-board Intel i7 computer provides an entire environment based on a Linux CentOS distribution to access to the FPGA memory registers as well as configure the FMC111 card. In fact, BEEcube has provided a complete set of tools to control the embedded registers and memory from the Matlab environment.

The FMC111 provides an RF front end to the FPGA and it is described in the document [9]. This front end comprises the Digital to Analogue Converters (DAC), filters, mixers and amplifiers required to transmit the generated signals in the FPGA.

Also, it also includes the required components for receive RF.

The top level of design is in Xilinx System Generator which uses Matlab Simulink to develop the code graphically. In addition, BEEcube provides a set of tools to add custom embedded memories and registers to the design under development. The main idea behind the software platform (BPS) is that the designer only has to design his own user logic and BEEcube software will generate all the environment for him.

The user logic is synthesized as a Internal Peripheral. Where the FMC111 controller, memory blocks (BRAM) and registers are attached to it. Moreover, all those components are connected to a Processor Local Bus (PLB) of the Microblaze embedded processor.

Moreover, the FPGA programming file generation flow is completely automated by BEEcube software (BPS). In fact, the developer only has to press a button and BPS will bring the file.

All the architecture described adobe is depicted in the figure 4.4.

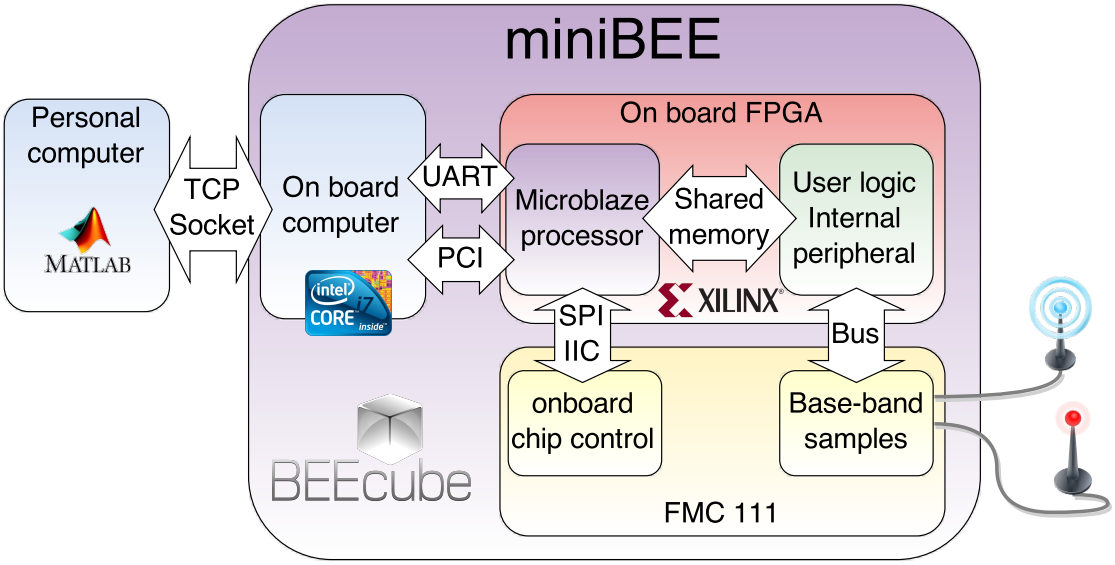


Figure 4.4: BEEcube Block diagramm

4.2.1 Programming file generation flow

As mentioned, in the previous section, the FPGA programming file generation is integrated in the BPS software and it is transparent to the developer. This sections

presents the FGPA programming file generation.

First of all, the Simulink Model is updated, compiled into VHDL and Netlist. In other words, the design blocks used in the Simulink code are translated into VHDL instances and Netlist files (**.ngc*). This first step generates a custom Internal Peripheral (IP) for the embedded system.

Secondly, BPS compiles the Microblaze embedded processor code which is in C language and generates the binary executable file (**.elf*). This file will be loaded into the FPGA programming file in the end of the process.

In the next step, BPS uncompress the Xilinx Platform Studio (XPS) base project from its folder. This base project is a template containing the basic components of the design. After that, the custom IP generated previously and other user components such as BRAM are added to the base project.

Once the XPS project is ready, it is synthesised and mapped. Finally, the Microblaze processor program code is added and the final FPGA programming file is copied into the *output* folder.

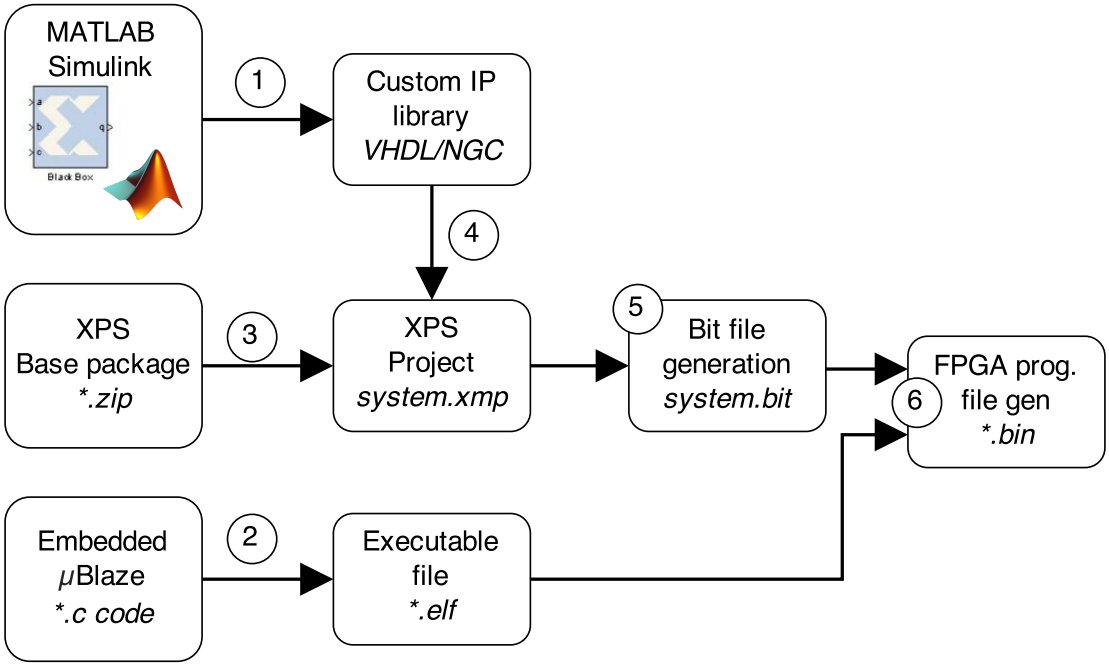


Figure 4.5: FPGA programming file process

4.2.2 Changing FPGA programming file options

BPS scripts use by default the configuration options that generate the FPGA programming file as *fast as possible*. This configuration generates the programming file in the minimum time without perform any extra effort to route the signals inside the FPGA. Also, the fact that designing with Simulink is at very high level, the number of timing errors due to clock routing are quite common. Because of that, the FPGA programming flow options should be modified.

Since BPS uses XPS for generating the FPGA logic file (bit file), it reads a option file (**.opt*). The documents [10] [11] contain the information regarded to the flow options. The custom options file must be loaded just before the XPS synthesis and mapping. It has been possible to override the default flow option with the desired ones. One can find a tutorial of how to patch BPS for customizing the flow options file in the appendix A.

4.3 Hardware prototyping of FBMC

Currently, there are a very few implementations of FBMC/OQAM implementations. A FBMC/OQAM transmitter implementation is described in the reference publication [12]. In fact, it is not only described the hardware algorithms used for the implementation but also it is described the design and prototyping flow.

The authors of this publication divided their implementation process in four phases. The first phase is the *algorithm simplification and optimization* where the algorithms, numeric representation, parallelisation and requirements are defined. The second phase explained in the publication is the *architecture exploration* where the algorithms of the previous phase are tested.

The third phase is *Hardware implementation* where the hardware code is generated. Finally, the forth phase is the *On-Board Validation and Demonstration* where the prototype is finished at the end of this phase.

The reference paper [13] describes the main differences between the emulation and the prototyping cycle. In fact, it is explained how time consuming is the emulation in front of the prototyping. Because of that, the hardware will be co-simulated in indivisible (atomic) modules before generating the main prototype.

Moreover, it is proposed an hybrid software and hardware co-simulation in the refer-

ence paper [12]. The said co-simulation uses Matlab for coefficients generation and waveform verification while ModelSim is used for the hardware simulation.

Due to the fact that the implementation of this project is done using Xilinx System Generator [14], a similar co-simulation architecture will be used. Therefore, Matlab is used instead of using ModelSim. The figure 4.6 shows the said co-simulation diagram where Matlab code contains the software DSP behaviour and uses Xilinx System Generator under Simulink environment to emulate the hardware algorithms.

