Universidade Federal de Juiz de Fora

Engenharia Elétrica

Programa de Pós-Graduação em Engenharia Elétrica

Diogo Fernandes

Low-Cost Implementation Techniques for Generic Square and Cross M-QAM Constellations

Juiz de Fora

2015

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Fernandes, Diogo.
Low-Cost Implementation Techniques for Generic Square and Cross M-QAM Constellations / Diogo Fernandes. – 2015.
116 f. : il.

Orientador: Moises Vidal Ribeiro
Dissertação de mestrado – Universidade Federal de Juiz de Fora, Engenharia Elétrica. Programa de Pós-Graduação em Engenharia Elétrica, 2015.

1. Modulação em amplitude e quadratura M-ária. 2. Comunicações digitais de dados. 3. Field programmable gate array. I. Ribeiro, Moisés, orient. II. Título.

Diogo Fernandes

Low-Cost Implementation Techniques for Generic Square and Cross M-QAM Constellations

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétricada Universidade Federal de Juiz de Fora, na area de concentração em sistemas eletrônicos, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Moises Vidal Ribeiro

Juiz de Fora

2015

Diogo Fernandes

Low-Cost Implementation Techniques for Generic Square and Cross M-QAM Constellations

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétricada Universidade Federal de Juiz de Fora, na area de concentração em sistemas eletrônicos, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Aprovada em: 31 de agosto de 2015

I dedicate this work to the Fernandes's family, my mother Maria Helena, my father Sebastião, my brothers Thiago and Victor, and my girlfriend Marcela.

ACKNOWLEDGMENTS

My father Sebastião Fernandes and my mother Maria Helena Fernandes for their support and dedication throughout my life.

My brothers Thiago Fernandes and Victor Fernandes for their patience and exchanges of experience.

My beloved girlfriend Marcela Fernandes, for all the love and encouragement.

My advisor Ph.D. Moises Ribeiro and my professor Ph.D. Weiler Finamore for guiding and leading me throughout the implementation and writing of this thesis.

To my company and laboratory colleagues for the important contribution in the project.

To my friends who always support and cheer me in moments of relaxation.

I would like to thank FINEP, FAPEMIG, CNPq, CAPES, P&D ANEEL, CEMIG, INERGE, and, specially, Smarti9 and Wari, for their financial supports.

“Um sonho é um sonho até que se acorde. Um homem pode acalentar esse sonho ou arregaçar as mangas e pensar no que é preciso para torná-lo realidade.”

(Eike Batista)

RESUMO

Este trabalho tem como objetivo apresentar técnicas com complexidade computacional reduzida para implementação em hardware do modulador de amplitude em quadratura M-ária (M-ary *quadrature amplitude modulation* - M-QAM) de elevada ordem, que pode ser viável para sistemas banda larga. As técnicas propostas abrangem as constelações M-QAM quadradas e cruzadas (número par e ímpar de bits), a regra de decisão abrupta (*hard decison rule*), derivação de constelações M-QAM de baixa ordem das de elevada ordem. A análise de desempenho em termos de taxa de bits errados (*bit error rate* - BER) é realizada quando os símbolos M-QAM são corrompidos por ruído Gaussiano branco aditivo (*additive white Gaussian noise* - AWGN) e ruído Gaussiano impulsivo aditivo (*additive impulsive Gaussian noise* - AIGN). Os resultados de desempenho da taxa de bits errados mostram que a perda de desempenho das técnicas propostas é, em média, inferior a 1 dB, o que é um resultado surpreendente. Além disso, a implementação das técnicas propostas em arranjo de portas programÃ¡veis em campo (*field programmable gate array* - FPGA) é descrita e analisada. Os resultados obtidos com as implementações em dispositivo FPGA mostram que as técnicas propostas podem reduzir consideravelmente a utilização de recursos de hardware se comparadas com as técnicas presentes na literatura. Uma melhoria notável em termos de redução da utilização de recursos de hardware é conseguida através da utilização da técnica de modulação M-QAM genérica em comparação com a técnica de regra de decisão heurística (*heuristic decision rule* - HDR) aprimorada e uma técnica previamente concebida, a tÃ©cnica HDR. Com base nas análises apresentadas, a técnica HDR aprimorada é menos complexa do que a técnica HDR. Finalmente, os resultados numéricos mostram que a técnica de modulação M-QAM genérica pode ser oito vezes mais rápida do que as outras duas técnicas apresentadas, quando um grande número de símbolos M-QAM (p. ex., > 1000) são transmitidos consecutivamente.

Palavras-chave: Modulação em amplitude e quadratura M-ária. Comunicações digitais de dados. Arranjo de portas programável em campo.

ABSTRACT

This work aims at introducing techniques with reduced computational complexity for hardware implementation of high order M-ary quadrature amplitude modulation (M-QAM) which may be feasible for broadband communication systems. The proposed techniques cover both square and cross M-QAM constellations (even and odd number of bits), hard decision rule, derivation of low-order M-QAM constellations from high order ones. Performance analyses, in terms of bit error rate (BER) is carried out when the M-QAM symbols are corrupted by either additive white Gaussian noise (AWGN) or additive impulsive Gaussian noise (AIGN). The bit error rate performance results show that the performance loss of the proposed techniques is, on average, less than 1 dB, which is a remarkable result. Additionally, the implementation of the proposed techniques in field programmable gate array (FPGA) device is described and outlined. The results based on FPGA show that the proposed techniques can considerably reduce hardware resource utilization. A remarkable improvement in terms of hardware resource utilization reduction is achieved by using the generic M-QAM technique in comparison with the enhanced heuristic decision rule (HDR) technique and a previously designed technique, the HDR technique. Based on the analyses performed, the enhanced HDR technique is less complex than the HDR technique. Finally, the numerical results show that the generic M-QAM technique can be eight times faster than the other two techniques when a large number of M-QAM symbols (e.g., > 1000) are consecutively transmitted.

Keywords: M-ary quadrature amplitude modulation. Digital data communication. Field programmable gate array.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

1D one-dimensional

2D two-dimensional

3D three-dimensional

AC alternating current

ADC analog to digital converter

AGC automatic gain control

AIGN additive impulsive Gaussian noise

AM amplitude modulation

AWGN additive white Gaussian noise

BER bit error rate

BPL broadband over power lines

BPSK binary phase shift keying

CAD computer aided design

CSI channel state information

DC direct current

DDR double data rating

DSP digital signal processor

FIFO first in, first out

FM frequency modulation

FPGA field programmable gate array

HDR heuristic decision region

LE logic elements

LPTV linear and periodically time-varying

LSB least significant bit

LTI linear and time invariant

LTV linear and time-varying

LUT look-up table

MAPSK multiple amplitude and phase amplitude and phase shift keyed

ML maximum likelihood

MSB most significant bit

MUX multiplexer

OFDM orthogonal frequency division multiplexing

PAM pulse amplitude modulation

PLC power line communication

PLL phase locked loop

PSD power spectral density

QAM quadrature amplitude modulation

QASK quadrature amplitude-shift-keyed

QDRII quad data rate

RAM random access memory

ROM read only memory

RX receiver

SRC silicon rectifier controlled

SDRAM synchronous dynamic random access memory

SM state machine

SNR signal to noise ratio

SRAM static random access memory

TX transmitter

VHDL very high speed integrated circuits hardware description language

CONTENTS

1 INTRODUCTION

The recent standardization efforts toward broadband communication systems [1–4] and the needed for increasing the spectral efficiency of data communication systems to fulfil demands related to smart cities [5,6], indicate that, for advancing in the current state of the art related to communication technologies, new modulation techniques have to be designed. However, the global demand for energy-saving telecommunication devices imposes severe hardware energy consumption restrictions. Recognizing the utmost importance of energy consumption in communication systems, [7] introduced parameters to quantify this feature in a communication system.

