Universidade de Aveiro
2013

Departamento de
Electrónica, Telecomunicações e Informática

Telmo Filipe Pedrosa Cavaco

Compensação da Não-Linearidade do Modulador-MZ-IQ baseada em FPGA

Compensation of MZ-IQ-Modulator Nonlinearity Based on FPGA Algorithms

Universidade de Aveiro Departamento de
Electrónica, Telecomunicações e Informática
2013

Telmo Filipe Pedrosa Cavaco

Compensação da Não-Linearidade do Modulador-MZ-IQ baseada em FPGA

Compensation of MZ-IQ-Modulator Nonlinearity Based on FPGA Algorithms

Universidade de Aveiro Departamento de
2013 Electrónica, Telecomunicações e Informática

Telmo Filipe Pedrosa Cavaco

Compensação da Não-Linearidade do Modulador-MZ-IQ baseada em FPGA

Compensation of MZ-IQ-Modulator Nonlinearity Based on FPGA Algorithms

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob orientação científica do Dr. Paulo Miguel Nepomuceno Pereira Monteiro, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Dr. Rui Fernando Gomes de Sousa Ribeiro, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

O Júri

Presidente	**Prof. Dr. José Rodrigues Ferreira da Rocha** Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Vogais	**Prof. Dra. Maria do Carmo Raposo de Medeiros** Professora Associada do Departamento de Engenharia Electrotécnica e de Computadores da Universidade de Coimbra
	Prof. Dr. Rui Fernando Gomes de Sousa Ribeiro Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Agradecimentos

Gostaria de agradecer à minha família o apoio incansável prestado ao longo da minha formação.

Agradeço também ao Professor Paulo Monteiro e ao Professor Rui Ribeiro por me terem dado a oportunidade de aprofundar conhecimentos na área das comunicações ópticas. Agradeço a orientação e apoio durante a realização da dissertação. Agradeço também ao Professor Arnaldo Oliveira pelos esclarecimentos na área de sistemas reconfiguráveis e ao aluno de doutoramento Ali Shahpari, pela disponibilidade e ajuda nas execuções laboratoriais.

Deixo ainda uma palavra de apreço a todos os meus amigos pelo apoio demonstrado ao longo destes cinco anos.

palavras-chave

Comunicações Ópticas, Sistemas de Transmissão Coerente, Modulador Mach-Zehnder, Pré-distorção Electrónica, FPGA

resumo

Nos últimos anos, a crescente necessidade de largura de banda e a evolução das técnicas de processamento digital de sinal renovaram o interesse pelos sistemas de comunicação ópticos coerentes. O modulador IQ assume-se como um componente chave nestes transmissores, sendo responsável pela conversão de informação do domínio eléctrico para o domínio óptico. Os moduladores Mach-Zehnder que constituem este dispositivo recebem sinais de *drive* com uma excursão controlada, garantindo a utilização de uma região aproximadamente linear das suas funções transferência e a geração de constelações sem distorções de fase. No entanto, existem vantagens em explorar a extensão completa da característica dos moduladores. Neste contexto, torna-se relevante efectuar um estudo acerca das técnicas de pré-distorção electrónica que permitem corrigir os efeitos das não-linearidades associadas a este método de transmissão.

Esta dissertação foca-se no estudo da compensação dos impactos que a característica não-linear do modulador Mach-Zehnder tem nos sistemas de transmissão ópticos coerentes. Após a identificação e desenvolvimento de soluções matemáticas para o problema, realizaram-se vários testes utilizando um simulador integrado em ambiente MATLAB. Um sistema de transmissão coerente utilizando formatos de modulação QAM e os respectivos algoritmos de compensação foram posteriormente implementados em FPGA. Desenvolveram-se também co-simulações que permitiram garantir que o *hardware* concebido produzia os resultados desejados. Para além disso, realizaram-se vários testes utilizando um modulador IQ disponível no “Laboratório de Óptica” do Instituto de Telecomunicações de Aveiro. O objectivo consistiu em operar o sistema em condições laboratoriais e analisar o desempenho dos algoritmos de compensação em ambiente real.

keywords

Optical communications, Coherent Transmission Systems, Mach-Zehnder Modulator, Electronic Predistortion, FPGA

abstract

In recent years, the ever-increasing bandwidth demand and the evolution of digital signal processing techniques renewed the interest for the optical coherent systems. The IQ-Modulator is a key component in optical coherent transmitters, being responsible for the conversion of information from electrical to optical domain. The Mach-Zehnder modulators that compose this device receive driving signals with a controlled excursion, in order to use an approximately linear region of their transfer function and produce constellations without phase distortions. However, there are advantages in exploit the full range of the modulators' characteristic. In this context, a study about the electronic predistortion techniques required to overcome the nonlinear effects associated to this transmission method becomes relevant.

The subject of this dissertation is the compensation of impairments related to the nonlinear characteristic of the Mach-Zehnder Modulator in coherent optical transmission systems. After the identification and development of mathematical solutions for the problem, several tests were made using a simulator that runs in a MATLAB environment. A QAM coherent transmitter system and the compensation algorithm were then implemented in a FPGA platform. Co-simulations were performed in order to prove that the designed hardware was generating correct results. Furthermore, some tests were conducted using an IQ-Modulator available in the "Optics Laboratory" at Telecommunications Institute of Aveiro. The goal was to operate the system under laboratorial conditions and analyze the performance of the compensation algorithm in a real case scenario.

Content

List of Acronyms

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPD	Balanced Photodetector
BPF	Band-Pass Filter
BRAM	Block Random Access Memory
BW	Bandwidth
CLB	Configurable Logic Blocks
CMT	Clock Management Tiles
CPLD	Complex Programmable Logic Device
DAC	Digital to Analog Converter
DD	Direct Detection
DDR	Double Data Rate
DLL	Delay-Locked Loop
DQPSK	Differential Quadrature Phase Shift Keying
DSP	Digital Signal Processing
EAM	Electro-Absorption modulator
EOM	Electro-Optic modulator
EVM	Error Vector Magnitude
FDM	Frequency Division Multiplexing
FIFO	First-In, First-Out
FIR	Finite Impulse Response filter
FMC	FPGA Mezzanine Card
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
HPC	High-Pin Count version of FMC connector
HSTL	High-Speed Transceiver Logic
I2C	Inter-Integrated Circuit
ICON	Integrated Controller
IIR	Infinite Impulse Response filter
ILA	Integrated Logic Analyzer
IM	Intensity Modulation
IOB	I/O Block
IPCORE	Intellectual Property Core
ISE	Integrated Software Environment
ISI	Intersymbol interference
JTAG	Joint Test Action Group
LDO	Low-Dropout Regulator
LO	Local Oscillator
LPC	Low-Pin Count version of FMC connector
LPF	Low-pass Filter
LUT	Look-Up Table
LVCMOS	Low-Voltage CMOS

LVDS	Low-Voltage Differential Signaling
LVPECL	Low-Voltage Positive/Pseudo Emitter-coupled Logic
MMCM	Mixed-Mode Clock Manager
MZM	Mach-Zehnder Modulator
OSA	Optical Spectrum Analyzer
OSIP	Optical Simulator Platform
OSNR	Optical Signal-to-Noise Ratio
PAL	Programmable Array Logic
PBS	Polarization Beam Splitter
PCI	Peripheral Component Interconnect
PLA	Programmable Logic Array
PLD	Programmable Logic Device
PM	Phase Modulator
PROM	Programmable Read-Only Memories
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
SMA	Sub Miniature version A connector
SNR	Signal-to-Noise Ratio
SPI	Serial Peripheral Interface Bus
SPM	Self-Phase Modulation
SSTL	Stub Series Terminated Logic
TDM	Time Division Multiplexing
UCF	User Constraints File
UUT	Unit Under Test
VCO	Voltage-Controlled Oscillator
VHDL	Very High Speed Integrated Circuits Hardware Description Language
WDM	Wavelength Division Multiplexing

List of Symbols

C_n	Filter's coefficients
d	Euclidean distance between two adjacent symbols of a constellation
E_{in}	Electrical field at the input of an optical modulator
E_{inc}	Photodiodes incoming electrical field
E_{out}	Electrical field at the output of an optical modulator
$h(n)$	Impulse response of a filter
$H(n)$	Transfer function of a filter
$p(t)$	Transmitted pulse
P_{in}	Power introduced to the fiber
P_{out}	Power at the output of an optical modulator
R	Responsivity of the photodiode
R_b, r	Bit Rate
RF_{low}, RF_{up}	Analog driving signals that will be applied to the electrodes of an optical modulator
T_s	Symbol Period
T_b	Bit Period
$u(t)$	External voltage applied to the electrodes of an optical modulator
$u_{DC}(t)$	External bias voltage applied to the electrodes of an optical modulator
Vdc_{low}, Vdc_{up}	Bias voltage applied to the electrodes of an optical modulator
V_{drive}	Predistorted analog driving signals that will be applied to the electrodes of an optical modulator
V_i	Sum of the voltages applied to an electrode of an optical modulator
V_{PM}	Voltage used in a phase modulator
V_π	Voltage required by a modulator to invert the electrical field of an optical signal
x_I	"in-phase" component of the complex envelope of a transmitted signal
x_Q	"quadrature" component of the complex envelope of a transmitted signal
α	Roll-off factor of the Raised-Cosine filter
β	Bandwidth that is added to the minimum bandwidth of a ideal low-pass filter
φ_1, φ_2	Phase shifts in the arms of an optical modulator
ψ_{ij}	Phase coefficients of the "2x4 90° Hybrid"

List of Figures

List of Tables

1. Introduction

1.1 Historical perspective of optical communication systems

In the 1970s, telecommunication networks were built essentially to carry voice traffic. The few data services that were available over global networks were carried on a voice-optimized infrastructure [1]. However, over the last forty years, the network services have suffered a dramatic change. The number of users has increased exponentially, the available services become more diverse and video is playing an important role in the dominant traffic on network backbones [2]. Traffic growth continues at a rate of almost 50% per year, without showing any signs of slowing [2, 3]. This network capacity increasing was only possible by continued developments in the capacity of fiber optic transmission systems. In the 1970s, the simultaneous availability of compact optical sources (GaAs semiconductor lasers) and low-loss optical fibers (losses could be reduced to 20 dB/km in the wavelength region near 1 µm) led to a worldwide effort for developing these fiber-based systems [4]. Thus, all the progress in these systems can be grouped into several distinct generations.

The first generation became commercially available in 1980 and used GaAs semiconductor lasers for operation near 850 nm, which was a low-loss transmission window of early silica fibers [5]. Those systems operated at 45Mbps, for a single channel, and were based in intensity modulation (IM) at the transmitter and direct detection (DD) at the receiver. They also allowed repeater spacings of up to 10 Km [4,6]. This larger repeater spacing, when compared with the spacing of typical coaxial systems, was an important advantage because it decreased the installation and maintenance costs associated with each repeater [4]. A few years later, the repeater spacing was increased again by moving the operation of the lightwave system to the region near 1300 nm, where the fiber loss is below 1dB/Km and a minimum chromatic dispersion is achieved [4].

The second generation of optical communication systems arose from the need of avoiding the 100Mbps bit rate limitation due to intermodal dispersion in multimode fibers. So, in 1980s, systems began to use single mode fibers and rates of 2 Gbps were achieved [4]. However, the fiber losses were a limitation of these systems, having the typical value of 0.5 dB/km. The solution was to change the lasers wavelength to the region around 1550 nm, where the fiber loss has its minimum [5]. However, this wavelength region has got associated a considerable dispersion. Thus, in order to solve this problem, the third generation of optical communication systems was created. The dispersion problem can be overcome by using dispersion-shifted fibers designed to have minimum dispersion near 1550 nm. Third-generation lightwave systems operating at a bit rate of up to 10 Gbps and with regeneration stages spaced by 70 Km became available commercially in 1990 [4].

The fourth generation is characterized by the invention of the Erbium-Doped Fiber Amplifiers (EDFA) and by the developments in the Wavelength Division Multiplexing (WDM) technology. In a simplified way, it is possible to say that optical amplifiers use a pump laser and a doped fiber to implement the principle of stimulated emission and amplify all the WDM channels contained in a certain spectral region. The WDM technology allowed the transmission of multiple channels simultaneously in a single fiber, which increased the transmission capacity by 10.000 times in twenty years [4]. WDM technology was only possible to develop due to advances in laser manufacturing.

Lasers with reduced phase noise and low linewidth have got high spectral coherence, allowing the implementation of coherent optical communication systems [7]. Until then, only IM/DD schemes that respond to changes in the power level of the optical carrier and ignore its phase or frequency content were considered. However, this method does not take full advantage of the bandwidth capabilities of optical fibers. Thus, coherent optical communication systems evolved, treating the light as a carrier medium which can be amplitude or/and phase modulated [5]. This allowed a symbolic representation of the transmitted data and the evolution of advanced data modulation formats. Therefore, an increase in the number of bits that represent each symbol guaranteed higher data rates (and lower symbol rates), resulting in narrower bandwidths. This also reduced the effects of fiber propagation issues and increased the spectral efficiency.

At the same time, two types of modulators were considered: the Electro-Absorption modulator (EAM) and the Electro-Optic modulator (EOM), where the Mach-Zehnder Modulator (MZM) is included (and is going to be explained in the next chapter). An advantage of the Electro-Absorption modulators is that they are made of the same semiconductor material that is used for the laser, and both can be easily integrated on the same chip. However, when compared with all types of MZMs, an EAM has got some disadvantages like low saturation power, large chirp, narrow optical bandwidth and need of temperature control [8]. In 1998, EOMs with a bandwidth of 10 GHz were available commercially [4, 9] and, in 2000s, the bandwidth increased to 40 GHz [9], making the MZMs almost exclusively used for transport systems working at 40Gbps and above. Their main advantages are the well-controllable modulation performance, the possibility of independently modulating intensity and phase of the optical field, broad optical bandwidth and zero or tunable chirp [8].

The fourth generation had also contributed for the development of new systems and fiber concepts that reduce the nonlinear effects caused by the large optical power arising from the presence of many channels in the fiber. New “non-zero” dispersion fibers and some dispersion-management techniques were conceived for this purpose [6]. The fifth generation is concerned with the extension of the wavelength range in which a WDM system can operate simultaneously. The availability of new fibers and Raman amplifiers allowed the operation in 1300-1650 nm with thousands of WDM channels [4].

1.2 Historical perspective of Digital Signal Processing in optical communication systems

All the optical communication systems have got some impairments that must be compensated in order to prevent the degradation of its operation. The earlier compensation techniques arose with the first optical systems and were performed in the optical domain. Those techniques were transparent to the data rate and modulation format of the carried signal, being suitable to systems working at high data rates [10]. For example, dispersion shifted fibers or dispersion compensation fibers were used to reduce the chromatic dispersion. However, this type of compensation was expensive since it uses optical discrete components, giving space to new developments in compensation techniques performed in the electrical domain.

In recent years, digital signal processing (DSP) techniques have played an important role in optical communication systems evolution, eliminating many problems related to coherent systems. They allow a flexible digital manipulation of data by implementing specific and complex algorithms in high-speed appropriate hardware [11]. Coherent detection with digital signal processing improves receiver sensitivity, spectral efficiency and impairment compensation, increasing the robustness and performance of the system [12]. The transmitter and receiver structures may remain the same regardless of the modulation format. This makes possible to develop flexible transponders that use varying modulation formats depending on the given reach in the system [11, 12].

Digital signal processors are complex integrated circuits that allow complex computations with strict time and accuracy constraints and the implementation of DSP algorithms for coherent optical systems in an efficient way. In 1982, Texas Instruments introduced the first successful "DSP processor", the TMS32010, which incorporated specialized hardware that enable it to compute a multiplication in a single clock cycle [13]. As might be expected, faster multiplication hardware yields faster performance in many DSP algorithms, and for this reason all modern DSP processors include at least one dedicated single-cycle multiplier or a combined multiply-accumulate (MAC) unit [13]. During 1980s, the developments in CMOS technology and the challenging needs in DSP applications influenced the way DSPs architectures evolved. However, today's manufacturers have got the same challenges: increase speed, decrease energy consumption, decrease memory usage, and decrease the cost of DSPs [13].

More recently, the "Field-Programmable Gate Array" has been proposed as the technology for DSP systems, since it offers the capability to develop the most suitable circuit architecture for the computational, memory and power requirements of a certain application [14]. Despite being fully described in Chapter 4, an FPGA can be seen as a complex and high speed integrated circuit that provides a method to map logical circuits into hardware. FPGA's are achieving great performances due to the possibility of parallel execution of multiple operations. Furthermore, FPGA-based designs also offer faster prototyping and the possibility of adapting the hardware during runtime [15].

Before FPGAs started to being developed and becoming viable alternatives in digital signal processing, all the reconfigurable computing technology went through decades of evolution. Before PLDs (Programmable Logic Devices) were invented, PROMs (Programmable Read-Only Memories) were used to store information and create combinational logic circuits. However, PLDs implemented the idea of creating reconfigurable digital circuits, where the combinational function of the device was undefined until its manufacturing time. The PLAs (Programmable Logic Arrays) can be seen as a set of programmable AND gate planes link to a set of OR gate planes, allowing the generation of complex combinational functions. In 1978, PAL (Programmable Array Logic) was introduced by "Monolithic Memories, Inc" [16]. It can be considered an improved PLA, where the OR-plane is fixed and only the AND-plane is programmable [16]. A few years later, some "Complex Programmable Logic Devices" (CPLDs) started to be manufactured. They can be seen as a continuation of the PLA architecture, since the chip includes logic blocks (*macrocells*) at the borders, and a connection matrix located at the central part. Each macrocell has a structure similar to a PLA. After years of evolution in FPGA technology, their popularity is growing, leading the manufacturers to make an effort to increase their set of features. Reconfigurable computing market segment is leaded by Xilinx, with a share of almost 51% [17]. The remaining competitors are Altera, Actel and Lattice Semiconductor [18]. In 2010, Xilinx released the ‘7’ version of FPGA families, which were built on 28-nm process. The “Virtex” family is optimized for high system performance, with this last version having up to two million logic cells, up to 2.8 Tbps total serial bandwidth, 5.335GMACs, 68Mb Block RAM and even lower power consumption [19,20]. Xilinx also introduced for the first time the “Artix-7” and the “Kintex-7” families which provide better coverage of the lower and mid-end applications previously covered by the “Spartan” family [19, 20]. Altera has made great progress in winning market share in recent years. Their 28-nm FPGAs cover the low, mid and upper end markets with the “Cyclone”, “Arria” and “Stratix” series, respectively [15]. In February 2013, Altera has developed the “Stratix-10”, a new generation of FPGAs manufactured on Intel's revolutionary 14-nm 3D Tri-Gate transistor technology, which can provide, for example, more than ten TFLOPs of single-precision floating-point DSP [21].

Nowadays, due to the standardization of the transmitter and receiver configurations of optical coherent systems, the majority of signal impairments are treated at the receiver, and the algorithms are being implemented using ASICs, DSPs or FPGAs. However, recent researches demonstrate that FPGAs can also be used to generate DSP algorithms at the transmitter [22]. For example, predistorted signals can be generated with the objective that, during the propagation in the transmission channel, nonlinear effects reverse the distortion, resulting in the desired signal waveform at the receiver [22, 23].

1.3 Motivation

In 2000, the ever-increasing bandwidth demand, the increased capacity in WDM systems and the evolution of high-speed digital signal processing renewed the interest for the optical coherent systems [7]. As will be described in the second chapter of this dissertation, the digital information that is transmitted by these systems must be pulse-shaped and converted to continuous-time voltage waveforms, before being applied to the electrodes of an optical IQ-Modulator. This device is responsible for the modulation of an optical carrier and it is composed by two Mach-Zehnder Modulators (MZMs) and a phase-modulator [24]. The MZMs must be biased accordingly to the constellation that it is intended to generate. For example, Square Quadrature Amplitude Modulation (QAM) constellations require the biasing of the modulators in the “Null Point” of their characteristic [25]. Since the optical field response of the MZM is nonlinear, the excursion of the analog driving signals must be reduced and carefully adjusted, in order to use an approximately linear region of the characteristic and produce correctly sized constellations, without phase distortions [26]. Several studies about the impacts of the operating conditions in the transmitted constellations are investigated in [25].

It would be useful to study the possibility of using electrical driving signals that exploit the full modulation depth of the modulators, i.e., that have an excursion equal to the entire range of modulator’s characteristic [26]. This transmission method minimizes the translation of electrical noise into the optical domain in some of the QAM symbols [26, 27]. It also brings benefits in terms of OSNR, since that receiver's sensitivity can be improved due to a significant performance gain observed for high modulation orders of M-ary QAM constellations [28]. Furthermore, small excursions of the driving signals will lead to a decrease of the optical power efficiency, since the mean optical output power of the MZM will be reduced significantly when compared to its mean optical input power. This will reduce the available OSNR at the receiver side, since the transmit laser power is limited [28].

Despite the increasing of complexity, the inclusion of digital signal processing techniques in the transmitter can be a solution to overcome the nonlinear effects of the IQ-Modulator and allow the use of those higher amplitudes of the 'I' and 'Q' waveform data streams [28]. In the past few years, several experiments using transmitter-based electronic predistortion (EPD) were demonstrated, having considerable interest in the compensation of nonlinearities such as self-phase modulation (SPM) or chromatic dispersion [22, 23]. In certain types of experiments, electronic predistortion can bring advantages when compared to receiver-based equalization. One of the reasons is the current availability of commercial high-speed digital-to-analog converters (DACs), while analog-to-digital converters (ADCs) required for the receivers are yet to become generally available [22]. Other important factor is the constant availability of a suitable clock for the DSP circuits at the transmitter without requiring clock recovery. This makes the scheme simpler to implement than receiver-based signal processing [22].

Electronic predistortion can be implemented using ASICs or FPGAs. As seen in the previous section, manufacturers are doing an effort to overcome the processing speed limitation of the FPGAs. Recent advances in several features are enabling their use in optical coherent systems.

1.4 Objectives and dissertation structure

The main purpose of this dissertation is the study of efficient ways of compensating the impairments caused by the nonlinear characteristic of the Mach-Zehnder Modulator in coherent optical transmission systems. The first steps consist in creating OSIP simulations using several QAM formats for development and test of different mathematical solutions. Then, a transmission system that simultaneously performs the compensation algorithm will be implemented in FPGA. The goal is to validate its operation and test the solution under real hardware limitations. This work will be focused in discussing the most efficient solutions, in order to guarantee an increase in the efficiency and a reduction in the hardware requirements.

This document is divided in six chapters. The first one presents a brief historical perspective of optical communication systems and the state of the art of the digital signal processing in coherent systems. In chapter two, details related to the transmission part will be studied. The impacts of the nonlinear characteristic of the Mach-Zehnder Modulator will be analyzed and a solution to compensate those impairments will be discussed. After the conception of an algorithm, the third chapter will present several OSIP simulations, where it is intended to analyze and validate the produced results. In chapter four, some solutions to efficiently implement the transmitter system and the compensation algorithm in a FPGA platform will be discussed. In the fifth chapter, the operation of the system will be tested, in order to assess if the designed hardware is generating proper results.

Furthermore, some tests will be performed with the IQ-Modulator available in the "Optics Laboratory" at Telecommunications Institute of Aveiro. The goal is trying to obtain results and get conclusions about the operation of the entire system in real laboratorial conditions. Finally, in the sixth chapter, conclusions and suggestions of possible future work will be presented.

2. Optical coherent systems

An optical coherent communication system is one of the most efficient ways to transmit information. It allows a symbolic representation of the transmitted data, enabling the evolution of advanced modulation formats. The increase of the number of bits represented by each symbol guarantees higher rates in signals transmission, since the symbol rates are lower than the corresponding binary rates. This leads to an increased spectral efficiency, with narrower spectral bandwidths.

An optical coherent system consists of a transmitter, a transmission channel and a receiver. The first one is responsible for generating the data and mapping it into a specific constellation. The data will be converted into the optical domain and sent through the transmission channel, where some linear and nonlinear effects will interfere with the signal. The coherent receiver is responsible for the signal´s conversion into the electrical domain and for the information decoding [5]. Generally, digital signal processing is necessary to have a better signal reconstruction, since it was distorted during the transmission.

In this chapter, all aspects related to the transmitter part of an optical coherent system will be discussed. The transmitter has got a fundamental structure composed by an optical and a digital part. In the next two sections both components are going to be studied in order to understand the main concepts related to an M-QAM transmitter system.

2.1 The Transmitter: Optical part

One of the key components of an optical transmitter is the IQ-Modulator. In this section, two fundamental components of the optical modulator will be introduced, providing a better understanding on the IQ-Modulator: the phase modulator (PM) and the Mach-Zehnder modulator (MZM).

In figure 2.1 an optical phase modulator is illustrated. The electrical field of the incoming optical carrier can be modulated in phase simply by applying an external voltage '$u(t)$', which is going to change the effective refractive index of the optical waveguide [24].

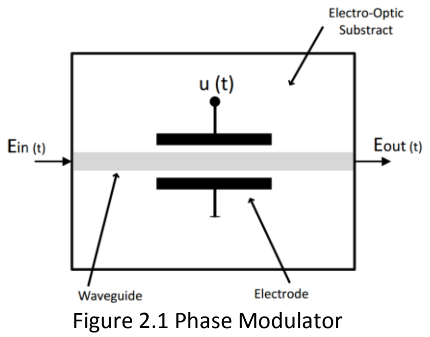


Figure 2.1 Phase Modulator

The following expression traduces the transfer function of a phase modulator, where 'V_π' is the voltage applied to the electrodes needed to generate a phase rotation of $\pi\ rad$:

$$E_{out}(t) = E_{in}(t).\, e^{j\left(\frac{u(t)}{V_\pi}\pi\right)} \tag{2.1}$$

In figure 2.2 is illustrated a Mach-Zehnder modulator. It is an external modulator typically used for high capacity systems, essentially due to its superior signal quality when compared with the less performing direct modulation or Electro-Absorption modulators. Furthermore, it provides a smaller frequency chirp, narrower spectrum and higher resilience to chromatic dispersion [29].

In a structural view, it consists of two phase modulators, one on each arm. Thus, the incoming optical carrier is split into two paths. The most commonly used material to fabricate this modulator is Lithium Niobate, which is an electro-optical crystal, whose refractive index also changes in response to an applied electric field. Therefore, the voltages from the modulating signals (‘$u_1(t)$’ and ‘$u_2(t)$’) will affect the propagation speed of the light waves. That speed will be higher or lower, depending if the refractive index decreases or increases, respectively. Thus, the two optical fields will acquire a certain phase modulation and will be recombined. Using the principle of interference, the optical fields can interfere in a constructive or destructive way, causing an intensity modulation [24].

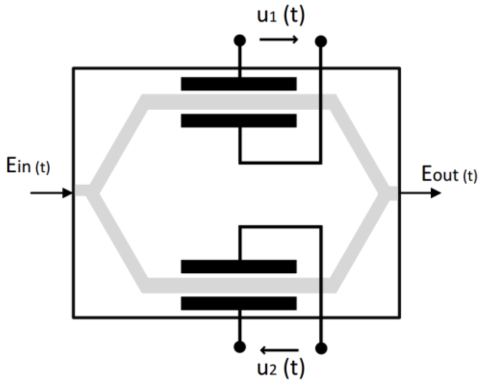


Figure 2.2 Mach-Zehnder Modulator

The transfer function of this modulator is given by the sum of two phase modulators transfer functions:

$$\frac{E_{out}(t)}{E_{in}(t)} = \frac{1}{2}.\left(e^{j\varphi_1(t)} + e^{j\varphi_2(t)}\right) \tag{2.2}$$

$\varphi_1(t)$ and $\varphi_2(t)$ are the phase shifts in the upper and lower arms of the modulator. Once again, $'V_\pi'$ will be the driving voltage needed to obtain a phase shift of π radians on a phase modulator. So, the expressions that relate phase shifts to the driving signals are:

$$\varphi_1(t) = \frac{u_1(t)}{V_{\pi 1}}\pi, \quad \varphi_2(t) = \frac{u_2(t)}{V_{\pi 2}}\pi \tag{2.3}$$

If identical phase shifts are applied in both arms (“push-push” mode), a pure phase modulation is achieved. However, if one arm gets the negative phase shift of the other arm (“push-pull” mode), an amplitude modulation is obtained:

$$E_{out}(t) = \frac{1}{2}.E_{in}(t).\left(e^{j\varphi_1(t)} + e^{j\varphi_2(t)}\right) = \frac{1}{2}.E_{in}(t).\left(e^{j\left(\frac{u(t)}{2V_\pi}\pi\right)} + e^{-j\left(\frac{u(t)}{2V_\pi}\pi\right)}\right) = E_{in}(t).\cos\left(\frac{u(t)}{2V_\pi}\pi\right) \tag{2.4}$$

By squaring the expression of the output electrical field of the modulator in "push-pull" mode, one may obtain the power transfer function, which is represented in figure 2.3, jointly with the electrical field transfer function:

$$\frac{P_{out}(t)}{P_{in}(t)} = \frac{1}{2} + \frac{1}{2} . \cos\left(\frac{u(t)}{V_\pi}\pi\right) \tag{2.5}$$

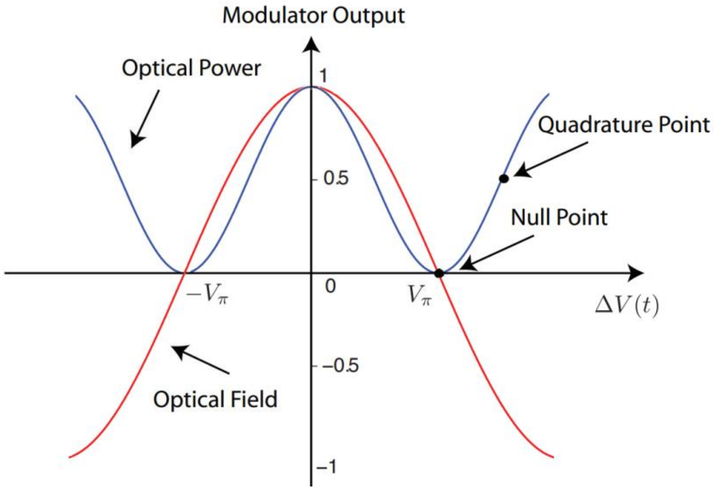


Figure 2.3 Representation of optical power and optical field characteristics as function of the input drive voltage [24]

By observing the two characteristics it is possible to conclude that the relationship between the modulating signal and the modulator's output is a nonlinear function.

The typical mode of operation consists in adding a bias voltage to the modulating signal in order to move the region of operation into a range centered in a certain point of the characteristic [24]. Usually, two points of operation are considered:

- Quadrature Point: extensively used in direct-detection systems, where the electrical driving voltage is converted into optical intensity.
- Null Point: typically used in coherent systems. It corresponds to the point of minimum optical power and it is used because those systems are based in the conversion from electrical driving voltage into optical field.

A multi-level modulation transmission system requires the mapping of bits into symbols from a constellation diagram [29, 30]. Each complex symbol is composed by "in-phase" and "quadrature" components. Thus, the transmitted signal will have the complex envelope $x(t) = x_I(t) + jx_Q(t)$, where [24]:

$$x_I(t) = \sum_k i_k . p(t - k.T_s) \tag{2.6}$$

$$x_Q(t) = \sum_k q_k . p(t - k.T_s) \tag{2.7}$$

Both i_k and q_k represents the "in-phase" and "quadrature" components of the symbol sequence, and T_s is the symbol period. Thus, if one can join two Mach-Zehnder modulators and even add a phase modulator, an *optical IQ- Modulator* can be created:

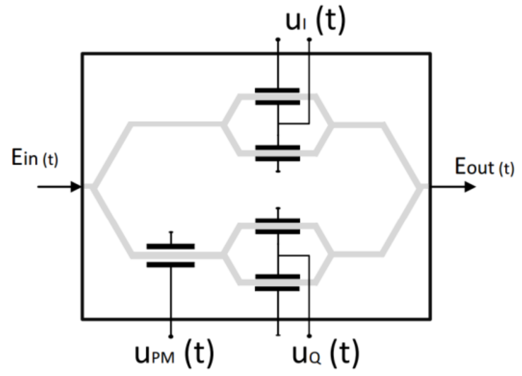


Fig 2.4 Optical IQ-Modulator

The IQ-Modulator is often used to create the usually called "Advanced Data Modulation Formats". The main reason is that with this kind of modulator one can modulate the optical carrier in amplitude and phase, creating data modulation formats as M-QAM (Quadrature Amplitude Modulation with 'M' constellation points), M-PSK, etc. These "Advanced Data Modulation Formats" bring benefits in terms of data rates and strength of the optical communications systems, for example. However, as seen in figure 2.3, the nonlinear characteristic of the MZM may be a problem when multilevel modulation data formats are implemented. Later on, solutions for this problem will be discussed.