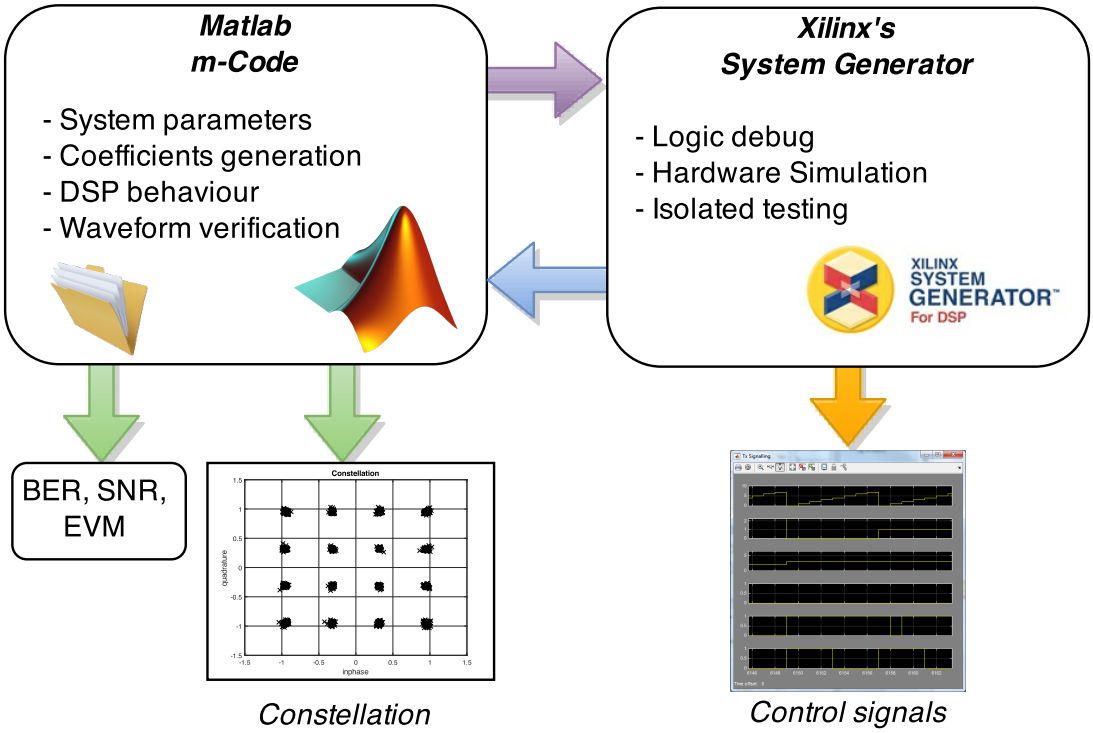


Figure 4.6: Simulation structure

Also, the Matlab simulation show some statistics and signals such as the Error Vector Magnitude (EVM), Bit Error Rate (BER), constellation and so on.

Chapter 5

Implementation

This chapter describes the hardware prototyping of the Filter Bank Multi-Carrier (FBMC) scheme. First of all, the frame structure is presented and the main system parameters with it.

Secondly, the transceiver architecture is presented. At this point the main modules are explained without entering in implementation details. Next, the dedicated FBMC/OQAM Digital Signal Processing (DSP) algorithms details will be given. They are the pre-coding, filter banks, time synchronization and channel equalization stages.

The explanations go with some block diagrams of the hardware implementation. Those diagrams are only graphical references of the implemented algorithms and they may not look like Simulink code.

5.1 Frame structure and system parameters

This section reviews the frame structure used for the transceiver prototype. The radio frame is composed of N_{symb} symbols of length M where the first symbol of the frame is dedicated to time synchronization. The length (N_{symb}) of the frame has a trade-off between time synchronization error and signalling overhead. It is due an excess of length of the frame causes a lack of synchronization (due to SFO) and consequently the received signal quality drops. In contrast, an excessive short frame would increase the synchronization overhead and degrade the performance of the system. The described frame is depicted by the figure 5.1.

The synchronization signal is generally mapped in the 62 central sub-carriers, around the DC. Nevertheless, it can be extended to 126 is the second zone of Nyquist is considered. The synchronization signal generation and mapping are explained later in the section 5.4.3.

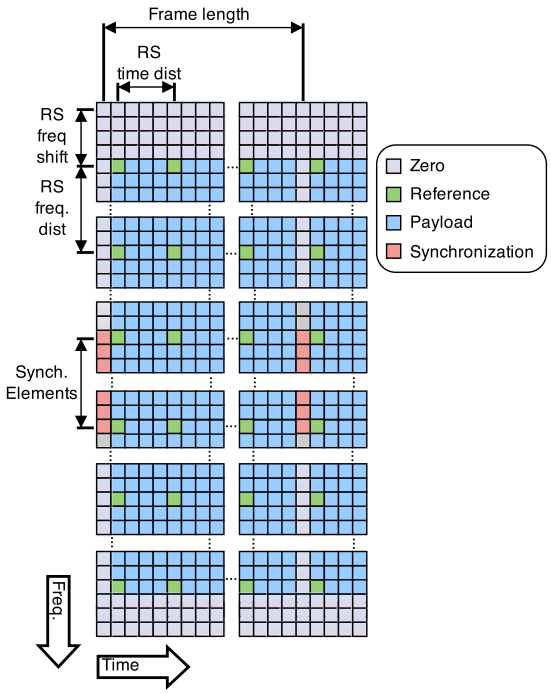


Figure 5.1: Proposed resource grid elements

A scattered reference signals scheme is used for estimating and equalizing the channel in the receiver where the distance between reference signals in frequency and time domain are fixed in the Matlab code. Also, the time and frequency domain RS positions shifts are defined there. Once again, these design parameters depends on the radio channel and they must be set according a trade-off between the received signal quality and the overhead they imply.

Moreover, some zeros are added in the edges of the band with the aim of filling the sub-carriers which are not covered by the reference signals. So, the number of errors in the edges is reduced.

5.2 Transceiver architecture

This section describes the architecture of the transceiver. The main Matlab code is based in frames. Particularly, every simulation generates a stream data which

contains two or more radio frames. This stream is processed once by each function. Despite the fact that the frame based is not an efficient way to simulate, it will make easier the hardware emulation afterwards. In contrast, the hardware code is sample based, it means that one sample of the radio frame is generated with every clock cycle.

The overall code has been divided in two big main blocks, the transmitter and the receiver. The transmitter includes the code related with the data generation, resource mapping, modulation and up conversion. While the receiver includes the code for down conversion, time synchronization, demodulation, resource extraction and data recovery.

5.2.1 Transmitter resource grid generation

The transmitter payload generator creates binary data, maps it into complex elements and places them in resource grid. In fact, this module is common for the FBMC/OQAM and OFDM schemes.

For this project, the binary data source generates the information to transmit using a Read Only Memory (ROM) with random data. The memory is generated when the Simulink model is updated. The main advantages of this generator are its simplicity and efficiency. Also, it always generates the same sequence. As result, the receiver can compare the received binary data and estimate the number of errors in each frame.

The generated bits are mapped into complex elements before being assigned to their resource positions in the frame. The codebook used for the bit mapping into symbols is the same described in the LTE Standard [15] and the supported mappings are the 4-QAM, 16-QAM and 64-QAM.

In order to map the payload into the sub-carriers the data symbols are queued and placed in the first available resource element. This stage not only maps the payload symbols, but also it adds the synchronization signal and Reference Signals (RS) into the resource grid as described previously in the section 5.1.

5.2.2 Filter Bank Multi-Carrier modulator

The modulator converts the resource grid provided by the resource generator to a time domain signal. Firstly, the reference signals are precoded so that the intrinsic interference is reduced, more details will be given in the section 5.4.1.

Secondly, the resource grid elements are mapped into the Inverse Fast Fourier Transform (IFFT). Where the lower band goes from the sub-carrier index 0 to $N_{SC}/2-1$ and the upper band goes from the index $N_{SC}/2$ to $N_{SC}-1$. The upper band is mapped above the DC while the down band is mapped in the last N_{SC} samples. The said mapping is described by the equation 5.1. Where $a_{k,l}$ is the resource element in the sub-carrier k and symbol l. While $x_{m,l}$ is the IFFT element where m is it mapping index in the IFFT.

$$x_{m,l} = \begin{cases} a_{(m+N_{SC}/2-1),l} & 0 < m \leq N_{SC}/2 \\ a_{(m-M+N_{SC}/2),l} & M - N_{SC}/2 < m < M \\ 0 & otherwhise \end{cases} \tag{5.1}$$

After that, the complex elements coming from the IFFT mapper are divided in two branches. The real and imaginary branches are multiplied by complex sequences. The branch split and phase shift is described by the equation 5.2. Where $x^{Re}_{m,l}$ and $x^{Im}_{m,l}$ are the real and imaginary branches.

$$\begin{aligned} x^{Re}_{m,l} &= \mathbb{R}e\{x_{m,l}\} \times j^{m} \\ x^{Im}_{m,l} &= \mathbb{I}m\{x_{m,l}\} \times j^{m+1} \end{aligned} \tag{5.2}$$

After the phase shift, the IFFT is performed. Once the data is in time domain, the IFFT window is applied in the filter bank. Its design details will be considered in the section 5.4.2.

5.2.3 Software Defined Radio Front end

The block between the transmitter filter bank and the Digital to Analog Converter (DAC) is the Digital Up Converter (DUC). The purpose of this block is up-sample the transmitted signal for matching the base band sampling rate with the DAC rate. Also, it is performed a Low Pass Filtering (LPF) which removes spurious images generated in the up-sampling stage.

The data coming from the Analogue to Digital Converter (ADC) feeds the Digital Down Converter (DDC). This block filters the received data using a Band Pass Filtering (BPF). It attenuates acquired interferences in the ADC such as the LO coupling in the DC. That the useful received signal is centred at one eight of the sampling frequency, so the DC can be removed.

After the filtering stage, the signal is down sampled. As a result, the received useful band is centred in DC instead of one eighth of the ADC sampling rate.

5.2.4 Filter Bank Multi-Carrier demodulator

First of all, the time domain synchronization is performed. It is described later in the section 5.4.3.

Once the received signal is synchronized the FFT window is performed in the receiver filter bank whose implementation details are given in the section 5.4.4.