Regarding communications channels, in a depth knowledge of the time-varying behavior of the channel can provide important information to feed dynamic resource allocation techniques or adaptive modulation [8,9]. As a result, the number of bits allocated in each subcarrier of a multi-carrier (number of bit transmitted in each subcarrier could surpass 15) or single carrier schemes [9,10], can be high and, as a consequence, the computational complexity associated with the use of high order M-ary quadrature amplitude modulation (QAM) constellations can be considerably large (in the new generation of transceivers, constellations with huge size, such as 2^{15}-QAM, are employed [11]).

Considering M-QAM techniques, one can state out that M-QAM modulating/demodulating techniques [12–14] are old challenges for digital communication experts. For instance, the improvements on bit error rate (BER) by using the Gray-coded square M-QAM constellations were reported in [15,16], while [17–20] addressed efficient ways to transmit odd number of bit in M-QAM symbols, the so called cross M-QAM constellations. The cross M-QAM constellations translates the bits from the corners of a rectangular M-QAM constellation to the top and bottom sides of it. Furthermore, [21,22] discussed concerning about computational complexity of M-QAM techniques, by focusing on efficient approaches to implement 16-QAM in an field programmable gate array (FPGA) device, while [23,24] paid attention to several approaches to modulate and to demodulate square and cross M-QAM symbols considering $M \in \{4, 8, 16, 32, \ldots, 65536\}$ by using the so-called heuristic decision region (HDR) technique. The HDR allows for a reduction of hardware resource utilization by M-QAM symbols in a FPGA device. This technique makes use of large memory storage at the transmitter, or large look-up table (LUT), and considerable number of logic elements because it is based on the state machine (SM) concept. The aforementioned works are based on hard decision rule, while a soft decision rule in a digital signal processor device was pointed out in [25]. In spite of improved performance of the soft decision, this approach demands high computational complexity.

It can be seen from the aforementioned references that if the additive noise is a white Gaussian random process and the hard decision rule is adopted at the receiver side, then hardware resource utilization and computational complexity can be reduced when the order of M-QAM constellation increases. These issues were initially discussed in [23, 24], however more investigations are needed to come up with techniques that considerably reduce the need for memory storage and logic elements, addressing both square and cross M-QAM constellations. Note that the current standard for broadband power line communication (PLC) systems, IEEE P1901 [1], only covers square M-QAM constellations, a raised hypothesis is that the computational complexity associated with current cross M-QAM techniques is too high for hardware implementation.

1.1 Objectives

Usually, the computational burden of implementing modulation and demodulation techniques relies on the mapping and de-mapping/detecting procedures [12–14]. The search for a low-cost hardware implementation of these procedures is a timely and important issue to be considered. In this context, this thesis focuses on the following main contributions:

1. To present a generic mapping and de-mapping/detecting procedures for M-QAM techniques, in order to transmit messages with even and odd number of input bits that satisfies a Gray mapping rule and presents low computational complexity, mainly, when M increases.
2. To implement the generic mapping and de-mapping/detecting procedures for M-QAM techniques in FPGA devices and comparatively verifies their BER performance and hardware resource utilization in terms of maximum operating frequency, energy consumption, delay, latency, memory usage and logic elements usage.

1.2 Thesis Outline

The thesis is organized as follows: Chapter 2 presents the problem formulation, describing the communication channel, the state of the art about M-QAM techniques, comments about high data-rate communication systems and the mathematical formulation of the addressed problem; Chapter 3 addresses the proposed techniques, and their implementation in FPGA devices; Chapter 4 focuses on the performance analyses; and Chapter 5 states the conclusions.

2 PROBLEM FORMULATION

The demands for high data-rate and low-cost transceivers are the main purposes to come up with simple and low computational complexity high order M-ary QAM ($M > 2^8$). In order to formulate the problem related to this challenging issue, this chapter is organized as follows: Section 2.1 addresses a brief discussion about the adopted mathematical channel model, Section 2.2 focuses on a review about M-QAM techniques, while the mathematical formulation of the problem investigated in this thesis is presented in Section 2.3.

2.1 Communication Channel

In the literature there are few mathematical models for the propagation of signals through communication channels. One of the most discussed model was proposed in [37], which represents the communication channel as a linear and time invariant (LTI) system with a frequency response expressed by

$$H(f) = \sum_{i=1}^{P} g_i e^{-(a_0+a_1 f^r)d_i} e^{-j2f\pi\tau_i}, \tag{2.1}$$

in which P is the number of channel paths, i is the index for the i-th path. The path gain of index i is represented by g_i. The length and the delay of the i-th route is represented by d_i and τ_i, respectively. a_0 and a_1 are attenuation parameters and r is the attenuation factor exponent.

In spite of taking the multipath propagation into account, the LTI model does not include the temporal variation of loads. In [38], the time dependent frequency response variation is observed in a electric power grid during an interval of 2 hours. One way to represent such temporal variation of the communication channel [39] is to introduce a change in (2.1) such that g_i is replaced by the random variable $g_{i,t}$ with a Gaussian probability density function, i.e.,

$$g_{i,t} \sim \mathcal{N}(g_i, \sigma_{g_i}^2), \tag{2.2}$$

in which g_i represents the average gain of the i-th path and $\sigma_{g_i}^2$ is the variance that determines the variability of the communication channel. This channel model is named linear and time-varying (LTV) [39]. Figure 1 shows the frequency response variation of communication channel as presented in [40].

According to [41], communication channels show abrupt temporal variations in their frequency responses, which are synchronous with the main frequency of the electric power grids. There variations occurs when silicon rectifier controlled (SRC) based devices are used to provide alternating current (AC)/direct current (DC) conversion. One way to address such periodicity is to consider that the communication channel is linear and

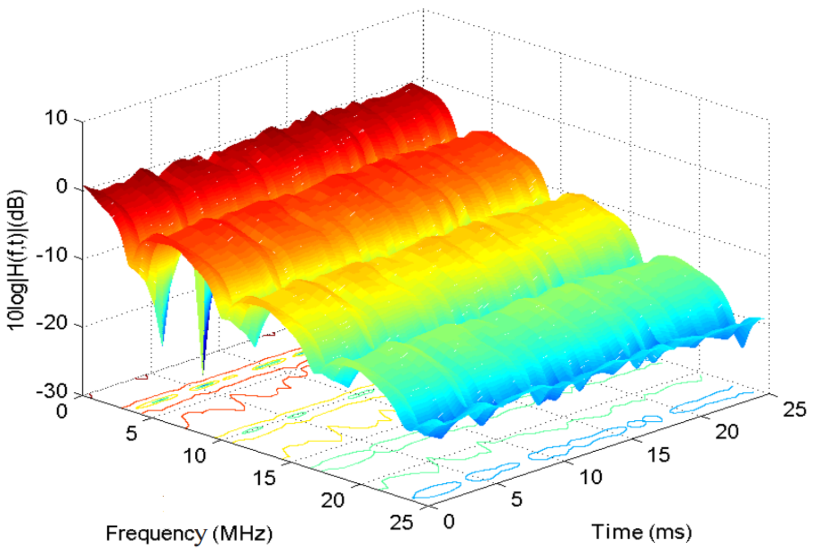


Figure 1: Frequency response variation in a communication channel.

periodically time-varying (LPTV) with frequency response given as [33]

$$H_{LPTV}(f,t) = \begin{cases} H_{LTV,1}(f,t), & \text{if } \left(\frac{k}{2f_0} - \frac{T}{2}\right) < t < \left(\frac{k}{2f_0} + \frac{T}{2}\right) \\ H_{LTV,2}(f,t), & \text{otherwise} \end{cases} . \tag{2.3}$$

in which $H_{LTV,1}(f,t)$ has different parameters of $H_{LTV,2}(f,t)$. The former represents the communication channel when the main voltage is close to zero crossing, and the latter one represents the communication channel at other time instants. It is considered that $k \in \mathbb{Z}$, f_0 is the mains frequency that, in Brazilian case, is 60 Hz, T is the interval time in which the channel response remains with the parameters of $H_{LTV,1}(f,t)$. Overall, the LPTV channel model is a combination of two distinct LTV channels.