As illustrated, the incoming electrical field is equally split into two arms: the "in-phase arm" and the "quadrature arm". In both arms, a field amplitude modulation is performed by operating the Mach-Zehnder modulators in "push-pull" mode. Moreover, a phase shift of '$\frac{-V_\pi}{2}$' is adjusted in one arm, given the possibility of reaching any constellation point in the complex IQ-plane, after recombining the electrical field of the two branches [24]. The transfer function of the IQ-Modulator is [24, 31]:

$$\frac{E_{out}}{E_{in}} = \frac{1}{2}\left(cos\left(\frac{u_I(t)}{2V_\pi}\pi\right) + jcos\left(\frac{u_Q(t)}{2V_\pi}\pi\right)\right) \tag{2.8}$$

This function proves that signals from both arms are intensity modulated and the signal from the quadrature arm will be also phase shifted by $\frac{\pi}{2}$ rad. Finally, this transfer function may be re-written in order to include the bias voltages applied to both arms of the modulator:

$$\frac{E_{out}}{E_{in}} = \frac{1}{2}\left(cos\left(\frac{u_I(t)+u_{DC,I}}{2V_\pi}\pi\right) + jcos\left(\frac{u_Q(t)+u_{DC,Q}}{2V_\pi}\pi\right)\right) \tag{2.9}$$

In order to modulate an optical continuous wave carrier produced by the laser, the IQ-Modulator must receive two electrical control voltages with encoded information. Those "driving signals" are generated in the digital part of the transmitter.

2.2 The Transmitter: Digital part

This is the part of the transmitter where binary serial data is mapped into specific constellation symbols of the Data Modulation Format that it is intended to implement. Depending on the symbol, two analog voltages will be generated and applied to the IQ-Modulator, allowing the transmission of the information through the optical domain. Therefore, it is necessary to convert the digital processed data into the analog domain. This can be done by using “Digital-to-Analog Converters” [23].

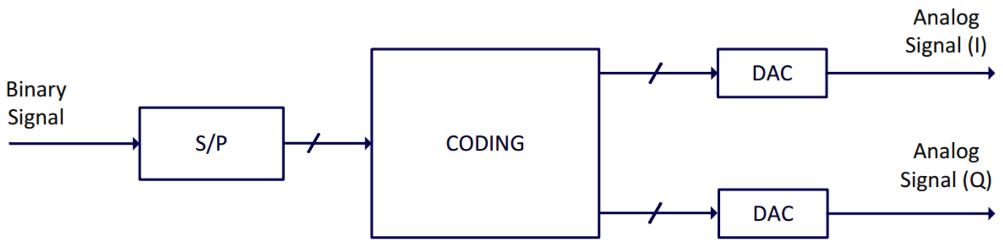


Figure 2.5 The Transmitter: Digital part

The incoming serial binary sequence represents the information that is going to be encoded and transmitted. In this work, pseudo-random binary information will be generated, since the goal is not to transmit any specific information. The “Serial to Parallel Converter” (‘S/P’) will receive the serial information and will parallelize it accordingly to the number of bits required to represent one symbol. The “Coding” block will receive each parallelized word and map it into the corresponding constellation symbol. It will generate two digital voltages, each one representing the projection of the symbol into the ‘I’ and the ‘Q’ axis of the complex IQ-plane. Finally, both DACs will receive digital voltages from the two arms of the transmitter and generate the corresponding analog voltages. It is possible to conclude that only the number of parallelized bits and the conversion of the symbols into two digital representations must be changed accordingly to the Data modulation Format that it is intended to implement.

In this work, more features will be added to the basic digital transmitter structure represented in the figure above. Each output from the “Coding” block can be seen as a “sample” that will be low-pass filtered by a digital implementation of the Raised-Cosine filter. This will allow studying the effects of this pulse shaping filter in the transmitted signal, in particular the spectral occupancy and the improved tolerances to nonlinearities. Furthermore, all the filtered samples will be predistorted in a “DSP” block. There, some mathematical operations will be applied to each sample in order to generate an electronic compensation of the nonlinear characteristic of the MZMs that constitute the IQ-Modulator of the optical part of the transmitter. The goal is to operate the MZMs in the “Null Biasing Point” and obtain non-distorted M-QAM constellations, regardless of the used signal’s excursion. This means that a high-quality data amplifier will be needed to fully exploit the modulation depth of the IQ-Modulator [26].

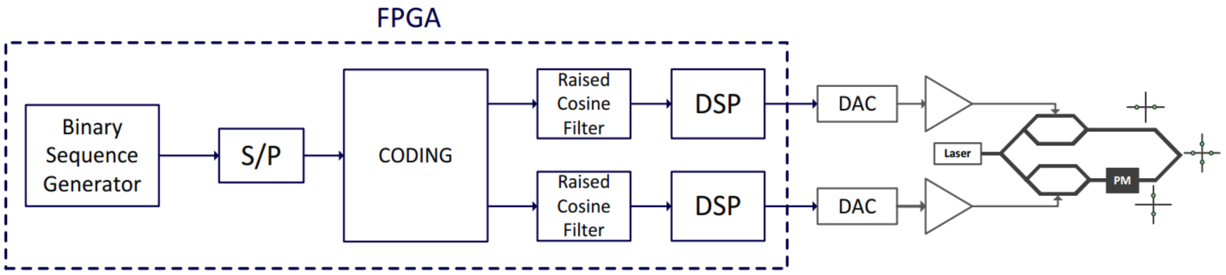


Figure 2.6 Transmitter's structure

2.3 Raised-Cosine filtering

2.3.1 Intersymbol interference

There are several ways to format binary logic information, in order to transmit it through a communication channel. However, those channels are not perfect, which will cause some signal distortion. If the amplitude and phase responses of a channel are non-ideal, the transmitted pulses will be spread. Intersymbol interference (ISI) occurs when a pulse spreads out in such a way that interferes with adjacent pulses at the sample instant [32].

This type of linear distortion is undesirable in TDM systems because it will cause interference between neighboring pulses. In FDM systems, each multiplexed signal will also be distorted, but no interference occurs with a neighboring channel. This is because each of the multiplexed signals occupies its own band. If the signal spectrums are non-overlapped, no interference occurs between them, since the channel issues will only affect the spectrum of each signal [32].

The main goal is to transmit a pulse at every T_b interval:

$$y(t) = \sum_k x_k . p(t - k.T_b) \tag{2.10}$$

If time-limited pulses are considered, they will not be band-limited, that is, the spectrum will be infinite. Thus, the band-limited communication channel will affect their spectrum and the transmitted pulses will be affected by dispersion effect. The solution is to generate pulses with band-limited spectrums. Despite these pulses are not time-limited, there is a way to format them in order to have no intersymbol interference at the sampling instants.

2.3.2 First Nyquist Criterion for null ISI

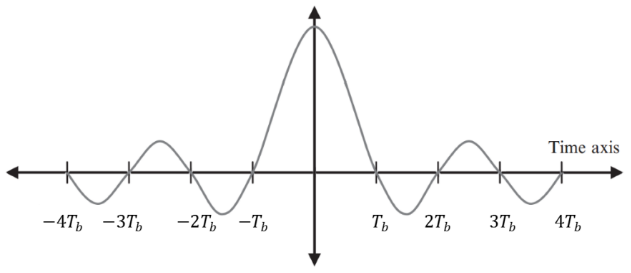
Nyquist has formulated a criterion which ensures that zero intersymbol interference may be achieved by choosing a pulse shape $'p(t)'$ with the following response:

$$p(t) = \begin{cases} 1, \ t = 0 \\ 0, \ t = \pm nT_b \end{cases} \qquad n = 1,2,3,4 \ldots \tag{2.11}$$

The Fourier Transform of $'p(t)'$ traduces the criterion in terms of frequency domain:

$$\sum_{n=-\infty}^{\infty} P\left(f - \frac{n}{T_b}\right) = T_b \qquad ,T_b = \frac{1}{R_b} \tag{2.12}$$

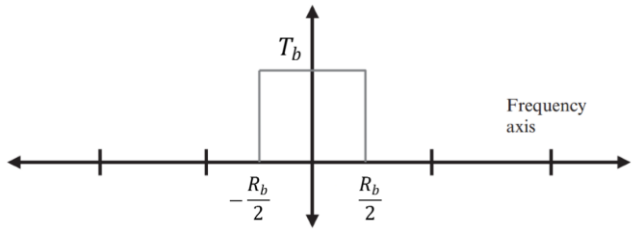
The impulse response $'p(t)'$ may be achieved with the "sinc" function:



$$p(t) = \frac{\sin(\pi.t.R_b)}{\pi t} = sinc(\pi.t.R_b) \quad (2.13)$$

Figure 2.7 Time Domain Representation of the 'sinc' function [32]

Thus, the Fourier Transform of a "sinc" function is:



$$P(f) = \begin{cases} \frac{1}{R_b}, & |f| \leq \frac{R_b}{2} \\ 0, & others \end{cases} = \frac{1}{R_b} rect(t.R_b) \quad (2.14)$$

Figure 2.8 Frequency Domain Representation of the 'sinc' function [32]

The reason why these pulses cause no ISI is that, at the sampling instants, each one has got a unitary value at its center and a null value at the points where other pulses are centered:

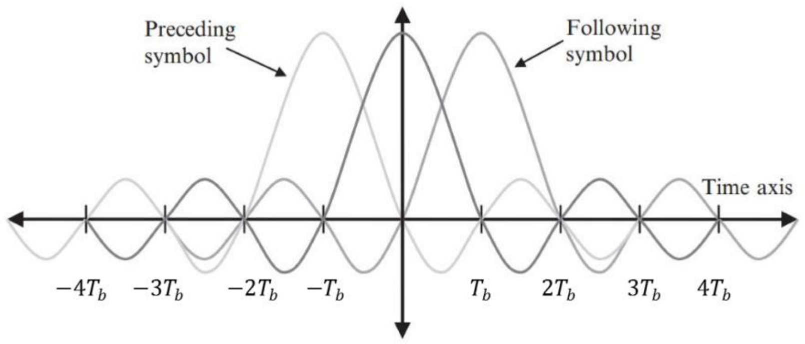


Figure 2.9 Zero Intersymbol Interference [33]

However, a "sinc" pulse is not possible to generate due to its infinite time duration and sharp transition band in the frequency domain, which is physically unrealizable. Furthermore, due to the slow decaying factor of $'p(t)'$, $\left(\frac{1}{t}\right)$, this pulse shape may cause significant intersymbol interference if the received signal is not sampled exactly at the bit instant - small margin for synchronization errors. The solution is to find a pulse that satisfies the equation above, but with a faster time decay and a softer transition in the frequency domain. There is a set of pulses known as "Raised-Cosine" that satisfy the Nyquist criterion and require a slightly larger bandwidth than the “sinc” pulse requires. The spectrum of these pulses is given by [30]:

$$H(f) = \begin{cases} T_b, & |f| \leq \frac{1-\alpha}{2T_b} \\ \frac{T_b}{2}.\left[1 + \cos\left(\frac{\pi T_b}{\alpha}.\left[|f| - \frac{1-\alpha}{2T_b}\right]\right)\right], & \frac{1-\alpha}{2T_b} < |f| \leq \frac{1+\alpha}{2T_b} \\ 0, & others \end{cases} \tag{2.15}$$

where ‘α’ takes values between ‘0’ and ‘1’, and is named as the “roll-off” factor.

The impulse response is given by the inverse Fourier Transform:

$$h(t) = \frac{\cos\left(\frac{\alpha.\pi.t}{T_b}\right)}{1-\left(\frac{2.\alpha.t}{T_b}\right)^2} \cdot \frac{sin\left(\frac{\pi.t}{T_b}\right)}{\frac{\pi.t}{T_b}} \tag{2.16}$$

The following figure intends to illustrate the impulse response and corresponding spectrum. The influence of the “roll-off” factor is also demonstrated:

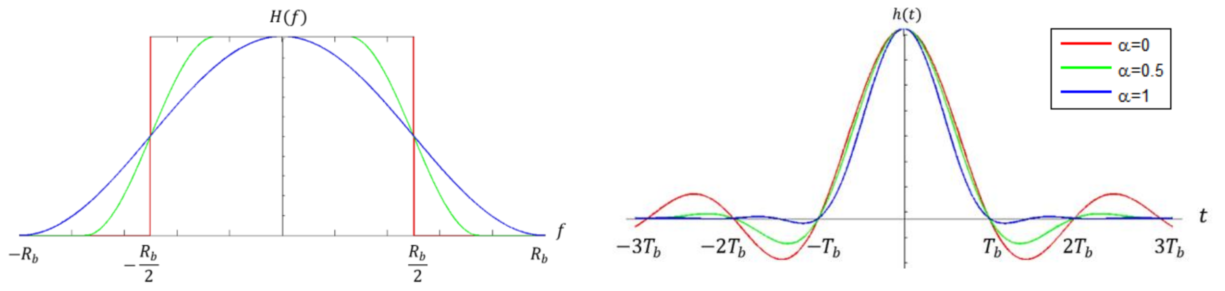


Figure 2.10 Raised-Cosine: spectrum and impulse response

As shown in figure above, the Raised-Cosine frequency response may be wider than the ideal low-pass filter response shown on figure 2.8, due to the transition band. This additional bandwidth can be adjusted by the “roll-off” factor. This parameter represents the percentage of bandwidth that is added to the defined minimum, or Nyquist, bandwidth. Thus, a null “roll-off” factor originates the ideal low-pass spectrum. On the other hand, if this parameter gets closer to the unitary value, the transition band becomes smoother and the bandwidth becomes higher. Therefore, since the minimum bandwidth is $\frac{R_b}{2}$ and the maximum is R_b, the following expression can represent the Raised-Cosine bandwidth:

$$BW = \frac{R_b(1+\alpha)}{2} \tag{2.17}$$

The value of the “roll-off” parameter has also consequences in terms of the Raised-Cosine impulse response. A null “roll-off” factor originates a pulse shaping similar to a “sinc” function, since the lobes become higher. However, if the “roll-off” factor gets closer to one, the lobes in the impulse response become smaller, improving the receiver tolerance to timing jitter.

Depending on the value of the “roll-off” factor, the impulse response will have different zero crossings. For example, when $\alpha = 1$, the pulse shape will have null values at $t = \pm\frac{(2n+1)}{2}T_b\ , n = 1,2,3\ldots$, in addition to the usual $t = \pm nT_b\ , n = 1,2,3 \ldots$ when $\alpha = 0$ [33].

Previously, it was stated that a pulse which satisfies the Nyquist Criterion, but with a faster decay, should solve the intersymbol interference problems due to receiver’s synchronization errors. In fact, as the “roll-off” factor increases, the impulse response goes to zero with a faster decay, and the adjacent lobes become very small. However, the corresponding frequency spectrum may be excessively wide ($BW = R_b$), depending on the applications. Thus, the “roll-off” factor results from a compromise between spectral bandwidth requirements, number of taps of the pulse-shaping filter and receiver sensitivity to intersymbol interference, and should be set depending on the applications [33].

2.3.3 Digital filtering

Ideal filters allow the transmission without distortion of a certain band of frequencies and suppress all the remaining ones. For example, the ideal Raised-Cosine filter (with $'\alpha' = 0$) allows all frequencies bellow $\frac{R_b}{2}$ to pass without distortion and suppresses all the components above $\frac{R_b}{2}$. However, many of the ideal filters are physically unrealizable, since they do not satisfy the causal condition [32]:

$$h(t) = 0\ , \quad t < 0 \tag{2.18}$$

In order to make possible a future hardware implementation of the Raised-Cosine filter, it will be useful to understand how can this filter be generated by a digital implementation.

In a generic way, digital filters are easier to design and simulate than the analog ones. They are capable of performance specifications that would be extremely difficult to achieve with an analog implementation. The actual procedure for designing digital filters has the same fundamental elements than the analog filters one’s. First, the desired filter responses are characterized, and the filter parameters are then calculated. Characteristics such as amplitude and phase response are derived in the same way. The key difference between analog and digital filters is that instead of calculating resistor, capacitor, and inductor values for an analog filter, coefficient values are calculated for a digital filter. These numbers reside in a memory as filter coefficients and are used with the sampled data values to perform the filter calculations. However, depending on the operation frequency and the filter specific requirements, digital filters usually require high performance DSP processors and Digital-Analog Converters.

There are two main types of digital filters: finite impulse response (FIR) and infinite impulse response (IIR). For the Raised-Cosine filter implementation, a FIR filter will be considered. Its impulse response has got a finite duration, settling to zero after a certain time. The next figure shows an example of the general structure of an ‘N’ order FIR filter:

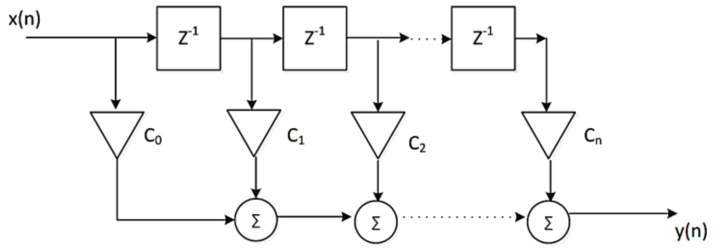


Figure 2.11 FIR filter Structure

This type of filter is usually composed by multipliers, adders and delay elements, which is particularly useful for hardware implementation. Its output can be defined as a weighted sum of the current and a finite number of previous values of the input:

$$y(n) = \sum_{k=0}^{N} C_k . x(n-k) \qquad (2.19)$$

Where 'n' is the sample's number, 'C_n' are the filter coefficients, 'N' is the filter's order, '$x(n)$' is the input signal and '$y(n)$' is the output signal. One can call to '$x(n-k)$' a "filter tap", where the name has its origins in the discrete taped delay line of digital filter theory. Thus, an 'N' order FIR filter generates 'N+1' taps [33].

The impulse response can be calculated by setting $x(n) = \delta(n)$:

$$h(n) = \sum_{k=0}^{N} C_k . \delta(n-k) = C_n \qquad (2.20)$$

The impulse response $h(n)$ has a finite length, i.e., is non-zero only for a finite range of indices '*n'*. For any FIR filter, $h(n) \neq 0$ only for $0 \leq n \leq N$. In this case, the filter is also causal [34]. The respective transfer function can be obtained by applying the Z-Transform to the impulse response:

$$H(z) = Z\{h(n)\} = \sum_{n=-\infty}^{+\infty} h_n . z^{-n} = \sum_{n=0}^{N} C_n . z^{-n} \qquad (2.21)$$

There are also other useful characteristics of FIR filters. One of the most important it is the FIR's linear phase response. In that case, the impulse response is symmetrical between its left and right side. The location of symmetry has been shifted from zero, which will result in the linear phase, being the slope of the phase's straight line directly proportional to the amount of the shift. It is also important to notice that a linear phase filter is equivalent to a zero phase filter. The main difference is that a zero phase filter is characterized by an impulse response symmetrical around sample zero. Therefore, the shift in the impulse response simply just produces an identical shift in the output signal [34].

2.3.4 FIR filter design: Raised-Cosine response

Since the frequency response of a digital filter is always periodic in the frequency variable '*w'* with a period of $'2\pi'$, the design specifications of the FIR filter need only to be specified for one period, usually, over the frequency region [-π, π]. Furthermore, when the frequency response is conjugate-symmetric (i.e. $H^*(w) = H(-w)$), then it is enough to specify the response only on the positive frequency interval $[0, \pi]$. The conjugate-symmetric case is the most common, because it corresponds to filters with real coefficients [34].

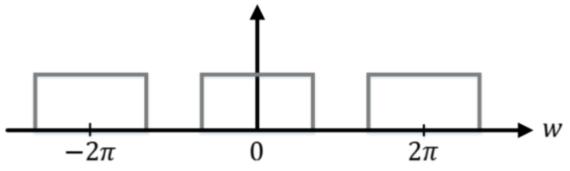


Figure 2.12 Periodic Frequency Response of the filter

The main goal is to design a FIR filter whose frequency response is the same as the Raised-Cosine filter. As seen before, the impulse response of a FIR filter is given by its coefficient values. Thus, the filter coefficients (and consequently, the impulse response) may be calculated by replacing the desired filter parameters in the expression of the impulse response of the ideal Raised-Cosine filter. However, the filter coefficients derived from the expression 2.16 will not produce a causal filter, since they do not satisfy the causal condition previously defined. This means that the system could not be implemented in real time. This can be verified if one considers the output of a discrete-time system produced by the convolution of the input signal with the impulse response coefficients as shown in equation above:

$$y(n) = \sum_{k=-M}^{M} C_k . x(n-k) \tag{2.22}$$

For example, when $k = -M$, the summation includes a term $x(n + M)$ that refers to an input value 'M' sampling periods behind of $y(n)$'s reference time. The problem can be handled by shifting all coefficient values to the right on the time axis so that only positive values of 'n' produce coefficients, as shown in the next figure. The disadvantage of this action is to increase the time delay between system input and output by 'M' sampling periods [35]:

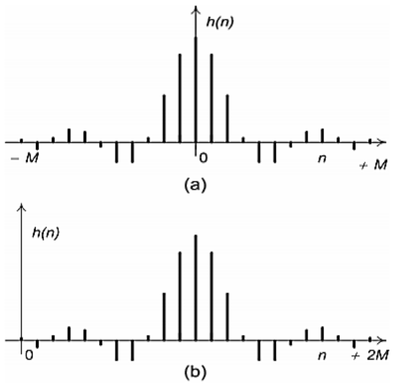


Figure 2.13 a)Non-Causal coefficients b) Causal Coefficients [35]

The causal coefficients can be determined from the non-causal coefficients by making some index adjustments. As the non-causal coefficients indices take on values from '–M' to '+M', the causal coefficient indices will take on values from '0' to '2M' [35]:

$$h(n+M)_{causal} = h(n)_{noncausal}\ , \ n = 0, \pm 1, \pm 2 \ldots . \pm M \tag{2.23}$$

Furthermore, it is also necessary to guarantee that filter coefficients are symmetric, in order to have a FIR filter with the desired linear phase response [35]:

$$h(n) = h(-n) \tag{2.24}$$

where a non-causal impulse has been considered.

During the filter's coefficients calculation, and depending on the desired characteristics of the filter, some mathematical indeterminate forms may arise, hindering the proper calculation of these coefficients. One of the multiple ways to solve the problem is to apply the well-known "L'Hôpital's Rule". The application (or repeated application) of this mathematical tool allows converting an indeterminate form to a determinate one, ensuring the correct evaluation of the impulse response expression [36]:

Let $f, g: (x_0, b) \rightarrow \Re$. *Suppose* $f(x_0) = g(x_0) = 0$ *and* f, g *are differentiable on* (x_0, b).
Let $g'(x) \neq 0$ *for all* $x \in (x_0, b)$. *Then,*

$$\lim_{x \to x_0^+} \frac{f(x)}{g(x)} = \lim_{x \to x_0^+} \frac{f'(x)}{g'(x)} \tag{2.25}$$

Thus, if an indeterminate form of the $\frac{\infty}{\infty}$ or $\frac{0}{0}$ type arises, one may derive both numerator and denominator of the expression and then complete the calculations. This technique was useful in the calculations of the filter's coefficients that were implemented in the experimental part of this work.

2.4 Digital Signal Processing

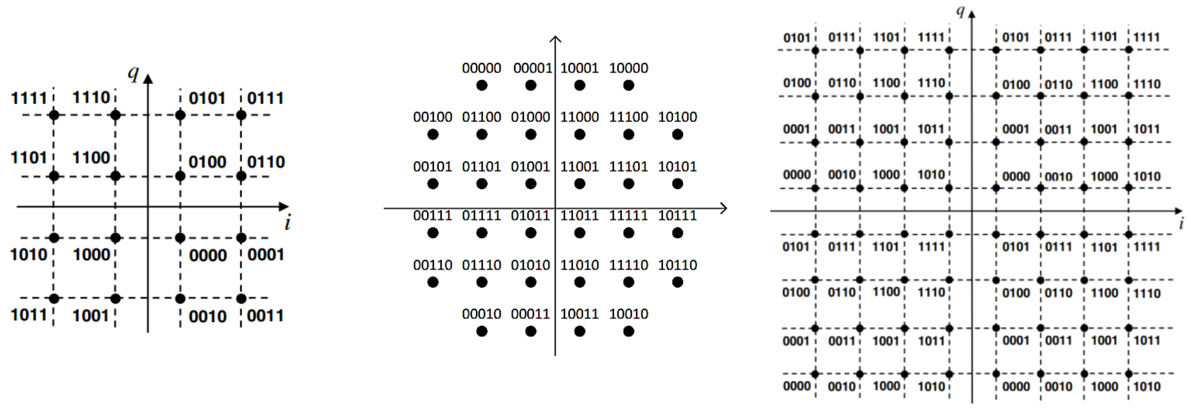
Before being converted into the analog domain, the filtered samples will be predistorted in the "DSP" blocks. As seen before, Mach-Zehnder modulators have got a nonlinear characteristic. This may be a problem when multi-level modulation data formats are implemented in optical coherent transmission systems. In this section, a solution to compensate the nonlinear response of the modulators is presented.

First of all, it is important to define the QAM data modulation formats. They will be studied in this work because they are widely used in today's transmission systems and also because they emphasize the MZM's nonlinearities, due to their multiple amplitude levels.

There are two possible types of QAM constellations: Star QAM and Square QAM. The first one will not be studied since the constellation is not optimal as regards noise performance, because symbols on the inner ring are closer together than symbols on the outer ring. In order to improve noise performance, Square QAM constellations will be generated since they have more balanced Euclidean distances between constellation symbols [24].

QAM modulation formats can be generated by the previously studied transmitter scheme. The signals of both transmitters' arms represent the "in-phase (I)" and "quadrature-phase (Q)" components of the resulting optical signal. The number of levels of those electrical driving signals is equal to the number of projections of the symbol points into the 'I' and the 'Q' axis of the complex IQ plane. For example, a Square 16-QAM requires two quaternary driving signals [24].

Depending on the value of the "M", different constellations may be generated. Below are shown the most common representations of M-QAM:



a) 16-QAM b) 32-QAM c) 64-QAM

Figure 2.14 M-QAM constellations

It is possible to notice that adjacent symbols are equally spaced by some previously defined distance (‘d’).

QAM increases the transmission´s efficiency, encoding $log_2(M)$ bits in one symbol, offering higher spectral efficiency than QPSK and reducing the required symbol rate to obtain the same overall bit-rate [26]. However, there are also disadvantages of using this data modulation format. Apart from the higher complexity, QAM also have reduced tolerance towards nonlinearity than QPSK because of the presence of multiple intensity levels and, hence, higher peak-to-average ratio [26]. QAM is also more sensitive to noise (requires an increased OSNR), since the constellation points are much closer, when compared to other modulation formats. Thus, at the receiver, greater accuracy is required for detecting a received symbol. For example, for the same symbol rate, 64-QAM allows transmitting at higher bit rates than 16-QAM because each symbol “encodes” six bits, against the four bits of 16-QAM. However, the 16-QAM Euclidean distance between symbols is greater, allowing detection with fewer errors (greater Quality of Service).

In order to illustrate the impact of the MZMs nonlinear characteristic in the generation of multi-level modulation formats, a simple example is shown below.

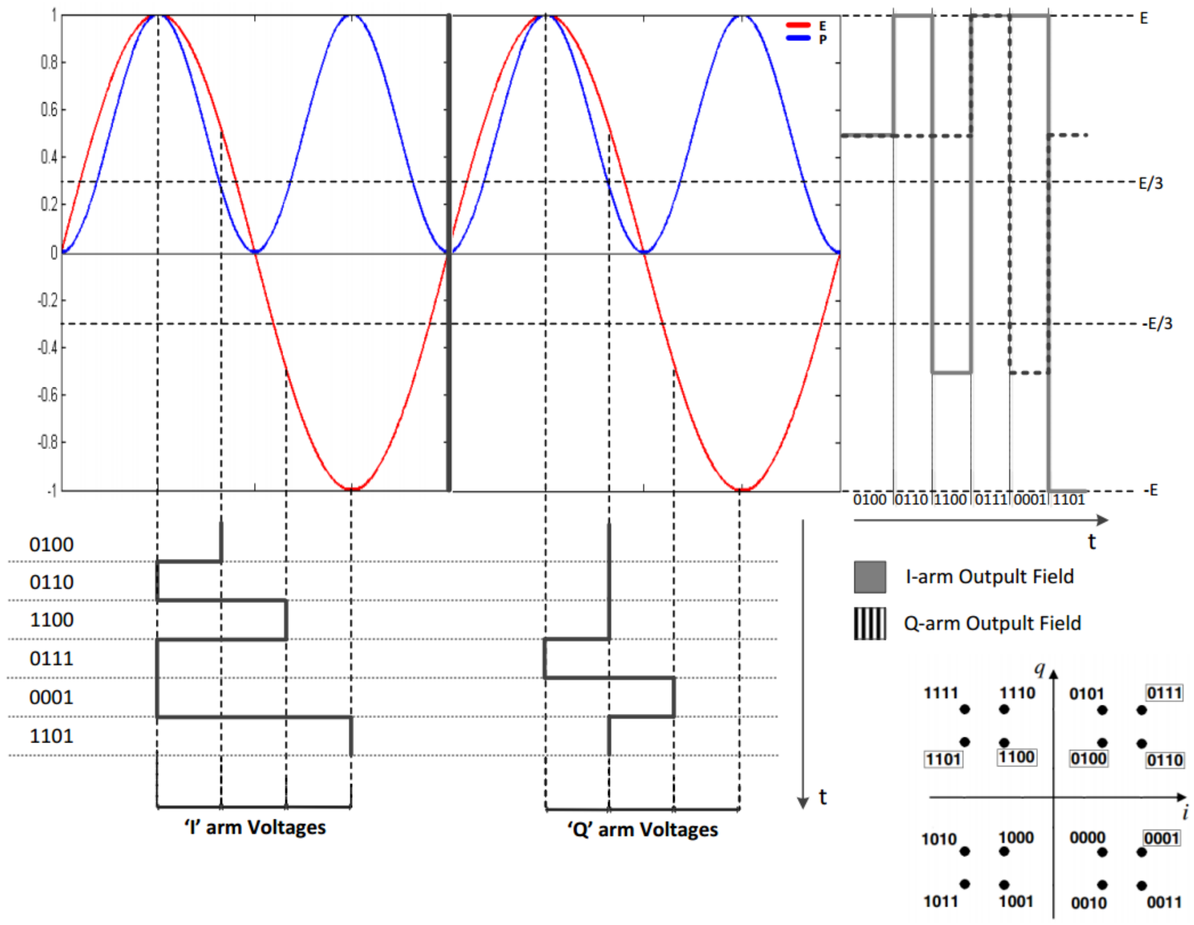


Figure 2.15 Example of the impact of the MZMs nonlinear characteristic in the generation of multi-level modulation formats

In the example, some random binary data is encoded in six different 16-QAM symbols, which are then converted into the proper analog driving signals and applied to the IQ-Modulator electrodes. The transmitter follows the structure given in the figure 2.6, where the samples are directly converted into the analog domain without any kind of filtering or digital signal processing. The driving signals will be applied to the IQ-Modulator electrodes, modulating an Optical Continuous Wave Carrier.

Accordingly to the constellation shown in figure 2.14-a), it is only possible to represent all the 16-QAM symbols if quaternary driving signals were generated in both arms of the transmitter. The voltage levels of each signal must be equally spaced, respecting the Euclidean distances ('d') of a 16-QAM constellation. However, the generation of these driving signals will not result in the expected constellation, since the transfer function of the MZM is nonlinear. Thus, the constellation will come slightly deformed, with unequal Euclidean distances between symbols [25], as can be seen in fig. 2.15.

Following the example, the symbols "0100" and "0110" of the resulting optical constellation will be spaced by a Euclidean distance lower than the expected one ('d'). On the other hand, the symbols "0100" and "1100" will be spaced by a distance greater than 'd'. These two examples allow explaining the reason why the resulting constellation becomes distorted: due to the nonlinear electrical field transfer function, the driving signals are leading both MZMs that compose the IQ-Modulator to generate electrical fields with unequally spaced levels (in terms of absolute value) [25].

In order to generate a correctly sized constellation, some electronic predistortion of the driving signals should be created in order to artificially linearize the transfer function of both MZMs that compose the IQ-Modulator. That is the main goal of the "DSP" block at the transmitter side: to compensate the nonlinear characteristic by adjusting the voltage levels of the driving signals right before they being converted into the analog domain and injected in the IQ-Modulator electrodes. The "junction" of the 'DSP' block transfer function with the IQ-Modulator characteristic should result in a linear function. This allows the "Coding" block to generate 'I' and 'Q' digital driving signals with equally spaced voltage levels, since they will result in correctly sized constellation symbols.