Next, the Fast Fourier Transform (FFT) is computed and a phase shift described by 5.3 is applied. The aim of this phase shift is recover the phase from the transmitter phase shift.

$$\begin{aligned} y_{m,l}^{Re} &= y_{m,l}^{Re} \times j^{-2n-m} \\ y_{m,l}^{Im} &= y_{m,l}^{Im} \times j^{-2n-m} \end{aligned} \tag{5.3}$$

After the phase recovery, before the channel estimation and equalization the frequency domain signal must be re-indexed. This stage removes guard bands and the DC component. The FFT de-mapping (re-index) is described in the equations 5.4. Where $y_{(n,l}^{Re}$ and $y_{(n,l}^{Im}$ are the real and imaginary output branches of the FFT for the frequency domain index n and the symbol l. While $b_{k,l}^{Re}$ and $b_{k,l}^{Im}$ are the resource elements at the sub-carrier k and symbol l.

$$\begin{aligned} b_{k,l}^{Re} &= \begin{cases} y_{(k+M-N_{SC}/2),l}^{Re} & 0 < k < N_{SC}/2 \\ y_{(k-N_{SC}/2+1),l}^{Re} & N_{SC}/2 \leq k < N_{SC} \end{cases} \\ b_{k,l}^{Im} &= \begin{cases} y_{(k+M-N_{SC}/2),l}^{Im} & 0 < k < N_{SC}/2 \\ y_{(k-N_{SC}/2+1),l}^{Im} & N_{SC}/2 \leq k < N_{SC} \end{cases} \end{aligned} \tag{5.4}$$

The last stage of the FBMC/OQAM demodulator before the payload recovery is

the channel estimation and equalization which is explained in the section 5.4.5.

5.2.5 Payload recovery

This section is in charge of recovering the binary payload from the resource grid. First of all, the payload is de-mapped from the resource grid. In other words, the synchronization signals and reference signals are removed.

The next step is convert the resource elements of the extracted payload into soft-bits. This is the opposite process of bit mapping in the transmitter where the mapping followed the codebooks described in [15].

Moreover, the conversion from symbols to soft-bits has been implemented using the equation 5.5 for 4-QAM, 5.6 for 16-QAM and 5.7 for 64-QAM. The three previous expressions assume that the channel equalization works properly and the received complex elements values are properly scaled.

$$\begin{aligned} y_b[2n] &= \mathbb{R}e\left\{y_{4QAM}[n]\right\} \\ y_b[2n+1] &= \mathbb{I}m\left\{y_{4QAM}[n]\right\} \end{aligned} \tag{5.5}$$

$$\begin{aligned} y_b[4n] &= \mathbb{R}e\left\{y_{4QAM}[n]\right\} \\ y_b[4n+1] &= \mathbb{I}m\left\{y_{4QAM}[n]\right\} \\ y_b[4n+2] &= \tfrac{2}{\sqrt{10}} - |y_b[4n]| \\ y_b[4n+3] &= \tfrac{2}{\sqrt{10}} - |y_b[4n+1]| \end{aligned} \tag{5.6}$$

$$\begin{aligned} y_b[6n] &= \mathbb{R}e\left\{y_{4QAM}[n]\right\} \\ y_b[6n+1] &= \mathbb{I}m\left\{y_{4QAM}[n]\right\} \\ y_b[6n+2] &= \tfrac{2}{\sqrt{42}} - |y_b[6n]| \\ y_b[6n+3] &= \tfrac{2}{\sqrt{42}} - |y_b[6n+1]| \\ y_b[6n+4] &= \tfrac{4}{\sqrt{42}} - |y_b[6n+2]| \\ y_b[6n+5] &= \tfrac{4}{\sqrt{42}} - |y_b[6n+3]| \end{aligned} \tag{5.7}$$

Finally, these soft-bits are converted to hard bits and they are compared with the transmitted sequence for counting number of errors. So the Bit Error Rate (BER) can be computed.

5.3 Implementation parameters

The transceiver implementation has been done using fixed design parameters. First of all, the number of active sub-carriers has been fixed to 137 (144-7) and the (I)FFT size is 256. The frame length has been set to 16 symbols so that the simulation time is reduced and it has a low synchronism signal overhead.

A frequency domain spacing of *8 sub-carriers* has been set between the scattered reference signals. This frequency reference signal spacing may be enough to equalize properly in the lab environment without increasing the overhead. Also, the time domain reference signals spacing has been fix to *4 symbols*.

Moreover, since the sub-carrier spacing is wanted to be the same than LTE (15 kHz), the (I)FFT data rate will be 3.84 MHz. Therefore, the control signalling of our design can be oversampled at a rate of $30.72MHz$ (8 times faster than the base sampling rate).

The table 5.1 summarizes the transceiver design parameters.

5.4 Dedicated FBMC/OQAM Algorithms

This section describes a few DSP hardware dedicated to FBMC/OQAM. The algorithms presented in this section provide feasible solutions to the hardware implementation. In fact, they are based in oversampling methods. In fact, they take advantage of the said oversampling to reuse hardware-expensive operations such as complex multiplications.

5.4.1 Reference signals pre-coding

Due to the fact that the used scheme (FBMC/OQAM) adds intrinsic interference in the receiver, it is not possible to estimate the channel using scattered reference signals without any additional processing. Because of that, a interference cancellation pre-coding technique in the transmitter is proposed.

The intrinsic interference in the channel estimator has been modelled as the ideal RS plus the addition of the adjacent resource elements multiplied by a complex constant. The equation 5.8 shows this interference model where $r_{k,l}$ is the reference

General	
Base sampling rate	3.84 MHz
Hardware Sampling rate	245.76 MHz
Sub-carrier spacing	15 kHz
Num. Active sub-carriers	137
Symbol duration	66.67 μs
Frame duration	1.067 ms
Constellations	4-QAM, 16-QAM and 64-QAM
Reference Signals	
Frequency spacing	8 sub-carriers
Time spacing	4 symbols
Filter Bank	
Generation algorithm	IOTA/PHYDYAS
Num. taps	4
Coefficient width	16 bit
Coefficient point	15 th
Reference Signals Pre-coding	
Num. sub-carriers	3
Num. symbols	5
Coefficient width	16 bit
Coefficient point	15 th

Table 5.1: System parameters summary

signal at the sub-carrier k and symbol l. While $\nu^{Re}_{n,m}$ and $\nu^{Im}_{n,m}$ are the complex interference coefficients for the real and the imaginary part at the index offset of n symbols and m sub-carriers from the RS.

$$r_{k,l} = a_{k,l} + \sum_{m=-d_f}^{d_f} \sum_{n=-d_t}^{d_t} \nu^{Re}_{n,m} \mathbb{R}e\{a_{k+m,l+n}\} + \nu^{Im}_{n,m} \mathbb{I}m\{a_{k+m,l+n}\} \tag{5.8}$$

The constants d_f and d_t are the maximum index offsets in frequency and time domain taken into account.

This function is fed with the output matrix of the resource elements mapping and its output is another matrix with the same dimensions.

Hardware algorithm

In order to pre-code the reference signals, their neighbour resources intrinsic interference components must be estimated and subtracted to the transmitted reference signal. In this case, 5 symbols and 3 sub-carriers are taken into account. Hence, it requires 15 complex multiplications to perform this operation (60 regular multiplication operations in total).

The algorithm represented by the block diagram in the figure 5.2 has been implemented with the aim of making this technique feasible in terms of FGPA resources. The algorithm is based in a series of shift registers which provide access to the reference signals resource neighbours. Every time that the control block detects a reference signal in the central register, the design holds all the neighbour values for 15 clock periods for estimating the intrinsic interference cancellation.

A multiplexer is used to select the resource element and a Look-Up Table (LUT) to select the corresponding coefficient. Both elements addresses are generated using a 4 bit width counter.

Finally, the results of the multipliers are accumulated for 15 clock cycles and stored in a First In First Out (FIFO) queue where they are read every time a reference signal is detected at the output of the shift registers.

As a result, the number of regular multiplications has been reduced from 60 to 4.

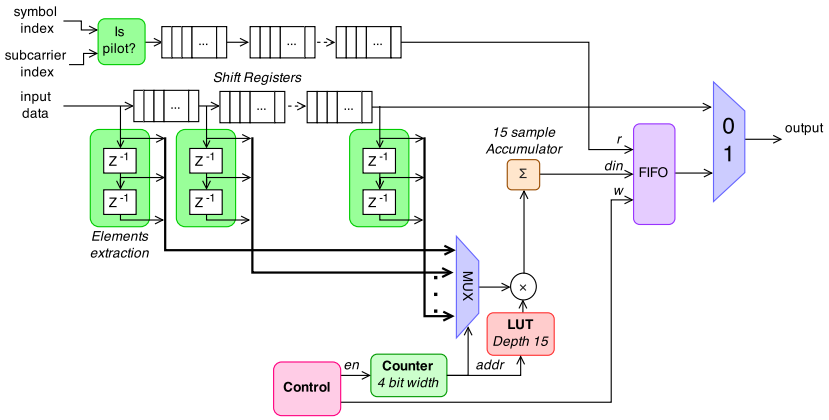


Figure 5.2: Reference signals pre-coding block diagram

5.4.2 Transmitter Filter bank

In the theory, the filter bank is a poly-phase filter network which applies a window to the FFT. The length of the IFFT window is longer than the symbol (M points). The said filter bank is composed by M filters of L coefficients each filter. In fact, L is the length of the window in symbols.