In order to better understand the communication channel model, a bus with multiple branches, as shown in Figure 2, can be used to illustrate the signal propagation in a wired communication system. At each end of branches there are time varying loads that can be the transmitter (TX) and/or the receiver (RX) device. These devices can be connected and disconnected randomly, some of then are synchronously switched on/off with mains frequency. The temporal and spectral variation lead to a impedance mismatch between the TX and RX devices, and, as a consequence, portions of the waves are reflected and retransmitted. The received signal is the sum of several reflected signals from distinct pathways, along with the directly transmitted signal [42], resulting in what is named multipath propagation model of the communication channel that is also present in wireless communications.

These kind of noises are presented, illustrated and discussed by [43, 44]. Usually, noises in the communication channels are constituted by components that are yielded by distinct loads. It is possible to classify these noises in two different groups: background

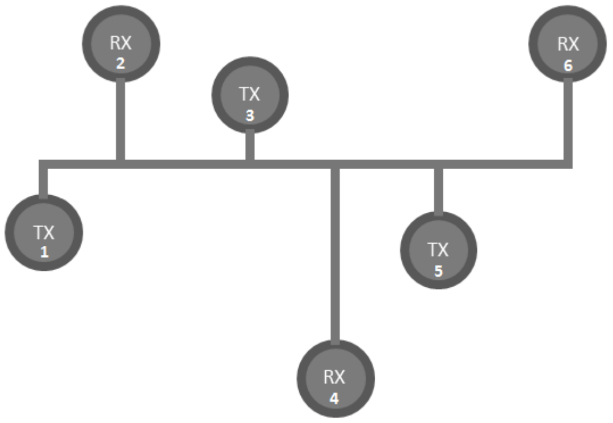


Figure 2: Typical wired communication system topology.

noise and impulsive noise (see detailed discussion in [43] and [44]). Figure 3 shows these types of noise in communication channels.

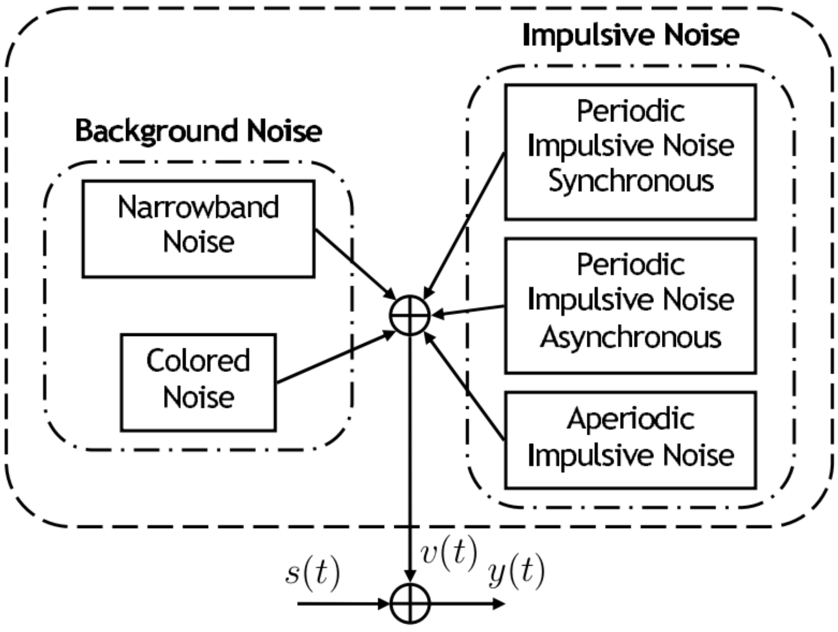

Impulsive Noise
Background Noise
Narrowband Noise
Colored Noise
Periodic Impulsive Noise Synchronous
Periodic Impulsive Noise Asynchronous
Aperiodic Impulsive Noise
$v(t)$
$s(t)$
$y(t)$

Figure 3: Types of noise in communication channels.

It is well established that the background noise can be considered as the sum of the following components:

- Narrowband Noise: it is yielded by the induction of radio broadcasting signals in the communication channel. The source of these signals are radio stations using amplitude modulation (AM) and frequency modulation (FM), and also amateur radio. The noise power may vary throughout the day and also in relation to the distance of the transmission source.

- Coloured Noise: it has low level of power spectral density (PSD), which decreases with increasing frequency. This noise is yielded by the sum of numerous low-power noise sources present in the communication channel, such as consumer electronics. Its PSD can vary in minutes or even in hours.

On the other hand, the impulse noise part can be considered as the sum of the following components. It is important to emphasize that most of them relies on PLC channels, see Annex A.

- Periodic impulsive noise asynchronous with mains frequency: it is caused by power supplies switching, usually with frequencies between 50 and 200 kHz, having frequency spectrum spaced components in accordance with the pulse repetition rate of the switching.
- Periodic and impulsive noise synchronous with mains frequency: it is yielded by SRC that are synchronous with the mains frequency, its frequencies either is 50 or 100 Hz in Europe and in the frequencies of 60 and 120 Hz in the United States and Brazil. It has a duration of only few microseconds and its PSD decreases with the increasing of the frequency.
- Aperiodic impulsive noise: This type of noise is produced by transients caused by the connection and disconnection of electric devices (loads) on the electric power grids as well as their switching. These kind of impulse has a random duration between few microseconds and few milliseconds with random interarrival time. The PSD of this noise can reach values up to 50 dB above the PSD of the background noise.

Following Figure 3, it is assumed that a model of the communication channel is given by $y(t) = s(t) + v(t)$, in which $s(t)$ is the transmitted signal and $v(t)$ is the additive noise. Assuming perfect synchronization and the use of a matched filter at the receiver side, then the vector representation of the i-th received signal is given by

$$\mathbf{y}_i = \mathbf{s}_i + \mathbf{v}_i, \tag{2.4}$$

in which $\mathbf{s}_i = \mathbf{s}_{m_i} \in \mathbb{R}^2$ is the constellation point associated with the $\mathbf{m}_i$ message. Note that $\mathbf{s}_{m_i}$ is constituted by in-phase and quadrature components. The vector $\mathbf{y}_i \in \mathbb{R}^2$ represents the channel output (demodulator input) and the vector $\mathbf{v}_i \in \mathbb{R}^2$ expresses the discrete time additive noise at the output of the communication channel. The vector $\mathbf{v}_i$ can be modeled as [45, 46]

$$\mathbf{v}_i = \mathbf{v}_{i,bkg} + \mathbf{v}_{i,nb} + \mathbf{v}_{i,pa} + \mathbf{v}_{i,ps} + \mathbf{v}_{i,imp}, \tag{2.5}$$

in which $\mathbf{v}_{i,bkgr}$ denotes the background noise, $\mathbf{v}_{i,nb}$ the narrowband noise, $\mathbf{v}_{i,pa}$ the periodic impulsive noise asynchronous to the mains frequency of the power system, $\mathbf{v}_{i,ps}$ the periodic impulsive noise synchronous to the mains frequency, and $\mathbf{v}_{i,imp}$ the asynchronous impulsive noise.