The region of interest of the IQ-Modulator electrical field transfer function is located between the curve's maximum and minimum points, since the driving signals are usually biased in the "Null-Point". So is quite enough to ensure a linear response of the modulators in that range. The 'DSP' component should convert a received digital voltage into a different value (V_{drive}). As will be explained in the following section, its transfer function should have an "arccosine" shape, since it must be the reversed transfer function of the modulator.

2.4.1 Predistortion algorithm

The Mach-Zehnder Modulators that constitute the IQ-Modulator have got an electrical field transfer function with a cosine shape, as stated in equation 2.8. The 'DSP' module will only perform a correct predistortion of the incoming voltages if its response is the reversed transfer function of the modulators. The inverse trigonometric function of a cosine is an arccosine function. However, an arccosine implementation requires some caution. The first point to take into account is the function's domain. It is restricted to incoming values from '-1' to '1', and the output range of the function is comprised between '0' and 'π'. Thus, since that the four possible input digital voltages of the "DSP" block will be restricted to the range from '0' to '1' volt, it will be necessary to guarantee that the predistorted values will also be limited to that range. To achieve this, the input digital voltages must be processed in order to ensure that their excursion becomes equal to the arccosine's domain. Furthermore, a normalization by a factor of $'\pi'$ must also be applied to the predistorted values:

$$V_{drive}\ (v) = \frac{1}{\pi}.\arccos(1-2v) \tag{2.26}$$

Where $'v'$ is the input digital voltage, restricted to the range from '0' to '1', and V_{drive} is the corresponding predistorted digital voltage, also comprised between '0' and '1' volt.

Each V_{drive} signal is not ready to drive the IQ-Modulator electrodes yet. After being converted into the analog domain, it is necessary to give some amplification and offset to both driving signals, so they can act in the desired region of the modulator's transfer function. For example, if we want to use the maximum excursion of the region of interest of the field transfer function of the MZMs, the driving signals must be amplified to have an excursion of "$2V_{\pi}$". The offset parameter will ensure that the driving signals will be biased in the "Null Transmission Point" of the characteristic. In laboratorial practice, the amplification and offset parameters must be permanently adjusted, since the IQ-Modulator characteristic changes during the operation time. There are some causes for this situation, like aging or temperature fluctuations.

In many applications it will not be necessary to use the entire excursion of the modulator's field transfer function. Thus, the 'DSP' module should be prepared to provide an "adaptive gain", which will allow the control of the driving signal's excursion. The gain factor will always have a value less or equal to one. This feature will be particularly useful when the incoming signal has got a Raised-Cosine shape. This signal will be certainly affected by overshoots and may exceed the limits of the arccosine's domain. Hence, the arccosine action will cause significant signal distortions (phase inversions). In this case, a gain factor less than one should be applied to the incoming signal before doing the predistortion, in order to guarantee that its excursion will be constrained to the required range. That is the reason why this gain factor should always be applied to the incoming signal right before doing the predistortion: if the excursion's control was made after that, the impairments caused by the overshoots will not be solved.

However, this "dynamic gain" factor will bring undesired issues to the resulting constellation, as will be demonstrated in the following calculations.

An ideal Mach-Zehnder modulator generates an electrical output field (E_{out}) given by:

$$E_{out} = \frac{Ein}{2}\left(e^{j\varphi_{up}} + e^{j\varphi_{low}}\right), \text{ where } \varphi_{up} = \pi\,\frac{RF_{up}+Vdc_{up}}{V_\pi} \text{ and } \varphi_{low} = \pi\,\frac{RF_{low}+Vdc_{low}}{V_\pi} \tag{2.27}$$

Note:
$RF_{up/down}$ is the driving voltage of the upper/down electrode
$Vdc_{up/low}$ is the DC voltage applied to the upper/down electrode

Assuming that $V_i = RF + Vdc$ is the voltage of the modulator driving signals and will take values from '0' to '$2V_\pi$', the electrical output field of the MZM operating in the "push-pull" mode is given by the equation 2.28:

$$E_{out} = \frac{Ein}{2}\left(e^{j\pi\frac{Vi}{2V_\pi}} + e^{-j\pi\frac{Vi}{2V_\pi}}\right) = \frac{Ein}{2}.2\cos\left(\pi\,\frac{Vi}{2V_\pi}\right) = Ein.\cos\left(\pi\,\frac{Vi}{2V_\pi}\right) \tag{2.28}$$

When the "dynamic gain" is unitary, the predistorted driving signals will have their maximum excursion and will use the entire range of the region of interest of the field transfer function. Therefore, no constellation impairments will be visualized. However, if the gain factor is not unitary, the driving signals will have smaller excursions than on the previous situation. This means that the DC level of the driving signals will change and 'V_i' will take values from '0' volts to a value lower than '$2V_\pi$' volts. Accordingly to the previous expression, E_{out} will not have symmetrical maximum and minimum values. Thus, these output electrical fields will lead to distortions on the symbol mapping, which will be more pronounced as lower is the value of the gain factor. Since the IQ-Modulator has got two Mach-Zehnder Modulators, this undesired effect will affect the resulting constellation, which will have some phase distortions on the symbol mapping due to the combination of both modulators action.

The solution to solve this problem is to convert the input unipolar signals of the “DSP” block to a polar representation, which is equivalent to remove the signal’s DC level. Considering that there are no overshoots, the polar signal will be restricted to the range from “-0.5” to “0.5” volts. The gain factor should only be applied to the signal after this conversion takes place. However, before being predistorted by the arccosine function, the previously removed DC level of the signal must be re-added. Therefore, changing the value of the “adaptive gain” factor will only produce a corresponding change on the driving signals excursion. The resulting constellation will not be affected and will appear always centered around the origin of the IQ plane, since the DC component of the driving signals will remain the same, keeping them biased in the “Null-Point” of the modulator’s transfer function. This phenomenon will be demonstrated in the next chapter, during the OSIP simulation.

3. OSIP Simulation

OSIP (Optical Simulator Platform) [37] is a simulation platform designed to run over MATLAB that was developed at University of Aveiro and at Telecommunications Institute. It includes a wide variety of models for optical communications, as building blocks, and provides a graphical interface to define the system topology, allowing the simulation of several optical communication systems [31]. The simulations results can be analyzed in the same graphical interface. The main advantages of using OSIP are the wide range of pre-defined functions available, its flexibility in performing complex scientific calculations, its compatibility with all the recent MATLAB releases and the possibility to review and change the code that describes each simulation component [31].

OSIP will be the basic tool used to implement an optical transmission system under ideal conditions. The main goal is to simulate some M-QAM optical transmitters, to design a Raised-Cosine filter and validate the algorithm to compensate the nonlinear characteristic of the Mach-Zehnder modulator.

3.1 Simulation setup and principles of operation

In the next figure is shown the block diagram of an optical transmission system implemented in OSIP.

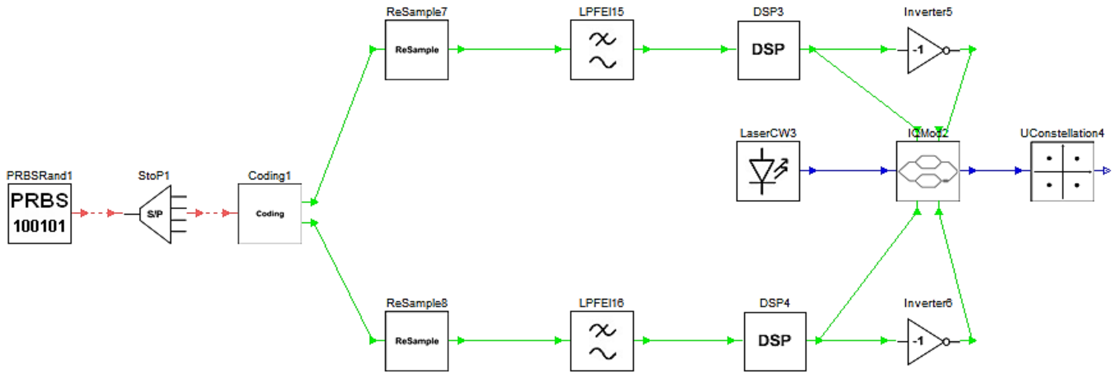


Figure 3.1 Block Diagram of the Transmitter

The structure of the block diagram is very similar to the one presented in section 2.2. In “PRBSRand” block, a pseudo-random binary sequence is generated, repeating itself with an “N” bit period that can be defined by the user in the “Numerical parameters” menu of the simulator. Those serial bits will be parallelized by the “StoP” block and mapped into the corresponding constellation symbol by the “Coding” block. There, two digital voltages will be generated, each one representing the projection of the symbol into the ‘I’ and the ‘Q’ axis of the complex IQ-plane. Before the filtering process, each original sample should be upsampled by a certain factor ‘L’. This operation is performed by the “ReSample” block and will be explained in the next section. It intends to represent both signals with more “samples per symbol”, avoiding intersymbol interference. After that, the “LPFel” component allows to apply one of the multiple available filters to the upsampled signals: Bessel, Butterworth or Rectangular filters. The filters will also work as pulse shapers, giving the temporal format to the samples. However, as explained before, the purpose of this simulation is to design and validate a Raised-Cosine filter, which is expected to bring higher performance to the transmission system, due to the reasons stated before. Thus, ideal and FIR-based implementations of this filter were designed and implemented in this simulator, improving its set of filtering features.

Before being converted into the analog domain, all the filtered samples will pass through the "DSP" blocks, where the predistortion algorithm described in section 2.4.1 will be implemented. The two resulting signals will drive an IQ-Modulator composed by two Dual-Drive MZMs. So, "Inverter" blocks will be needed to define the "push-pull mode", guaranteeing that the voltages applied to the electrodes of each MZM are symmetric. In the following simulations, the IQ-Modulator will have a field transfer function with a $'V_\pi'$ of three volts and an offset voltage of six volts. The value chosen for $'V_\pi'$ intends to simulate the IQ-Modulator present at the laboratory. The offset (positioning of the curve on the voltage axis) is an arbitrary value that intends to simulate a real case scenario, where offsets are always present.

OSIP also provides a group of blocks that allows the user to visualize the simulation results in specific points of the transmitter block diagram. They will be included in the schematics of the simulations performed in this chapter. For example, the "UConstellation" block displays the transmitted constellation after the modulator and the "Scope" and "ElectSpectrum" blocks allow the user to analyze the signals in time and frequency domain, respectively.

3.2 16-QAM Simulation

The 16-QAM simulation allows to assess the general operation of an optical transmitter system. It will be possible to design several Raised-Cosine filters and compare their different impacts on the signals. Furthermore, the previously studied mathematical operations that must be applied to the driving signals in order to linearize the IQ-Modulator response will also be implemented and the results will be analyzed. If the results become improved and the predistortion operation is validated, a hardware implementation of the transmitter system will be designed to run in a FPGA platform, allowing the completion of some tests in a laboratorial environment.

The reason why the simulations will be performed with M-QAM transmitters is because these advanced modulation formats emphasize the MZMs nonlinear impairments. If a QPSK transmitter was simulated, the effects will not be noticed, as will be explained later. The table 3.1 resumes the main parameters introduced in the simulator to generate a correct 16-QAM simulation:

Block Name	*Introduced Parameters*
PRBSRand	100 Mbps
StoP	4 parallelized bits
Coding	16-QAM constellation
ReSample	Factor L=4
LPF el	Raised-Cosine filter; Adjustable Roll-off Factor
LaserCW	193.5 THz; 1549.32 nm
IQM	V_π=3v; V_{PM}=-3/2 v

Table 3.1 16-QAM Simulation parameters

As will be seen in the next chapter, the “Digital-to-Analog Converter” Evaluation Board, which will be used to convert the processed digital data into the analog domain, distorts the output signals (due to differentiation effect), since their spectra extend to lower frequencies than the cutoff frequency of the DAC’s output circuits (which have a high-pass behavior). Due to some speed limitations of the FPGA and the DAC, it will not possible to generate those driving signals at higher bit rates. Therefore, one valid solution is to create a specific binary sequence that reduces the low-frequency content of the driving signals, making the distortion effect being less noticed.

The sequence must ensure the mapping of all the 16-QAM constellation symbols, which gives rise to a minimum length of 64 bits (there will be no repeated symbols). This minimizes the number of samples that will be generated in each period of the driving signals, achieving their maximum possible output frequency. Furthermore, the binary sequence must also ensure that the voltage levels in both ‘I’ and ‘Q’ DAC outputs commute in every symbol transitions.

The 100MHz sampling frequency that will be used in the following simulations was set by taking into account several hardware constraints, like FPGA’s and DAC’s speed limitations and also the maximum acceptable duration time of a sample in both DAC’s outputs: 40 ns. This value was experimentally obtained using an oscilloscope and corresponds to the time where a voltage level of the DAC’s output can remain constant without suffering a significant distortion by the differentiation effect. Therefore, this sampling frequency ensures that each sample will have a temporal duration of 10 ns, which is inferior to the maximum requirement of 40 ns.

Thus, the OSIP transmitter system was configured to be continuously generating the following fixed binary sequence at 100Mbps:

“1000110101110010010100001111101011001001111010110100000101100011”

In the “StoP” block, these serial bits will be parallelized in groups of four bits, allowing the mapping to the 16-QAM constellation. Thus, the symbol rate is:

$$Symbol\ Rate = \frac{BitRate}{log_2(M)} = \frac{100Mbps}{log_2(16)} = 25\ MSymbols/s \tag{3.1}$$

The “Coding” block will use conversion tables to map the parallelized bits into 16-QAM constellation symbols. Therefore, each constellation symbol will be converted into the two corresponding digital output voltages (which can be called “samples”), with values comprised between zero and one volt.

Before the filtering process, it is necessary to upsample the signals from both “I” and “Q” arms of the transmitter, in order to increase their sampling rate and to provide a temporal pulse shape to the signal symbols. Therefore, "Upsampling" can be defined as the process of inserting zero-valued samples between original samples to increase the sampling rate of a signal. If we consider $x(n)$ the signal that is going to be upsampled by an *‘L’* factor, the resulting signal is [38]:

$$y(n) = \begin{cases} x\left(\frac{n}{L}\right), & n = kL,\ k \in \mathbb{Z} \\ 0, & otherwise \end{cases} \tag{3.2}$$

Thus, an *‘L’* upsampling factor means that *‘L-1’* zeros will be inserted between two consecutive original samples. In this simulation, an upsampling factor of ‘4’ was chosen, which means that, after the "ReSample" block, in each four samples, only one contains the original information. The remaining three samples are simply added zeros and do not contain any information. They will only allow a correct shaping of each original sample by the Raised-Cosine filter, avoiding intersymbol interference.

After upsampling by a factor of '4', the new sampling period will become $'\frac{T}{4}'$, leading the new sampling frequency to be four times greater than the previous one. In the frequency domain, this indicates that the spectral images originally centered at multiples of the symbol rate ($\pm 25MHz, \pm 50MHz$ ...) will be included in the frequency range from 0 Hz to the new Nyquist limit of $' \pm \frac{L \cdot f_{symb}}{2}'$, i.e., $' \pm 50MHz'$ [38]. Therefore, the spectral content of the signal will remain unchanged, since no information is added to the signal. The upsampling process will only "contract" the frequency axis, increasing the range of representable frequencies from 0 Hz to $\frac{f_{sample}}{2}$ Hz.

In order to preserve the original baseband content of the signal, a lowpass filter with a cutoff frequency of $'\frac{f_{symb}}{2}'$ must be applied to the upsampled signal. The next figure shows the resulting spectrum after filtering the signal with an ideal Raised-Cosine filter (unitary roll-off factor), which transfer function was shown in the expression 2.15.

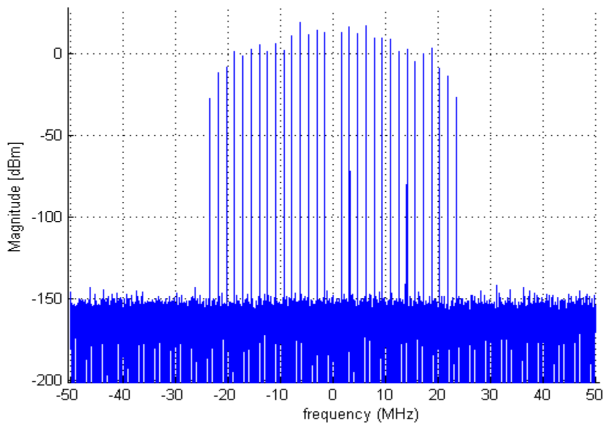


Figure 3.2 Simulation Results: Spectrum after filtering with an ideal filter

Naturally, this is an ideal filtering and don´t simulates the real case scenario that is supposed to study. Thus, the solution is to design a Raised-Cosine Filter based on digital FIR filters. The calculation methods of the filter's coefficients were described in section 2.3.4 and can be proven by MATLAB's filter design tool called "FDAtool". This software makes the filter design easier and intuitive by allowing the selection of "Raised-Cosine" filter type, the sampling rate, cutoff frequency and filter order. This tool also generates the filter's coefficients for the introduced parameters. The following figure shows the resulting spectrum after filtering the signal with an FIR Raised-Cosine filter (unitary roll-off factor, 10th order):

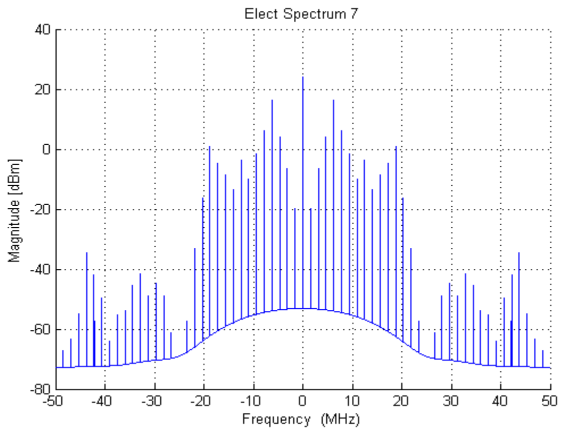


Figure 3.3 Simulation Results: Spectrum of the filtered signal

Here, the rejection band is more realistic and depends on the filter's order. As expected, the main spectral lobe has its first null at 25MHz. In the time domain, it is possible to see the four different voltage levels of the driving signals:

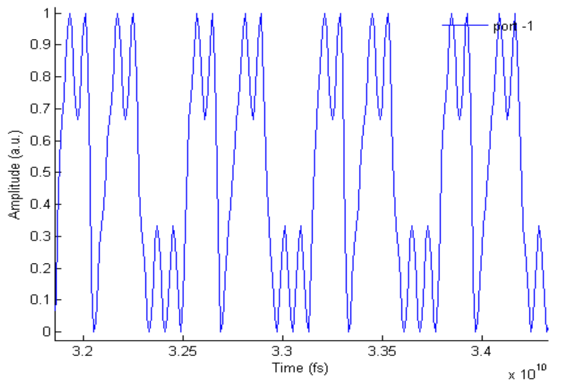


Figure 3.4 Simulation Results: Time Domain Representation of the filtered signal

The first simulations intend to evaluate the impacts of the Mach-Zehnder Modulators' nonlinear characteristic in the generation of 16-QAM constellations. Thus, the "DSP" block was configured to apply an "Adaptive Gain" to the incoming samples, amplify the signals by a factor of $'V_\pi'$ and ensure the biasing in the null point of the transfer function. Predistortion was not performed. The first results were obtained by configuring the excursion of the driving signals for using only 5% of the entire region of interest of the characteristic. Then, the "Adaptive Gain" was gradually increased until reach its maximum unitary value (see figure 3.5). In order to quantify the impact of the MZM characteristic in the obtained constellations, a study of the Error Vector Magnitude (EVM) was performed. Considering that the received signal vector $'E_r'$ deviates by an error vector $'E_{err}'$ from the ideal transmitted vector $'E_t'$, the EVM can be defined as the mean root square of $'E_{err}'$ for a number of $'I'$ randomly transmitted data [39, 40]:

$$EVM_m = \frac{\sigma_{err}}{|E_{t,m}|},\ \sigma^2{}_{err} = \frac{1}{I}\sum_{i=1}^{I}\left|E_{err,i}\right|^2, E_{err,i} = E_{r,i} - E_{t,i} \tag{3.3}$$

The power of the longest ideal constellation vector with magnitude $'|E_{t,m}|'$ served for normalization. Informally, EVM is a measure of how far the received symbols are from the ideal locations. In the following graphic each value of the "Adaptive Gain" is associated with an average value of the EVM, quantifying only the symbol deviations from its ideal positions:

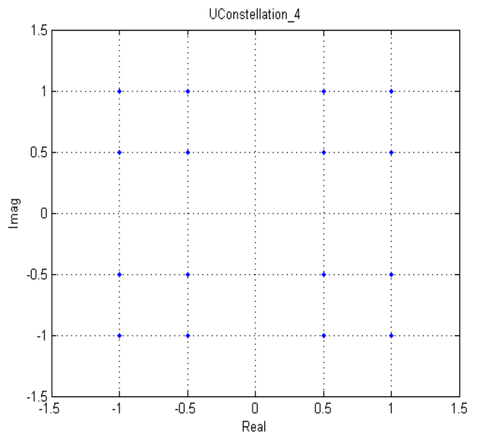


Figure 3.5 Simulation Results: incorrect 16-QAM constellation for the full excursion

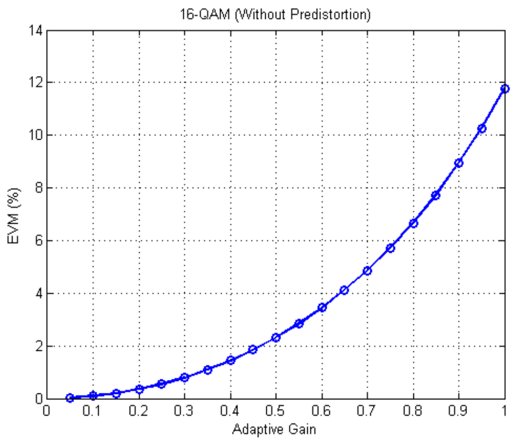


Figure 3.6 EVM (mean) for different values of "Adaptive Gain"

It is possible to verify that gradual increases in the "Adaptive Gain" were traduced in higher percentages of the EVM, since the driving signals started to operate in nonlinear regions of the characteristic and distortions in the constellations started to being noticed. The maximum distortion was obtained for the full excursion (EVM of almost 12%). Therefore, it is only possible to use driving signals with higher excursions if the predistortion is applied, simulating a linear response of the modulator in the previously defined region of interest of its characteristic and ensuring a null EVM for all the excursions. In another OSIP simulation, the 'DSP' block was configured to perform all the intended operations to the filtered samples. The "Dynamic Gain" factor had to be configured to be less than '1', reducing the signal's excursion before being predistorted by the arccosine function. If this factor is not correctly defined, the signal can be affected by overshoots that will make it to exceed the domain boundaries of the arccosine function, which will create a phase distortion in the signal, usually called "clipping". An Eye Diagram is a useful tool to analyze the performance of the system. This visualizer block was included in the transmitter diagram right after both "DSP" blocks and will allow studying some characteristics of the processed signals. As can be seen in the next figure, the four different voltage levels of the IQ-Modulator driving signals are now unequally spaced. In this particular situation, where the signal excursion is at its maximum (amplitude of $'V_\pi'$ volts), one can verify that there is no ISI and parameters like "Eye Amplitude "and "Eye SNR" are satisfactory. However, it is possible to see that some "Jitter" is affecting the signal:

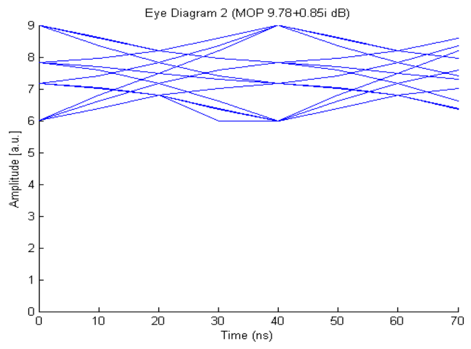


Figure 3.7 Simulation Results: Eye Diagram of the IQ-Modulator driving signals

It is important to notice that the excursion of the driving signals does not have the amplitude of $'2V_\pi'$, but only $'V_\pi'$ volts. This is due to the fact that we are simulating a transmitter with a dual-drive IQ-Modulator. Thus, an "Inverter" is present to define the "push-pull mode" and to guarantee that the signal excursion that drives each modulator has got $'2V_\pi'$ amplitude.

The resulting constellations are now correctly sized for all the excursions, validating the predistortion algorithm under ideal operating conditions. We can conclude that the transfer function of the implemented 'DSP' block is the reversed transfer function of the modulator.

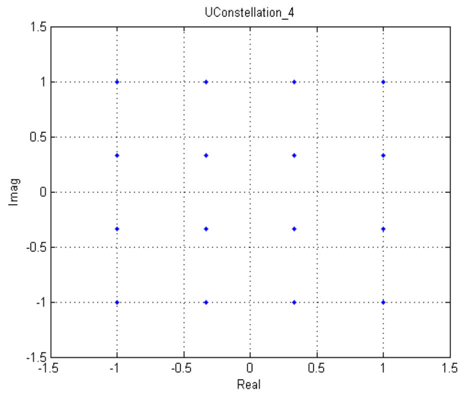


Figure 3.8 Simulation Results: 16-QAM constellation for unitary "Adaptive Gain"

3.2.1 Analysis of the resulting optical spectrum

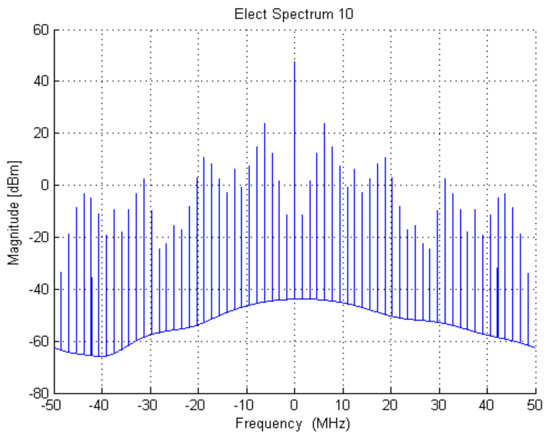


Figure 3.9 16-QAM Simulation Results: Spectrum of the driving signals

As can be observed in the figure above, the predistortion algorithm has got adverse effects on the spectrum of the signals that are going to drive the IQ-Modulator. This is an expected result, since the natural shape of a “Raised-Cosine” signal is being manipulated by some nonlinear operations. However, the optical spectrum of the transmitted signal has got a well-defined “Raised-Cosine” shape:

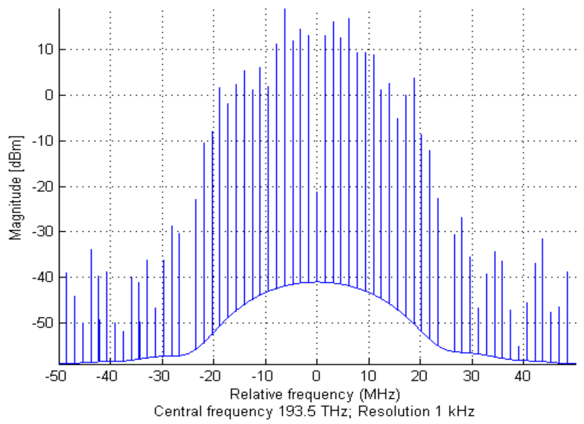


Figure 3.10 Simulation Results: Optical transmitted spectrum

This result clearly confirms that the nonlinearities of the IQ-Modulator response affect the transmitted spectrum “in an opposite way”, when compared to the ‘DSP’ block effect. As was previously explained, the ‘DSP’ transfer function should be the inverse transfer function of the IQ-Modulator. Thus, the junction of the ‘DSP’ transfer function with the IQ-Modulator characteristic should result in a linear function. So, the spectral distortion caused by the ‘DSP’ operation is compensated by the IQ-Modulator action, and the resulting spectrum is almost similar to the original “Raised-Cosine” one. The necessary condition to observe these results is to have an IQ-Modulator with a bandwidth sufficiently higher than the signal bandwidth. The following simulation intends to proof this statement:

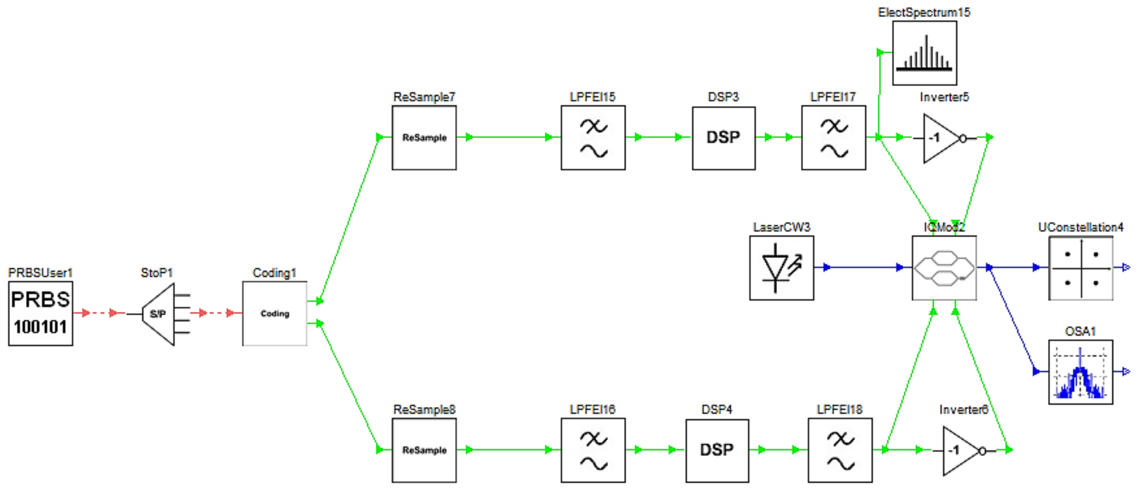


Figure 3.11 Block Diagram of the transmitter with pre-filtering

Right before the IQ-Modulator driving signals reach the electrodes, an ideal rectangular filter with a cutoff frequency of 25MHz was applied, in order to simulate an IQ-Modulator with a bandwidth equal to the driving signals one (after the performance of a Raised-Cosine filtering with a unitary roll-off factor). It is important to notice that both MZMs in the simulation have got infinite bandwidth, since they are ideal modulators. The spectrum of the driving signals is represented in figure 3.12:

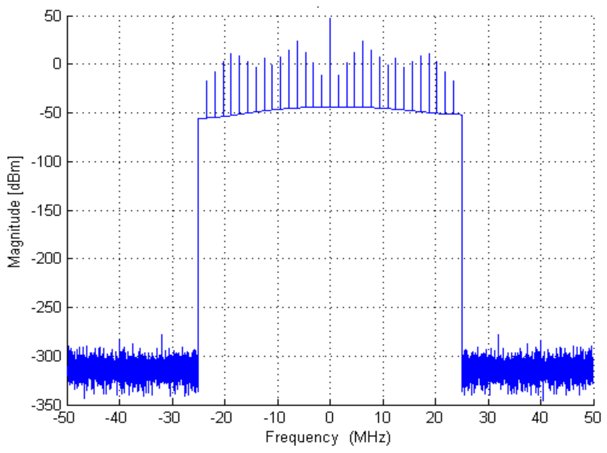


Figure 3.12 Simulation Results: Spectrum of the driving signals after ideal low-pass filtering

The optical spectrum of the transmitted signal has got some changes, as it can be seen in figure 3.13:

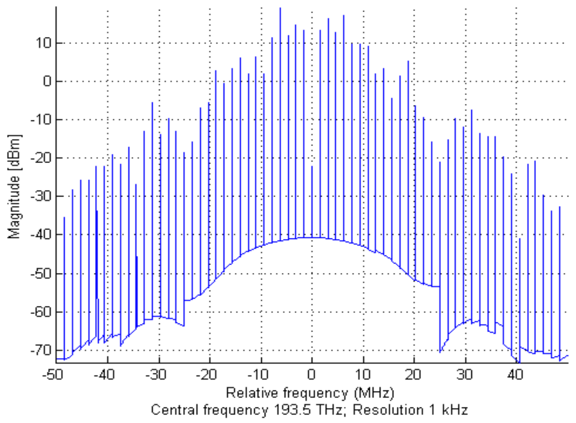


Figure 3.13 Simulation Results: impacts of the filtering in the resulting optical spectrum

As can be seen in the spectrum above, the power density in the rejection bands is now much stronger, and the spectrum is not quite acceptable. Therefore, the bandwidth condition for the IQ-Modulator is required just to ensure that the signal's frequency range in which the modulator will operate is the same range where the 'DSP' has operated. This is the only way to guarantee that the nonlinear effects will cancel each other, and the resulting optical spectrum will remain almost unchanged, similar to the original Raised-Cosine.