The coefficients are extracted from the prototype filter of length $L \times M$. The poly-phase filter coefficients extraction from the prototype filter follows the equation 5.9. Where g_n is the prototype filter and m the index of filter. While $g^{Re}_{m,n}$ and $g^{Im}_{m,n}$ are the real and imaginary coefficients of the filter bank.

$$\begin{aligned} g^{Re}_{m,n} &= g_{Mn+m} \\ g^{Im}_{m,n} &= g_{Mn+m-M/2} \end{aligned} \tag{5.9}$$

Furthermore, the output of the filter bank is given in the equation 5.10. Where m is the filter index (IFFT output index), n is the index of symbol and l the index of the coefficient.

$$x_{m,n} = \sum_{l}^{L-1} x^{Re}_{m,n-l} \cdot g^{Re}_{m,l} + x^{Im}_{m,n-l} \cdot g^{Im}_{m,l} \tag{5.10}$$

It means that for every IFFT performed and a prototype filter of length $M \times L$, it requires $2 \times M$ filters of N taps. If every filter is implemented without any optimization, the design would require $2 \times M \times N$ multiplications which is not possible to implement.

However, since FPGA clock rate is faster than the sample rate of the filters, the entire poly-phase filters network can be implemented with a pair of generic multiplications per branch and a complex accumulator as illustrated in the figure 5.3.

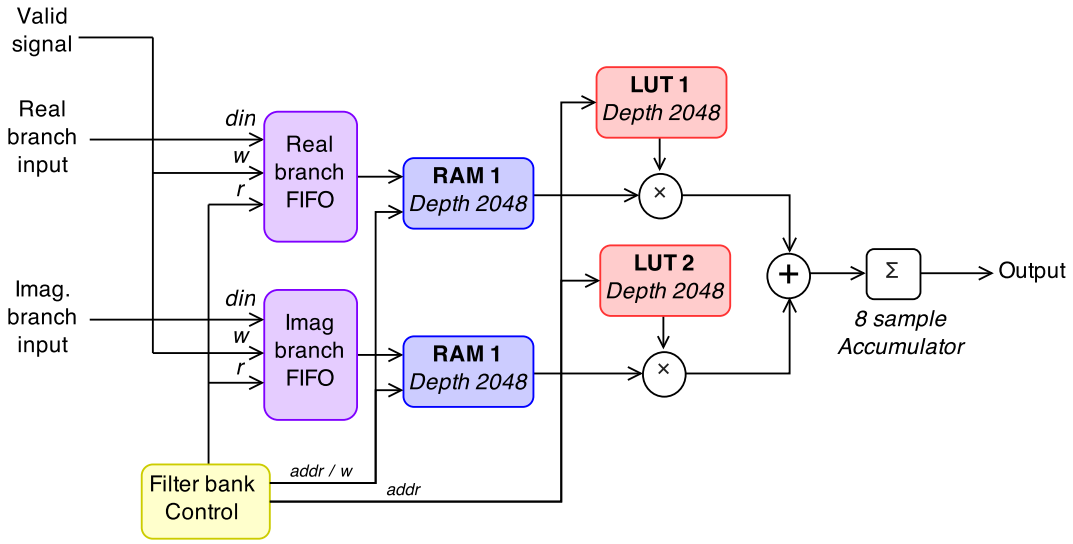


Figure 5.3: Transmitter filter bank block diagram

First of all, two FIFO queues are fed with the burst IFFT time domain samples so that it buffers and matches the rate of the incoming data. Next, the time domain samples are read every eight clock cycles from the FIFO queues and written into Random Access Memories (RAM).

Next, the output values of the RAM are multiplied by the filter coefficients given by the LUTs. Finally, every 8 samples are accumulated.

All the blocks described previously are controlled by the *filter bank control* block which generates the control signals for the RAMs and LUTs. The expression in 5.11 describe the access addresses to the LUTs and RAMs. Where k is the clock index ($0 \leq k, < 8$), m is the filter index ($0 \leq m, < M$) and l the symbol index ($0 \leq l, < N_{symb}$). Moreover, $N = L \times M$, m is incremented by one when k reaches its limit and l is incremented by one when m is equal to $M - 1$.

$$\begin{aligned} a^{LUT}_{k,m,l} &= mod(m + Lk, N) \\ a^{RAM}_{k,m,l} &= mod(a^{LUT}_{k,m,l} - Ml, N) \end{aligned} \tag{5.11}$$

Despite the fact that the prototype filter is four tap long (256×4), the RAMs and LUTs depths are 2048 each (256×8). It is due to the oversampling factor

is eight times higher than the data rate (3.84 MHz). Moreover, the imaginary branch time shift is performed by shifting the filter coefficients instead of using *Shift Registers*. Since, the scope of this project is find feasible hardware algorithms for FBMC/OQAM scheme, memory depth reduction will be considered when optimizing the design.

5.4.3 Time synchronization

For synchronizing in time, it is perform a correlation in time domain. Since the time correlation is computationally expensive on FPGA, it would be worth to reduce the number of coefficients to the minimum.

The frequency domain signal is mapped in the 62 centre sub-carriers, satisfying the Nyquist-Shannon sampling theorem the minimum length of the sequence is 64 samples. Considering that the FBMC/OQAM modulator delays half symbol the imaginary branch, the autocorrelation properties of the ZC sequence are degraded. Consequently, the dime domain sequence has been lengthen to 96 samples in order to keep its autocorrelation properties.

Frequency domain generation and mapping

The signal used for time synchronization in this frame is based on the LTE Primary Synchronization Signal described in [15]. The signal is generated from a well-know sequence called Zadoff-Chu (ZC) which is a constant amplitude and zero autocorrelation sequence. This sequence is generated once, stored in a Read Only Memory (ROM) and placed in the resource grid at the transmitter.

Despite the fact that the FBMC/OQAM modulator degrades the ZC autocorrelation property, it is still acceptable for this prototype.

As said previously, the ZC sequence is generated once and stored. In fact, it is composed by a 62 complex sequence $d_u[n]$ which is generated with the equation 5.12. Where n is the index of the sequence and u is the root.

$$d_u[n] = \begin{cases} e^{-j\frac{\pi u n(n+1)}{63}} & n = 0, 1, ..., 30 \\ e^{-j\frac{\pi u(n+1)(n+2)}{63}} & n = 31, 32, ..., 61 \end{cases} \tag{5.12}$$

The generated sequence is mapped in the 61 central resources as described by the

equation 5.13. Where $a_{k,l}$ is the resource element, k is the sub-carrier, l the number of symbol and N_{SC} is the number of active sub-carriers.

$$\begin{gathered} a_{k,l} = d[n], n = 0, 1, ..., 61 \\ k = n - 31 + \frac{N_{SC}}{2} \end{gathered} \tag{5.13}$$

Besides, considering that no payload is allocated in the synchronization symbol, the same ZC sequence can be mapped at same time in the second Nyquist zone so to increase the correlation performance. The ZC mapping in the second Nyquist zone is described by the equation 5.14.

$$\begin{gathered} a_{k,l} = d[n], n = 0, 1, ..., 61 \\ k = \begin{cases} -n - 32 + \frac{N_{SC}}{2} & n = 0, 1, ..., 30 \\ -n + 93 + \frac{N_{SC}}{2} & n = 31, 32, ..., 61 \end{cases} \end{gathered} \tag{5.14}$$

Time domain sequence generation

The time domain signal is generated using two 64 point IFFT. The frequency domain sequence is mapped as expressed in the equation 5.15. Where k is the inverse FFT algorithm index and $d_u[n]$ the ZC generated sequence.

$$X[k] = \begin{cases} d_u[k + 30] & k = 1, 2, ..., 31 \\ d_u[k - 33] & k = 33, 35, ..., 63 \\ 0 & otherwise \end{cases} \tag{5.15}$$

Before applying the inverse FFT algorithm, the frequency domain sequence phase is shifted with the same criteria applied in the modulator. The equation 5.16 shows the real and imaginary branch split and phase shift applied to $X[k]$.

$$\begin{aligned} X_{Re}[k] &= \mathbb{R}e\{X_k[k]\} \cdot j^k \\ X_{Im}[k] &= \mathbb{I}m\{X_k[k]\} \cdot j^{k+1} \end{aligned} \tag{5.16}$$

Next, the inverse FFT is applied as expressed in the equation 5.17. Where M is the

FFT size and n the time domain index.

$$\begin{aligned} x_{Re}[n] &= \frac{1}{M}\sum_{k=0}^{M-1} X_{Re}[k] \times e^{j2\pi\frac{nk}{M}} \\ x_{Im}[n] &= \frac{1}{M}\sum_{k=0}^{M-1} X_{Im}[k] \times e^{j2\pi\frac{nk}{M}} \end{aligned} \tag{5.17}$$

Once the signal is in the time domain, it is multiplied by a single tap prototype filter. Finally, both conjugated branches are added as it is given by the equation 5.18. The resultant sequence is 96 samples length and its autocorrelation is almost zero.

$$x[n] = g[n] \cdot x^*_{Re}[n] + g[n - M/2] \cdot x^*_{Im}[n] \tag{5.18}$$

The figure 5.4 shows the comparison between the autocorrelation of ZC modulated using FBMC (96 length) and modulated using OFDM (64 samples).