In this thesis it is assumed that

$$\mathbf{v}_i = \mathbf{v}_{i,bkg} + \mathbf{v}_{i,ps} + \mathbf{v}_{i,imp}, \tag{2.6}$$

because $\mathbf{v}_{i,nb}$ can be mitigated by filtering and $\mathbf{v}_{i,pa} + \mathbf{v}_{i,ps}$ can be considered as a single noise component. If one assumes that (2.6) is modeled as additive impulsive Gaussian noise (AIGN), then $\mathbf{v}_{i,bkgr} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ is a random vector with Gaussian distribution and mean vector equal to $\mathbf{0}$, and variance σ^2, that represents the background noise; $\mathbf{v}_{i,ps} \sim \mathcal{N}(\mathbf{0}, K_1\sigma^2\mathbf{I})$ denotes the discrete time version of an impulsive noise in the continuous time domain with interarrival interval $t_{arr,ps} = (1/2f_0)$ seconds, in which f_0 is the mains frequency of electric power systems (60 Hz in Brazil). Then, the duration of $t_{w,ps} = 100\mu$s is located in $(n/2)f_0$, $n = 0, 1, 2, ...$ And also $\mathbf{v}_{i,imp} \sim \mathcal{N}(0, K_2\sigma^2\mathbf{I})$ is the discrete time version of an impulsive component in the continuous time domain with interarrival time $t_{arr,imp}$ modeled as an exponential random variable with mean equal to $100ms$ and time lasting of $t_{w,imp} = 100\mu$s. The constants K_1 and K_2 are specified in order to generate noises with different levels of severity. This model is capable of representing communication channel scenarios with high severity once the impulsive noise can be modeled as white and Gaussian, which makes it possible to emulate the worst case scenario [23, 24]. In all the simulation results shown in this work, $K_1 = K_2 = 20$ dB. The adopted values are the same of [23, 24] and, as a consequence, comparative analyses can be easily performed.

2.2 State of The Art of M-QAM

Considering digital modulation techniques, it is well known that phase and amplitude mapping and detection techniques are old challenges for digital communication. The contributions [12–14] from 60's discussed an approximation to the probability of error derived for both coherent and phase-comparison demodulation by taking into account, efficient use of transmission power, minimum peak power or minimum average power, most of them with theoretical results only in [12]. Another paper, at the same decade, presented some simulation results [14] regarding two types of digital phase and amplitude modulated signals as well as approximated expressions for the symbol error probability and the channel capacity for a digital phase-modulated scheme. It was shown that the phase and amplitude modulation scheme offer power advantages over the phase-only scheme. Some configurations of these schemes that change the constellation points based on the data input were presented in [47]. Figure 4 shows four suggested constellations when the number of bits transmitted in each symbol is equal to 2. As a result of these investigations, digital modulation schemes started to evolve considering the in-phase (I) and quadrature (Q) components in many different configurations.

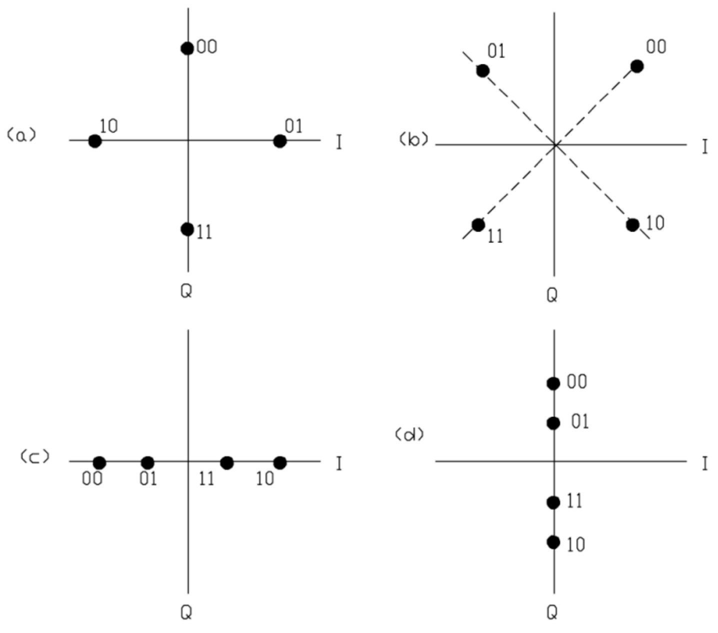


Figure 4: Possible phase and amplitude modulated constellations.

Among the suggested configurations, square M-QAM techniques were developed in 60's as well. In [13] it is presented a combination of amplitude and phase modulations for digital communication, under the assumption of phase coherence and perfect synchronization between transmitter and receiver. The challenge was to implement a technique that involve simultaneous and independent amplitude modulations of two carriers of the same frequency and in quadrature with each other, consisting in the separation of the in-phase and quadrature components followed by amplitude detection. Figure 5 shows the square 16-QAM constellation analysed in [13].

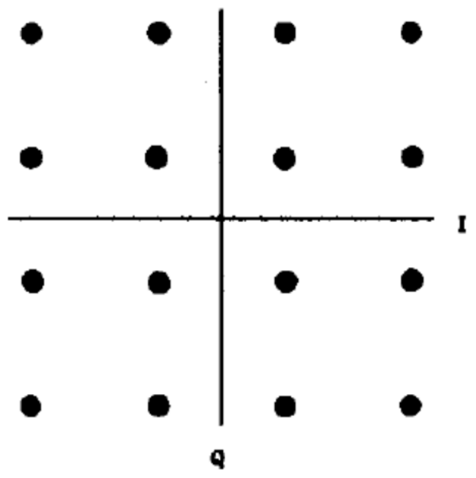


Figure 5: Square 16-QAM constellation.

In order to reduce the error probability of square M-QAM, a binary encoding technique was employed, the so called Gray coding. This technique was created by the researcher Frank Gray, from the Bell Labs, who developed it to be used in a vacuum tube, in order to prevent spurious outputs in electromechanical switches [48]. Nevertheless, a patent of Gray coding was issued in the 50's entitled "Pulse Code Communication"[48], which describes a method to build code from the conventional binary code through a reflection process, known as binary reflected code, as exemplified in Figure 6.

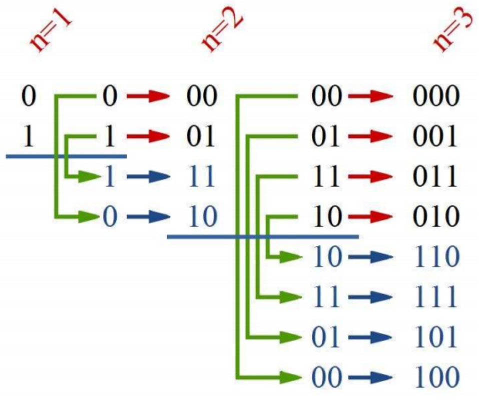


Figure 6: Code construction method of the binary reflected code.

Different kinds of Gray encoding were researched. For example, in [16] a technique generates the binary reflected Gray code efficiently in terms of the required time to generate multiple and consecutive symbols. These kind of Gray codes can be applied in digital modulation techniques that are being used in the current communication technology. Improvements on BER by using the Gray coded square M-QAM were reported in [15, 16]. The constellation with Gray codes was applied in the main telecommunication application of the 60's, which at the time was digital television [15].

At the beginning of the 70's, digital modulation techniques were a hot topic and several new ideas arose. For instance, [19] introduced the multiple amplitude and phase amplitude and phase shift keyed (MAPSK) modulation scheme and a performance analysis over a linear additive Gaussian noise channel, showing the influence of the peak signal to noise ratio (SNR) for a given symbol error probability. Also, the simplified generation and detection of the symbol and the bit error probabilities using Gray code technique were presented. The main contributions of [19] was to use two quadrature amplitude-shift-keyed (QASK) signal sets when the number of bits per symbol is odd, resulting in a rectangular QASK set with low computational complexity.

Regarding rectangular M-QAM constellation, $k = \log_2(M)$ in which $k \in \mathbb{N}$ is the number (even or odd) of transmitted bits. The constellation points are arranged in $2^{\lceil k/2 \rceil}$ columns and $2^{\lfloor k/2 \rfloor}$ rows, in which $\lceil k/2 \rceil = \min \{n \in \mathbb{Z} \mid n \geq k/2\}$ and $\lfloor k/2 \rfloor = \max \{m \in \mathbb{Z} \mid m \leq k/2\}$. This asymmetric configuration results in average power unbalance of the I and Q components of the signal [20]. On the other hand, Gray mapping can be applied easily. Figure 7 shows 32-QAM constellation with a rectangular geometry.