Several simulations where the bandwidth of the ideal rectangular filter was gradually increased were also performed. However, it was noticed that the secondary lobes of the transmitted spectrum (located beyond the cutoff frequency of the rectangular filter) appeared always distorted. Thus, it is possible to conclude that the modulator must have a bandwidth of at least half of the sampling frequency, which may be difficult to achieve at high bit rates.

3.2.2 Study Case: Constellation's phase distortions

Previously, on section 2.4.1, it was discussed the fact that a "dynamic gain" cannot be applied to the filtered samples if those unipolar signals were not firstly converted to a polar representation. In this section it is intended to simulate the situation where the "adaptive gain" is applied to the signals without any other signal processing first, and analyze the resulting constellations to confirm the expected distortions. Thus, the transmitter configuration is equal to the one represented in the figure 3.1. The only change occurs in the 'DSP' block, which is applying the "dynamic gain" factor to the incoming samples without acting in the DC level of the signals.

As explained before, the resulting constellation is only affected by distortions if the "dynamic gain" takes values lower than '1'. For example, if the gain is set to '0.9', the driving signals will begin to be decentered relatively to the middle point of the Mach-Zehnder Modulator's electrical field transfer function (the "Null Biasing Point"). Thus, the resulting constellation comes distorted, as figure 3.14 shows:

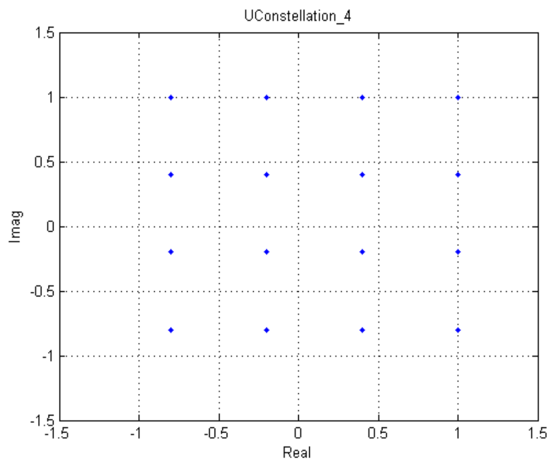


Figure 3.14 Simulation Results: Impact of the gain in the resulting constellation

The reason for this distortion is that the generated driving signals lead the modulators to produce unsymmetrical output electrical fields. Since the IQ-Modulator has got two Mach-Zehnder Modulators, this undesired effect will affect the resulting constellation. For example, if the gain is reduced for '0.6', the result should be even worse, which is shown in figure 3.15. This happens because the distortion on the symbol mapping will be more pronounced as lower is the value of the gain factor:

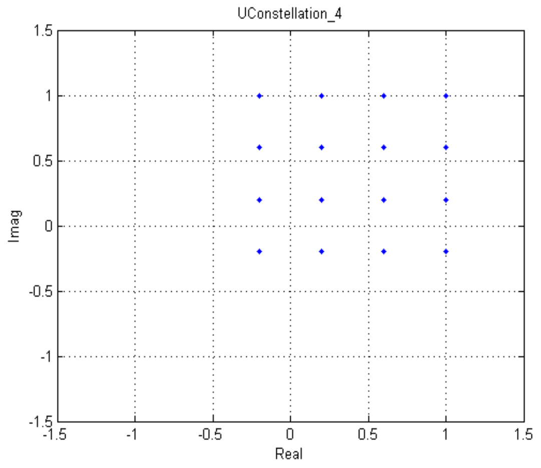


Figure 3.15 Simulation Results: Increase of the constellation's distortion

A solution to solve this problem is to convert the incoming unipolar signal to a polar representation, which is equivalent to remove the signal's DC level. Only then, the "dynamic gain" should be applied. This way, changing the value of the gain factor will only produce the respective change in the driving signals excursion. The resulting constellation will not be affected and will appear always centered, since the driving signals are biased in the null-point of the referred transfer function.

3.3 M-QAM Simulations

The nonlinear response of a Mach-Zehnder Modulator affects the generation of any M-QAM format. In this section, it is intended to prove that the predistortion algorithm is independent of the data modulation format that is going to be transmitted, always producing the correct results. Thus, 32-QAM and 64-QAM simulations were performed. The following table resumes the main parameters introduced in the OSIP simulations:

Block	*32-QAM Parameters*	*64-QAM Parameters*
PRBS	100Mbps	100Mbps
StoP	5 parallelized bits	6 parallelized bits
Coding	32-QAM constellation	64-QAM constellation
ReSample	Factor L=5	Factor L=6
LPF el	Raised-Cosine filter; Configurable Roll-off Factor	Raised-Cosine filter; Configurable Roll-off Factor
LaserCW	193.5THz; 1549.32 nm	193.5THz; 1549.32 nm
IQM	V_π=3v; V_{PM}=-3/2	V_π=3v; V_{PM}=-3/2

Table 3.2 32-QAM and 64-QAM Simulation parameters

The topology of the transmitter system is the same as before, presented in figure 3.1. In each simulation, a fixed binary sequence was created to ensure that the driving signals meet the requirements explained in section 3.2, allowing a future hardware implementation without having distortion effects produced by the output circuits of the DAC Evaluation Board. Simulating the systems without any digital signal processing applied to the filtered samples ("DSP" block is only applying the "adaptive gain", amplification and offset factors to the samples), the constellations obtained for 32-QAM and 64-QAM are sketched in figures 3.16 and 3.17, respectively.

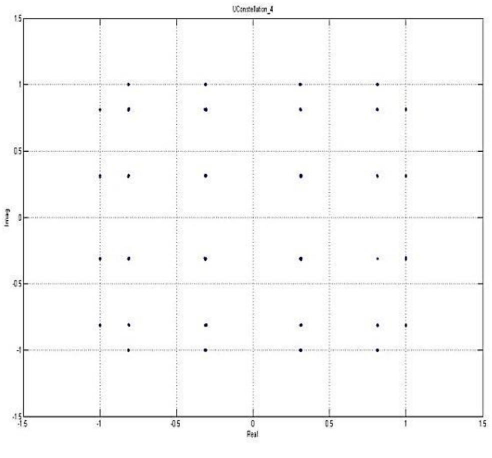


Figure 3.16 Simulation Results: Incorrect 32-QAM Constellation

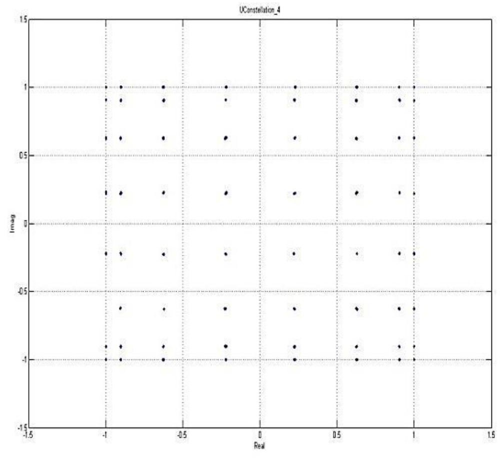


Figure 3.17 Simulation Results: Incorrect 64-QAM Constellation

The results show that the constellation's impairments are even more pronounced than on a 16-QAM simulation because the generation of these modulation formats requires driving signals with a higher number of different voltage levels. In order to compensate these problems, the 'DSP' block was configured to perform the predistortion algorithm. The constellations obtained are presented in figures 3.18 and 3.19:

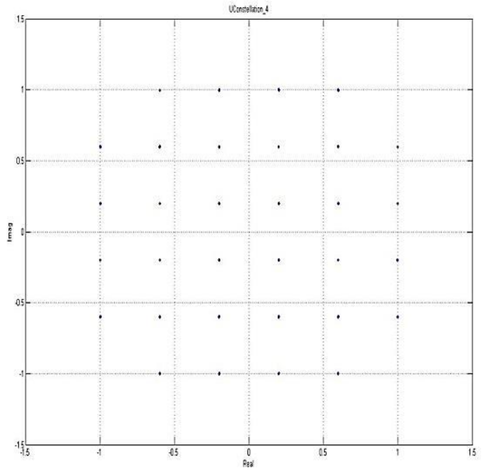


Figure 3.18 Simulation Results: Correct 32-QAM Constellation

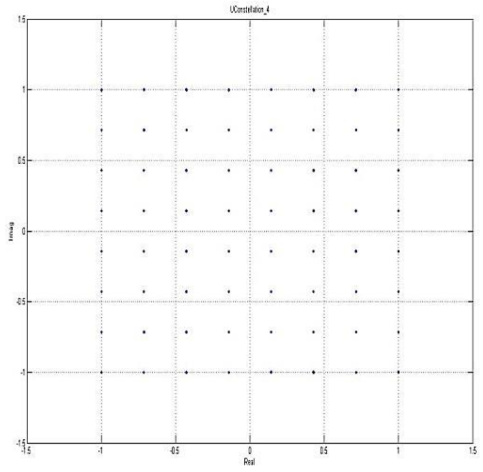


Figure 3.19 Simulation Results: Correct 64-QAM Constellation

The resulting constellations are correctly sized (all the adjacent symbols are equally spaced), proving that the 'DSP' allows to "simulate" the existence of a modulator with a linear transfer function. The results also prove that the predistortion algorithm is independent of the QAM format considered.

The optical spectra of the transmitted signals were also correct, having a well-defined "Raised-Cosine" shape, with high attenuated rejection bands. The figure 3.20 illustrates the results obtained for a 32-QAM simulation:

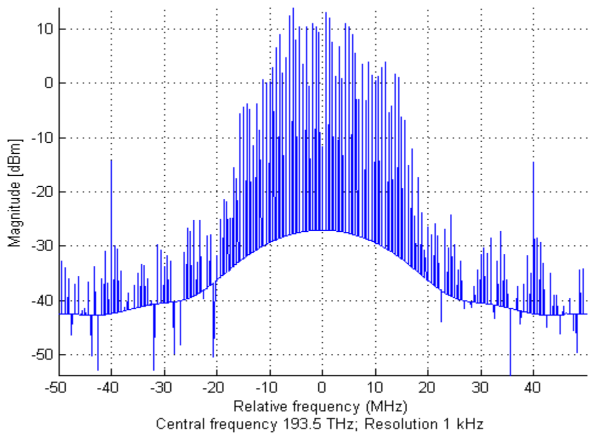


Figure 3.20 Simulation Results: 32-QAM Optical transmitted spectrum

These results also confirm what was stated before: the spectral distortion caused by the 'DSP' operation is compensated by the IQ-Modulator action, and the resulting spectrum is almost similar to the original "Raised-Cosine" one. The only condition that must be taken into account is the IQ-Modulator bandwidth, which should be enough to accommodate the range of frequencies necessary to represent the main and secondary lobes of the signal's spectrum. This condition ensures that the frequency range in which the modulator will operate is the same range where the 'DSP' has operated.

4. Hardware implementation

The purpose of this chapter is to identify some solutions to efficiently implement the digital part of an optical coherent transmitter system in a FPGA platform. Several features like complexity, synchronism, parallelism, resources utilization or speed performance will be discussed in order to ensure that the design will generate precise results.

Before discussing the details of the hardware implementation, an overview of the FPGAs structure and principles of operation is provided. This allows understanding how can the designer take advantage of the FPGA's architecture to create more effective solutions.

4.1 FPGA

A "Field Programmable Gate Array" is a complex and high-speed integrated circuit. The term "Field Programmable" is due to the fact that this product can be configured by the user to do a certain function right after has been installed in the field [41].

FPGAs are widely used in reconfigurable computing development, since they allow combining some of the software's flexibility with its own high performance and flexible hardware platform. Similar to an ASIC ("Application-Specific Integrated Circuit"), FPGA-based systems provide a method to map logical circuits into hardware, achieving great performances as a result of parallel execution of multiple operations and bypassing the fetch-decode-execute cycle of traditional microprocessors [42]. Furthermore, the ability to update the functionality after shipping offers advantages for many applications. Although ASICs provide full custom capability and smaller form factor (since the devices are manufactured to design specifications), FPGA-based designs also offer faster prototyping, lower risk, short manufacturing time and are cheaper for small and medium markets [41]. It is also possible to adapt the hardware during runtime through dynamic reconfiguration approaches. In fact, FPGA based-systems have a wide range of target applications, which include the telecommunications area, digital signal processing, network applications or medical solutions, for example [14].

An FPGA can be seen as an array containing thousands of logic blocks that can be programmed and interconnected as desired by the developer. Those fundamental blocks are called "CLBs" ("Configurable Logic Blocks") and its internal structure depends on the technology implemented by each manufacturer:

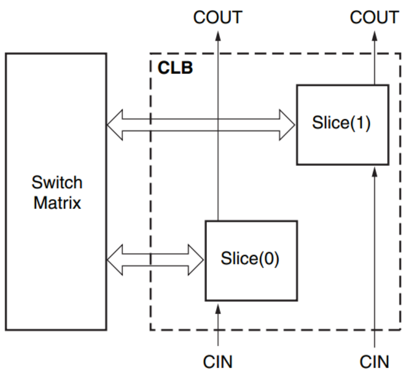


Figure 4.1 CLB example for a Xilinx Virtex- 6 [43]

Each CLB is connected to a "Switch Matrix", as depicted in figure 4.2, which ensures low-skew connections between those logic blocks.

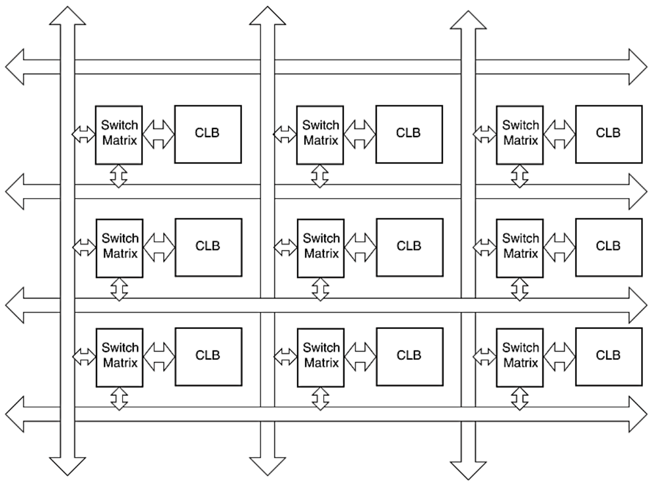


Figure 4.2 CLBs interconnection example

Generally, the CLBs arrangement in the board depends on the manufacturer. Despite the following descriptions are based in the "Xilinx Virtex-6" FPGA technology, all the other FPGA families have similar structures and follow identical principles of operation [43].

A CLB element usually contains a pair of slices that are not interconnected. As can be seen in figure 4.1, each slice is located in a column and has its own carry chain, as figure 4.3 shows. "Xilinx" also provides an identifier to each slice within an FPGA [43]. This will help the developer to perform a low-level analysis of its project, for example.

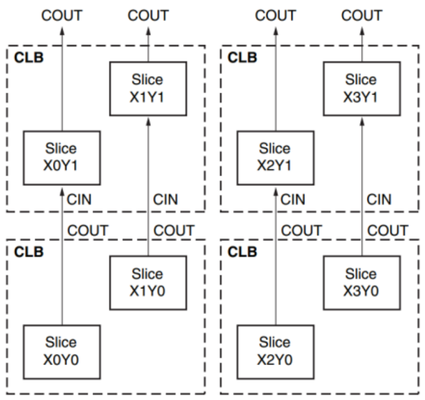


Figure 4.3 Row and Column Relationship between CLBs and Slices in Virtex-6 [43]

Each slice contains several "Look-Up Tables" (logic-function generators), storage elements, wide-function multiplexers and carry chain logic (see figure 4.4). However, some different types of slices may be available, depending on the additional supported features. In "Xilinx Virtex-6" FPGA, the slices that support storing data as distributed RAM and shifting data with 32-bit registers are called "SLICEM". All the others are called "SLICEL" [43].

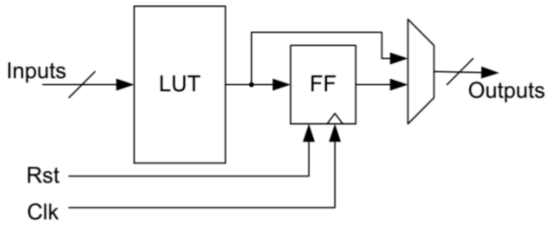


Figure 4.4 Simplified Block diagram of a generic slice

A "Look-Up Table" (LUT) is a digital memory that can implement any arbitrarily defined Boolean function of 'N' input variables. The 'N' inputs are used to address a '2^N x 1' bit memory that stores the truth table of the Boolean function. The propagation delay through a LUT is independent of the function implemented, which makes it a valid solution for implementing combinatorial functions [43].

All the LUTs present in "SLICEMs" can also perform other functions like acting as a 32-bit shift register or even as distributed RAM. The last one is called "distributed" because all the LUTs available in the CLBs can store information that can be addressed by 'N' input bits. This is an efficient way of storing data in a memory and having high-speed access to the stored information. However, this type of RAM should be avoided if there is a large quantity of information to be stored. There, the access times will cause undesirable delays, and a dedicated RAM (usually called "Block RAM") should be considered, since large quantities of data can be stored and accessed with high performances and dedicated functionalities [44].

The remaining elements that compose each slice present many features that make an FPGA more efficient and versatile. For example, carry chains and other additional logics provide the capability of executing arithmetic functions in an efficient way. The outputs of a CLB are also very flexible, since it is possible to use multiplexers for selecting the element of the slice that will be the source of the output data.

The information flow between the internal signals of an FPGA and other peripherals is performed through the "I/O Blocks" (IOB). This interface may be configured as input or output, and its basic structure is presented in figure 4.5 [45]:

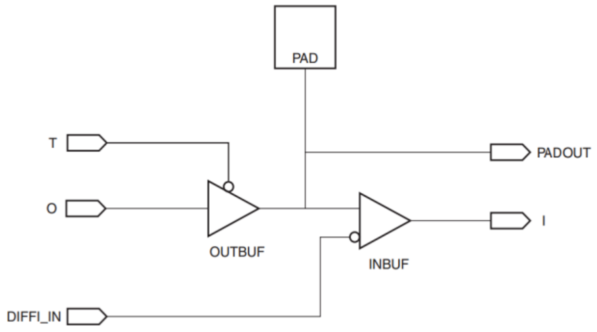


Figure 4.5 Basic IOB Block Diagram [45]

Different FPGA families can support a wide variety of standard interfaces: single-ended I/O standards like LVCMOS, HSTL and SSTL, PCI, I2C or differential I/O standards like LVDS, LVPECL, differential HSTL and SSTL, etc. Differential I/O standards generally use two IOBs grouped together in a single tile [45].

Technological advances allow the manufacturers to produce devices with more complex "I/O Blocks". They have got even more features, like the possibility to have a programmable control of output strength and slew rate, use of DDR registers, on-chip termination using Digitally Controlled Impedance (DCI), and the ability to internally generate a reference voltage, for example [45]. This shows the importance that these blocks have in current FPGAs, allowing the communication with external devices.

Today's manufacturers also include several dedicated slices in FPGAs, in order to improve their features and capabilities. For example, in the last years, FPGAs have played an important role in digital signal processing. Thus, manufacturers have designed and improved the usually called "DSP slices". The basic feature of these slices is to implement a MAC (Multiply and Accumulate) operation: the product of two numbers is computed and added to an accumulator block. However, depending on the FPGAs' families, the "DSP slices" may have many other features, like multiply-add, magnitude comparators, bit-wise logic functions, pattern detectors, wide counters, etc. Several architectures, like "Xilinx Virtex-6", also support cascading multiple "DSP48E1" slices to form wide math functions, DSP filters or complex arithmetic without the use of general FPGA logic [46]. The architecture of the DSP48E1 slice is shown in figure 4.6. This dedicated block will be used to implement some steps of the project, as will be explained later.

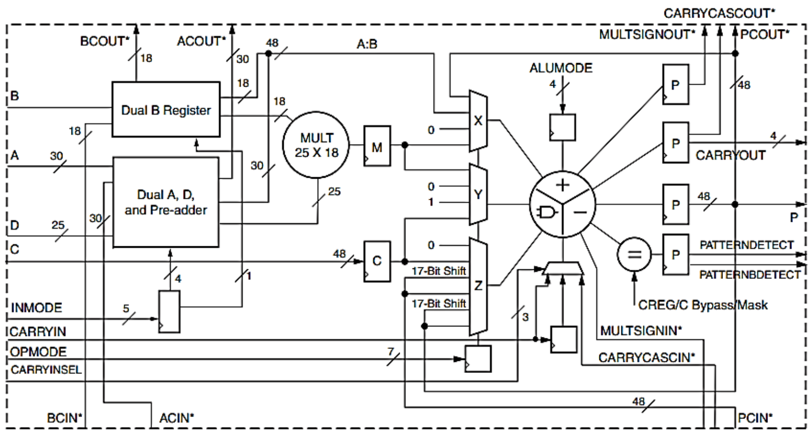


Figure 4.6 Xilinx Virtex-6 DSP48E1 slice general schematic [46]

Although there are several dedicated blocks that could be explored in this text, the "Clock Management and Distribution" network will be the last subject to be studied, since it plays an important role in the FPGA design that is going to be discussed later.

Almost all medium or high complexity FPGA designs require multiple clocks to process data at different working rates. Thus, modern FPGAs provide several clocking resources, allowing the creation of complex systems. Usually, those resources are classified as "Global", "Regional" or "I/O" clocks.

Global Clock Lines are designed to clock all sequential resources in the entire device. These lines are usually driven by a global clock buffer, which can also be used as a clock enable circuit, or a glitch-free multiplexer [47]. They also guarantee a low-skew clock, low duty-cycle distortion and improved jitter tolerance. In Xilinx Virtex-6, the global clock buffers are usually driven with a Clock Management Tile (CMT) to eliminate the clock distribution delay, or to adjust its delay relative to another clock [47].

Regional clocks are a set of differential clock networks independent from the global clock network. They are restricted to some regions, whose size depend on the FPGA technology and will define the number of supported clock domains [47]. Also the I/O Banks placement and its relationship with the regional clocking resources will depend on the FPGA technology. In figure 4.7, the Global Clocking architecture of "Xilinx Virtex-6" device is shown:

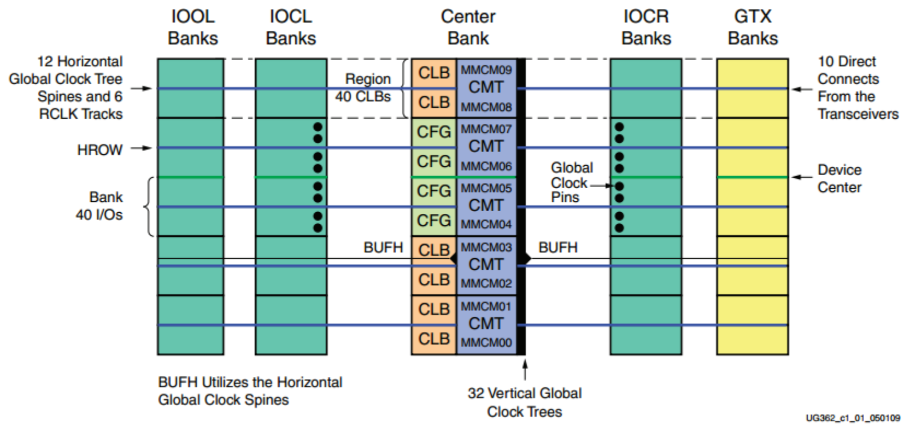


Figure 4.7 Xilinx Virtex-6 Global Clocking architecture [47]

Each global and regional clock trees are driven by specific buffers. There are also clocking modules that perform the clocking network management. For example, in Virtex-6, each one of the nine CMTs (Clock Management Tiles) contains two MMCM (Mixed-Mode Clock Manager) Modules that work as frequency synthesizers for a wide range of frequencies and as jitter filters for either external or internal clocks [48]. An MMCM takes input clocks and generates several output clocks. Each one can be configured to have a different frequency, which is always dependent on the input clock rate and the frequency of the "Voltage-Controlled Oscillator" (VCO) [48, 49].

4.2 Evaluation Kit

The hardware implementation of the previously studied coherent transmitters will be performed with the "Xilinx ML605 Evaluation Board" (device "XC6VLX240T" and package "1FFG1156"). This evaluation kit includes a "Xilinx Virtex-6" FPGA chip and many other peripherals that facilitates the configuration and communication with the corresponding chip. ML605 also provides some key features common to many embedded processing systems [50]:

- Xilinx FPGA Virtex-6 XC6VLX240T-1FFG1156;
- Power supply through transformer;
- Support for several configuration modes: Slave SelectMAP, Master BPI-Up and JTAG (using USB cable or via CompactFlash card);
- 512MB DDR3 Memory SODIMM;
- 128Mb Platform Flash XL;
- 32MB Linear BPI Flash;
- Fixed 200MHz differential oscillator;
- Socketed 2.5V oscillator (single-ended), at 66MHz;
- SMA connectors for external clock or user GPIO;
- Multi-Gigabit Transceivers: FMC connectors, SMA, PCI Express, Ethernet…
- LCD display, Status LEDs, Pushbuttons, Switches and other User I/O;

An overview of the evaluation board is illustrated in figure 4.8:

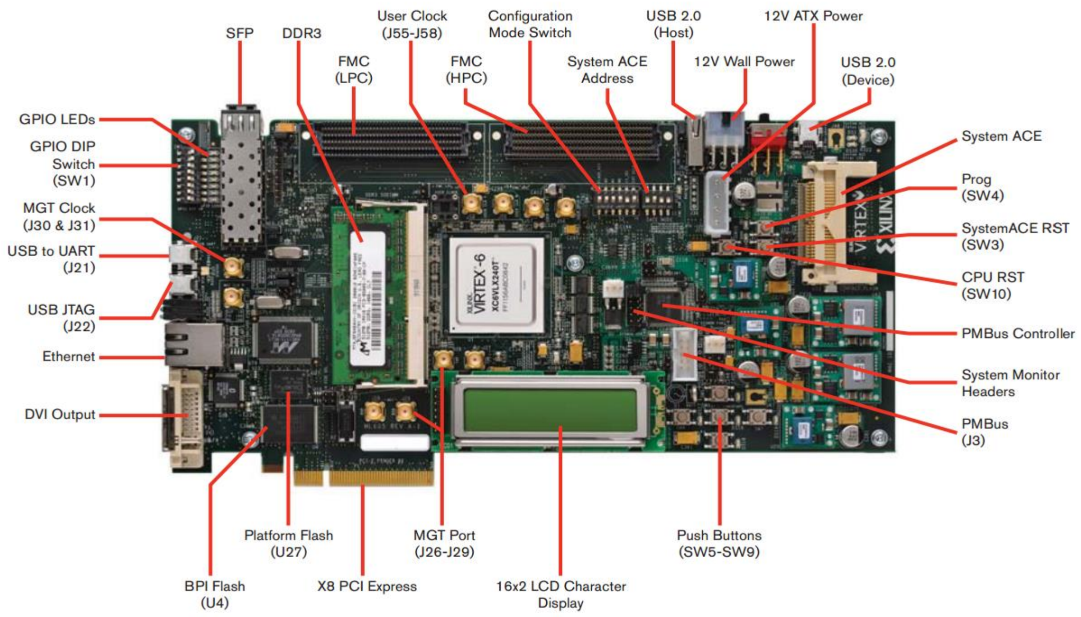


Figure 4.8 Xilinx ML605 Evaluation Board overview [51]

This evaluation board uses a “LXT” sub-family of “Xilinx Virtex-6 FPGAs”, providing high performance logic with advanced serial connectivity. Each sub-family contains a different ratio of features to most efficiently address the needs of a wide variety of advanced logic designs [52]. The table 4.1 resumes the most relevant features of the Virtex-6 FPGA [52]:

Logic Cells	**CLBs**		**DSP48E1 Slices**	**Block RAM Blocks**			**MMCMs**	**Ethernet MACS**	**Interface Blocks for PCI Express**	**Maximum Transceivers**		**Total I/O Banks**	**Max User I/O**
	Slices	Max Distributed RAM		18 Kb	36 Kb	Max (Kb)				GTX	GTH		
241,152	37,680	3,650	768	832	416	14,976	12	4	2	24	0	18	720

Table 4.1 Most relevant features of the Virtex-6 FPGA

Each CLB is formed by two slices and the corresponding carry chains. Each slice has got an identifier and is composed by four Look-Up Tables, eight Flip-Flops and multiplexers. The LUTs can be configured as either one 6-input LUT with a single output, or as two 5-input LUTs with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a Flip-Flop. Finally, four of those Flip-Flops can optionally be configured as latches, which imply that the remaining four flip-flops in the slice must stay unused [52].

4.3 Implementation of the coherent transmitter system

In this work, 16-QAM and 32-QAM transmitter systems were implemented. Both projects were developed using VHDL. More details about this hardware description language are given in Appendix 1. Furthermore, a brief approach to the software tools used to execute some data processing and perform debugging techniques is provided in Appendix 2.

Since the transmitters' structure is similar, the following discussions will be centered in the 16-QAM project. However, details about the 32-QAM implementation will be given whenever it is necessary. The simplified block diagram presented in figure 4.9 intends to illustrate the main VHDL modules that compose both coherent transmitter systems:

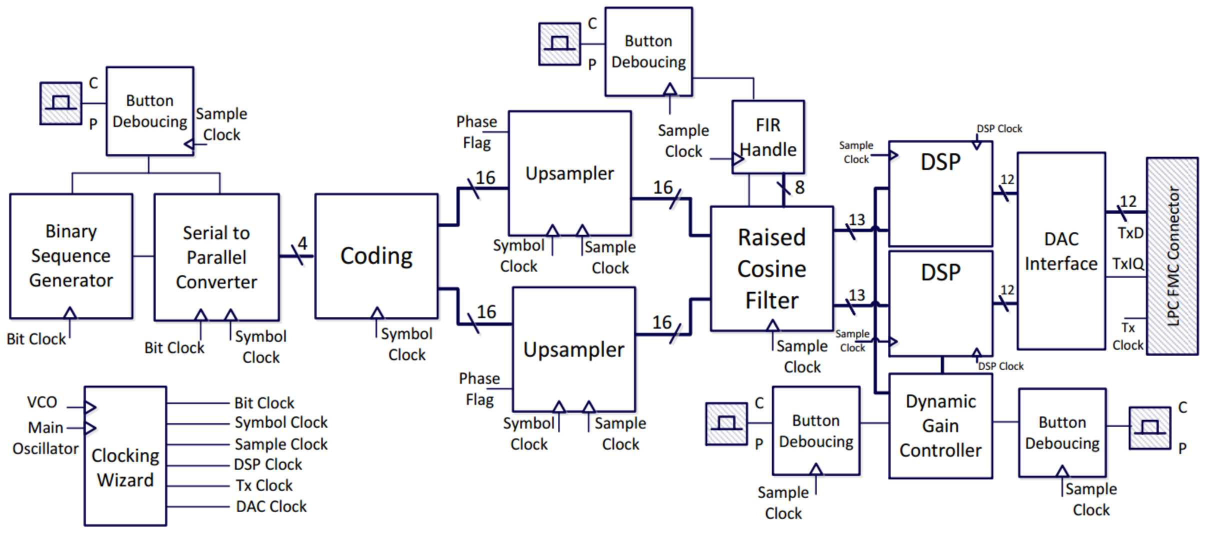


Figure 4.9 Simplified Block Diagram of the transmitter system implemented in FPGA

4.3.1 Clock generation

The different blocks that compose the transmitter system must read or generate data at different rates. Thus, it is necessary to generate different clock signals to make the entire system work properly. However, despite having different frequencies, those clocks must be synchronous. If that does not happen, some data may be loss (due to wrong sampling instants) and incorrect results may be produced.

In fact, it is important to guarantee that those multiple clocks are generated from the same source and are distributed through specific FPGA clocking lines to minimize the clock skew issue, which makes a clock signal switch at different instants in different parts of the system, depending on the distance from the clock source. Thus, in addition to generate synchronous clocks, it is also a good practice to use dedicated buffers (global buffers) to route clock signals in an efficient way [47].

The "Xilinx LogiCORE™ IP Clocking Wizard v1.8" was the solution found to implement a customized clock circuit in an efficient way. Intellectual Property (IP) refers to preconfigured logic cores designed by "Xilinx", which offer a more efficient implementation of a certain function, since they were designed and optimized for “Xilinx” FPGAs and Evaluation Boards. Thus, this IPCORE accepts up to two input clocks and generates up to seven output clocks. It allows the user to configure the buffering, feedback and timing parameters for the clocking network. All the frequency, phase shift and duty-cycle requirements are implemented based in the input clocks, in the “Voltage-Controlled Oscillator” frequency and in multiply and divide counter values calculated by the “Clocking Wizard” [47, 53].