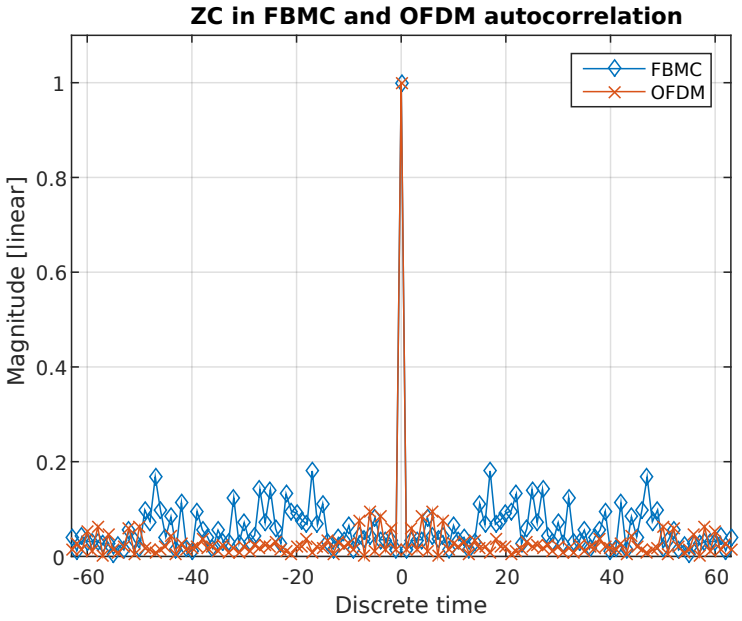


Figure 5.4: ZC Autocorrelation, comparison OFDM versus FBMC

Time domain correlation

In order to synchronize in time, the received signal ($y(t)$) is correlated by the time domain complex sequence.

Due to the time domain correlation sequence is generated at a lower sampling rate, the received signal requires being decimated. Therefore, the received data will match with the synchronization sequence sampling rate (960 kHz). Consequently, the correlator will have an uncertainty of $\frac{M}{2\times 64}$. In other words, the synchronizer has a very low granularity.

The proposed correlation algorithm in this project is based in the reference [16] which provides an area efficient synchronizer for LTE. The designed correlator comprises four Correlation Units (CUs) which are fed at the sampling rate of $960kHz$. Each CU is delayed one sample at the rate of $3.84MHz$, hence it reduces the synchronism uncertainty to less of a FFT sample in the window alignment. In addition, all the CUs are operating at same clock rate and share the same coefficients memory. The simplified block diagram of the correlator is depicted by the figure 5.5.

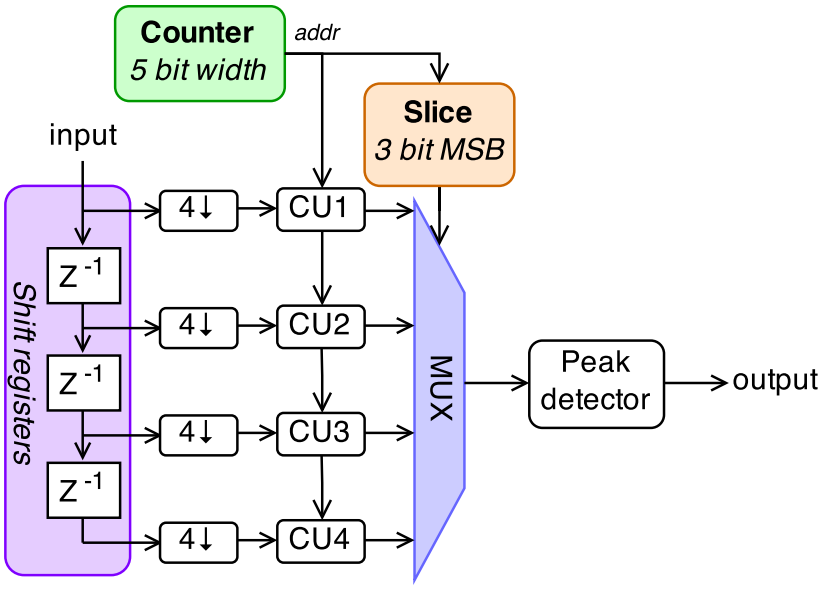


Figure 5.5: Simplified time synchronism correlator block diagram

The output correlation of each CU is multiplexed for aligning them in time. The multiplexor selection port is fed from the same counter which provides the LUT addresses to the CUs.

Because of the time sequence of each CU is 96 samples and the oversampling rate is 32, the correlation must be performed using three complex multiplications (12

regular multiplications). Besides, 3 complex addressable shift registers select the input samples.

The figure 5.6 shows the CUs simplified block diagram. Where the single input counter signal provides to the multiplexers and the LUTs the same addresses. The product is accumulated for 32 cycles and finally the squared module is computed.

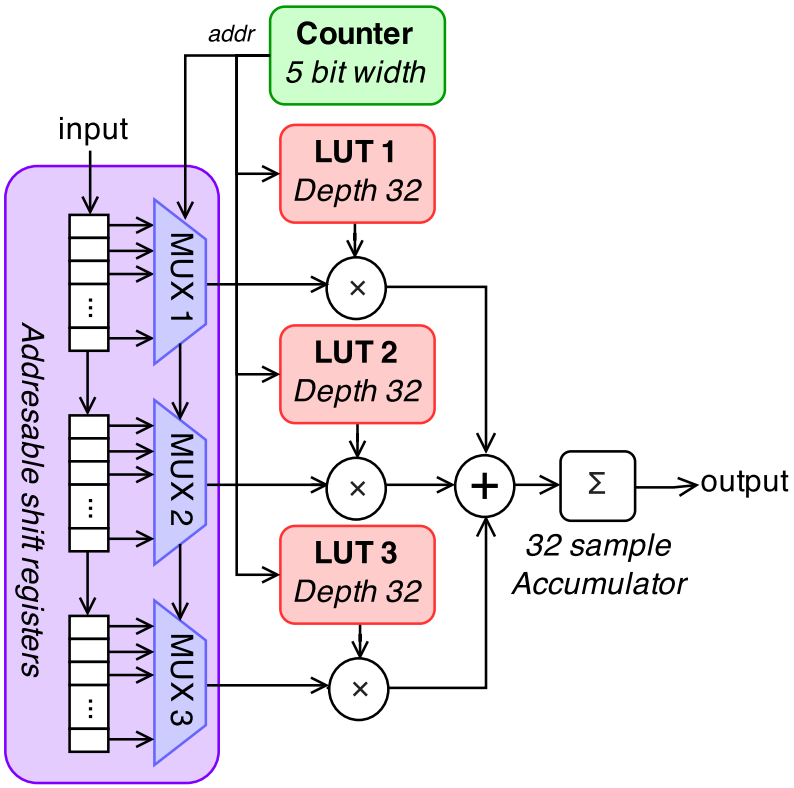


Figure 5.6: Simplified correlation unit block diagram

Peak detection

Once the correlation squared magnitude is computed, the peak must be detected. As a result of the received power is not constant, a moving average filter with a constant gain is introduced with the aim of generating a variable threshold. The signal will be considered aligned when the output of the moving average filter is below the squared magnitude of the correlation.

5.4.4 Receiver Filter bank

The algorithm behind the receiver filter bank is the same than the transmitter one. However, there are a few differences in the coefficient addressing and the data buffering. As expressed in the equation 5.19, the coefficients of the filters are flipped.

$$\begin{aligned} y^{Re}_{m,n} &= \sum_{l}^{L-1} y_{m,n-l} \cdot g^{Re}_{m,L-l-1} \\ y^{Im}_{m,n} &= \sum_{l}^{L-1} y_{m,n-l} \cdot g^{Im}_{m,L-l-1} \end{aligned} \tag{5.19}$$

Furthermore, the synchronization algorithm synchronizes with the central coefficients of the prototype filter. Therefore, several symbols have to be buffered so that the coefficients before the synchronization peak are integrated too.

A first approximation would be the use of a 2-3 symbol length buffer at the synchronizer. Nevertheless, it would increase the memory usage and latency. In contrast, a second solution is reuse the receiver filter bank RAM for buffering.

So, the receiver filter bank fills the RAM continuously with the incoming data. In fact, the RAM is used as a circular buffer where the addresses are periodically repeated as expressed in 5.20. Where the difference with the transmitter filter bank are that RAMs addresses increase instead of decreasing with l. Also, the term n_o has been added. It is the RAM address offset.

In addition, when the synchronizer detects the beginning of the frame sends a strobe signal to the filter bank. Consequently, the receiver filter bank resets its filter states (it does not erase the RAM) and starts the process of filtering from the beginning. At the beginning, the RAM address offset (n_o) is zero. When the correlation peak is detected, the current base position ($M \times l + m$) is stored in n_o and the m, k and l values are set to zero.

Therefore, every time that synchronism is detected, the offset is updated and the receiver keep being synchronized.

$$\begin{aligned} a^{LUT}_{k,m,l} &= mod(m + L \times k, N) \\ a^{RAM}_{k,m,l} &= mod(a^{LUT}_{k,m,l} + M \times l + n_o, N) \end{aligned} \tag{5.20}$$

5.4.5 Channel estimator and equalizer

The used estimation and equalization algorithm in this project is a Zero Forcing Equalizer (ZFE). The algorithm estimates the channel by extracting the reference signals and interpolating the resource elements corrections using a 1-D linear filter.

The estimated channel $h_{k,l}$, for the sub-carrier k and the l symbol, is calculated from the weighted average of the two closest reference signals. The equation 5.21 expresses how are computed the reference signal positions u_1 and u_2 in function of k and the interpolating coefficients v_1 and v_2.

$$\begin{aligned} h_{k,l} &= v_1 \left(b^{Re}_{u_1,l} + b^{Im}_{u_1,l}\right) + v_2 \left(b^{Re}_{u_2,l} + b^{Im}_{u_2,l}\right) \\ u_1 &= N^{RS}_{SC} \left\lfloor \frac{k-\nu^{RS}_{SC}}{N^{RS}_{SC}} \right\rfloor + \nu^{RS}_{SC} \\ u_2 &= u_1 + N^{RS}_{SC} \\ v_1 &= 1 - \frac{k-u_1}{N^{RS}_{SC}} \\ v_2 &= 1 - v_1 \end{aligned} \tag{5.21}$$

The channel is estimated for every symbol containing reference signals and it is kept constant in time up to the next symbol containing reference signals. Also, the incoming symbols are coming in bursts. As consequence, a sequential channel estimator and equalizer depicted in the figure 5.21 have been design.