The cross QAM was investigated [17, 20]. A 32-QAM constellation with cross

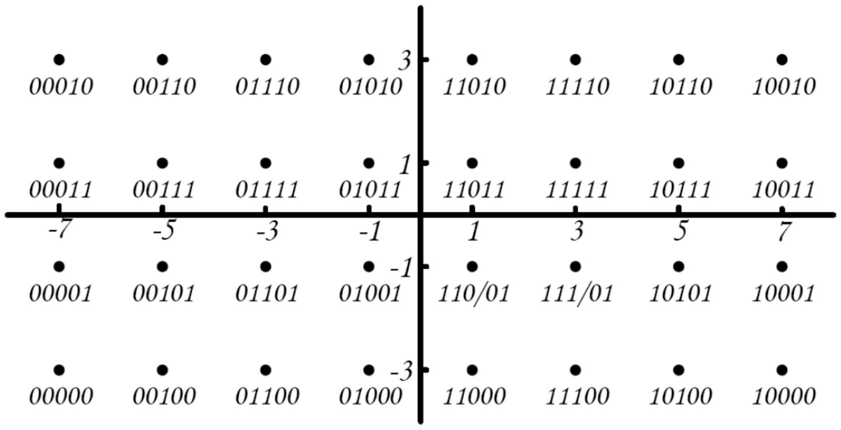


Figure 7: Rectangular 32-QAM constellation.

geometry, as shown in Figure 8, was presented in [20]. As it was discussed, perfect Gray coding is not possible for cross QAM. Also it was shown that there are two ways to generate the cross M-QAM, the most common is to obtain the values from a LUT with pre-defined values and the second one is made by the translation of the bits located in the corners of a rectangular QAM constellation to the top and bottom sides of it, as described by [17]. By doing that and by using a new technique, which is called the Smith-style Gray coding with minimum bit difference between neighbor [17], the authors derived the exact BER for cross QAM constellations over additive white Gaussian noise (AWGN) and Rayleigh fading channels. Based on that, [18] reported the first demonstration of a 512-QAM coherent optical transmission by using an optical phase locked loop (PLL). A polarisation-multiplexed 54 Gbps data signal was successfully transmitted with an optical bandwidth of 4.1 GHz, including a tone signal, showing that it is possible to achieve high data rates using cross QAM.

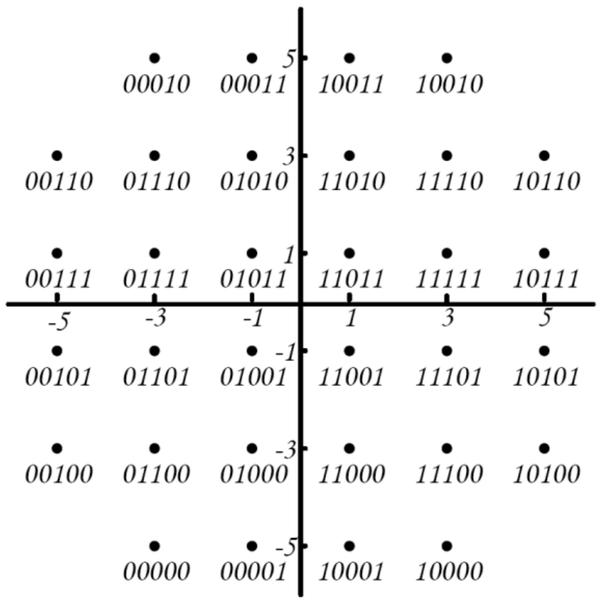


Figure 8: Cross 32-QAM constellation.

2.2.1 High Data-Rate Communication Systems

Nowadays, communication systems making use of digital modulation scheme based on up to 2^{15} constellation points are being standardized and designed [11] to fulfil the increasing demand for high data-rate. In order to achieve high data rate and/or spectral efficiency with digital communication systems operating in frequency selective channel, multi-carrier modulations is the most suitable one. A simple and well-known way of designing a high data-rate data communication system for data communication through frequency selective channels is to combine an orthogonal frequency division multiplexing (OFDM) scheme with digital modulation, such as M-QAM and binary phase shift keying (BPSK) [49]. In addition, OFDM scheme [49] is attractive because of the simplicity associated with channel equalization due to the use of frequency domain equalization technique [50], hardware clock speed reduction in comparison with single carrier modulations, efficient power or bit allocation in the subchannels if complete channel state information (CSI) is available at the transmitter.

A bitloading algorithm, which is one of the available techniques to increase the efficient of power and bit allocation in the subchannels, that considers the prior knowledge of complete CSI made available to the transmitter may be applied to maximize data rate for a given transmission power or minimize the transmission power given a target data rate [9]. On the other hand, the spectral efficiency of adaptive modulation, which do not require complete CSI at the transmitter side, is constant when compared to the bitloading technique. It means that the average spectral efficiency of the bitloading technique is improved, while, at the same time, the BER is better tailored to the application requirement. Thus, bitloading techniques are much more effective when it comes to maximize the use of channel resources for data communication if complete CSI is available at the transmitter [51].

The bitloading technique for digital communication systems based on OFDM scheme chooses the digital modulation scheme and constellation based on the knowledge of normalized SNR at each subchannels, which is perceived by the receiver [9]. From the negotiation between the base stations and modem, the digital modulation is dynamically adapted to fullfil the link conditions [8]. This technique gives greater flexibility to data communication system, with respect to maximizing spectral efficiency as well as the fairness of resource allocation. Figure 9 shows an example of bitloading technique for OFDM scheme when distinct M-QAM constellations are chosen for data communication through sets of subchannels. It is important to emphasize that bitloading techniques are mandatory in this context, since each subcarrier can transmit a different number of bits to adapt to the subchannel conditions.

Based on the aforementioned context, it is clear that the development of algorithms that demand low complexity to perform M-QAM (encoding, decoding and detecting) in

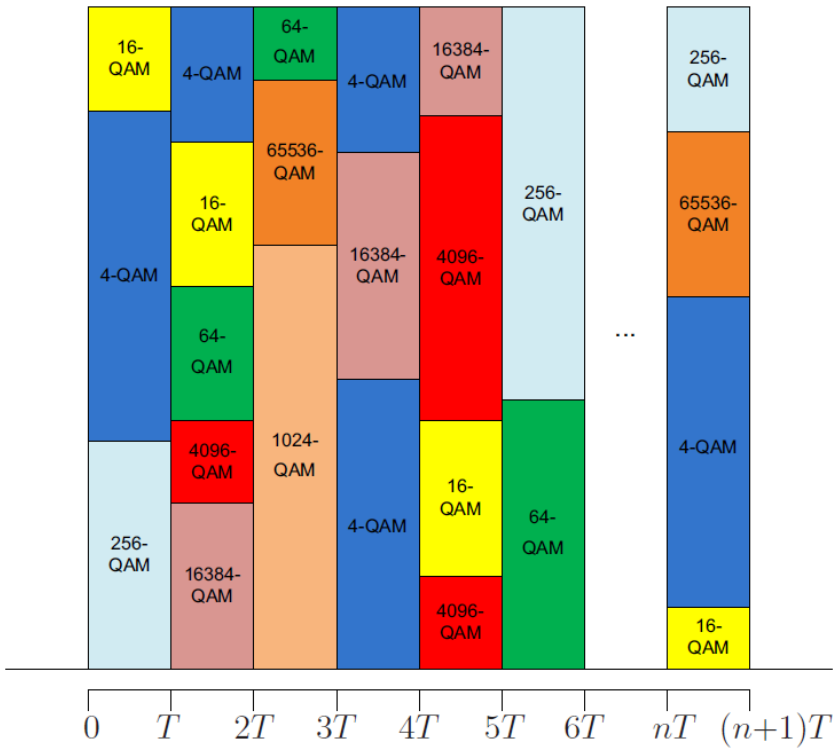


Figure 9: Allocation of M-QAM constellation in groups of subchannels when bitloading technique is applied with OFDM scheme.

OFDM scheme is a timely and important issue to be addressed. In fact, the need for the transmission of variable number of bits through the subchannels, the lager number of subcarriers and large size of QAM constellations (reaching 2^{15}) makes the computational complexity of M-QAM a very important issue to be addressed due to the impact on the choice of the hardware or technology used to implement the transceiver.