In this project, the input clock source will be the 200MHz main oscillator. The six created output synchronous clocks are:

- “Bit Clock”: this clock signal has got the frequency of 100MHz and defines the working rate of the “Sequence Generator” block, where binary data is generated.
- “Symbol Clock”: defines the rate at each new parallelized symbol is available: 25MSymbols/s in 16-QAM transmitter and 20MSymbols/s in 32-QAM.
- “Sample Clock”: 100MHz clock that defines the rate at each new sample is available.
- “DSP Clock”: 400MHz clock that defines the working rate of “DSP” Modules.
- “TX Clock”: clock with a working rate two times higher than the sampling clock. Allows data interleaving for the DAC’s interface.
- “DAC Clock”: defines the DAC sampling rate. For a unitary interpolation factor, this clock has got the same frequency of the sampling clock.

All the output clocks were buffered with the “BUFG” primitive (Global clock buffer), allowing them to be routed in the global clocking nets and minimize the clock skew issue.

Another possible solution for the clock generation was to implement a “Synchronous Clock Divider” VHDL Module. It could be created simply by having an input clock source (the 200MHz oscillator) and associating a counter and a variable called “clock divider factor” to each one of the output clocks. This variable could define the factor by which the main clock source would be divided, in order to generate lower frequency clocks. In a general way, if a ‘reset’ button was pressed by the user, all the output clocks would get a unitary logic value and the counters would be set to zero. After that, the “Synchronous Clock Divider” could start its normal dividing procedure. In every rising edge of the input clock, the counters associated to each output clock would be incremented. When a counter reached half of the “clock divider factor” value, the corresponding output clock would get an opposite logic level. Furthermore, if the counter reached the final value of the respective “clock divider factor”, the associated output clock would get an opposite logic level again, and the counter would be restarted to a null value again. This procedure ensures the correct generation of different clocks, all of them synchronous.

However, the “clock divider factors” must be integer values, hindering the generation of certain duty-cycles or output clock frequencies. Furthermore, if “Timing Constraints” are applied to a clock signal that is going to be translated through a clock modifying block (as MMCMs), the ISE design tool will constrain the generated output clocks with new constraints. This is an advantage of using the “Clocking Wizard” IPCORE over the “Synchronous Clock Divider” solution: all the system clock domains will be automatically constrained by the ISE Implementation tool if a timing constraint is applied to the input clock source [54].

"Timing Constraints" communicate all the project timing requirements to the implementation tools. Thus, it is important to understand the performance requirements of the system and the features of the target device. This knowledge allows the user to use proper coding techniques utilizing the features of the device to achieve the best performance [54]. In the system that will be described below, several clock domains and synchronous hardware elements will be related. Thus, it will be important to create constraints that will cover the synchronous data paths, in order to ensure that the entire system works properly. The "PERIOD" constraint is included in the "Register-to-Register Timing Constraints" category and will be applied to the entire design. It will define the timing requirements of the system and analyze all the paths between related clock domains, taking into account all frequency, phase and uncertainty differences [54]. This timing constraint is defined in the UCF file of the project and will be applied to the 200MHz oscillator, which is the input clock of the "Clocking Wizard" IPCORE. Thus, as was previously explained, the Xilinx ISE Implementation tool will automatically derive a new "PERIOD" constraint to every MMCM output clocks. It will also determine the relationships between the clock domains and perform an analysis for any existent paths between these domains [54]. Thus, the implementation process performed by the ISE tool will map and route the design into specific FPGA slices in order to meet the timing requirements.

After the design implementation, the ISE tool summarizes the "Timing Report", allowing the designer to verify if the timing constraints were met. If some errors occur, it is possible to analyze which are the failing paths covered by the constraint. Depending on the type of failure, different solutions may be adopted. During the design of this system, the timing issues were solved by increasing the parallelism of the troubled paths.

4.3.2 Binary Sequence Generator

In the previous chapter, all the OSIP simulations were performed with a finite number of generated bits. However, in a hardware implementation, the visualization of the desired results will only be possible if the transmitter is operating in "real-time". Thus, it is necessary to create a system where binary information is continuously being generated and processed.

In this work, it is not intended to encode and transmit any specific information. However, as explained in the previous chapter, both 16-QAM and 32-QAM transmitters must generate a specific fixed binary sequence, created to ensure that the produced driving signals meet several requirements, avoiding the distortion effects produced by the output circuits of the DAC Evaluation Board. Thus, the main function of the "Binary Sequence Generator" Module is to generate a specific binary sequence that will be repeated indefinitely over time.

The most efficient way to generate the serial binary data is by creating a "Look-Up Table" with a number of entries equal to the number of bits that compose the sequence. This table will store the studied sequence. Then, a counter will be incremented in the rising edges of the "Bit Clock", defining the index of the table that will be accessed. Therefore, the content of the entry that is addressed by the counter will be synchronously provided to the next Module in the chain: the Serial-to-Parallel Converter. This VHDL Module also includes an enable button which allows the user to start the sequence generation in a synchronous way.

It is important to notice that this table will not be physically implemented by the ISE tool in a LUT with sixty four entrances, since there are no LUTs of that size in the device. The ISE Implementation tool will map the table into several LUTs, accordingly to the previously described technology employed in the Xilinx Virtex-6 FPGA. For example, the 64-bit sequence of the 16-QAM transmitter will occupy eleven of the 6-input LUTs with a single output available in the FPGA.

4.3.3 Serial to Parallel Converter

Considering that the serial output of the "Binary Sequence Generator" Module provides a new bit in the rising edges of "Bit Clock", the "Serial to Parallel Converter" block must receive those bits and parallelize them. The number of parallelized bits should be in accordance with the constellation that will be mapped in the "Coding" block. In a 16-QAM transmitter, this module must have a 4-bit parallel output, as represented in figure 4.10.

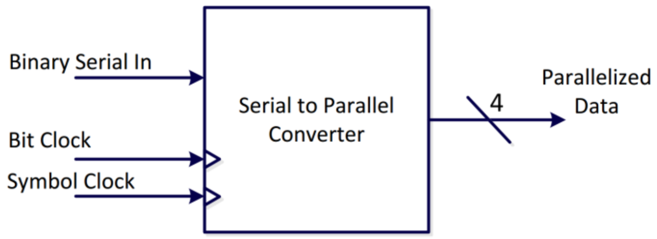


Figure 4.10 "StoP" for a 16-QAM Coding Block

This block can be efficiently implemented using a Shift-Register in "Serial-In, Parallel-Out" configuration, as figure 4.11 shows:

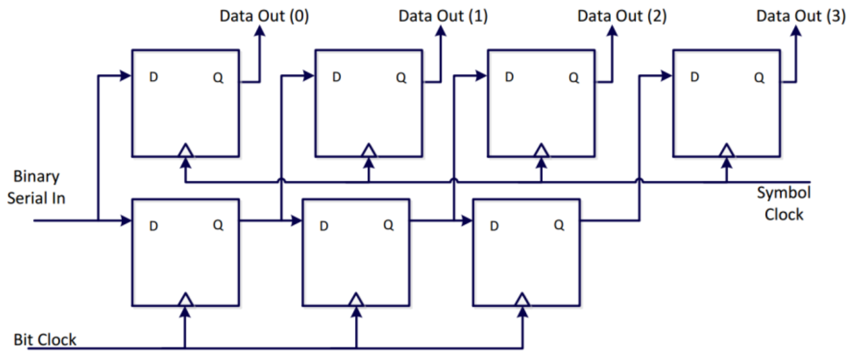


Figure 4.11 "StoP" for a 16-QAM Coding Block: Shift Register

This Shift-Register has its serial input connected to the output of the "Binary Sequence Generator" block. In the rising edges of the "Bit Clock", the input of the first Type-D Flip-Flop will be updated with a new incoming bit, and the information stored in the remaining Flip-Flops will be "right-shifted" at the same time. This synchronous update is possible since the cascade of Flip-Flops shares the same clock.

However, the “Serial to Parallel Converter” output data should not be always available. After a valid parallelized word, the new parallelized word must only be available after four complete periods of the clock (in a 16-QAM transmitter). Thus, the most efficient way to synchronize information and validate the output of this block is to have a second clock, called “Symbol Clock”, with a frequency four times lower than the “Bit Clock”. This solution ensures that the output of the “Serial to Parallel Converter” becomes valid in the rising edges of the “Symbol Clock”, i.e., in every four cycles of “Bit Clock” signal. Similarly, in a 32-QAM transmitter, the “Symbol Clock” signal must have a frequency five times lower than the “Bit Clock”, since the parallelized words will composed by five bits.

The “Clock Wizard” clock generator ensures that these two clocks are synchronized. However, the “Serial to Parallel Converter” must also take into account the importance of the “Binary Sequence Generator” enable signal. The corresponding “push-button” can be pressed at any time, which can lead to the generation of incorrect parallelized words. Thus, the “Serial to Parallel Converter” should only start operating after the serial binary sequence started being generated, that is, after the enable button being pressed.

4.3.4 Coding

The “Coding” block will map the parallelized bits into symbols of a specific constellation, accordingly to the Grey’s Code. This mapping is achieved through conversion tables, which make the Code’s alteration easier. The block will have two outputs, in order to forward information to both branches (‘I’ and ‘Q’) of the transmitter.

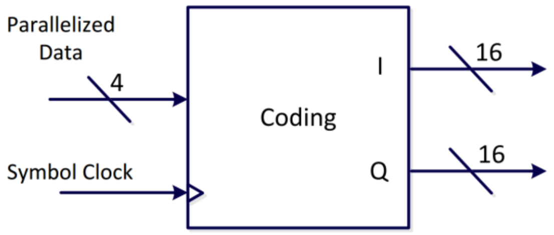


Figure 4.12 16-QAM Coding Module

On the rising edges of the “Symbol Clock”, this module will receive a parallelized word from the “Serial to Parallel Converter” and generate the corresponding two output “voltages”. The most efficient way to make this convertion is by creating two “Look-Up Tables”, each one relative to one arm of the transmitter. Both tables should be composed by a number of positions equal to the number of constellation symbols. In 16-QAM, for each incoming 4-bit symbol (which will be the index of the “Look-Up tables”), both tables will be simultaneously accessed and output the two “voltages” associated with the symbol. Naturally, the converted data should be available at the outputs on the rising edges of the “Symbol Clock”, in order to ensure the synchronism of the transmitter system.

Each position in those tables contains 16-bit words that represent the “voltage” level that should be forwarded to each branch of the transmitter. Those values are comprised between zero and one volt and represent the projection of the symbols into the ‘I’ and the ‘Q’ axis of the complex

IQ-plane. As it will be seen later, the filtering block will only process data with a bitwidth multiple of a byte. Thus, in those tables, 13-bit words represented in the Q1.11 2's complement format (one signed bit, one integer bit and eleven fractional bits) are concateneted with three null most significant bits, in order to output 16-bit words, ready for filtering [55].

4.3.5 "Phase Flag" Signal Generator

This VHDL Module generates a signal (called "Phase Flag") that will be updated in the rising edges of the "Sampling Clock". The "Phase Flag" signal is totally synchronous with all the clocks of the transmitter and its value repeats itself after a certain number of complete periods of the clock. As will be discussed in the next section, this signal will act in the "Upsampling" block as a 'flag', providing an easy way of inserting null samples between two consecutive 16-bit words generated by the "Coding" block. For example, in a 16-QAM transmitter, the "Phase Flag" signal will be composed by two bits, and its value will be repeating itself after four complete periods of the "Sampling Clock, helping the "Upsampling" Module performing the upsampling by a factor of '4'.

4.3.6 Upsampler

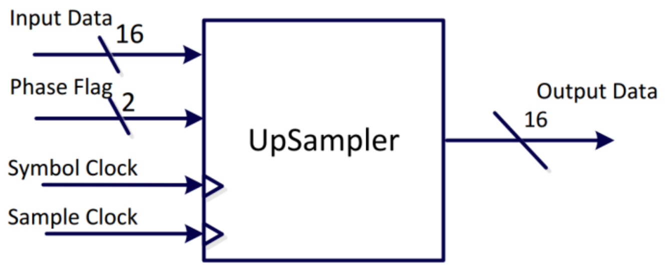


Figure 4.13 16-QAM UpSampler Module

On the rising edges of the "Symbol Clock", 16-bit samples are provided by the "Coding" block to both arms of the transmitter. However, before the filtering stage, it will be necessary to perform an upsampling process. Thus, both "Upsampler" Modules will insert 'L-1' null samples between two consecutive original samples, increasing the signal's rate by an 'L' factor. Therefore, the upsampled data will be available at the output of this block at the rate defined by the "Sampling Clock":

$$f_{Sample\ Clock} = L \cdot f_{Symbol\ Clock} \tag{4.1}$$

The insertion of 'L-1' null samples between two consecutive original samples must be done in a synchronous way. The "Phase Flag" signal will play an important role in the process, as shown in figure 4.14:

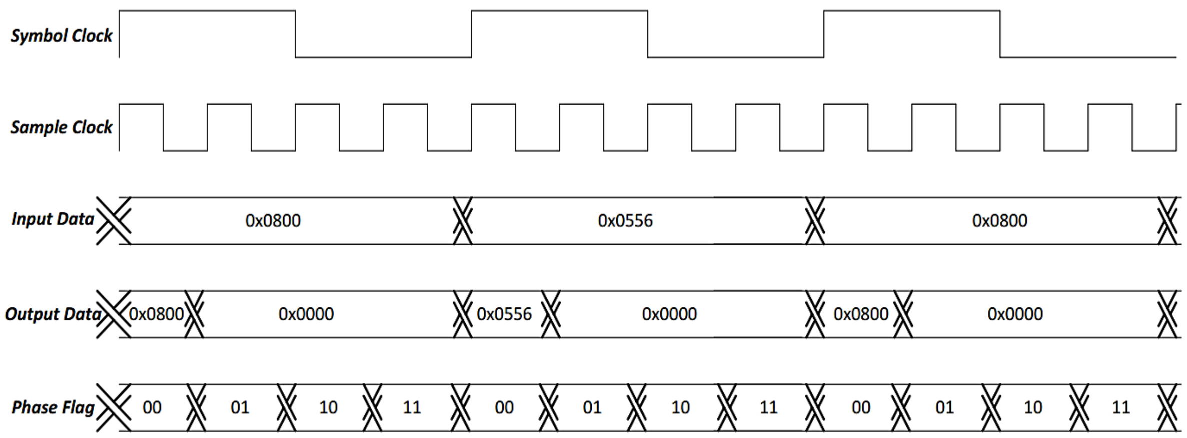


Figure 4.14 16-QAM Upsampling Process by L=4

When the “Phase Flag” signal has got a null value, each “Upsampler” Module outputs the 16-bit sample provided by the “Coding” block. In the three remaining values of the ‘flag’, null samples are synchronously provided to its “Output Data”.

4.3.7 Raised-Cosine Filter

Before being processed in the predistortion block, the upsampled signals from both arms of the transmitter should pass through a filtering process. In this stage, a Raised-Cosine filter will be designed based on the structure of a FIR filter, since it is usually composed by multipliers, adders and delay elements, which are particularly useful for hardware implementations.

The “Xilinx LogiCORE™ IP FIR Compiler v6.1” was the solution found to implement the filter in an efficient way [56]. During the CORE generation, using the “CORE Generator Graphical User Interface” tool, it is possible to define all the desired characteristics for the FIR filter, while its frequency response is simultaneously visualized. The first point that must be defined is the working rate. The clocking source of this single-rate filter will be a 100MHz “Sampling Clock”, which allows the sampling of the input data at the same rate.

The filter coefficients must be supplied to the “FIR Compiler” using a file with a “.coe” extension. This file must be created using the structure shown in figure 4.15:

$$\begin{array}{l} radix = coefficient\_radix; \\ coefdata = \\ a(0), \\ a(1), \\ a(2), \\ \ldots \\ a(N-1); \end{array}$$

Figure 4.15 FIR Filter Coefficient file format [56]

The “radix” field was set to “10”, indicating that the filter coefficients were introduced in a base-10 representation. These $a(n)$ coefficients were inserted as real numbers and were obtained using the previously described “Matlab FDA Tool”. Their symmetry can also be exploited to save resources in the DSP48E1 module of the FPGA [56].

Then, the coefficients must be quantized and represented in binary fixed-point, manageable in hardware. They will be converted into Q1.11 format (13-bit 2´s complement value with one signed bit, one integer bit and eleven fractional bits), which it is the same fixed-point representation of the input data. The common fixed-point representation of the coefficients and the input data allows achieving precise results, without using too much space in the DSP48E1 module of the FPGA. The use of more fractional bits in the coefficient quantization will occupy unnecessary resources, since the filter’s output will always be truncated to thirteen bits.

The VHDL Top Module that will implement the Raised-Cosine filter will have the general structure shown in figure 4.16:

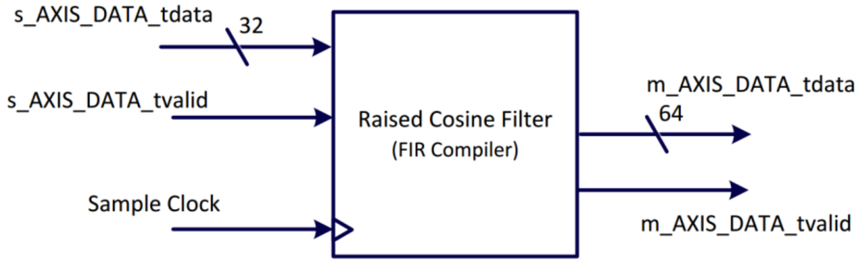


Figure 4.16 16-QAM FIR Compiler Module

The input flag named “s_AXIS_DATA_tvalid” indicates if the data available at “s_AXIS_DATA_tdata” is ready for processing. The processed data is provided in “m_AXIS_DATA_tdata”, and it is validated by the CORE through the flag “m_AXIS_DATA_tvalid”.

Some additional logic must be created to control the data flow in the filter. Since the transmitter has got two different paths (‘I’ and ‘Q’), the filter can process those two inputs using the same implementation, and filter the paths with the same filter configuration. This multi-channel filter implementation is very efficient in terms of resource utilization. A filter with two or more channels can be implemented using similar amounts of logic resources of a single-channel version of the same filter, with proportionate increase in data memory requirements [56]. Thus, each new 16-bit samples provided by both “UpSample” blocks must be concatenated, originating a single 32-bit word. This word must be sent to the filter in the rising edges of the “Sampling Clock”, keeping the system synchronous.

The full precision output width can be defined as the input data width plus the bit growth due to the application of the filter coefficients, as represented in figure 4.17:

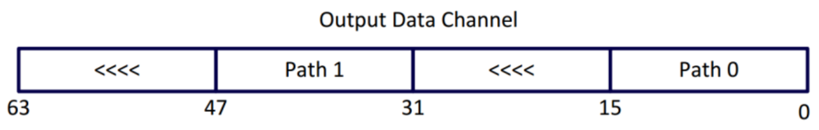


Figure 4.17 Bit Growth for each channel of the FIR Compiler

Bit growth from the original sample width occurs as a result of the many multiplications and additions that form the basic function of the filter. Therefore, the accumulator result width is significantly larger than the original input sample width [56]. The valid output provided in "m_AXIS_DATA_tdata" will have sixty four bits. The desired thirteen bits for each 'I' and 'Q' output must be selected and provided to the next block in the chain by the referred additional logic block.

The good functioning of the filter and its additional logic must be tested. One way to verify if the filtering is being correctly performed is to obtain the impulse response of the filter. Thus, a single 13-bit word representing the digital voltage of one volt, concatenated with three null most-significant bits was sent to each input of the filter during a single period of its clocking source. The expected outputs were the coefficients specified in the ".coe" file, i.e., its impulse response. The period of time between the unitary input impulse and the output of the desired coefficients is called "latency" of the filter. This value will not affect the system performance and will just define the delay between input and output data.

In the laboratorial experiments, it will be useful to change the "roll-off" factor of the Raised-Cosine filter and see what will be the impact of that change in the results. However, a change in the "roll-off" factor implies an alteration in the coefficients of the filter. Thus, it will be useful to find a way of implementing the filter reconfiguration without have to recompile the entire VHDL design everytime the "roll-off" factor must be changed.

The "Config" control channel of the CORE allows the specification and selection of multiple filter coefficient sets. When this feature is enabled, multiple coefficient sets may be specified in the ".coe" file. For selecting a certain set, a binary index must be provided to the CORE via "s_AXIS_config_tdata", and validated via "s_AXIS_config_tvalid". The "s_AXIS_config_tdata" signal will be zero padded to an 8-bit boundary. This feature was chosen due to its simplicity and efficiency. In this project, it is intended to store only about ten or twenty filters. However, other applications may require much more filter coefficient sets and each set may be composed by a larger number of coefficients. This will require more storing space and the "Config" control channel may not be a valid solution for those cases. The entire user interface for the selection of specific filter coefficient sets is represented in figure 4.18:

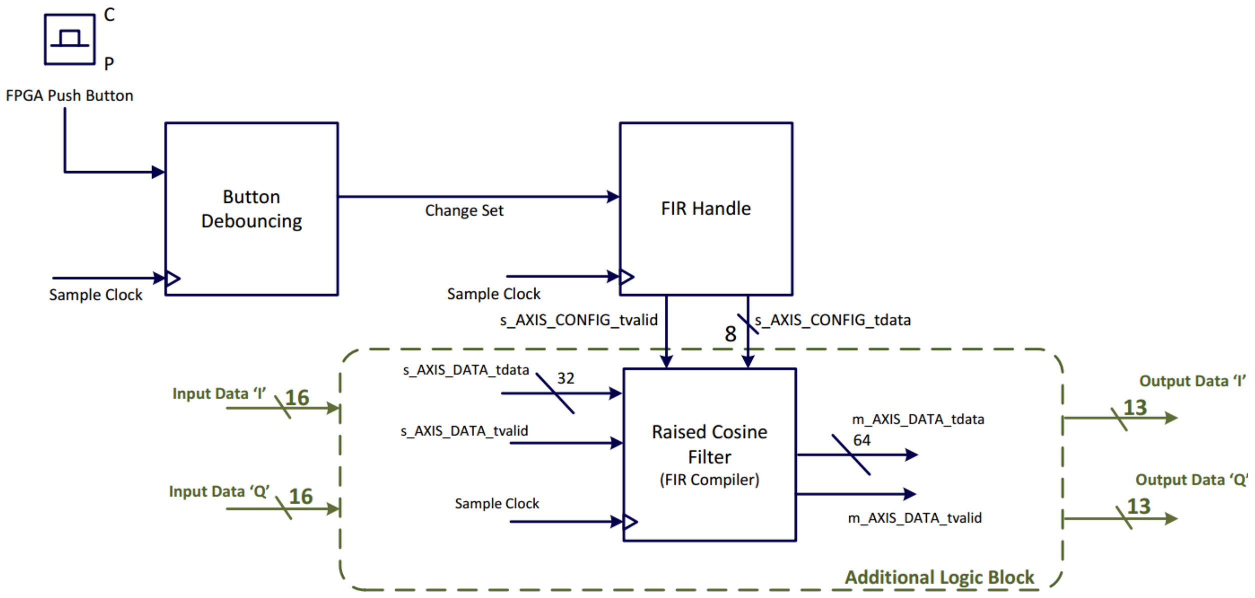


Figure 4.18 16-QAM FIR Filter interface

The user should press the correct push-button in the ML605 Evaluation Kit to promote a change in the "roll-off" factor of the Raised-Cosine filter. However, when the push-button is actuated, the contacts often bounce, leading to a temporary ambiguity in the effective logic level. The method used to solve this typicall issue in asynchronous hardware interface is shown in figure 4.19:

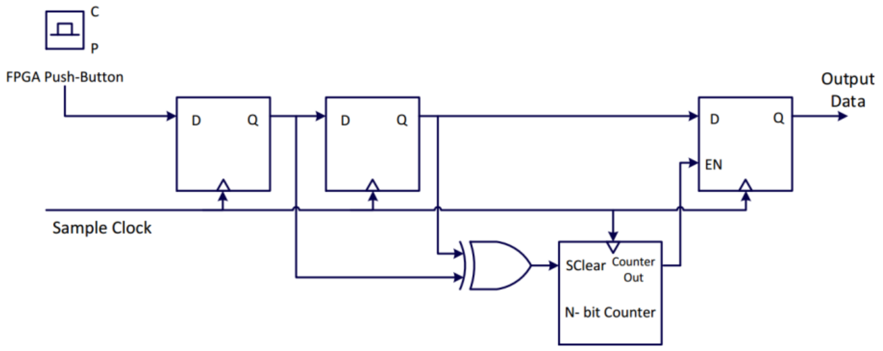


Figure 4.19 Button Debouncing Block: Simplified schematic

The logic level defined by the push-button is sampled at the sampling rate of 100 MSPS and is stored in the first Flip-Flop of the chain. So, the last two samples are stored in the first two Flip-Flops. When the logic level changes, the outputs of the first two Flip-Flops will differ for a clock cycle and the "XOR" gate produces a positive logic level, clearing the N-bit Counter. If the button's level remains unchanged, the "XOR" gate releases the counter's synchronous clear and the counter begins to count. When it reaches a specific value will enable the third Flip-Flop, and the logic level defined by the push-button will be provided to the output. The circuit remains in this state until a different value is clocked into the first Flip-Flop, clearing the counter via "XOR" gate. The counter's size and the clock frequency will determine the time required to validate the button's stability.

The “FIR Handle” block will define which coefficient set (specified in the “.coe” file) will be activated by sending the corresponding index to the filter, through the signal “s_AXIS_config_tdata”. In the rising edges of the “Sampling Clock”, the output of the “Button Debouncing” block will be synchronously read. If its logic level is positive, the index of the filter coefficient set will be incremented, padded to achieve the 8-bit boundary and synchronously provided to the filter. If the last index was reached, the next increment will select the first index again, providing a circular selection of the index. This allows the use of a single button of the Evaluation Kit, saving the remaining buttons for other important functions.

The push-button will be pressed asynchronously by the user, and the signal provided by the “Button Debouncing” block will last for several “Sampling Clock” cycles. This will lead to multiple index changes for a single time pressed button. To avoid this undesired situation, a “Disabled Mode” was implemented inside the “FIR_Handle” module. If the push-button is pressed, the index is incremented and an internal flag is activated, avoiding any other index increments. This flag will start the “Disabled Mode” process, where an n-bit counter will be incremented at the “Sampling Clock” rate. The flag will only be disabled when the counter reaches its final value. The counter’s size determines how long the system ignores the output of the “Button Debouncing” block. In this project, a 24-bit counter was used, generating a “Disabled Mode” of 0.15 seconds.

In a 32-QAM transmitter, the structure of the designed hardware remains the same. However, new filter coefficient sets were calculated and introduced in the “.coe” file, since the Symbol Rate will be lower than in the 16-QAM.

4.3.8 DSP

This VHDL Module will be present in both arms of the transmitter and aims to perform some digital signal processing to the samples provided by both outputs of the FIR filter. The predistortion process intends to guarantee a linear response of the IQ-Modulator in the previously defined region of interest of its characteristic.

The application of the “Dynamic Gain” to each 13-bit sample will be the first step in the processing chain. Before applying a specific gain, the incoming unipolar digital signal must be converted to a polar representation. All the required mathematical operations will be implemented using a two’s complement representation in Q1.11 binary fixed-point. The gain will also be composed by thirteen bits and its multiplication by the incoming samples must be carefully done. If a 13-bit sample, already in bipolar representation, has got a positive value (which corresponds having null most-significant bit), the multiplication by the gain must be done normally. However, if the signed bit of the sample is a logical high, the sample represents a negative value. Thus, the two’s complement conversion must be applied to the sample right before the multiplication by the gain. The two’s complement conversion should also be applied to the multiplication result, to obtain the correct value. This process is usually called “Two’s Complement Signed Multiplication”.

As expected, bit growth from the original sample width occurs as result of the signed multiplication. Therefore, is necessary to extract from the multiplication result the desired thirteen bits for a sample representation. After that, the sample must be converted into a unipolar representation again.

After that, all the resulting samples are ready for being predistorted by the method presented before. Thus, an efficient way of implementation must be studied, since the algorithm is based in the arccosine function. There are several ways to generate those kinds of trigonometric functions in hardware designs. One available solution was to use the CORDIC IP CORE [57], provided by “Xilinx”. This CORE implements a generalized coordinate rotational digital computer (“CORDIC”) algorithm to iteratively solve trigonometric (and other) equations. However, another solution was taken into account, since it takes advantage of the internal design of the FPGA to efficiently implement the desired function.

As it was previously explained, the fundamental elements of the FPGA’s slices are constituted by LUTs that allow the implementation of combinatorial functions. These LUTs may also store information that can be easily indexed, working as “distributed RAM”. However, if the size of the information that will be stored is significant, a “Block RAM” based solution should be considered, reducing the access times and the occupied resources. Therefore, the practical way found to implement the predistortion was to generate a previously studied “Look-Up Table” and store it in a “Block RAM” [23].

The predistortion “Look-Up Table” can be seen as a “blackbox” that receives a 13-bit sample and outputs the correspondent 12-bit predistorted sample. The incoming sample will be the access index of the “Look-Up Table”: for example, the 13-bit word for a zero voltage representation will address the first position of the table and the word for a unitary voltage representation will address the last position. Thus, the “Block RAM” only has to output the corresponding content of the addressed position. This is a very efficient way to predistort the samples, since the time spent in the predistortion execution is only the time required to access and read a certain memory position.

The incoming samples that represent voltages higher that one volt or lower than zero volt (very common situation due to existent overshoots caused by the Raised-Cosine filter action) will be truncated, and set to one volt and zero volt, respectively. It is important to remember that only voltages between zero and one volt will be accepted by the “Look-Up Table”, due to arccosine function mathematical domain. Thus, 13-bit samples representing voltages between zero and one volt, represented in Q1.11 format, leads to the existence of “$2^{11} + 1$” different words. This will be the effective number of entrances of the “Look-Up Table”. The content of each entrance was calculated using “MATLAB”: for each possible input sample, the corresponding predistorted value was calculated by applying the operations described in Chapter 2 to the sample, and then represent the result in the fixed-point format. These were the values that will fill the “Look-Up Table” and that will be synchronously provided to the next blocks in the chain: the DAC interface.

In this work, the “DSP” module was the first block to be studied and designed. In order to validate and analyze its performance in FPGA, a “Test Platform” was conceived, and is carefully explained in “Appendix 3”. This platform is based in RAM memories and Finite-State Machines and intends to provide a flexible and global model that allows the validation of any block of the transmitter. After generate and send data to the “Test Platform” through serial port, the “Unit Under

Test" VHDL Module will process the data, which will be sent back to the computer through serial port again. A comparison between the generated input data and the processed output data may be done with created "MATLAB" scripts.

The validation of the "DSP" block operation begins with the generation of the data that is going to be processed. As illustrated in the scheme of figure 4.20, a "16-QAM" OSIP simulation was created and the data after the Raised-Cosine filtering was captured.

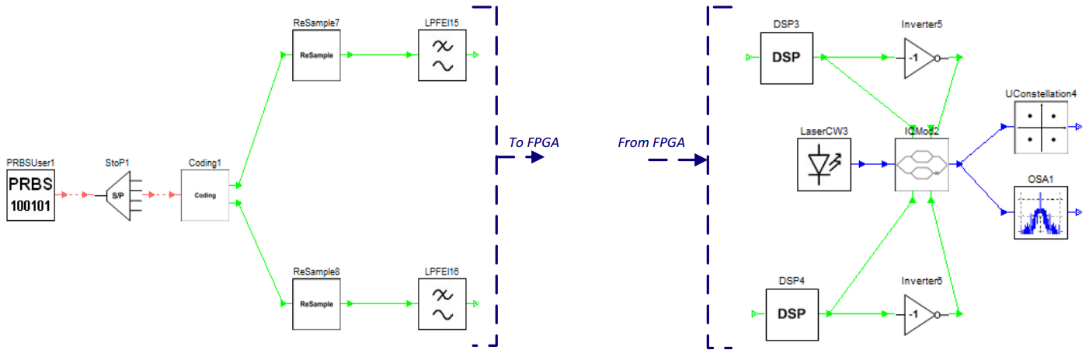


Figure 4.20 Co-Simulation using the "Test Platform"

The captured data was included in a "MATLAB" script. There, it was converted into a binary Q1.11 representation, in order to be sent to FPGA through serial port. However, the serial port communication only supports 8-bit words. Thus, before sending the binary information, it must be split in two fragments.

In the "Test Platform" the fragments will be grouped and the 13-bit words will be stored in a memory. After that, each word will be processed by the "DSP" block, which is going to apply a certain "Dynamic Gain" and predistort the stored information. The results will be stored in another memory and sent back to "MATLAB", where the appropriate decoding and plotting of the results will be done.