First of all, the real and imaginary branches are combined and hold in a register when a reference signal has been detected. Next, it is perform it's complex reciprocal ($\frac{1}{x}$). Since division is the most expensive of the four basic operations, it must be avoided as much as possible. In fact, the complex reciprocal has been implemented using a single division and four multiplications.

After the computing the complex reciprocal, it is performed a linear interpolation between the reference signals. Once the channel correction has been estimated, it must be keep the same for several symbols up to the next symbol containing reference signals. So, a RAM is used to keep the channel in memory.

Finally, the incoming signals are multiplied by the estimated inverse of the channel. Moreover, since the imaginary of the real branch and the real of the imaginary are discarded, the number of required regular multiplications is four.

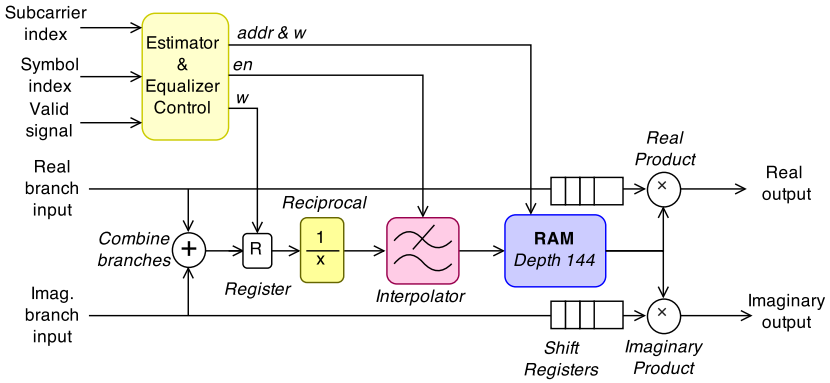


Figure 5.7: Channel estimator block diagram

5.5 Code troubleshooting

Once all the code has been written and its behaviour is verified, all the blocks are integrated and the overall programming file is generated. When this file is being generated, XPS estimates the signal paths delays. The details about timing constrains and timing closure are available in [17].

If in the mentioned estimation any signal path delay is too long, an error is reported. In this case, the design may not behave as intended. Consequently, those errors must be solved before loading the code into the FPGA. The procedure described in the figure 5.8 has been followed with the aim of removing the timing issues and correct glitches in hardware. In addition, this is an iterative procedure that starts with reading the timing report and applying changes in the atomic modules. Those changes must be verified before integrating the module. Also it must be verified once again after the integration. This procedure guarantees that another error is not created when the previous one is solved.

As a result of forcing the developer to change always the code in the atomic benches, it is not possible have the same block in two different places with incompatible code.

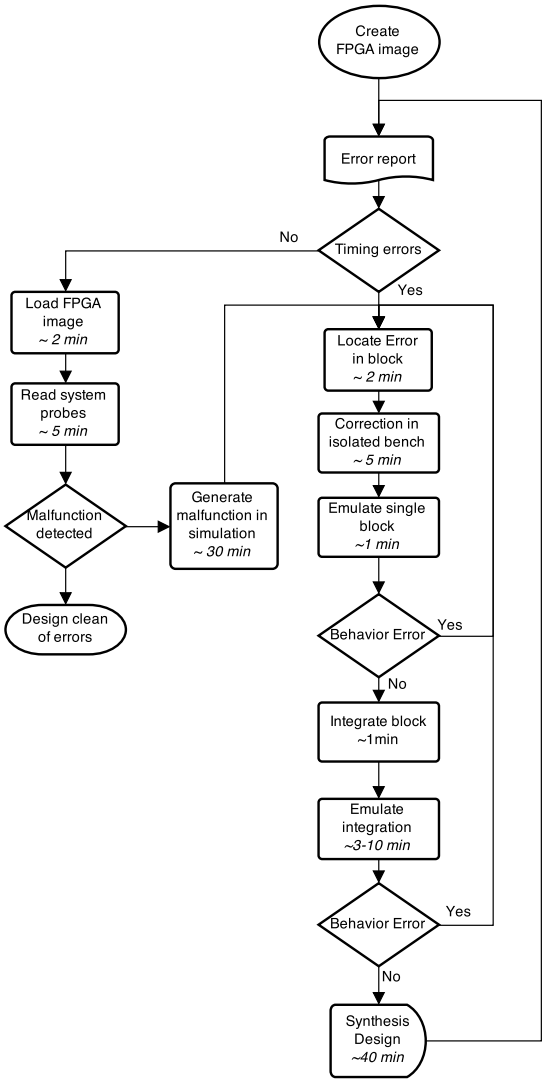


Figure 5.8: Timing error troubleshooting procedure

Chapter 6

Implementation results and discussion

This chapter presents the results of the implementation. Firstly, the simulated and analogue frequency domain signals are shown and discussed. Secondly, the synchronizer correlation is depicted as a prove that the correlation performs as intended. Next, the received constellation in simulation and internal FPGA loop-back are given. Finally, the overall design latency and FPGA resources are provided and discussed.

6.1 Transmitted spectrum

The figure 6.1 shows the comparison between the transmitted frequency domain of the software simulation (64 bit floating point precision) and the Simulink hardware simulation (16 bit fixed point precision). It can be perceived how the noise floor has been increased due to the use of fixed point arithmetic.

The spectrum of the transmitted signal is shown in the figure 6.2. Where the centre frequency is *2.45 GHz*, the span is *5 MHz* and the vertical scale is 10 dB/div. The noise floor is 50 dB below the signal power.

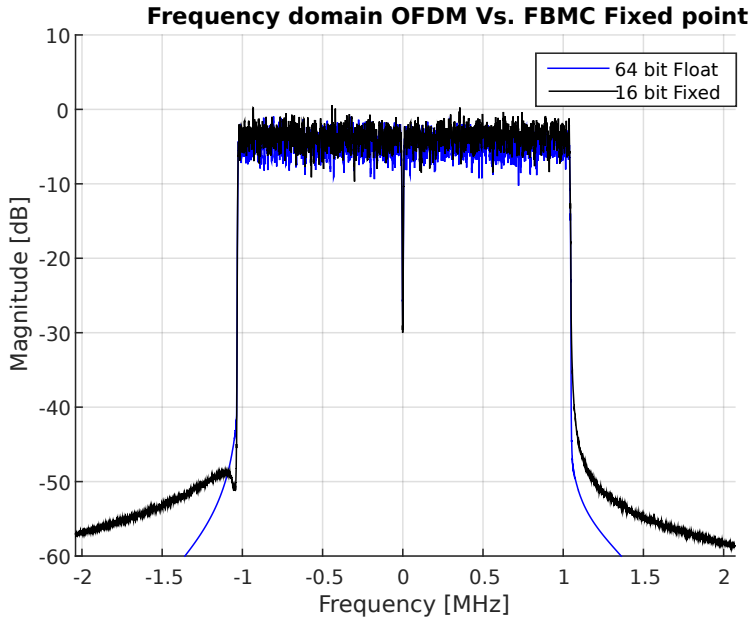


Figure 6.1: Simulated transmitted spectrum, fixed Point comparison

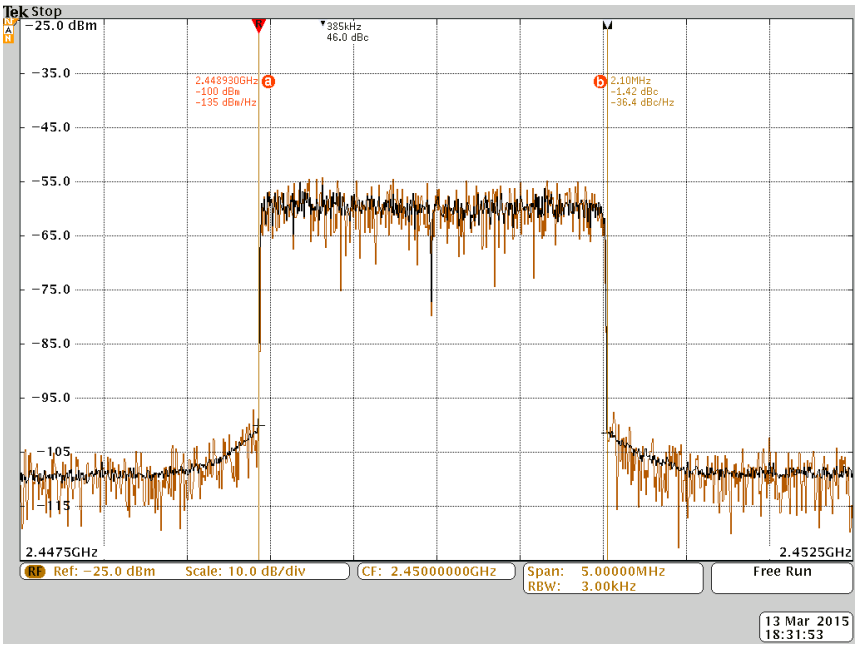


Figure 6.2: Transmitted spectrum

6.2 Synchronizer correlation

The figure 6.3 shows the correlation at the receiver. In this case, the receiver has been fed with the IQ data coming from the ADC. In addition, the transmitter and receiver RF front ends were connected with a coaxial cable.