Recent contributions about M-QAM have focused on computational complexity reduction. For instance, [21, 22] discussed the implementation of 16-QAM with low computational complexity in an FPGA device. The advantages in favor of FPGA technology are flexibility, accuracy and reconfigurability for designing and implementing digital systems, and capacity to process complex algorithms at clock rates of hundreds of MHz. Taking advantage of the FPGA technology, [21] considered the real time physical layer level of data communication systems, such as symbol mapping, source encoding, channel coding and receiver decoding, presenting the advantages and disadvantages of the FPGA technology. Also, [22] discussed experiments with Virtex 4 FPGA development kit to show a promising platform for developing algorithms for the real time implementation and analysis of digital modulation schemes.

In addition, [23, 24] paid attention to several approaches to encode, decode and detect square M-QAM symbols considering $M \in \{4, 8, 16, 32, \ldots, 65536\}$ by using the so-called HDR technique that allows a reduction of hardware resource utilization by M-QAM scheme in a FPGA device in comparison with maximum likelihood (ML) criterion when hard decision rule, AWGN and AIGN channel are taken into account. This technique makes use of large memory storage, requiring large LUT at the transmitter, and considerable number of logic elements because it is based on the SM concept to implement the symbol detection at the receiver.

The authors of [23, 24] addressed a M-QAM technique based on hard decision. However, a soft decision rule can be applied to improve the detection performance, as discussed in [25]. The drawback associated with soft decision is the increased computational complexity due to the fact that soft decision rules demand more hardware resource utilization then hard decision one. The authors of [25] investigated techniques such as the max-log approximated detection and subregion based soft-demappers, but these techniques are also too complex to be implemented due to the need for calculating two-dimensional (2D) euclidean distances. As a result, the authors proposed what they named two one-dimensional (1D) soft-demappers. By reformulating a QAM signal as two layered pulse amplitude modulation (PAM) signals, the proposed QAM symbol detection technique was simplified to two 1D soft de-mapping and, as a consequence, it demands 73% less computational complexity. Also in [52], an optimum bits-to-symbol mapping for square constellations is introduced which allows for a simple detection of M-QAM symbols, considering the division of a M-QAM constellation in four sub-regions, reserving the two most significant bit (MSB) to map the sub-regions, representing the so-called $(4M)$-point constellations.

Considering the papers in the literature about the M-QAM modulation, it seems to be clear that if one come up with a procedure that eliminates the need for storage memory and avoids the use of state machine, which considerably increase the resources usage when high order M-QAM is considered [23, 24]; and addresses both square and cross M-QAM constellations with independent in-phase and quadrature modulation and detection, to simplify the demodulation process [25], then hardware resource utilization as well as computational complexity can be considerably reduced. Novel M-QAM techniques that combines these issues are detailed in this thesis by considering Gray coded constellations and hard decision rule. In order to present them, the mathematical formulation is described in the next section.

2.3 Mathematical Formulation

The main objective in digital communication systems is to deliver to a destination blocks made of k bits, $\mathbf{u}_i = [u_{i,k-1} \dots \ u_{i,1} \ u_{i,0}] \in \mathbb{B}^k$ in which $\mathbb{B} = \{0, 1\}$. To express mathematically the electrical signal transmitted through the channel, let us first introduce the notation ψ_{10} to indicate the function which maps a general binary block

$\mathbf{b} = (b_{K-1} \ldots\ b_1\ b_0) \in \mathbb{B}^K$ into its decimal representation[1]

$$\psi_{10}(\mathbf{b}) = 2^{K-1} b_{K-1} + \ldots + 2b_1 + b_0. \tag{2.7}$$

Using this notation it is said that the block $\mathbf{u}_i$ corresponds to the message $m_i = \psi_{10}(\mathbf{u}_i)$ and it is mapped, by the modulating function f, to the symbol $\mathbf{s}_{m_i} = f(\mathbf{u}_i) = (s_{m_{i,R}}, s_{m_{i,I}}) \in \mathcal{C}_k \subset \mathbb{R}^2$ in which the set $\mathcal{C}_k$ with cardinality $|\mathcal{C}_k|$ is called the signal constellation and $m_i \in \{0, 1, \ldots, |\mathcal{C}_k - 1|\}$ is a set of integer indices—in this development $|\mathcal{C}_k|$, the number of points in the constellation, is restricted to powers of two i.e., $|\mathcal{C}_k| = M_k = 2^k$. Let $\mathcal{L}_k = \{\pm 1,\ \pm 3,\ \ldots\ \pm 2^k - 1\}$, then this thesis will be considering constellations $\mathcal{C}_k = \mathcal{L}_{k_R} \times \mathcal{L}_{k_I}$, in which $k = k_R + k_I$, that lies on a rectangular grid.

Transmitters which use modulating functions with M-QAM constellations, when sending the block $\mathbf{u}_i$, transmits through the waveform channel the signal

$$s(t) = A \sum_{i=-\infty}^{\infty} (s_{m_{i,R}} \cos(2\pi f_c t) + s_{m_{i,I}} \sin(2\pi f_c t)) p(t - iT), \tag{2.8}$$

which considers $p(t)$ to be a square pulse of unitary amplitude and duration T_s; f_c is the carrier frequency; and $s_{m_{i,R}}$ and $s_{m_{i,I}}$ are the in-phase and quadrature components of the message. It should be noticed that the average power of the transmitted signal, P_s, can be adjusted by varying the value of A.

The block diagram depicted in Figure 10 illustrates the modulation and demodulation part of a vectorial modeled digital communication system. A simple two-dimensional function f which Gray maps a modulator input into the i-th symbol $\mathbf{s}_{m_i} = f(\mathbf{u}_i)$, $0 \leq m_i \leq |\mathcal{C}_k| - 1$, belonging to some constellation $\mathcal{C}_k$, can be constructed as described next. One define first a partitioning function that splits $\mathbf{u}_i$, the vector of input bits, in the two parts $\mathbf{u}_{i,0} = \left(u_{i,k-2}, \ldots, u_{i,2}, u_{i,0}\right)$ and $\mathbf{u}_{i,1} = \left(u_{i,k-1}, \ldots, u_{i,3}, u_{i,1}\right)$.

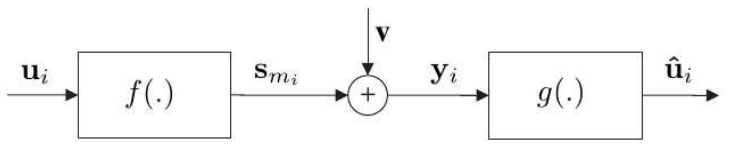


Figure 10: Vector model of a digital communication system.

Then $\mathbf{s}_{m_i} = f(\mathbf{u}_i) = (s_{m_{i,R}}, s_{m_{i,I}})$ is considered, in which

$$\begin{aligned} s_{m_{i,R}} &= (-1)^{u_{i,0}} \left(2^{k/2-2} u_{i,k-2} + \ldots + u_{i,2} + 1\right) \text{ and} \\ s_{m_{i,I}} &= (-1)^{u_{i,1}} \left(2^{k/2-2} u_{i,k-1} + \ldots + u_{i,3} + 1\right). \end{aligned}$$

Letting the vector $\mathbf{v} \in \mathbb{R}^2$ expressing the discrete time additive noise at the transmission, one will have at the channel output (demodulator input), the vector $\mathbf{y}_i \in \mathbb{R}^2$, in which

[1] It should be noticed that, in (2.7), b_0 is the least significant bit (LSB).

$\mathbf{v}_i = \mathbf{v}_{i,bkg}$ (background noise) for AWGN or $\mathbf{v}_i = \mathbf{v}_{i,bkg} + \mathbf{v}_{i,ps} + \mathbf{v}_{i,imp}$ for AIGN, in which $\mathbf{v}_{imp}$ represents impulsive noise and $\mathbf{v}_{ps}$ the periodical impulsive noise (see [45, 46] and Section 2.1). For the sake of simplicity, in this section it is assumed that only the AWGN model applies.