The processed data will be introduced in the OSIP simulation, into both "DSP" blocks of 'I' and 'Q' arms of the transmitter. Therefore, after re-running the simulation, all the information generated by OSIP will be discarded in those "DSP" blocks and substituted by the manually inserted data. These two blocks will not perform any mathematical algorithm: they will only give certain amplification and offset to the inserted data, preparing the signals for driving the modulator. The resulting constellation and optical spectrum at the transmitter output are presented in figures 4.21 and 4.22, respectively.

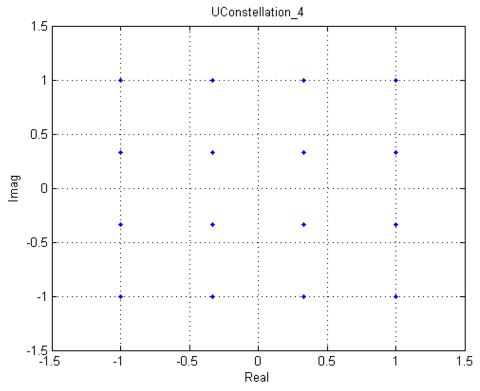


Figure 4.21 16-QAM Constellation after FPGA processing

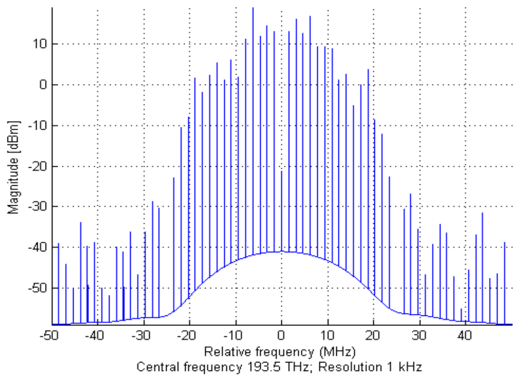


Figure 4.22 16-QAM Optical Spectrum after FPGA processing

The results are correct (for any "Dynamic Gain") and confirm the correct implementation of the "DSP" block. However, the first designed "DSP" block was ready to process 8-bit input samples, represented in Q1.6 format. This solution can save more FPGA resources but, as will be explained now, didn´t produce correct results.

The validation of that "DSP" block followed the same steps: a "16-QAM" OSIP simulation was created and the data right after the Raised-Cosine filter was captured. In the "MATLAB" script, the data was converted into a binary Q1.6 representation. After FPGA processing and back to the OSIP, the constellation obtained at the transmitter output is plotted in figure 4.23.

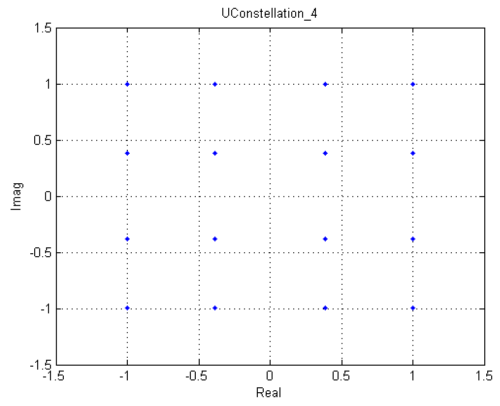


Figure 4.23 16-QAM Constellation after FPGA processing, for g=0.4 with 8-bit precision

As can be seen in the constellation of figure 4.23, the symbols are not equally spaced, which means that the response of the IQ-Modulator was not correctly linearized. This happens because the fractional part of each sample only has a 6-bit resolution, which is insufficient to represent the slight distortions made by the arccosine function. That is the reason why the "DSP" block was redesigned to support 13-bit words with a 11-bit fractional part: the precision is considerably increased and the results are very similar to the OSIP complete simulations.

The validated "DSP" block should now be included in the transmitter system and it must work synchronously with all the other blocks. Therefore, additional logic must be created to ensure the synchronism and avoid processing errors.

The first solution was to create an auxiliary block, which will work at the "Sampling Clock" rate and has the function of providing 13-bit samples to the "DSP" block. The "DSP" block will work at a much higher rate, when compared with the "Sampling Clock" frequency, in order to ensure that the predistortion of each sample is always finished before the "Sampling Clock" completes an entire period. Thus, in one complete period of "Sampling Clock", the auxiliary block should receive a 13-bit sample from the filter and send it to the "DSP" block, where the sample is going to be processed and returned to the auxiliary module again. Naturally, a processed sample will only be available to the following block in the chain in the next rising edge of the sampling clock.

However, this type of architecture required that the frequency of the "DSP Clock" was four times higher than the sampling clock's one, in order to complete all the 'DSP' tasks on time. As it was previously analyzed, for an upsampling factor of 4, the system must work with a 100MHz sampling clock. This constraint implies that the 'DSP' Module works at 400MHz. Although the main clock oscillator has got a frequency of 200MHz, the “Clock Wizard” IPCORE can use the 800MHz VCO to generate the desired frequency. However, after the design implementation with the ISE tool, the “Timing Reports” showed that some timing requirements of the system were not met. These error reports arose because some critical paths between synchronous hardware elements of the “DSP” Module do not support the 400MHz clock rate.

The way to solve the problem was to identify those critical paths and design another solution with increased parallelism. The majority of the critical paths were related to the logic required to implement the signed multiplication. Thus, that processing part was substituted by the Xilinx “LOGICORE IP Multiplier v11.2”. This multiplier was specially designed for high performance multiplications, minimizing area occupation and giving the possibility of having working rates up to 450 MHz in Virtex-6 FPGA Families [58]. For a signed multiplication of two 13-bit operands, the IPCORE uses a single “XtremeDSP” slice and requires three pipelined stages, which corresponds to three latency cycles.

Therefore, the final version of “DSP” block includes a “DSP1” block, which will work at the “Sampling Clock” rate and has the function of receiving the 13-bit samples from the FIR filter and providing them to the “Multiplier” block. Each sample must be subtracted by the fixed-point representation of “0.5” before become available to the multiplier. The “DSP1” block also provides a “Clock Enable” signal to the “Multiplier”, in order to start its operation. After three periods of the 400MHz “DSP Clock” signal, the result of the signed multiplication between the sample and the dynamic gain will be available and it must be provided to the output of “DSP1” in the next rising edge of the sampling clock. As expected, bit growth from the original sample width occurs and it is necessary to extract from the multiplication result the desired thirteen bits for a sample representation.

The second block that compose the “DSP” module is called “DSP2” and will add the “0.5” value to the incoming samples before truncating all the samples that represent voltages higher that one volt or lower than zero volts. Those samples will be the access indexes of the “Look-Up Table” and the “DSP2” block only has to output the corresponding content of the addressed position. Due to parallelism, it is possible to complete these operations using a single sampling clock period.

Finally, it is important to remind that despite the “DSP” block (composed by “DSP1”, “Multiplier” and “DSP2” VHDL Modules) is using two sampling clock cycles to perform the predistortion operation of a single sample, the sample throughput remains 100 MSPS, since only the latency between the FIR filter output and the end of the predistortion process was increased.

In laboratorial experiments, it will be useful to adjust the “Dynamic Gain” that the “DSP” block applies to the filtered samples. In this way, the user can simultaneously control the “roll-off” factor of the filter (as was previously explained) and adjust the gain applied in the predistortion block, defining the bandwidth and maximum excursion of the modulator driving signals, respectively. However, an efficient solution must be found in order to change the value of the “Dynamic Gain” without have to recompile the entire FPGA design. A method similar to the one that was used in the FIR filter reconfiguration may be considered, and is schematized in figure 4.24:

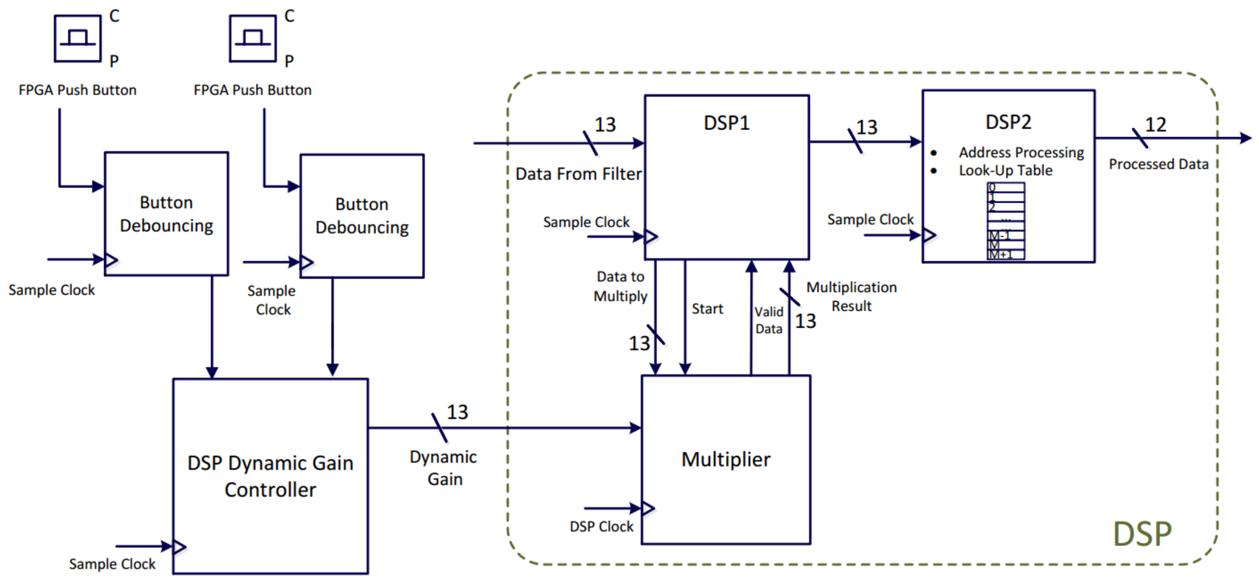


Figure 4.24 16-QAM DSP Module Interface

The user should press the correct push-buttons in the ML605 Evaluation Kit to promote a "real-time" change in the "Dynamic Gain" parameter of the "DSP" block. There are two push-buttons available: one for increasing the gain and another one to decrease it. However, as explained before, when a push-button is actuated, the contacts usually bounce, leading to a temporary ambiguity in the effective logic level. The method used to solve this issue in asynchronous hardware interface was described with the "Button Debouncing" block solution. A clocking source of 100 MHz and a 22-bit register were considered again.

The "DSP Dynamic Gain Controller" is the VHDL Module responsible for receiving signals from both "Button Debouncing" blocks and for providing the correct adaptive gain to the signed multiplier of "DSP" block. As explained, the "Dynamic Gain" is represented in a 13-bit word in Q1.11 format and can take values from zero to one volt, which allows the existence of "$2^{11} + 1$" different gains. Naturally, it is not necessary to have so many different gains and it will be difficult for the user to browse all the gains using only two buttons. The solution was to generate a "Look-Up Table" with a lower number of gains, each one spaced by "0.05". Using this method, a pressed push-button will lead to an increase or decrease of an "index" variable. Then, the LUT will output the content of the entrance addressed by the "index", in a synchronous way. This solution was chosen because the required logic is simple and the time spent in the adaptive gain calculation is only the time required to access and read a certain memory position. The information of the LUT was stored in a "distributed RAM", since the number of positions of the table is relatively small.

Once again, the buttons will be pressed asynchronously by the user, and the signal provided by both "Debouncing Button" blocks will last for several "Sampling Clock" cycles, which will lead to multiple index changes for a single time pressed button. This undesired situation will be avoided by implementing a "Disabled Mode" again. Therefore, every time an index is incremented or decremented (depending on which button is pressed), an internal flag inside "DSP Dynamic Gain

Controller" is activated, avoiding any other index changes. This flag will start the "Disabled Mode" process, where an n-bit counter will be incremented at the "Sampling Clock" rate. The flag will only be disabled when the counter reaches its final value. The counter's size determines how long the system ignores the output of the "Button Debouncing" block. Once again, a 24-bit counter was used, generating a "Disabled Mode" of 0.15 seconds.

In a 32-QAM transmitter, the structure of the designed hardware remains the same, since the predistortion algorithm is independent of the Data Modulation Format that is going to be transmitted.

4.3.9 DAC Interface

The subsystem of the optical transmitter that is being implemented in the FPGA processes digital data. However, as was explained in the second chapter, the IQ-Modulator requires driving signals in the analog domain. Therefore, a Digital-to-Analog Converter will be needed to create an interface between these two domains and generate the required signals. In a general way, the DAC will receive an 'N'-bit digital word from the FPGA and will perform the conversion to the corresponding analog voltage.

The Analog Devices® IC AD9963 was the adopted solution to perform the data conversion. This product provides two ADCs (Analog-to-Digital Converters) channels with sample rates of 100 MSPS and two 12-bit DAC channels with sample rates up to 170 MSPS. The AD9963 also offers five auxiliary analog channels, high integration, low power consumption and a flexible digital interface [59]. The block diagram in figure 4.25 helps to illustrate the main features of the "Data Converter":

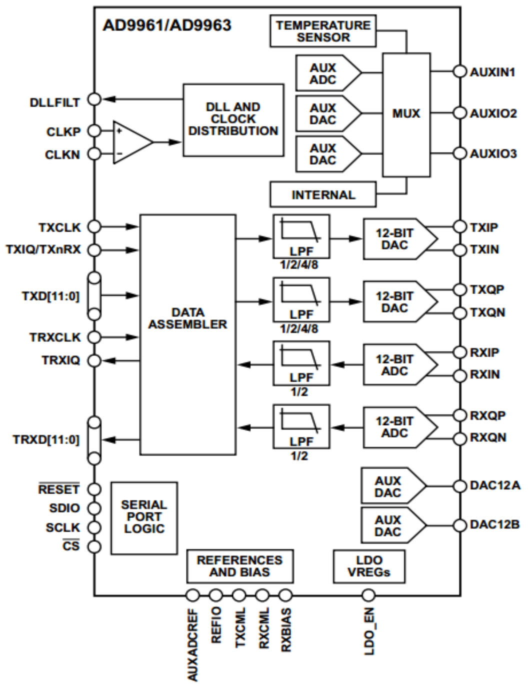


Figure 4.25 Functional Block Diagram of the AD9963 [59]

This chip takes part of the “AD9963-EBZ” Evaluation Board, which allows the use of nearly all of the AD9963’s capabilities. The board provides an adapter with the LPC (low-pin count) version of the “Vita 57.1 FMC” connector, so it can be used on either LPC or HPC (high-pin count) hosts of Xilinx Evaluations Boards, such as the ML605 (used in this work) [60, 61].

Apart from connecting the Evaluation Board to the FPGA, it is necessary to connect a +5Vdc power supply into a specific SMA input. This will provide an energy supply to all the internal LDO regulators, which can generate 3,3V or 1.8V analog voltages to all the internal components of the AD9963. Finally, a main clock source must be connected to the appropriate SMA “Main Clock Input”. The purpose of this clock signal will be explained later on.

The AD9963 has several “Configuration Registers”. Their purpose is to configure parameters like the operation modes, clocking sources, current gains, FIFO offsets, etc. These registers are configured through a serial control port using an SPI protocol. Thus, a USB cable must be connected between the Evaluation Board and a computer with the “AD SPI Application” previously installed. This software implements the serial communications interface, allowing the user to transfer data into the right addresses of the 8-bit “Configuration Registers”. It is also possible to activate a real-time “readback” of the content of each register in order to verify the correct configuration of the AD9963, or even save the entire content of those registers for a quick configuration of the Evaluation Board in future uses.

The AD9963 has two different paths available: the receive path and the transmit path. The Digital-to-Analog data conversion is made through the transmit section, which consists of two complete paths of interpolation filters stages, each followed by a high-speed current output DAC:

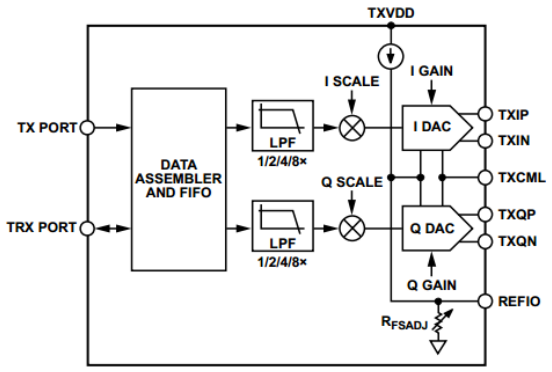


Figure 4.26 Transmit Path Block Diagram [59]

The ‘I’ and ‘Q’ transmit paths contain three interpolation filters designated as “INT0”, “INT1” and “SRRC”. Each of the interpolation filters provides a double increase in output data rate. These filters can be bypassed or cascaded to provide ‘2x’, ‘4x’ or ‘8x’ upsampling ratios [59]. They are also useful to reduce the requirements on the analog output reconstruction filter, after the DAC. The schematic in figure 4.27 shows the general transmit path data flow and clock generation.

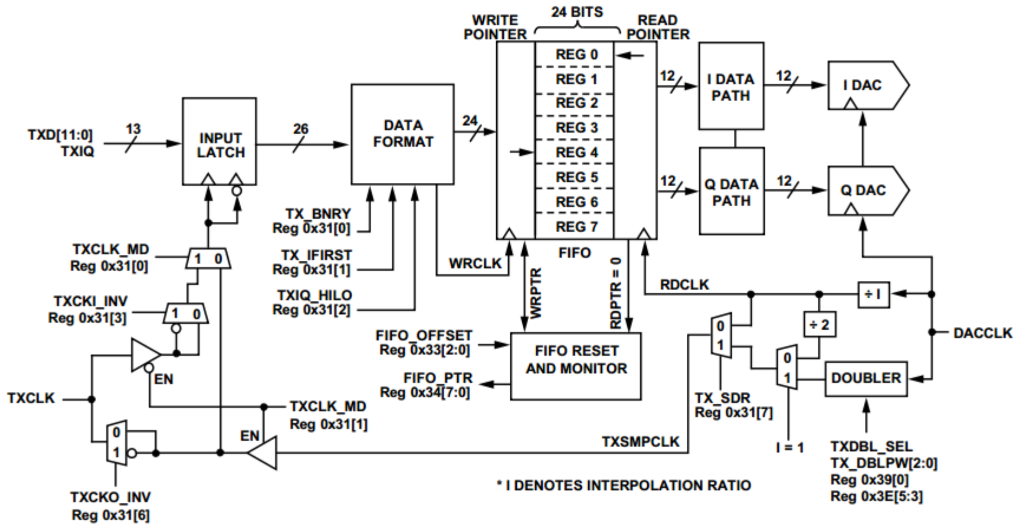


Figure 4.27 Transmit Path Data Flow and Clock Generation [59]

The interleaved 'I' and 'Q' 12-bit input data samples (TXD[11:0]) are accompanied by the "TXIQ" signal that identifies to which transmit channel ('I' or 'Q') the data is intended. Both signals are provided by the FPGA and are latched in the rising edges of the "TXCLK" [59]:

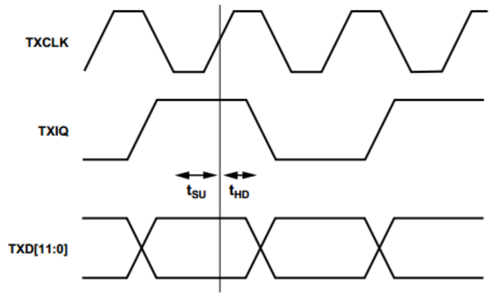


Figure 4.28 Transmit Path Timing Diagram [59]

The data is then formatted and buffered in the 8-deep FIFO of 24-bit words by the "WRCLK" signal, which has one half of the "TXCLK" speed because all the 'I' and 'Q' samples are interleaved. Then, the data is read from the FIFO by the "RDCLK" signal (which is always the "DACCLK" signal divided by the interpolation ratio) and is processed by the enabled interpolation filters in both 'I' and 'Q' paths. The data is then sampled by the transmit DACs [59]. The design assumes that "WRCLK" and "RDCLK" are at the same frequency. Thus, the FIFO plays an important role in the data processing, absorbing any phase drift between the two clock domains that drive the transmit data.

The device clocking source must be supplied through the SMA "Main Clock Input" of the AD9963 Evaluation Board. Thus, in the FPGA project, a 100MHz clock is generated and routed to one of the two available "SMA User I/O" pins of the Xilinx ML605 Evaluation Board. An appropriate SMA cable provides the single-ended clock to the AD9963 "Main Clock Input" pin. The schematic in figure 4.29 shows the general clock distribution of the AD9963.

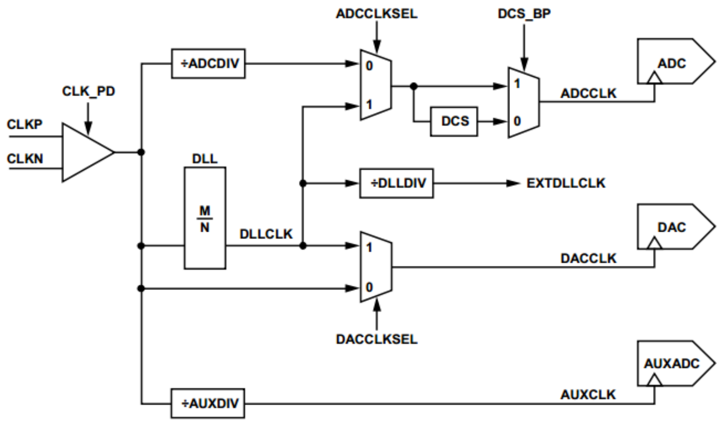


Figure 4.29 Clock Distribution Diagram [59]

The single-ended input clock is converted to a differential representation ("CLKP/CLKN") before clocking all the receive, transmit and auxiliary paths. In the specific case of the transmit path, it can be clocked directly from the "CLKP/CLKN" inputs or from the outputs of the DLL, where the clock can be multiplied by a factor of $\frac{M}{N}$. Therefore, the transmit path clock is called "DACCLK" and is represented in figure 4.27. It clocks the transmit DACs (acts as sampling clock) and allows the generation of the "RDCLK" signal. In this project, the direct clocking method was used and the DLL was disabled.

As can be seen in the figure 4.27, the "TXCLK" pin can be set as an input or an output, allowing the user to choose which clock is going to latch the input data. If the pin is set as an output (register "TXCLK_MD = '10' "), the "DACCLK" rate will be doubled before latch the input data, since 'I' and 'Q' samples are interleaved and need to be latched at a frequency two times higher than the sampling rate. On the other hand, if the "TXCLK" pin is set as an input, an appropriate 200MHz clock must generated and routed to the last available "SMA User I/O" pin of the ML605 Evaluation Board. An SMA cable must be connected to the "TXCLK" SMA input pin of AD9963 Evaluation Board. In both situations, the "TXCLK" signal must ensure the correct setup and hold timing requirements of the transmit path digital interface [59]. In this project, the "TXCLK" pin was used as an output since it requires reduced hardware solutions.

Both DACs have complementary output current pins, meaning that the sum of the two currents is always equal to their full-scale current [59]. The AD9963 Evaluation Board has got circuits that convert the differential output current of each DAC into single-ended voltage signals. This allows establishing an SMA connection between 'I' and 'Q' output channels of the Evaluation Board and an oscilloscope or the IQ-Modulator.

After several laboratorial experiments of the Evaluation Board operation, it was observed that the DAC output signals appeared slightly distorted. It was concluded that the low-frequency content of those output signals was being filtered due to the high-pass behavior of the Evaluation Board's output circuits. In order to illustrate this problem, an FPGA square wave generator project was created and the DACs analog output signals were analyzed. The results were similar to a typical behavior of an "RC Differentiator Circuit", as figure 4.30 illustrates.

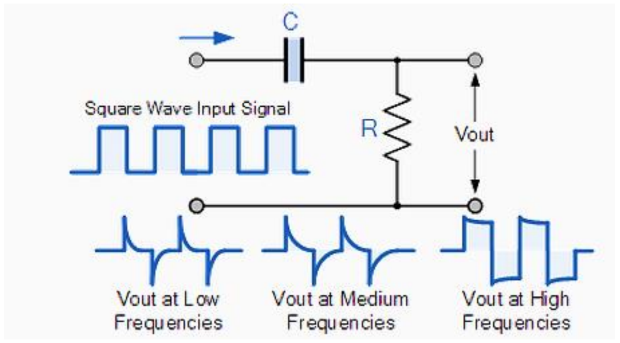


Figure 4.30 Distorted Square Wave at Evaluation Board's Output [62]

This type of high-pass filters does not change the shape of an input sine wave signal. However, if the input wave is a complex signal, the effects of these circuits are very similar to the mathematical function of differentiation. The output wave of a differentiator circuit is ideally a graph of the rate of change of the voltage at its input. Thus, each cycle of the square wave input waveform produces two spikes at the output, one positive and one negative, coinciding with the rising and falling edges of the wave. The decay of the spikes depends upon the time constant, "RC", and the value of the input frequency. Thus, by measuring the decay time and voltage variation it is possible to obtain the constant "RC" and the cutoff frequency of the filter based in the following expression:

$$f_c = \frac{1}{2\pi RC} \tag{4.2}$$

The lower cutoff frequency obtained is approximately 4MHz. However, due to speed limitations of the FPGA and AD9963, it is not possible to increase the frequencies of the 16-QAM or 32-QAM signals, in order to reduce the significance of the differentiation effects. Therefore, after several laboratorial experiments, it was concluded that 40 ns is the time that the DAC can hold on the same output voltage level without significant differentiation effect. This situation leads to the solution explained in the third chapter: the transmitter must generate a specific binary sequence that maps all the constellation symbols and simultaneously ensures that the resulting voltage levels in both 'I' and 'Q' DAC outputs are commuting in every symbol transition. This is equivalent to reduce the low-frequency content of the output signals. Furthermore, the sampling frequency must also be set to ensure that the time duration of a symbol does not exceed 40 ns. In a 16-QAM transmitter, the sampling rate must be set to 100 MSPS for an upsampling factor of '4', in order to ensure that the time duration of the symbol and the three added zeros is limited to 40 ns.

In either 16-QAM or 32-QAM implementations of the transmitter, it will be necessary to develop a VHDL Module that performs the correct data formatting in order to fulfill the transmit path timing diagram requirements:

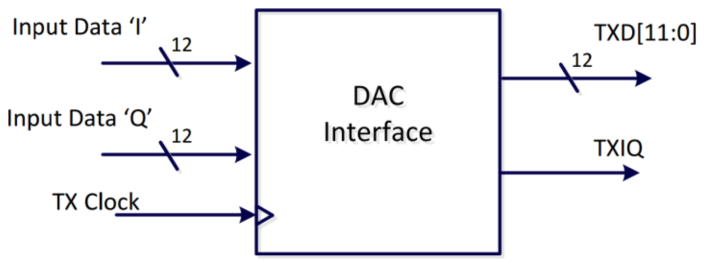


Figure 4.31 DAC Interface VHDL Module

The 12-bit processed samples from the "DSP" VHDL Modules of both 'I' and 'Q' arms of the transmitter are generated at a sampling rate of 100 MSPS. Thus, the "DAC Interface" Module will receive those samples and interleave them. The input "TX Clock" is a clock signal synchronous with the sampling clock, but with a frequency two times higher. Therefore, when two new 12-bit samples become available, the "DAC Interface" Module uses two rising edges of "TX Clock" to interleave and output the samples. The "TXIQ" signal is also generated, signaling to which DAC transmit channel ('I' or 'Q') the data is intended. The time diagram of these signals is represented in figure 4.32.

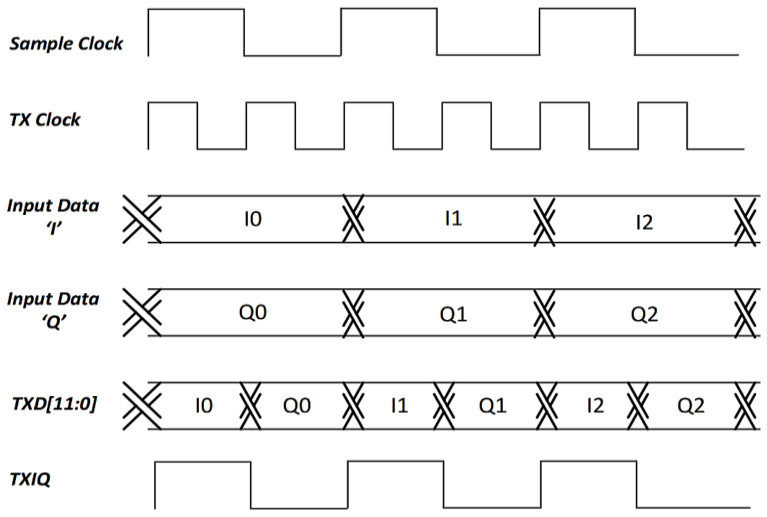


Figure 4.32 DAC Interface Timing diagram

Both "TXD[11:0]" and "TXIQ" signals must be routed to specific pins of the "Vita 57.1 FMC" connector of the ML605 Evaluation Board, respecting the "IOSTANDARD" "LVCMOS_25" [45, 59]. These constraints are specified in the "User Constraints File" of the ISE Design Project. The correct FMC pins can be determined by analyzing the AD9963 Evaluation Board Schematic Datasheet ([63]) and the "AD FMC Adapter" datasheet ([61]). A conversion table can be created in order to match the pins of the ML605 Evaluation Board with the corresponding internal input signals of AD9963 Data Converter.

4.4 Resources utilization

Considering the resources available in the ML605 Evaluation Board (presented in section 4.2), the following table shows the amount of hardware needed for the implementation of the transmitter systems. The percentage of occupied resources depends on the FPGA used and the internal structure of its logic.

Logic	*Utilization (%) 16-QAM*	*Utilization (%) 32-QAM*
Slice Registers	853 (1%)	1101 (1%)
Slice LUTs	541 (1%)	829 (1%)
Occupied Slices	266 (1%)	361 (1%)
DSP48E1	14 (1%)	14 (1%)
RAMB36E1	2 (1%)	6 (1%)
Bounded IOB	30 (5%)	30 (5%)
BUFGs	7 (21%)	8 (25%)
MMCM_ADVs	1 (8%)	1 (8%)

Table 4.2 Total hardware usage

5. Laboratorial results

5.1 Co-Simulation

In this section it is intended to validate the operation of the transmitter system implemented in a FPGA and to prove that the predistortion concept can be assembled in a hardware platform. Thus, the digital information generated by the transmitter will be captured and introduced in OSIP simulations, allowing to verify if those processed samples will produce the expected results, i.e., create correct constellations and optical spectrums. The schematic in figure 5.1 shows a simplified block diagram of the transmitter and identifies the part that was implemented in hardware and the one that was simulated using OSIP:

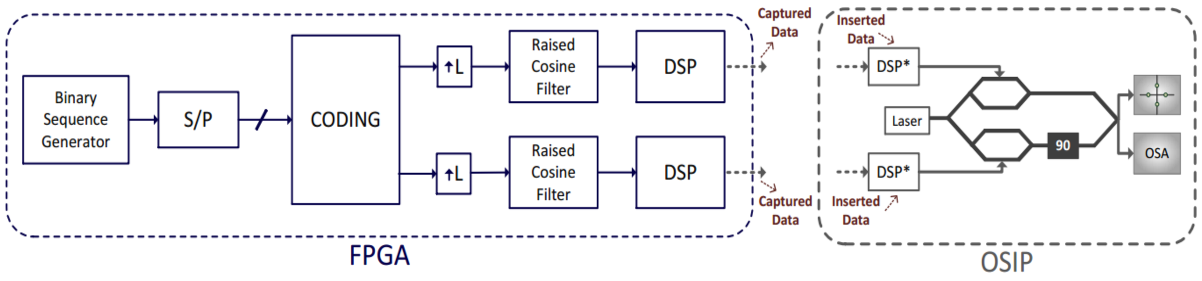


Figure 5.1 Co-Simulation: Transmitter (*DSP block will only amplify and offset the inserted data)

As explained in the previous chapter, the real-time monitoring and capture of any internal signals of the FPGA design can be performed by including an “ICON” core and the corresponding “ILA” units into the project hierarchy. Then, during the system’s operation, the “ChipScope Pro Analyzer” software allows the user to capture and visualize data. In this work, the ILA units were configured to capture some thousands of 12-bit words that were being provided by the “DSP” blocks to both ‘I’ and ‘Q’ channels of the DAC. The “ChipScope Pro Analyzer” tool also allows exporting the acquired data into a file, for future viewing or processing [64]. Thus, a MATLAB script was created to read the information from that file and to make the appropriate conversion of the data into real numbers.

In the OSIP simulation of the transmitter system (whose block diagram was presented in figure 3.1), it is possible to insert arrays with the captured data into both “DSP” blocks of ‘I’ and ‘Q’ arms of the transmitter. Therefore, during the simulation, all the information generated by OSIP will be discarded in those “DSP” blocks and substituted by the manually inserted data. These new driving signals will be amplified and biased in order to operate the IQ-Modulator in the “Null biasing point”. The results will be visualized by the “UConstellation” and “OSA” blocks.