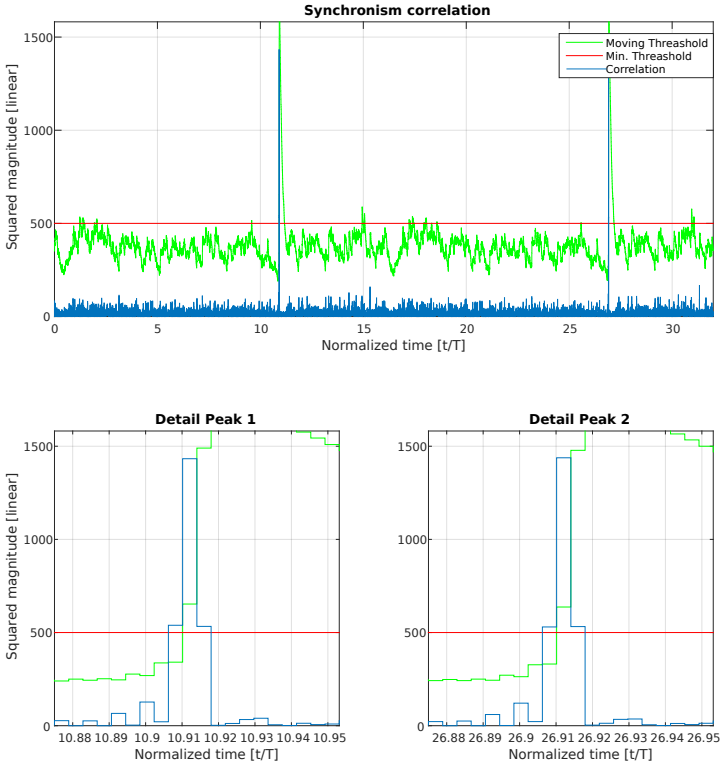
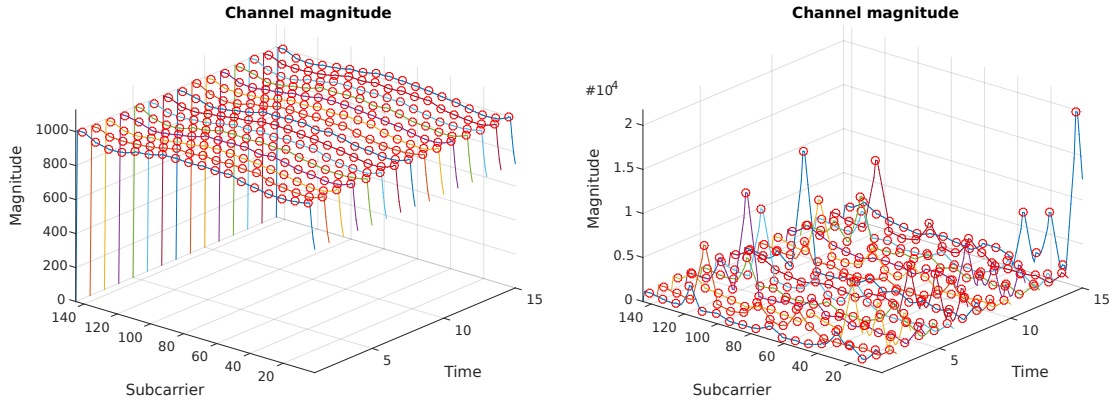


Figure 6.3: Synchronizer correlation

6.3 Channel equalization

The channel correction in time and frequency domain on hardware is depicted by the figure 6.4. The sub-figure 6.4a is the ideal channel magnitude correction in the FPGA internal loop-back. In fact, one can see the DUC and DDC frequency responses in the channel correction. In contrast, the figure 6.4b corresponds to the signal received through the loop-back cable. It can be seen that there are some deep fading (peaks in the corrections) which prevent the receiver from recover properly the payload.



(a) Internal loop-back

(b) Cable loop-back

Figure 6.4: Estimated channel correction (Magnitude)

There are two hypothesis for this problem. The first one is that a spurious signal is coupled in the ADC and it is not properly removed by the DDC. The second one is that there is a phase unbalance in the FMC111 mixers.

6.4 Received constellation

The figure 6.5 shows the received constellations. It proves that the internal FPGA loop-back works as intended and the bit error rate is zero. One can see how the EVM has been degraded due to the use of fixed point operations. In fact, the average EVM in the simulation is 0.75 % while the internal FPGA loop-back is 2.63 %.

The modules which have a biggest effect on the received quality are the pre-coding and channel equalization. In other words, a change of precision in one of those blocks can substantially degrade the received quality.

6.5 End to End Latency

The latency has been measured in the hardware baseband emulation without take into account the delay caused by the RF front end. The measured end-to-end latency is 32100 clock cycles at *30.72 MHz*. It means that the elapsed time since the first bit is generated until it is received is *1.044 ms*.

The approximated delay details are summarized in the table 6.1. They have been calculated in function of the number of symbols that they are buffering.

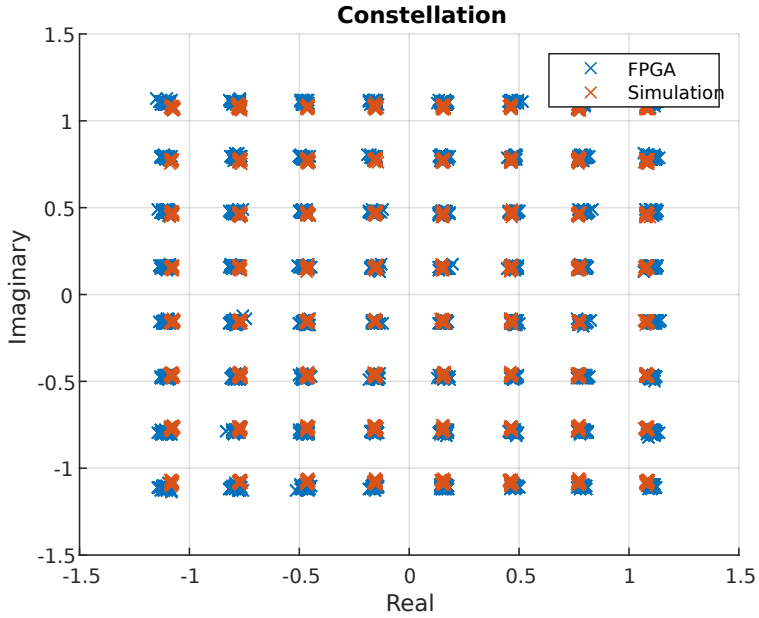


Figure 6.5: Simulated and FPGA internal loop-back 64-QAM constellations

	Delay		
Module	Symbols	Clock Cycles	Time [μs]
Pre-coding	5	10240	333
IFFT	1	2048	67
Tx Filter Bank	4	8192	267
Rx Filter Bank	4	8192	267
FFT	1	2048	67
Others	-	1380	45
TOTAL	+15	32100	1044

Table 6.1: Modules approximated latency

The most expensive module in terms of delay is the reference signals pre-coder. It is because it has to store 5 symbols. The delay caused by this module can be decreased by reducing the number of interfering symbols. Consequently, the intrinsic interference in the received reference signals will increase.

Besides, the filter banks delay is also high. It is due to the number of coefficients. If the number of coefficients were fitted to the prototype filter, the filter banks latencies would be reduced by half.

The latency contained in the row others is distributed along all the design. It is the result of intermediate buffers and pipelined operations along the overall processing chain.

6.6 Design resources

The FPGA programming file has been generated for the target device *Virtex-6 XC6VLX550T FF1759-2*. The resource utilization summary is given in the table 6.2. Where the first row is the total available resources in the FPGA and system is the overall design including the base BPS project plus the FBMC transceiver. The transceiver includes the base band transmitter and receiver. The DUC and DDC are considered in the transmitter and receiver resources.

In addition, between brackets there are the percentages of the relative FPGA resources. Also, in *Specific Features* have been included the *BRAM/FIFO* and the *DSP48* which are specific resources available in Virtex-6 device series [18].

	Slice Logic Utilization		**Specific Features**	
Module	**Slice Registers**	**LUTs**	**BRAM/FIFO**	**DSP48**
Total	68736	343680	1896	864
System	20493 (29.81%)	15730 (4.58%)	103 (5.43%)	131 (15.16%)
Transceiver	15126 (22.01%)	11407 (3.32%)	3260 (3.29%)	128 (14.81%)
Transmitter	5024 (7.31%)	3820 (1.11%)	27 (1.42%)	16 (1.85%)
Tx Filter Bank	344 (0.50%)	695 (0.20%)	10 (0.53%)	4 (0.46%)
Pre-coding	2394 (3.48%)	636 (0.64%)	4 (0.21%)	4 (0.46%)
Receiver	6763 (9.84%)	1353 (1.36%)	23 (1.21%)	74 (8.56%)
Equalizer	1877 (2.73%)	183 (0.18%)	2 (0.11%)	12 (1.39%)
Rx Filter Bank	349 (0.51%)	297 (0.09%)	4 (0.21%)	4 (0.46%)
Synchronizer	2156 (3.14%)	2313 (0.67%)	6 (0.32%)	50 (5.79%)

Table 6.2: Design physical resources utilization of FBMC/OQAM

At first sight, the *slice registers* is the most limiting resource. In fact, if every added transceiver requires the same area, only three transceivers can be placed in one single device. However, if some optimizations were done, four transceivers could fit in.

The reference signals pre-coder is the most expensive in terms of slice registers. It is because it contains a number of shift registers.