The demodulating function g, in Figure 10, corresponds to a hard-decision rule-based detector followed by a de-mapping (signal to binary block map). When $\mathbf{v}_i = 0$, i.e., the received signal is $\mathbf{y}_i = \mathbf{s}_{m_i}$, since f is a bijective function, the received block would be $\widehat{\mathbf{u}}_i = g(\mathbf{y}_i) = f^{-1}(\mathbf{s}_{m_i})$ or, in other words, since $f^{-1}(\mathbf{s}_{m_i}) = \mathbf{u}_i$, the recovered message $\widehat{\mathbf{u}}_i$ is an exact replica of the message sent. But, since the noise is modeled, in this development, as an *i.i.d.* (independent and identically distributed) discrete time random variable $\mathbf{V}_i$ with Gaussian distribution having expected value $\mathbb{E}\{\mathbf{V}_i\} = 0$, the decision rule, which provides an optimal (in the ML sense) solution to the problem of mapping the received signal back into some constellation signal is,

Decision Rule

Find some signal $\mathbf{s}_{\widehat{m}_i} \in \mathcal{C}_k$ such that, $\forall \ell \in \{0, 1, ..., M-1\}$,

$$\left\|\mathbf{y}_i - \mathbf{s}_{\widehat{m}_i}\right\|^2 \leq \left\|\mathbf{y}_i - \mathbf{s}_\ell\right\|^2. \tag{2.9}$$

Decide then the received block of bits is

$$\widehat{\mathbf{u}}_i = g(\mathbf{y}_i) = \mathbf{s}_{\widehat{m}_i} \in \mathcal{C}.$$

An interpretation for this M-QAM decision rule systems (ML Rule) is related to the detector output $\mathbf{s}_{\widehat{m}_i}$ as the symbol which, in terms of Euclidean distance, corresponds to the nearest neighbour to the channel output vector $\mathbf{y}_i$. If M, the constellation size, is small, the computational complexity to evaluate Equation (2.9) can be small. On the other hand, when M is very large (in the new generation of PLC transceivers constellation with huge size, 2^{15}-QAM, [11] are used) the burden to directly compute (2.9) leads to a quite high computational complexity. On the other hand, to design high-speed communication systems, which makes use of high-order QAM constellations, commercially feasible, the design of transceivers demanding as low as possible hardware resource utilization and computation complexity is a challenging issue to be addressed.

Usually, the computational burden of implementing modulation and demodulation techniques relies on the mapping (function f) and de-mapping/detection (function g) procedures. The search for a low-cost hardware implementation of f and g is a important issue to be considered. In this context, the following questions can arise:

1. How to implement a generic mapping/detection technique for M-QAM constellations, to transmit messages with even and odd number of input bits, that satisfies a Gray mapping rule?

2. How to implement this generic mapping/detection technique for M-QAM symbols with low-computational complexity in a FPGA device to analyze its improvement in terms of computational complexity reduction without affecting BER performance?

In this thesis, the use of the adjective **generic** is to emphasize that the proposed techniques, which will be introduced in Chapter 3, are capable of dealing with square and cross M-QAM constellations and can provide low-order QAM constellations from a higher order one. Chapter 3 presents techniques that address both posed questions.

3 THE PROPOSED TECHNIQUE FOR M-QAM

The techniques introduced in this chapter address the problem of designing adaptive M_k-QAM techniques which switches the size $M_k = 2^k$ of the corresponding constellation $\mathcal{C}_k$ by varying the value of $k \in \mathbb{N}$ in the range from 2 to K (i.e., $4 \leq M_k \leq 2^K$). These techniques can be applied to even and odd number of input bits, satisfying the Gray mapping rule, representing a symbol of any M-QAM constellation order with the same procedure. Due to their characteristics, the proposed techniques can be easily applied to multi carrier modulation scheme based on orthogonal frequency division multiplexing (OFDM) scheme when bit loading or adaptive modulation techniques are employed [53].

In this regard, this thesis outlines two novel mapping and detection techniques for high-order square and cross M-QAM constellations when hard decision rule is taken into account. The techniques eliminate drawbacks related to the technique proposed in [23, 24], which are the need for several LUT's and several SM's. The novelties related to this technique are flexibility to handle both square and cross M-QAM constellations and simplicity to adapt to any constellation order based on a high-order one.

In order to present the proposed techniques, this chapter is organized as follows: Section 3.1 introduces a new M-QAM technique, named Generic M-QAM (technique #1), and it is FPGA implementation. In addiction, Section 3.2 introduces the use of HDR for M-QAM and the so-called Enhanced HDR technique (technique #2).

3.1 Technique #1 - Generic M-QAM

This section introduces a new M-QAM technique, named Generic M-QAM (Technique #1) which, in addiction of being a simple technique to construct an adaptive Gray mapping modulating function, it is amenable to easy hardware implementation.

This technique is designed to cover even and odd number of input bits $k = 2, 3, \ldots, K$, satisfying the Gray mapping rule, and to represent a symbol of any M-QAM order, just changing the value of k, from 2 to K, depending of the application and the available hardware. The main innovations behind this technique are: i) low-cost modulation/demodulation based on Gray mapping rule; ii) low-order square and cross M-QAM constellations can be derived from a high-order square M-QAM; iii) cross to rectangular and rectangular to cross translations to cover M-QAM symbols with odd number of bits; iv) modulation and demodulation of cross and square M-QAM symbols; and v) memory storage avoidance. Overall, this technique can modulate/demodulate any Gray-coded M-QAM without the need of any storage memory for M-QAM constellation points and the use of sophisticated detection approach to estimates of the transmitted M-QAM symbols.

The Gray mapping rule is always considered in any modulation/demodulation technique [15,16,48]. In Tech #1 the approach that converts symbols in Gray codes is used to generate the constellations indexes in the transmitter and to recover the demodulated sequence in the receiver. By doing that, the implementation of the Tech #1 can represent any M-QAM constellation order, by changing the parameter K, which is the maximum number of bits used by the transmitter. This technique becomes very simple when the modulation/demodulation of the in-phase and quadrature components are independently performed [13, 19, 25], because one M-QAM symbol is represented by two $\sqrt{M}$-PAM symbols by spliting the input bits sequence $\mathbf{u}$ into two bits sequence $\mathbf{r}_R$ and $\mathbf{r}_I$, and, as a consequence, low-order M-QAM ($k = 2, 3, \ldots, K - 1$) constellation points can be derived from high-order ones (K).

For square and rectangular M-QAM constellations it is straightforward to derive lower-order constellations because the decimal representation of the input bits $\mathbf{u}$ shall receive the same index, independently of the M-QAM order. For example, $\mathbf{u} = 0011$, which represents the decimal number 3, is splitted into $\mathbf{r}_R = 01$ and $\mathbf{r}_I = 01$ to receive the index constellation point $s_m = (-1, -1)$ if 16-QAM constellation applies; $\mathbf{u} = 000011$ that represents the decimal number 3 is splitted into $\mathbf{r}_R = 001$ and $\mathbf{r}_I = 001$ to receive the index constellation point $s_m = (-1, -1)$ if 64-QAM constellation applies, and so on. In the case of rectangular M-QAM constellations, the procedure is the same, but the number of bits in the in-phase component (k_R) is different from the number of bits of the quadrature component (k_I), that is $k_R = k_I + 1$, so, for example, $\mathbf{u} = 10011$, that represents the decimal number 19, is splitted into $\mathbf{r}_R = 101$ and $\mathbf{r}_I = 01$ to receive the index constellation point $s_m = (-7, -1)$ if 32-QAM constellation applies.