Finally, it is important to notice that this manual insertion of data is only possible if the number of inserted samples is equal to the number of samples generated by the simulator. If this simulation parameter does not match with the number of inserted samples, a simulation error will occur, since the number of samples is a predefined value that cannot be changed during the simulation.

Figures 5.2 and 5.3 show the results obtained after capturing data from the 16-QAM transmitter implemented in the FPGA and using the OSIP simulation to apply these driving signals to an ideal IQ-Modulator.

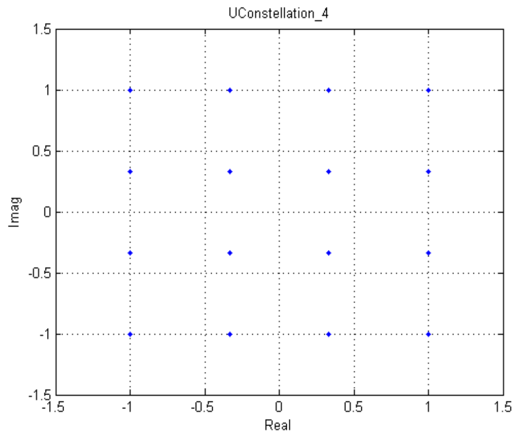


Figure 5.2 Co-Simulation Results: 16-QAM Constellation

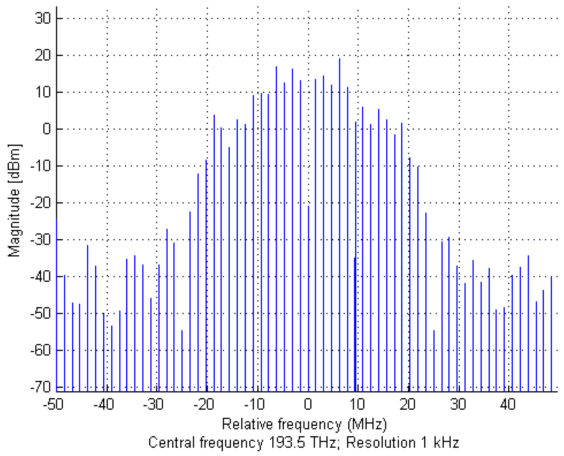


Figure 5.3 Co-Simulation Results: 16-QAM Optical Spectrum

These results validate the operation of the 16-QAM transmitter system and the predistortion concept. The constellation obtained right after the IQ-Modulator presents correct Euclidean distances between symbols for all the different excursions of the driving signals. These excursions can be directly adjusted in the FPGA through the corresponding push-buttons available on the Evaluation Kit. Furthermore, the results obtained with the "Optical Spectrum Analyzer" block are also correct, proving that the predistortion operation does not affect the transmitted spectrum. The figure above shows that the spectrum has a well-defined Raised-Cosine shape with a unitary roll-off factor. The first null point is located at 25MHz, which is the symbol rate of the transmitter.

Figures 5.4 and 5.5 shows that the same conclusions can be applied to the results obtained with the 32-QAM transmitter, which proves that the predistortion concept is independent of the number of constellation symbols of the M-QAM format:

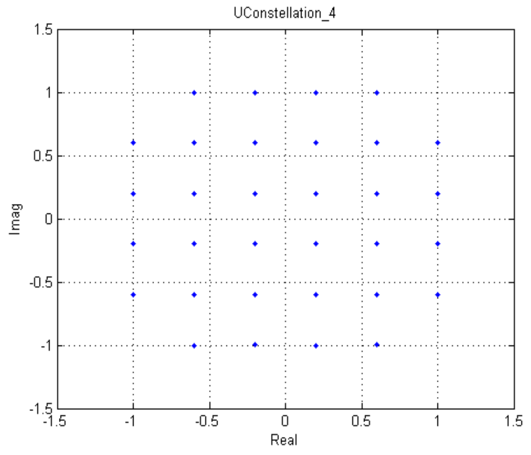


Figure 5.4 Co-Simulation Results: 32-QAM Constellation

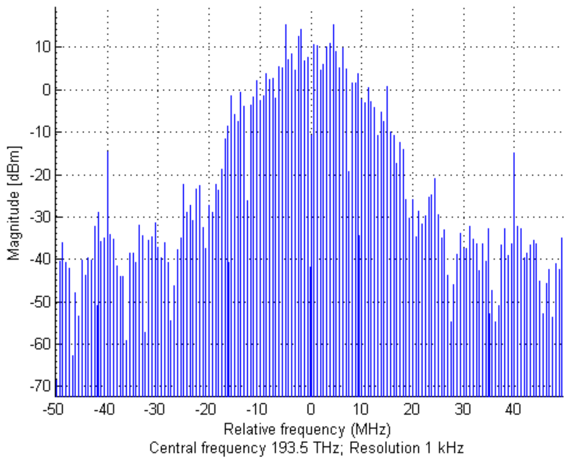


Figure 5.5 Co-Simulation Results: 32-QAM Optical Spectrum

5.2 16-QAM Laboratorial experiments

In the previous section it was also proven that the hardware implementation of the transmitter system was generating digital information with the required precision, producing correct results with an ideal IQ-Modulator. However, this ideal representation has features (like infinite bandwidth or an ideal transfer function) that cannot be achieved with the modulators present in real transmitter systems. Thus, it is interesting to perform some tests with the IQ-Modulator available in the “Optics Laboratory” at Telecommunications Institute of Aveiro. The “Fujitsu’s Optical Modulator FTM7962EP” [65] is composed by single-drive MZMs and its operation is controlled by the “Bias Controller D0156” [66]. This device was specially designed to control the biasing positions of the IQ-Modulator in DQPSK/QAM applications. The user can define the specific null biasing point by configuring some parameters in a software application, which will program the device through USB connection. Furthermore, the electrodes of the modulators are connected to the outputs of the “SHF 807 Linear Broadband Amplifiers”, which are optimized for amplification of multi-level analog signals [28]. The laboratorial setup followed the typical structure of an optical coherent system and is shown in figure 5.6:

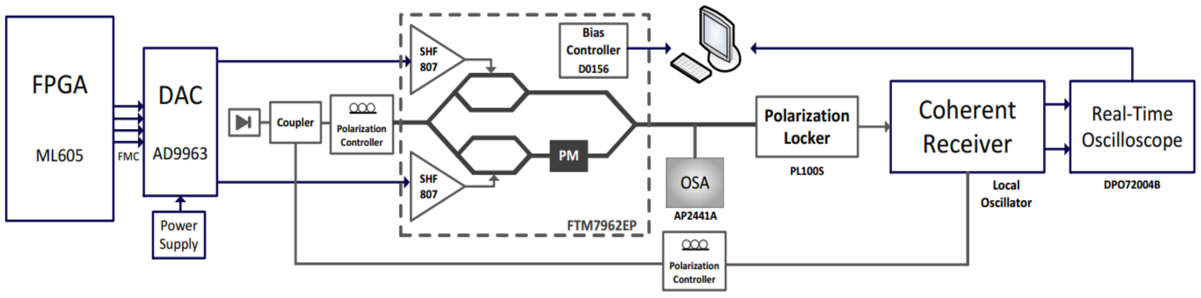


Figure 5.6 Laboratorial Setup

As it can be seen in figure 5.6, a coherent receiver was used to process the transmitted data and provide two electrical signals to the Real-Time Oscilloscope. Due to the important role played in the laboratorial setup, it is important to get some considerations about the coherent receiver. Figure 5.7 shows a simplified block diagram of a homodyne coherent receiver with phase and polarization diversity [24]:

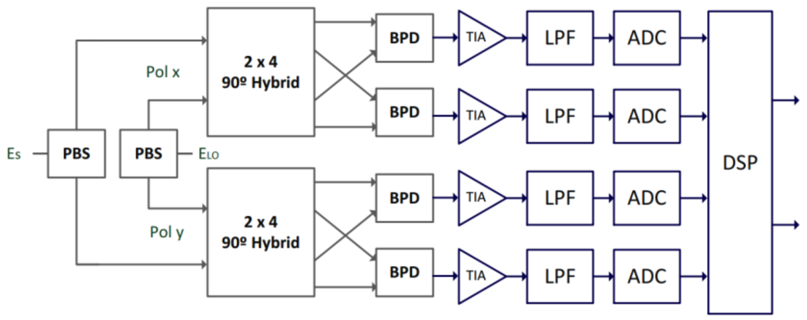


Figure 5.7 Homodyne Coherent Receiver with Phase and Polarization Diversity

A polarization beam splitter (PBS) will capture the optical signal and separate the polarizations ‘x’ and ‘y’. Another PBS will perform the same process to the local oscillator (LO).

Furthermore, the LO beam incident at this PBS must be polarized with an angle of 45°, in order to obtain the same LO power in both polarizations. Each "2x4 90° Hybrid" will enable the detection of the in-phase and quadrature components of its inputs, which can be generally described by the following equations [24]:

$$E_{in_1}(t) = \left|E_{in_1}(t)\right|.e^{j\phi_1(t)}, \quad E_{in_2}(t) = \left|E_{in_2}(t)\right|.e^{j\phi_2(t)} \tag{5.1}$$

where $'\left|E_{in_1}(t)\right|'$ and $'\left|E_{in_2}(t)\right|'$ represent the intensity information of the signal and $'\phi_1(t)'$ and $'\phi_2(t)'$ represent the phase information. The following output powers are desired at the four hybrid outputs (n = 0, 1, 2, 3) to enable the detection of the in-phase and quadrature components:

$$P_{out_n}(t) = E_{out_n}(t)\,.E^*{}_{out_n}(t) =$$
$$\tfrac{1}{4}\left|E_{in_1}(t)\right|^2 + \tfrac{1}{4}\left|E_{in_2}(t)\right|^2 + \tfrac{1}{2}\left|E_{in_1}(t)\right|\left|E_{in_2}(t)\right|.\cos[\phi_1(t) - \phi_2(t) - n.90° + \psi] \tag{5.2}$$

where the phase shift ψ is allowed to be arbitrary [24]. To provide the desired powers specified by the equation (5.2) at the four outputs, the "2x4 90° Hybrid" must exhibit the following field transfer function [24]:

$$\begin{bmatrix} E_{out_1}(t) \\ E_{out_2}(t) \\ E_{out_3}(t) \\ E_{out_4}(t) \end{bmatrix} = \frac{1}{2}.\begin{bmatrix} e^{j\psi_{11}} & e^{j\psi_{12}} \\ e^{j\psi_{21}} & je^{j\psi_{22}} \\ e^{j\psi_{31}} & -e^{j\psi_{32}} \\ e^{j\psi_{41}} & -je^{j\psi_{42}} \end{bmatrix}.\begin{bmatrix} E_{in_1}(t) \\ E_{in_2}(t) \end{bmatrix} \tag{5.3}$$

where the phase coefficients ψ_{ij} must satisfy the phase condition $\psi_{11} - \psi_{12} = \psi_{21} - \psi_{22} = \psi_{31} - \psi_{32} = \psi_{41} - \psi_{42} = \psi$. The balanced photodetectors will perform the conversion of the information from the optical to the electrical domain. The transfer function of the photodiodes is given by [24]:

$$I_{out} = R.E_{inc}.E^*{}_{inc} \tag{5.4}$$

where 'R ' is the responsivity and 'E_{inc}' is the incoming optical field. A balanced photodetector is composed by two photodiodes and its transfer function is given by [31]:

$$I_{out} = I_1 - I_1 = R.(E_{inc1}.E^*{}_{inc1} - E_{inc2}.E^*{}_{inc2}) \tag{5.5}$$

The trans-impedance amplifiers will amplify the electrical signals (now represented in their in-phase and quadrature components) and convert currents into voltages. The lowpass filters (LPF) will avoid spectral superposition due to the sampling operation (anti-aliasing filters) and will also limit the amount of noise and include the frequency response of the photodiodes. However, the photodiodes response may be dimensioned to limit the bandwidth of the signal and the noise. In this case, the lowpass filters will not be necessary [31]. The ADCs will sample the signals at a rate at least two times greater than the highest frequency of the corresponding signal, accordingly to the "Nyquist-Shannon Sampling Theorem" (which states that the sampling frequency should be greater than twice the highest frequency component of interest in the signal ($L_{min} = 2$) [38]). Finally, the 'I' and 'Q' components are ready for digital signal processing (DSP). In this work, a Real-Time Oscilloscope was connected to a computer were a software (developed at Telecommunications Institute of Aveiro) was performing the signal equalization, allowing the extraction of the transmitted information and the visualization of constellations before and after compensation. This signal processing comprises timing recovery, adaptive linear and nonlinear equalization, frequency offset or phase correction [25].

5.2.1 Transmission without Predistortion

These first laboratorial measurements intend to illustrate the real impacts of the Mach-Zehnder Modulators' nonlinear characteristic in the generation of 16-QAM constellations. For that purpose, the transmitter system implemented in FPGA was generating information without any kind of filtering or electronic predistortion and the structure of the setup shown in figure 5.6 remained unchanged. The first captured results were obtained by configuring the excursion of the driving signals for using approximately 30% of the entire region of interest of the characteristic (which has a full range of $'2V_\pi'$). As it can be seen in figure 5.8, the Euclidean distances between adjacent points are almost equal, despite the existing impairments like phase noise or slightly misaligned points. Then, the "adaptive gain" was gradually increased by using the corresponding FPGA buttons. This expansion in the excursion of the driving signals took the operation of the modulators into nonlinear regions, and the expected distortions in the constellations started to being noticed. In figure 5.9 is shown the most misaligned constellation, obtained when the full range of the modulators' characteristic was exploited.

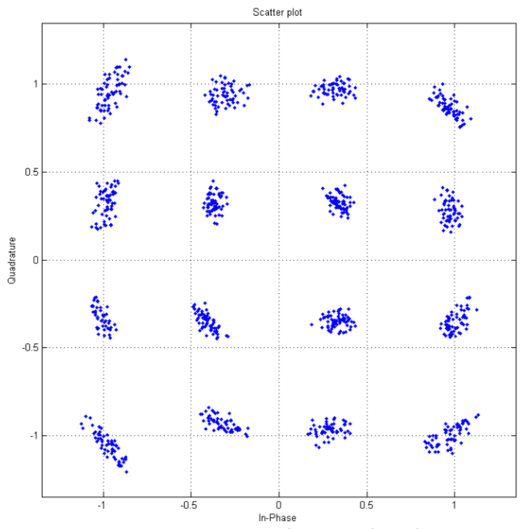


Figure 5.8 16-QAM Constellation for 30% of the full excursion

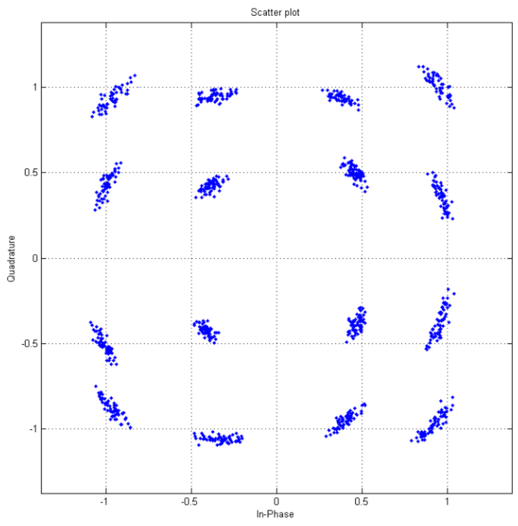


Figure 5.9 16-QAM Constellation for $'2V_\pi'$ excursion

In this experiment it was verified that an increase in the driving signals' excursion led to an evolution of the symbols' positioning that matched with the simulation results. This demonstrates that the modulators have a nonlinear characteristic with a cosine-like shape. It was also demonstrated that it is only possible to generate correct QAM constellations without using predistortion if a small linear operating region of the characteristic was considered.

5.2.2 Transmission with Predistortion

Until now it was proven that the impairments caused by the nonlinear characteristic of an ideal modulator can be overcome by the electronic predistortion of the driving signals. However, the following laboratorial experiments intended to analyze the performance of the compensation algorithm under laboratorial conditions. The same experimental setup was considered, with the FPGA implementing a transmitter system with the predistortion algorithm but without the filtering process. The figure below shows an example of an eye diagram captured at one of the DACs' outputs:

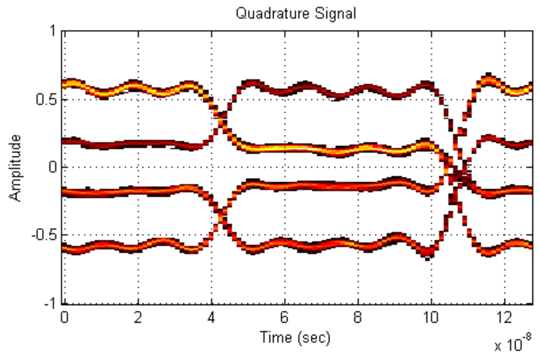


Figure 5.10 Eye Diagram at DACs' "Q" output for an "Adaptive Gain" of 60%

The voltage levels are unequally spaced due to the predistortion effect and the level transitions demonstrate the transmission of the short predefined binary sequence. The procedure to capture and analyze new results was the same used before: the excursion of the driving signals was defined as approximately 30% of the characteristic's full range. Then, the amplitude of the signals was gradually increased until reach the maximum value of $'2V_{\pi}'$. Figures 5.11 to 5.13 illustrate some of the best obtained constellations after performing DSP in computer software:

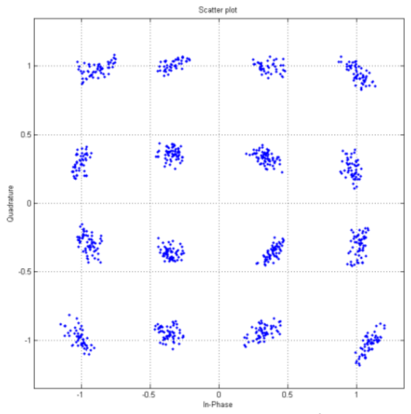


Figure 5.11 16-QAM Constellation for 30% of the full excursion

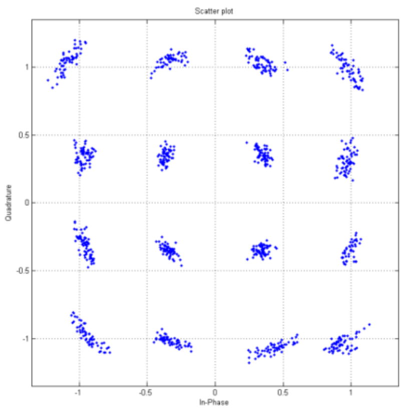


Figure 5.12 16-QAM Constellation for 80% of the full excursion

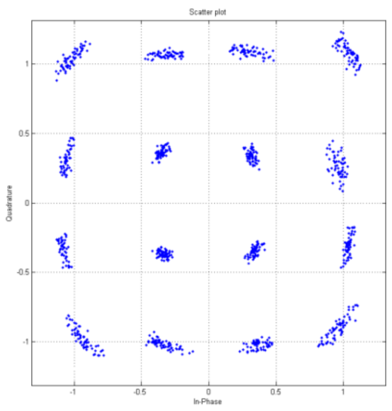


Figure 5.13 16-QAM Constellation for the full excursion ($'2V_{\pi}'$)

These constellations have almost equal Euclidean distances between adjacent symbols. All the results are very similar and independent from the driving signals' excursion, validating the operation of the 16-QAM transmitter system and the predistortion concept in a real-case scenario. The possibility of simulating a linear response of the MZMs and adjusting the excursion of the driving signals may bring advantages in terms of optical power efficiency in the IQ-Modulator and also allows increasing the OSNR in the system.

5.2.3 Discussion

The previous two sections proved the impacts of the nonlinear characteristic of the MZM in the generation of 16-QAM constellations and the possibility of using an electronic predistortion technique to suppress those impairments and increase the performance of the transmission system. However, all the obtained constellations have some problems in common. One of them is the phase noise. In optical coherent systems there are several noise sources, but the degradation of the receiver's sensibility is clearly associated with the phase noise introduced by the transmitter laser and the local oscillator (LO) laser at the receiver. Phase noise can be quantified in the frequency domain by its power spectral density and can be considered a Wiener process caused by the spontaneous emission of the laser [67]. The "C-Band Tunable Laser" used in the setup was configured for operating at 1550 nm. However, the output of any single-frequency laser is not perfectly monochromatic but rather exhibits some phase noise, which leads to its finite linewidth. All the constellations above are degraded due to this phenomenon, where the effective phases of the constellation symbols are rotated, reducing the noise margins between adjacent points. Besides the phase noise, also frequency deviations that may exist between the received signal and the local oscillator can have impacts in the constellations. This impairment generally arises from the frequency control of the lasers [68]. In this laboratorial setup, the oscillator of the transmitter was intentionally used as the same oscillator of the receiver, in order to reduce the phase noise effects and to ensure the synchronism of the homodyne receiver. Digital signal processing techniques for phase estimation and recovery ("Viterbi and Viterbi" algorithm) were also performed in computer software to compensate these impairments. As was previously stated, the plotted constellations already illustrate the results after the compensation. These constellations also have some misaligned points, i.e., the Euclidean distances between adjacent symbols aren´t totally correct in certain regions of the IQ plane. Despite the clear benefits of the predistortion algorithm, these imperfections are mainly caused by the fact of the field transfer functions of the MZMs may differ from each other, with their natural shape and symmetry changing during the system operation. Some regions of the field transfer function may slightly differ from the ideal cosine shape, leading the predistortion algorithm to perform less effective compensations.

It is necessary to establish a quality metric that allows quantifying the performance and real advantages of the predistortion algorithm and also the impacts of the previously described impairments in the 16-QAM constellations. The Error Vector Magnitude (EVM) was already defined as the mean root square of the difference between an ideal constellation point and an actual received symbol in the IQ plane [40]. In the following figure each excursion of the driving signals is associated with an average value of the EVM, which results from several captured constellations during the laboratorial experiments.

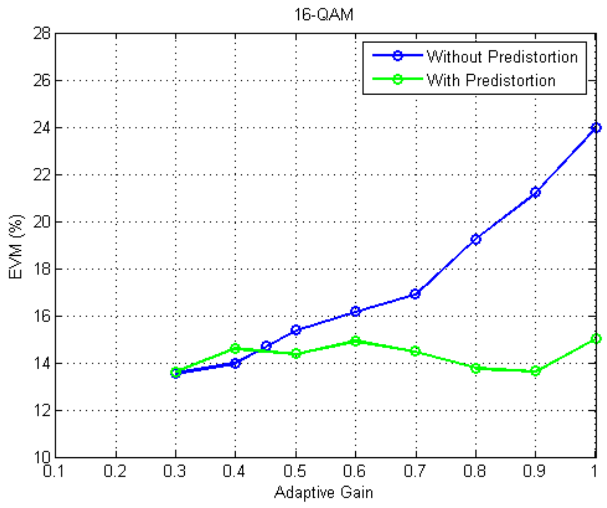


Figure 5.14 EVM (mean) for different excursions of the IQ-Modulator driving signals

The results show that when the transmitter system did not perform any electronic predistortion, the lower EVM values were registered only for small excursions of the driving signals, which was coincident with the operation in an approximately linear region of the modulator's characteristic. Then, gradual increases in the "Adaptive Gain" were traduced in higher percentages of the EVM, since the driving signals started to operate in nonlinear regions of the characteristic, enhancing the distortions in the constellations. On the other hand, when the predistortion algorithm was applied, the results were improved, especially for high excursions of the driving signals. The green curve shows that EVM mean values remain nearly constant for all the different values of "Adaptive Gain", mainly because the positioning of the symbols in the constellations is kept almost unchanged. However, instead of presenting the theoretical null values, the percentages swing around 14%. These values are particularly influenced by the already described impairments and also by laboratorial limitations that caused a instable operation of the system. Considering a certain "Adaptive Gain", the captured constellations sporadically presented small displacements around the center of the IQ-plane. This situation was caused by difficulties in the biasing control of the modulator and led to an increase of the registered EVM mean values. Therefore, the constellations above illustrate the best achieved results, presenting EVMs around 12%. Furthermore, the EVM percentages of the green curve (transmitter with predistortion) might even decrease for greater excursions of the driving signals, since the improvements in OSNR should led to the reduction of the symbols' spreading [39]. This evolution of EVM was verified only for "Adaptive Gains" from 0.6 to 0.9, but was almost unnoticed simply by looking at the constellations. Despite the non-expected increase of the EVM when reaching full excursion, this value stills 9% lower than the EVM mean value registered for the same excursion without compensation. These results prove that it is possible to increase the OSNR without distort the positioning of the symbols in the constellations. However, the precision of the algorithm cannot be totally evaluated since that several laboratorial limitations are impairing the system's performance.

5.2.4 Laboratorial limitations

The first experiments with the setup presented in figure 5.6 were not well succeeded. The results only started to appear after replacing the laser. The reason for this improvement was the linewidth of the lasers: the first laser used had a linewidth of a few "MHz", while the new "C-Band Tunable Laser" had a linewidth of 100 KHz. In fact, the linewidth of the laser seems to be one of the main reasons for the impairments in the results. As seen before, the spectrum of the transmitted signal had a bandwidth of tens of "MHz", even without filtering. This bandwidth can be considered small when compared to the typical bandwidths of the WDM channels transmitted in optical systems. Thus, the bandwidth of the signal can be "compared" with the linewidth of the laser, which brings consequences in terms of the main frequency content of the transmitted signal. Such a narrower bandwidth can also be less tolerant to the effects of the optical part of a coherent system, and distortions in the 16-QAM signal can arise, as stated in ref. [69], where studies about the impacts of laser phase noise in 16-QAM systems were performed.

The IQ-Modulator can also have negative impacts in the transmitted signal. As stated before, the transfer function of each MZM may present small changes during the operation, especially due to temperature variations [65]. It will be also difficult to predict losses, the noise introduction and the transfer function of the modulator. There were also difficulties in managing the "Bias Controller" of the modulator, and the bias positions had to be defined in the "Manual Mode", which introduced errors due to the always changing features of the modulator's characteristic. The DACs of the "AD9963" also present some limitations. The symbols of the constellations plotted above present clouds around the ideal positioning. This indicates the presence of amplitude and phase noises mainly caused by the amplifiers at the DACs' outputs, although some AWGN might also be present. It is also noticeable from the previous figures that no results were captured for "Adaptive Gains" below 30%. The DAC's maximum amplitude of 170mV required the drivers to apply a gain factor in order to reach the $'2V_{\pi}'$ excursion of approximately 3 volts. An excursion 30% lower corresponds to 50mV. Signals with amplitudes lower that this impaired the driver's correct amplification and decreased the SNR of the received signal. If DACs could achieve greater excursions, the SNR would be improved and better results could be obtained. Some of the misaligned points of the constellations were also caused by distorted voltage levels sporadically generated by both DACs, possibly caused by distortions in the high-pass behavior of their output circuits.

The only concept that was not proved in laboratorial conditions was the behavior of a Raised-Cosine spectrum right after the IQ-Modulator. The Optical Spectrum Analyzer (OSA) available at the laboratory [70] could only get a resolution of 100MHz for the presented working conditions, not allowing the study of the transmitted spectrum, which has a smaller bandwidth. The laboratorial limitations presented in this section and the consequences that they had in the results of the 16-QAM system prevented the analysis of the 32-QAM system performance. High-order QAMs require greater SNRs, since Euclidean distances between symbols grow shorter. Their tight constellations also increase sensitivity to phase noise, one of the main impairments affecting these experiments.

6. Conclusions

This dissertation was focused in the compensation of impairments related to the nonlinear characteristic of the Mach-Zehnder Modulator in optical coherent transmission systems. In the second chapter of this document, several aspects related to coherent transmitter systems were studied. Due to the advantages of exploiting the full swing of the modulators' characteristic ($'2V_{\pi}'$), an example that shows the impact of the nonlinearities in the generation of multi-level modulation formats was discussed. It was proven that if the excursion of the driving signals was not constrained to an approximately linear region of the transfer function, constellations with unequal Euclidean distances between symbols would be produced. The most efficient way to avoid this scenario was creating an electronic predistortion of the driving signals in order to compensate the nonlinear response of the modulators. It was concluded that the application of the inverse function of the modulators' response to the driving signals would lead to the generation of correct results. Thus, a mathematical solution for the problem was presented and carefully discussed.

In the third chapter, QAM coherent transmitter systems and the compensation algorithm were simulated in the OSIP platform. After demonstrating the impact of the nonlinear characteristic in the resulting constellations, several simulations including the predistortion algorithm were performed. The deviations from ideal constellations were reduced independently of the M-QAM format, bringing benefits for bandwidth variable transceivers. These simulations also allowed the study of the Raised-Cosine filtering. OSIP tools made possible to verify the null intersymbol interference, to visualize the correct shape of the spectrums and to study the impacts of changing the roll-off factor. However, it was stated that the predistortion algorithm had adverse effects on the spectrum of the driving signals, since they were being manipulated by nonlinear operations. Simulations proved that the electronic predistortion will compensate the nonlinear response of the modulators, while ensuring that the transmitted spectrum maintains the desired Raised-Cosine shape. It can be concluded that the necessary condition to observe these results is to have an IQ-Modulator with enough bandwidth to ensure that the signal's frequency range in which it will operate is the same range where the 'DSP' has operated. However, such bandwidth requirements may be unachievable in practical systems.

After validating the transmitter's operation, the algorithm was tested under real hardware limitations. The fourth chapter presented an overview of the FPGA's principles of operation and the main features of the ML605 Evaluation Kit used in the implementation stages. Then, solutions were identified and discussed in order to implement the digital part of the optical coherent transmitter system in FPGA. It was necessary to develop a clocking network that ensures "real-time" operation of the system without data loss. It was concluded that the application of "Timing Constraints" to the system was essential to guarantee the mapping of the design into specific FPGA slices and ensure the meeting of timing requirements. Another conclusion was the fact that, in this work, the majority of the hardware solutions had been conceived to exploit parallelism and the benefits of the internal structure of the FPGAs. For example, by using "look-up tables" it is possible to ensure the system's efficiency, since the time spent in accessing information was only the time required to access and read a certain memory position. Furthermore, all the digital samples were represented in binary fixed-point representation. When compared with floating-point representation, the FPGA design benefits from the superior speed that fixed-point offers as well as from the minimum amount of hardware resources it demands. Considering the hardware available in the ML605 Evaluation Board, the implementation of the transmitter systems did not requires the occupation of significant resources. Besides that, it is possible to conclude that 16-QAM and 32-QAM transmitters need almost the same amount of logic.

In fifth chapter, the operation of the transmitter system implemented in FPGA was validated by performing co-simulations. There, “real-time” information generated by the FPGA was captured and introduced in OSIP simulator. The results showed that the generated data had got the required precision, compensating the nonlinear effects of an ideal IQ-Modulator. The performance of the compensation algorithm under laboratorial conditions was also analyzed by connecting the FPGA and the DAC to an IQ-Modulator. The laboratorial setup allowed capturing 16-QAM constellations after performing DSP in the receiver. Without electronic predistortion, the first experiments showed that the modulators had a sinusoidal field transfer function. Undistorted constellations were only possible to achieve if small excursions of the driving signals were used. When the compensation algorithm was applied, correctly shaped 16-QAM constellations were obtained independently from the driving signals’ excursion. Although the predistortion algorithm was validated, all the constellations were affected by several impairments, which were quantified by “EVM”. The most relevant ones were phase rotations caused by the laser phase noise, cloudy symbols due to amplitude noise and small misalignments of the points. In fact, the quality of the results was affected by several laboratorial limitations. The main problem was related with the linewidth of the laser, which was “comparable” with the narrow bandwidth of the transmitted signal. There were also difficulties in managing the “Bias Controller” of the modulator and both DACs presented several limitations like low amplification, distortion of certain voltage levels, low working rate or amplitude noise. Due to lack of experimental equipment, it was not possible to measure the transmitted optical spectrum and also perform experimental tests to 32-QAM.

6.1 Future work

The availability of faster and precise DACs will allow to overcome the previously reported impairments and to analyze the performance of high-order QAM transmitter systems. Future work could be related with the implementation of an adaptive system based on monitoring the signals at the IQ-Modulator output through a feedback loop between the device and an ADC connected to the FPGA. The last one could process information from the modulator and adjust eventual misaligned points in the constellation. Finally, it is important to assess and quantify the impacts of the predistortion algorithm in the transmitted Raised-Cosine spectrum under limited bandwidth conditions.

7. Appendix 1: VHDL

VHDL (“**V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage”) is a hardware description language used for modeling, simulation and synthesis of digital systems. It allows the developer to describe the behavior of a system in an easy way, avoiding the complex manual configuration of all the connections and logic blocks that will be implemented in the FPGA [71].