Despite the fact that the transmitter and receiver filter bank implementations are based in a similar algorithm, the transmitter one is slightly more expensive in terms of memory. It is because it has four FIFO queues to match the rate of the IFFT (bursts at clock rate of 30.72 MHz) with the base band rate (3.84 MHz). Moreover, if the zero multiplications were avoided the memory utilization of the filter banks would be considerably reduced.

Another expected fact in the table is that the most expensive block in terms of multiplications is the synchronizer. It utilises 50 DSP48 (specific regular multiplications cores in Virtex-6 FPGA models).

Chapter 7

Future Work

After proofing that the FBMC/OQAM prototype works as intended, the future work is proposed. First of all, the FMC111 mixers must be calibrated properly and make the system work using antennas instead of internal or cable loop-back.

Secondly, binary data encapsulation and de-encapsulation must be provided to the system. It will enable transmit the received packets in the Ethernet interface. At the end of the day, a video stream transmission will be used for proof that the radio-link work.

Next, a fancy Graphical User Interface (GUI) may be developed in order to show in near-real time FPGA internal waveforms such as estimated channel corrections, correlation, EVM and so on.

Moreover, the same hardware code should be able to bypass filters and add the Cyclic Prefix to the IFFT. Therefore, the waveform could be changed from OFDM to FBMC/OQAM and viceversa in live. It would proof in execution time the gain of FBMC in front of OFDM.

For proofing that using FBMC/OQAM the carrier spacing can be reduced, carrier aggregation can be performed using several transmitters at different carrier frequencies. It would force them to interfere with each other.

At same time, the transceiver resources may be optimized in order to provide available resources for more complex designs. In fact, the fixed point should be analysed carefully and study the impact of each data width on the overall EVM. Moreover, FIFO, RAM and LUT sizes and their widths may be checked and removed redundant memories and registers.

In a long term, the overall transmitter and receiver can be moved to Multiple Input Multiple Output (MIMO).

Another future work would be change the custom frame structure to the LTE physical layer.

Since FBMC/OQAM interest resides in the reduction of interference, the increase the bandwidth and the rate with it is not considered as a priority. Indeed, the increase of bandwidth would considerably increase the simulation and verification times of the hardware.

Another point that should take into account would be the use of channel coding. It would increase the performance of the overall system.

Conclusion

In this project, a FBMC/OQAM scheme prototype has been developed. It constitutes an experience related with FBMC/OQAM prototyping for the miniBEE hardware platform and a proof that FBMC/OQAM receiver can be implemented on hardware. Moreover, a design flow with Simulink and BEEcube 's BPS software has been proposed.

The proposed design flow facilitated the overall verification of the design. In fact, the division of the design in indivisible (atomic) modules decreased the prototyping time. Moreover, it enabled us to evaluate the quantification error caused by the use of fixed point arithmetic.

The implemented algorithms take advantage of the hardware clock oversampling to re-use area-expensive operations such as multipliers. As a result, the FBMC/OFDM scheme modules implementation is feasible.

The pre-coding technique in the transmitter removes the intrinsic interference at the received reference signals. As a result, it is possible to use a simple channel estimator and a Zero Forcing Equalizer. However, this pre-coding technique increases the latency of the overall system.

The Zadoff-Chu sequence has been used in FBMC/OQAM as it is used in LTE downlink. Nevertheless, the time domain sequence was lengthen half symbol more so that it keeps good autocorrelation properties. Also, the mapping of the Zadoff-Chu sequence in the first and second Nyquist zones increases the correlation performance.

To sum up, a complete FBMC/OQAM scheme is working on a hardware platform and the transmitted binary data is successfully recovered at the receiver.

Bibliography

[1] Gartner, "Gartner says worldwide pc, tablet and mobile phone combined shipments to reach 2.4 billion units in 2013," Feb. 2015. [Online]. Available: www.gartner.com/newsroom/id/2645115

[2] P. Siohan and C. Roche, "Specification and design of a prototype filter for filter bank based multicarrier transmission," *Signal Processing, IEEE Transactions on (Volume:48 , Issue: 11)*, Aug. 2002.

[3] M. G. Bellanger, "Specification and design of a prototype filter for filter bank based multicarrier transmission," *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on (Volume:4)*, May 2001.

[4] Y. Medjahdi, D. L. Ruyet, D. Roviras, H. Shaiek, and R. Zakaria, "On the impact of the prototype filter on fbmc sensitivity to time asynchronism," *Wireless Communication Systems (ISWCS), 2012 International Symposium on*, Aug. 2012.

[5] Y. Zeng, Y.-C. Liang, M. W. Chia, and E. C. Y. Peh, "Unified structure and parallel algorithms for fbmc transmitter and receiver," *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, 2013.

[6] E. Kofidis, D. Katselis, A. Rontogiannis, and S. Theodoridis, "Preamble-based channel estimation in ofdm/oqam systems: A review," *Signal Processing,An International Journal*, 1 2012, a publication of the European Association for Signal Processing (EURASIP).

[7] A. Dziri, C. Alexandre, R. Zakaria, and D. L. Ruyet, "Sdr based prototype for filter bank based multi-carrier transmission," *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, Aug. 2014.

[8] J. Du, P. Xiao, J.-J. Wu, and Q. Chen, "Design of isotropic orthogonal transform algorithm-based multicarrier systems with blind channel estimation," *Communications, IET (Volume:6 , Issue: 16)*, Nov. 2012.

[9] BEEcube, *Software Defined Radio Using BPS and the FMC111.*

[10] Xilinx, *Command Line Tools User Guide.*

[11] ——, *XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices.*

[12] J. Nadal, C. A. Nour, A. Baghdadi, and H. Lin, "Hardware prototyping of fbmc/oqam baseband for 5g mobile communication," *IEEE International Symposium on Rapid System Prototyping (RSP)*, Oct. 2014.

[13] M. Wannemacher, M. Munteanu, S. Perret, and R. Singer, "Taking the best out of two worlds: prototyping and hardware emulation," *High-Level Design Validation and Test Workshop, 2002. Seventh IEEE International*, Oct. 2002.

[14] Xilinx, *System Generator for DSP.*

[15] ETSI, "Lte evolved universal terrestrial radio access (e-utra) physical channels and modulations (3gpp ts 36.211 version 9.0.0 release 9)," 3GPP, Tech. Rep.

[16] H. Cao, C. Ma, and P. Lin, "An area-efficient implementation of primary synchronization signal detection in lte," *Communication Technology (ICCT), 2010 12th IEEE International Conference on*, Nov. 2010.

[17] Xilinx, *Timing Closure User Guide.*

[18] ——, *Virtex-6 Family Overview.*

Appendix A

Tutorials

This appendix provides a few manuals with the steps to follow in order to generate the FPGA program file, download the program file into the FPGA and monitor the hardware probes.

The tutorials have been done using the following software:

- Microsoft Windows 7 64 bit
- Matlab version 7.12.0 (R2011a)
- A licensed Xilinx ISE Design Suite version 13.4
- BPS from BEEcube version 4.2 with its license under the directory $C:/.Xilinx$
- Cygwin Terminal with SSH.

A.1 Patching BPS

In order to provide the XPS the XFLOW desired options. The following steps must be followed:

1. Open with Windows explorer the folder *bps_path* which is under the project directory.
2. Open with WinRar or similar the compress file *XPS_13.4_mBEE4_base.zip* which is located under the folder *boards* in the BPS installation directory.

3. Copy the files *bps.Makefile* and *run_xps.bat* in the in the compress file *XPS_13.4_mBEE4_base.zip* root.

4. Copy the file *5g_poct_custom.opt* in the compress file *XPS_13.4_mBEE4_base.zip* subdirectory *./etc/* .

A.2 Generate FPGA program file

This section describes the steps of generating the FPGA program file from the project folder.

1. Start *SystemGenerator*13.4 in the directory where BPS is installed (this directory must contain a file called *startup.m*).

2. Change the Matlab *current folder* to the project's directory.

3. Run the script *FPGA_FBMC_Init_hw.m*. This script will generate all the parameters and memories for the FBMC/OQAM Simulink model.

4. Open the transceiver hardware *BEEcube_fbmcTxRx.mdl* which is under the directory *hw_synthesis* (do not change Matlab 's current folder).

5. Press $Ctrl + D$ in order to update the design. It will propagate sampling times and data-types.

6. Run the command *bps*, select *Complete bild* in the *ISE Design Flow Choice*, check *Fork processes* and click *Run BPS*

A.3 Download code into the FPGA via SSH

This section describes the steps of downloading the FPGA programming file using SSH.

1. Connect your personal computer through Ethernet cable to BEEcube hardware. Be careful, do not use the FPGA ethernet port.

2. Open Cygwin Terminal.

3. Ensure that you are in the same IP network (10.0.0.0/16).

```
ping 10.0.0.1
```

4. Change to the directory where the programming file is located.

   cd /cygdrive/c/<project dir>\
   /hw_synthesis/BEEcube_fbmcTxRx/output/<version>/

5. Copy them in the miniBEE using SCP.

   scp BEEcube_fbmcTxRx.bin BEEcube@10.0.0.1:/home/BEEcube

6. Access to miniBEE using SSH.

   ssh BEEcube@10.0.0.1

7. Change file permissions (only first time).

   sudo chmod +rw BEEcube_fbmcTxRx.bin

8. Download FPGA programming file into the FPGA.

   SelectMap BEEcube_fbmcTxRx.bin

9. Check that NectarOS is running using the serial port (Press Ctrl+A and after Q for exit).

   minicom fpga

10. Run BEEcube host software.

    b4d