This language provides several abstractions for hardware constructions. It is possible to instantiate a certain entity, specify its internal architecture and the corresponding signals or logic gates, determine the number and function of the input and output ports, etc. However, the characteristic that distinguishes VHDL from most programming languages is the support to parallelism. Thus, multiple processes may run simultaneously, which was not possible to implement with a software language, due to its inherent sequential nature [71].

In the conception of this project, two main VHDL abstraction levels were used. The highest abstraction level in VHDL is called “Behavioral model”. Here, the system is described by the way it behaves, instead of a lower abstraction of its connections. Therefore, this model can describe the relationship between input and output signals in two different ways: the “Register Transfer Level” (RTL), where data flow within the systems is represented like data flow between registers, and the “Algorithmic Level“, where a specific instruction set of statements define the sequence of operations in the system. The first way is commonly used for design of combinational logics, whereas the latter is mostly used for designs of sequential logics.

The second abstraction level is the “Structural model”. Typically, it describes the systems as gates or interconnected blocks, having several instantiated components in its architecture. That is the reason why it is called “Structural”: its architecture is based in a hierarchal representation of the system. It has the advantage of allowing a better management and control of what is really implemented and the possibility of reusing several modules of the system’s hierarchy.

8. Appendix 2: Development tools

Several software tools play an important role during the implementation stages of the project. They perform some data processing and provide debugging methods to the developer, allowing to take advantage of all the FPGAs features. In the following sections, the most relevant tools used in this project will be analyzed.

8.1 Overview of Xilinx ISE Software

Xilinx ISE ("Integrated Software Environment") is a software tool produced by Xilinx for synthesis and analysis of HDL project designs. This tool guarantees support for every FPGA and Evaluation Kits produced by Xilinx and provides many design solutions and utilities to the designer.

This work was implemented using the version "12.4" of this software. There, when starting a new ISE project, the designer should select the FPGA family and the respective Device and Package references. The default language (VHDL) should also be specified. Thereafter, the developer should start the design, creating new modules, adding already existing modules and eventually generating IPCORES. The design hierarchy must have a top-level design file where all the other low-level macros must be instantiated.

After the conclusion of the design, the XST ("Xilinx Synthesis Tool") will be used to synthetize the project. The tool will use the VHDL code to generate a netlist. All the hardware and interconnections that implement the modeled system will be generated and optimized for the target architecture. Possible syntax errors will also be detected and should be corrected by the designer before moving to the next steps. A detailed synthesis summary will be provided by ISE.

After the synthesis step is concluded, the ISE Software allows the developer to analyze the RTL and Technology schematics, where the connections between various components that XST has inferred are displayed. The designer can also perform a "behavioral simulation", simulating the system behavior using the "iSIM" simulator. This debugging method will be explained in the next section.

The next step in the ISE Design Flow, represented in the figure 8.1, is the "Design Implementation". Before beginning this process, a "Constraints File" must be created by the developer. There, physical input and output pins of the FPGA are specified and assigned to the corresponding VHDL design input and output ports. Furthermore, timing constraints may be added to ensure that the design follows specific timing requirements.

The "Translating Process" is the first sub-process in this second step. Here, the constraints from "Constraints File" (UCF) will be added to the merged netlist, timing specifications will be performed and some logical design rules will be checked.

The second sub-process, "Map", will ensure the correct allocation of CLBs and IOBs for all basic logic elements in the design. Furthermore, all the location and timing constraints will be processed and some target device optimizations will be performed. This step will output the resulting mapped netlist.

At the final sub-process, “Place and Route”, all the resources will be placed in particular locations of the chip and the interconnections between them will be routed. Here, the ISE software will proceed to a calculation in order to ensure that the chosen location for the hardware satisfies the timing constraints and guarantees the using of minimal hardware resources and minimum power consumption.

The last step in the ISE Design Flow is the “Device Programming”. After the design is completely placed and routed, the “BitGen” tool (“Xilinx bitstream generation program”) will produce a bitstream (“.bit” file) for Xilinx device configuration. This file must be downloaded to the ML605 board in order to test the project while operating in real hardware. The “iMPACT” tool will use a “USB” connection between the computer and the FPGA to implement a JTAG chain using the “Digilent Plugin” [72]. After initializing the chain, the configuration file will be used to program the device.

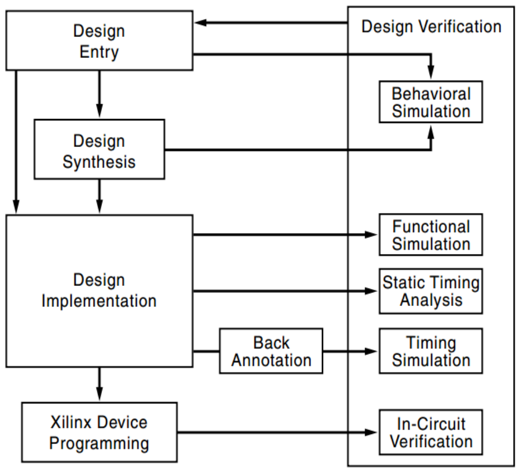


Figure 8.1 ISE Design Flow

ISE software also provides other useful tools that help the developer creating the best design. The “Timing Simulation” tool, for example, uses the block and routing delay information from a routed design to give a more accurate assessment of the behavior of the circuit under worst-case conditions.

8.2 Overview of Behavioral Simulation (using iSIM)

During the conception of a VHDL project, the developer must make specific tests just to know if a certain module has got the expected behavior. It would not be practical to make those primary tests directly in the FPGA, since the analysis of the results would be longer and difficult. Thus, the Xilinx ISE Software tool provides a simulator called “iSIM” that significantly simplifies the test and debugging steps in a project conception.

The Xilinx ISE Simulator (iSIM) is a HDL simulator that helps the developer to perform functional (behavioral) and timing simulations for VHDL, Verilog and mixed-language designs. This ISE Simulator environment is comprised by the following key elements [73]: VHDL Compiler, Verilog Compiler, HDL elaborator and linker, Simulation Executable and Graphical User Interface.

After synthetize the design that is going to be simulated, a “Testbench” should be created. A “Testbench” is a VHDL program that allows testing one or more modules of the project design (usually called “UUT: Units Under Test”). All the inputs and outputs of the UUT will be connected to VHDL signals. Thus, the designer can specify the value that will be assigned to each signal in a certain instant – this is called “stimulus insertion”. Finally, the simulator will run the created testbench and will display the resulting output values during the time interval specified by the user. The developer can now verify if the UUT meets the desired specifications, analyzing all the waves displayed in the “Waveform Window” and the results showed in the “Consol Panel”.

It is also important to notice that the described simulation environment is based on discrete events. Thus, the simulation time is composed by “simulation cycles”, which are much smaller than a system clock period, for example. The simulation time will advance due to repeatedly execution of those “simulation cycles”. In each one of them, all the processes that are sensitive to signals that have just experienced events will be resumed (activated) and executed. After the completion of all the executed processes, signals will be assigned sequentially during model execution but updated (conceptually) in parallel by the simulation engine [71]. The next “simulation cycle” will be scheduled accordingly to the time when another process will resume or another driver will become active.

This simulator also contains useful debugging tools that provide controlled execution of the source code to determine where problems may be occurring. The developer may “Step through the code line by line”, and verify if the design is working as expected or “Set breakpoints” on the specific lines of HDL code and run the simulation until a break point is reached [74].

8.3 Overview of Xilinx ChipScope Pro tools

After completing the design, it is useful to analyze its operation in real hardware. The simulation results can be considered “ideal results”, since technological limitations are not taken into account. Thus, the behavior of a certain design in a real hardware scenario may be different from the simulation one. For example, signal delays are always present and are dictated by the used technology and the system’s layout. They may interfere in the desired operation of the system. Therefore, it will be useful for the developer to have a complete “on-chip” access, in order to analyze some specific internal signals of the project. This flexible and real-time system verification may be done by using “Xilinx ChipScope Pro Tools”, which were designed to integrate test and measurement hardware components with the target design inside the supported Xilinx FPGA devices [64].

The figure 8.2 shows a typical block diagram of a system containing debug cores added by “Xilinx ChipScope Pro Tools”. These cores may be generated and instantiated into the design hierarchy. Then, the ISE Software will apply all the “Implementation Design” steps explained in section 8.1 and create the needed programming file to configure the FPGA. The developer may now analyze the system behavior in a host computer with the “ChipScope Pro Analyzer” tool.

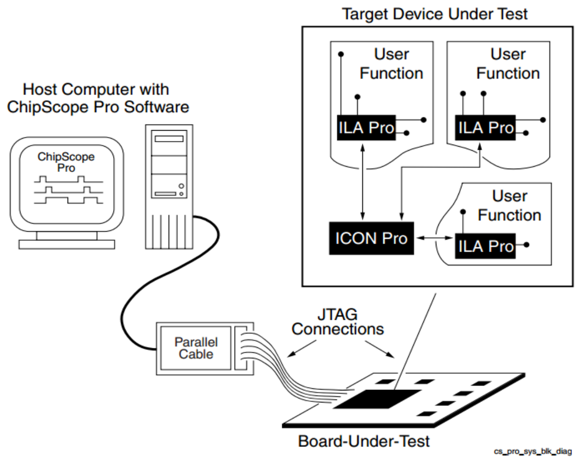


Figure 8.2 ChipScope Pro System Block Diagram [64]

The debugging cores use a "JTAG Boundary Scan port" to communicate with the host computer through the "JTAG download cable". In this project, a "USB Cable Connection Platform" was used.

The first module that must be added to the design hierarchy is the ICON core (Integrated Controller). It provides the interface between the "JTAG Boundary Scan port" of the FPGA and the "ChipScope Pro" debug cores, which are directly attached to the ICON core control ports (see figure 8.2) [75].

In this project, the ILA ("Integrated Logic Analyzer") was the chosen debug core type. It allows monitoring any internal signals of the design and its specifications should be defined by the developer. Naturally, it is possible to add multiple ILA cores to the design hierarchy. However, these cores will occupy several FPGA resources [76]. Thus, the maximum number of ILA cores that can be used may be different, depending on the FPGA.

The ILA "Trigger Ports" must be assigned to one or more internal signals of the design, allowing them to independently start the ILA data capture. Other features like "trigger width" or "trigger conditions" should also be specified. After that, the developer must define which signal will be the clocking source of the ILA unit, and specify the edges to trigger and capture data samples. Finally, the user must select the signals that will be captured and introduce the "Sample Data Depth" value. This parameter controls the number of data sample words that the ILA core can store in the sample buffer [76]. Therefore, those signals can be captured during real-time operation of the system and be visualized with the "ChipScope Pro Analyzer" software.

9. Appendix 3: Test platform

During the implementation stages of the project, the developer must make specific tests just to know if a certain module has got the expected behavior. Although it is possible to simulate the operation of a certain designed block using the previously described “Xilinx iSIM” tool, the results of the operation in real hardware scenario not always match with the simulated ones. Thus, to avoid unexpected and incorrect results, a “Test Platform” was created. It has got a flexible structure, allowing the introduction of any VHDL module in the chain, making possible the test and validation in hardware. Therefore, some of the blocks that compose the transmitter system were designed and tested with the help of this platform.

After the “Unit Under Test” had been inserted in the block chain, the “MATLAB” software tool was used for creating a script that is going to generate some specific data, represent it in the desired fixed-point binary format and send it to the FPGA through a serial connection. The data processed by the “Unit Under Test” will be sent back to MATLAB, which will decode and show the results. In the end of this section, a simplified block diagram of the entire system is illustrated. Each represented block is going to be described:

- MATLAB Script

  The data that is going to be processed must be inserted in a vector at the beginning of the script. Then, the data will be converted into the desired fixed-point binary representation. Finally, the binary values will be sent to FPGA through serial port by using the available functions *fopen, fwrite* and *fclose.* It is important to notice that if the binary representation of the data has got more than eight bits, the script will fragment each value before sending, since the “UART” block in the VHDL project only accepts data transfers with the maximum of eight bits.

  After the sending process is completed, the script will wait until the input buffer starts to receive the processed 8-bit words sent by the UART module. Those words will be concatenated to restore the original binary word length. The script will also plot the processed results.

- UART

  The “Universal Asynchronous Receiver/Transmitter” is a VHDL module that translates 8-bit values between serial and parallel forms. In this “Test Platform”, the RS-232 communication standard allows the communication through serial port between the UART and the computer, at a transfer rate of 115200 bps.

- Input Memory

  This module receives the 8-bit words provided by the UART and stores each value in a position of the memory array. Its size is limited to the maximum space available in the Block RAMs of the FPGA and can be configured by the user before compiling the design.

- Auxiliary Block 1

  The data that is going to be processed may have more than eight bits. Thus, this module has got the function of reading a certain number of memory positions and aggregate all those 8-bit words, in order to restore the original binary value that should be processed by the “Unit Under Test”. The maximum size of an aggregated word is 120 bits.

- Unit Under Test

  This is the module where the block that is going to be tested should be inserted. The connections of this module with the rest of the system are fixed. Thus, when inserting a new block for testing, the user should create the needed additional logic to correctly interconnect the block with the “Unit Under Test” module.

- Auxiliary Block 2

  This module was conceived to receive each processed word by the “Unit Under Test” module (with a maximum size of 120 bits) and fragment the data in multiple 8-bit words. Without this stage, the processed values cannot be stored in the “Output Memory”, since that it can only store words with 8-bit length.

- Output Memory

  Each position of this memory will store an 8-bit word provided by the “Auxiliary Block 2”. When all the values were processed, fragmented and stored in this memory, the sending process through serial port can be started. The size of this memory is also limited and can be configured by the user before compiling the design.

- Control Unit

  This is the unit that controls all the system’s operation, ensuring the needed synchronism and handshaking. It uses four counters to define addresses and a Finite-State Machine (FSM) that can be conceptually divided in four steps:

  - Step 1: Reading Process

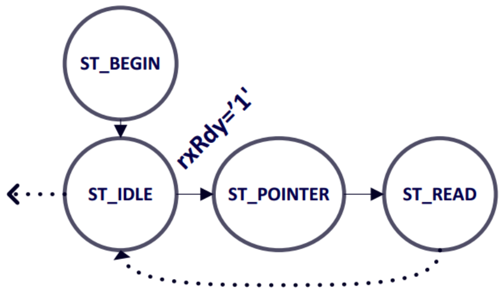


Figure 9.1 Reading Process

The initial state, “ST_BEGIN”, initializes a counter with an “Input Memory” address. Then, in “ST_IDLE”, the FSM waits for the UART signal (“rxRdy”) to read data from the serial port and forward it to the “Input Memory”. The “ST_POINTER” state increments the counter that defines the address of the “Input Memory” that is going to be written. Finally, in “ST_READ”, the value read from the UART module will be stored into the correct “Input Memory” position. The “Reading Process” will be repeated until the reception of the data is completed.

- Step 2: Processing Data

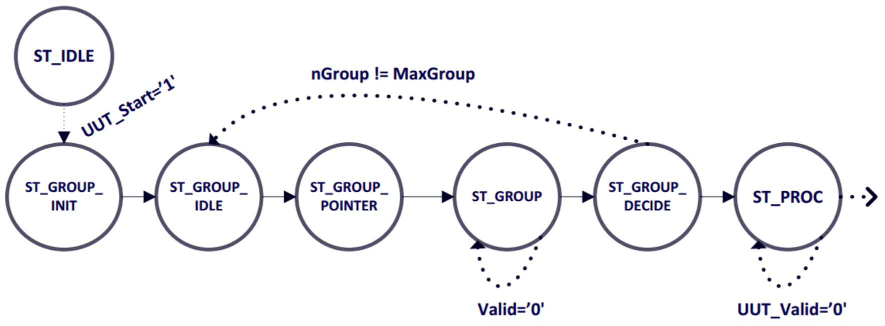


Figure 9.2 Processing Data

When the beginning of the “Processing Stage” is signalized, the FSM goes to the “ST_GROUP_INIT” state, where the counter that defines the number of the fragment that must be aggregated will be initialized. Then, the “ST_GROUP_POINTER” state will increment the number of the fragment that is going to be stored and the address of “Input Memory”. The “ST_GROUP_IDLE” is just a transition state. In “ST_GROUP”, the “Auxiliary Block 1” will be signalized to store the fragment. This state will wait the process is finished. If the final word with all the fragments aggregated is not completed yet, the process must be repeated. Thus, the FSM will leave the “ST_GROUP_DECIDE” state and will go to “ST_GROUP_IDLE” again. On the other hand, if the aggregation is completed, the word provided by the “Auxiliary Block 1” module will be processed by the “Unit Under Test” (“ST_PROC”). This state will wait until the processing phase reaches the end.

- Step 3: Fragmenting Process

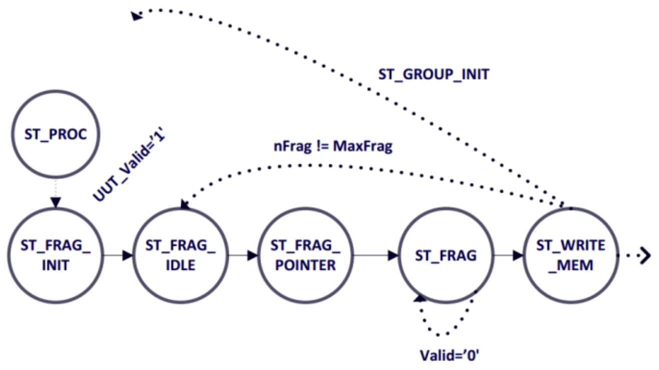


Figure 9.3 Fragmenting Process

This stage begins when the "UUT" finishes the processing of a single value. In "ST_FRAG_INIT", the counter that defines the number of the fragment that must be stored will be initialized. Then, in "ST_FRAG_POINTER", the number of the fragment that is going to be stored and the address of "Output Memory" where that fragment is going to be written will be incremented. The fragmenting process will be started, and the FSM will stay in the "ST_FRAG" state until the "Auxiliary Block 2" signals the availability of the desired fragment. After that, in "ST_WRITE_MEM", the fragment will be written in the "Output Memory". If all the fragments of the word are not stored yet, the process must be repeated, by going to "ST_FRAG_IDLE" state again. However, if the fragmenting and storing of a certain word is finished, another input data must be processed. Thus, the FSM must go to the state "ST_GROUP_INIT".

- Step 4: Sending Process

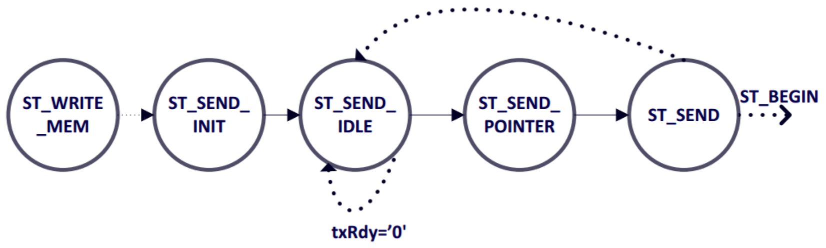


Figure 9.4 Sending Process

After all the processing and storing in the "Output Memory" is concluded, this set of states ensures the correct sending through UART module of the content of each position of the memory. Therefore, the "txRdy" signal provided by UART will be essential to control the sending of the data through serial port. When the sending process is finished, the FSM goes to "ST_BEGIN", and is ready for receiving new values, if it is necessary.

This "Test Platform" was designed and tested in two different "Xilinx" Evaluation Boards: *Spartan6-XC6SLX45(CSG324)* and *Virtex6 –XC6VLX240T-1FF1156 (ML605 Evaluation Kit).* The following table summarizes some of the resources consumed by this project:

Logic	*% Utilization (Spartan-6)*	*% Utilization (Virtex-6)*
Slice Registers	3	1
Slice LUTs	19	1
Occupied Slices	24	3
Bounded IOB	5	2
BUFG/BUFGMUXs	18	12

Table 9.1 Total hardware usage by the "Test platform"

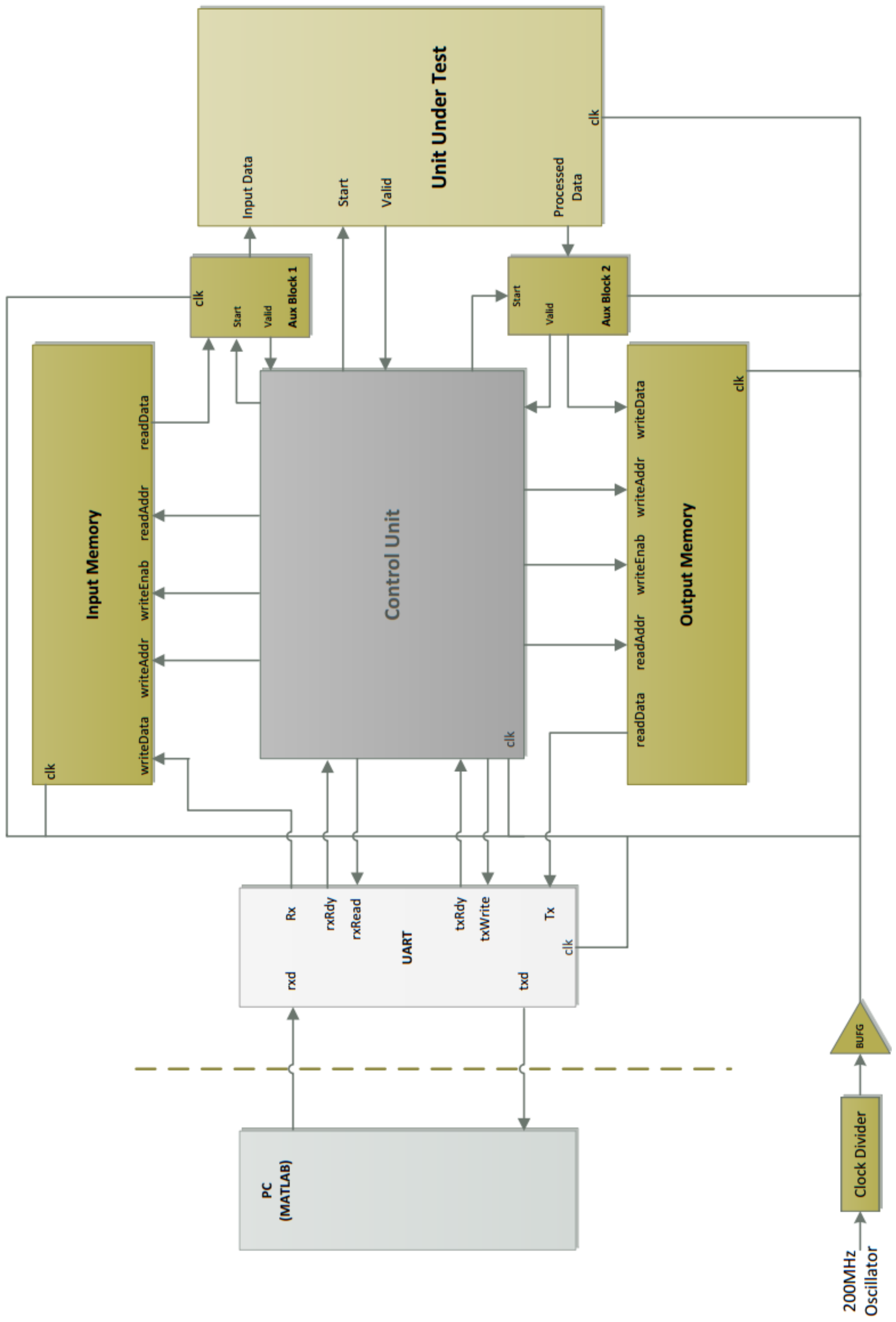


Figure 9.5 Simplified Block diagram

10. References

[1] Berthold, J.; Saleh, A.A.M.; Blair, L.; Simmons, Jane M., "Optical Networking: Past, Present, and Future," *Lightwave Technology, Journal of* , vol.26, no.9, pp.1104-1118, 2008

[2] Sato, K.; Hasegawa, H., "Optical Networking Technologies That Will Create Future Bandwidth-Abundant Networks", *J. Opt. Commun. Netw*., A81-A93, 2009

[3] Minnesota Internet Traffic Studies. URL http://www.dtc.umn.edu/mints/home.html

[4] Agrawal G. P., *Fiber-Optic Communication Systems*: Wiley-InterScience, 2002

[5] Keiser G., *Optical Fiber Communications*, 3rd Edition: McGraw-Hill, 2000

[6] Alferness R. C., et al., "The Evolution of Optical Systems: Optics Everywhere," *Bell Labs Technical Journal*, vol. 5, pp. 188-202, 2000

[7] Kikuchi K., *Digital Coherent Optical Communication Systems: Fundamentals and Future Prospects*: IEICE, 2011.

[8] Li, Yu, "Optical Intensity Modulators for Digital and Analog Applications", *Lightwave Technology, Journal of*, Vol. 21, 2003

[9] Howerton, M.; Moeller, R.; Greenblatt, A. and Krahenbuhl, R., "Fully packaged, Broad-Band LiNbO3 Modulator with Low Drive Voltage," *IEEE Photon. Technol. Lett. 12*, pp. 792-794, 2000

[10] Lau, A.; Barros, D.; Kahn, J., "Coherent Detection in Optical Fiber Systems," *Opt. Express*, vol. 16, pp. 753-791, 2008.

[11] Gilad Goldfarb, "Digital Signal Processing Techniques for Coherent Optical Communication", University of Central Florida, 2005

[12] Nokia Siemens Networks, "Space Division Multiplexing: A New Milestone in the Evolution of Fiber Optic Communication", White Paper, 2013

[13] Eyre, J.; Bier, J., "The Evolution of DSP Processors," *Signal Processing Magazine, IEEE* , pp.43,51, 2000

[14] Woods R., *FPGA-based Implementation of Complex Signal Processing System*: Wiley, 2008

[15] Altera FPGAs. URL http://www.altera.com/devices/fpga/fpga-index.html

[16] Birkner, John M., *PAL Programmable Array Logic Handbook*: Monolithic Memories, 1978

[17] Xilinx. URL http://www.xilinx.com

[18] EDN "No Room for Second Place" URL http://www.edn.com/electronics-news/4320763/No-room-for-Second-Place

[19] *7 Series FPGAs Overview*, Xilinx, DS180, Rev. 1.14, 2013

[20] Xilinx FPGAs. URL http://www.xilinx.com/products/silicon-devices/fpga/index.htm

[21] Altera Stratix-10. URL http://www.altera.com/devices/fpga/stratix-fpgas/stratix10

[22] Waegemans R.; Herbst S., "10.7 Gb/s Electronic Predistortion Transmitter using Commercial FPGAs and D/A Converters Implementing Real-time DSP for Chromatic Dispersion and SPM Compensation", *Opt. Express,* May 2009

[23] Watts, P.; Waegemans, R., "An FPGA-Based Optical Transmitter Design Using Real-Time DSP for Advanced Signal Formats and Electronic Predistortion," *Lightwave Technology, Journal of*, vol.25, no.10, pp.3089,3099, Oct. 2007

[24] Seimetz M., *High-Order Modulation for Optical Fiber Transmission*: Springer, 2009

[25] Al-Bermani, A.; et al., "Nonlinear effect of IQ Modulator in a Real-time Synchronous 16-QAM Transmission System," *Photonics Conference (PHO)*, 2011 IEEE, pp.763,764, 9-13 Oct. 2011

[26] Makovejs S., "High-speed Optical fiber Transmission using Advanced Modulation Formats", University College London, 2011

[27] Behrens C., "Mitigation of Nonlinear Impairments for Advanced Optical Modulation Formats", University College London, 2012

[28] *SHF 807 Linear Broadband Amplifier,* SHF Communication Technologies AG, Version 3, 2011

[29] Luís Manuel de Sousa Pessoa, "Compensation of Fibre Impairments in Coherent Optical Systems", Universidade de Aveiro, 2010

[30] Proakis J., *Digital Communications, 4th Edition*: McGraw-Hill, 2001

[31] Bruno Jorge Ramos Pereira, "Simulador de Sistemas de Comunicações Ópticas (OSIP)", Universidade de Aveiro, 2012

[32] Lathi B., *Modern Digital and Analog Communication Systems, 3rd Edition:* Oxford University Press, 1998

[33] Parker M., *Digital Signal Processing*: Elsevier Inc., 2010

[34] Karam, L.J.; et al., "Digital Filtering", *Digital Signal Processing Handbook*, CRC Press LLC , 1999

[35] Thede L., *Practical Analog and Digital Filter Design*: Artech House Inc., 2004

[36] Tanton J., *Encyclopedia of Mathematics:* Facts On File Inc., 2005

[37] OSIP. URL http://www.av.it.pt/osip/

[38] Tan L., *Digital Signal Processing: Fundamentals and Applications*: Elsevier Inc., 2008

[39] Shafik, R.A.; et al., "On the Extended Relationships Among EVM, BER and SNR as Performance Metrics," *ICECE International Conference*, 2006

[40] Schmogrow R., et al, "Error Vector Magnitude as a Performance Measure for Advanced Modulation Formats", *IEEE Photonics Technology Letters*, Vol. 24, pp. 61-63, 2012

[41] Altera. URL http://www.altera.com/products/fpga.html

[42] Compton K., Hauck S., "An Introduction to Reconfigurable Computing", *Northwestern University, Dept. of ECE Technical Report*, 1999.

[43] *Virtex-6 FPGA Configurable Logic Block,* Xilinx, UG364, Version 1.2, 2012

[44] *Virtex-6 FPGA Memory Resources*, Xilinx, UG363, Version 1.6, 2011

[45] *Virtex-6 FPGA SelectIO Resources,* Xilinx, UG361, Version 1.3, 2010

[46] *Virtex-6 FPGA DSP48E1 Slice*, Xilinx, UG369, Version 1.3, 2011

[47] *Virtex-6 FPGA Clocking Resources,* Xilinx, UG362, Version 2.2, 2013

[48] *Virtex-6 FPGA Clocking Resources,* Xilinx, Xilinx Training Presentation, 2012

[49] *Virtex-6 Mixed-Mode Clock Manager (MMCM) Module,* Xilinx, DS737, Version 1.00a, 2009

[50] *ML605 Hardware User Guide,* Xilinx, UG534, Version 1.5, 2011

[51] *ML605 Evaluation Kit - Hardware Setup Guide,* Xilinx, 2011

[52] *Virtex-6 Family Overview,* Xilinx, DS150, Version 2.4, 2012

[53] *LogiCORE IP Clocking Wizard,* Xilinx, DS709, Version 1.8, 2010

[54] *Xilinx Timing Constraints User Guide,* Xilinx, UG616, Version 1.0.0, 2008

[55] Fixed-Point Tutorial. URL http://www.biccari.com/docs/Fixed_point_tutorial.pdf

[56] *LogiCORE IP FIR Compiler v6.1,* Xilinx, DS795, 2010

[57] *LogiCORE IP CORDIC v4.0,* Xilinx, DS249, 2011

[58] *LogiCORE IP Multiplier,* Xilinx, DS255, Version 11.2, 2009

[59] *AD9961/AD9963 Datasheet,* Analog Devices, Rev. A, 2012

[60] *AD9963-EBZ Quick Start Guide,* Analog Devices, Rev. A, 2010

[61] *AD-DAC-FMC-ADP Adapter Board Quick Start Guide,* Analog Devices, 2010

[62] "Electronics-Tutorial: RC Filters" URL: http://www.electronics-tutorials.ws/filter/filter_3.html

[63] *AD9961/AD9963-EBZ Schematic,* Analog Devices, Rev. A, 2009

[64] C*hipscope Pro Software and Cores,* Xilinx, UG029, Version 12.3, 2010

[65] *Fujitsu FTM7962EP,* Fujitsu Optical Components Limited, Rev. C, 2012

[66] *Single Polarization QPSK Modulator Bias Controller for DQPSK/QAM Application,* YY Labs Inc., Rev. 1.0, 2012

[67] Asad Abidi, "How Phase Noise Appears in Oscillators", University of California, 1997

[68] Ricardo Ferreira, “Processamento Digital de Sinal em Sistemas de Detecção Coerente”, Universidade de Aveiro, 2012

[69] T. Huynh, et al., "Effects of Phase Noise of Monolithic Tunable Laser on Coherent Communication Systems," *Opt. Express*, 2012

[70] *Apex AP2440A Series OSA,* Apex Technologies, 2005

[71] Ashenden P., *The Designer’s Guide to VHDL, 2nd Ed.:* Systems on Silicon, Morgan Kaufmann Publishers, 2001

[72] Digilent Plugin. URL http://www.digilentinc.com/Products/Detail.cfm?Prod=DIGILENT-PLUGIN

[73] *ISE Simulator In-Depth Tutorial,* Xilinx, UG682, Version 12.3, 2010

[74] *iSIM Userguide,* Xilinx, UG660, Version 12.4, 2010

[75] *LogiCORE IP ChipScope Pro Integrated Controller (ICON)*, Xilinx, DS646, Version 1.05a, 2011

[76] *ChipScope Pro Integrated Logic Analyze*, Xilinx, DS299, Version 1.03a, 2010