On the other hand, for cross M-QAM constellations, deriving low-order M-QAM constellation points from high-order ones is not a simple task to be accomplished because a cross constellation have to be derived from a rectangular constellation with the same order. Essentially, for any M-QAM constellation, in which k is odd, the rectangular to cross translation assumes that the side edge constellation points are divided into two subregions, in each quadrant in the M-QAM transmitter, and then translated to the top and bottom corner of it. At the receiver, it is recovered the right constellation points by doing the reverse operation, which is the cross to rectangular translation. As a result, lower-order cross M-QAM constellations are derived from higher ones. Note that this is a generalization of the approach discussed in [17,20] for generating cross M-QAM symbols by changing the position of the side edge points of a rectangular constellation to the top and the bottom corners when $M = 32$ and $M = 128$. Following the previous example, $\mathbf{u} = 10011$ that represents the decimal number 19 is splitted to $\mathbf{r}_R = 101$ and $\mathbf{r}_I = 01$ to receive the rectangular index constellation point $m = (-7, -1)$ and then generate the cross index constellation point $s_m = (-1, -5)$, if 32-QAM constellation applies.

Figure 11 shows the flowchart of Tech #1. As it can be observed, the transmitter starts by an input bit sequence $\mathbf{u}$ that goes to the Modulation block to map the square and rectangular M-QAM and then it is analyzed how many bits k are going to be transmitted. If k is odd (cross constellation is chosen) then the modulated sequence is passed to the Position Switcher block, and then, for both cases (even and odd number of bits) the M-QAM symbol is generated. The receiver side checks how many bits k are going to be received, so that if k is odd (cross constellation was chosen), the M-QAM symbol is passed to the Combined Saturator and Inverse Position switcher blocks, and then the Demodulation block handles the detection of the actual output bit sequence $\hat{\mathbf{u}}$. All mathematical details about Tech #1 and the proposed algorithms to implement it are as follows.

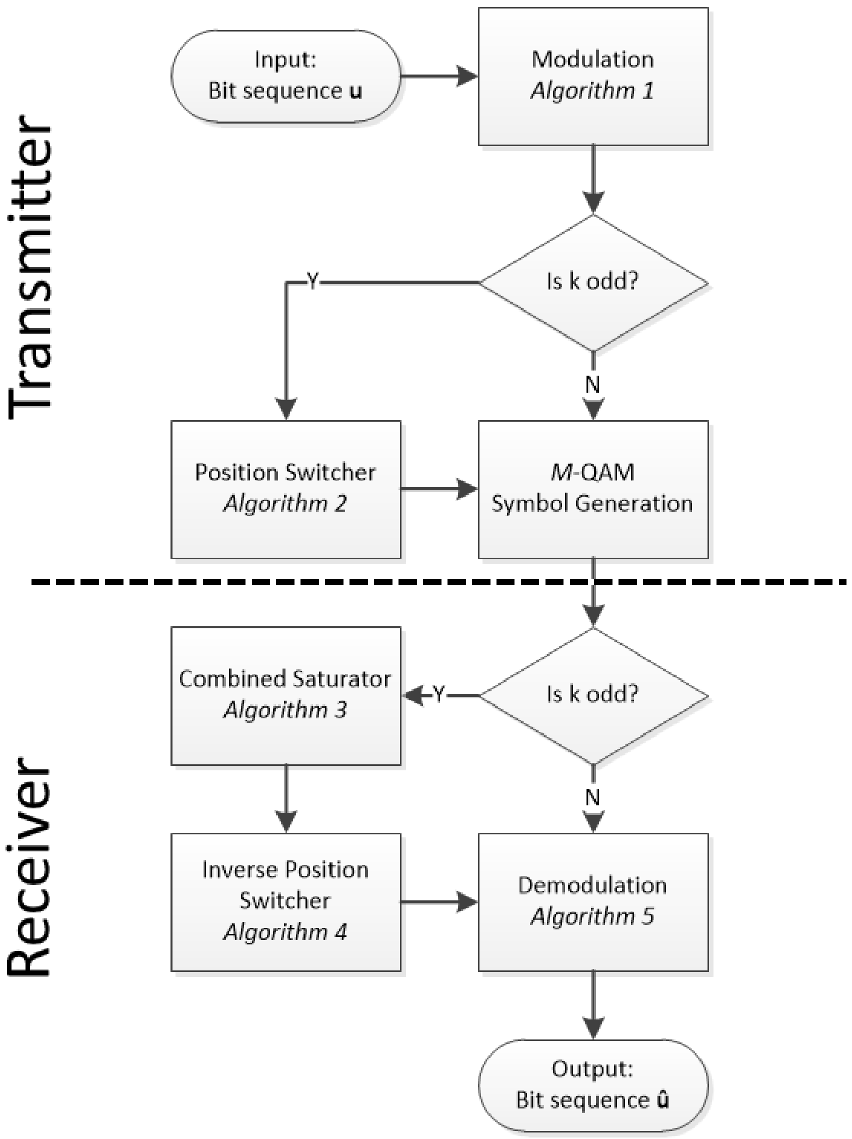


Figure 11: Flowchart for Tech #1.

Many are the ways to map $\mathbf{u}_i$ into points $\mathbf{s}_{m_i} \in \mathcal{C}_k$. Often, in practice, data communication systems are built around the modulation scheme based on the well known

"Gray Mapping Modulating function," for its improved probability of bit error performance. The definition of a Gray Mapping, which holds in general, is presented next for the case of a two-dimensional signal constellation with points lying in a rectangular grid. From now on, the i-th block of bits $\mathbf{u}_i$ and the i-th transmitted symbol $\mathbf{s}_{m_i}$ will be represented simply by $\mathbf{u}$ and $\mathbf{s}_m$, suppressing the index i and letting implicit the time dependence.

Definition 1 (Gray Mapping Modulating Function)**.** Consider that $\mathbf{u} \in \mathbb{B}^k$ and $\mathcal{C}_k \subset \mathbb{R}^2$ is a signal constellation of cardinality $|\mathcal{C}_k| = 2^k$. It is possible to say that the function f which maps the binary vector $\mathbf{u}$, at the modulator input, into a signal $\mathbf{s}_m = f(\mathbf{u})$, is a two-dimensional, Gray Mapping Modulating Function if for any distinct pair of signals $\mathbf{s}_m$ and $\mathbf{s}_{m'} = f(\mathbf{u}')$, which are nearest neighbours[1], the corresponding block of bits, $(\mathbf{u},\ \mathbf{u}') \in \mathbb{B}^k \times \mathbb{B}^k$, differs in a single bit — i.e., their Hamming distance is $d_H(\mathbf{u}, \mathbf{u}') = 1$.

In other words, for a Gray Mapping Modulating function (Gray function, for short) f, it holds that for $(\mathbf{u}, \mathbf{u}')$ with $d_H(\mathbf{u}, \mathbf{u}') = 1$, the Euclidean distance between the corresponding signals (or symbols) $\mathbf{s}_m$ and $\mathbf{s}_{m'}$ is $d_E(\mathbf{s}_m, \mathbf{s}_{m'}) = \|\mathbf{s}_m - \mathbf{s}_{m'}\| = d_{\min}$ where $d_{\min}$ is the minimum distance between any two points of the constellation.

For now, it is presented a theorem with the construction of a simple one-dimensional Gray function, based on [48], namely γ_k which maps the binary block $\mathbf{u}$ into a (scalar) symbol belonging to $\mathcal{L}_{k-1}$.

Theorem 1 (A Gray mapping function)**.** Consider the binary vector $\mathbf{u} = (u_{k-1}, \ldots, u_1, u_0)$ and let $\mathbf{u}^- = (u_{k-1}, \ldots, u_1)$. Let β be the boolean function which maps the block $\mathbf{u}^-$ to the $(k-1)$-bit binary vector $\tilde{\mathbf{u}} = \beta(\mathbf{u}^-)$, with components given by

$$\begin{aligned} \widetilde{u}_{k-2} &= u_{k-1}, \\ \widetilde{u}_{k-\ell-1} &= u_{k-\ell} \oplus \widetilde{u}_{k-\ell};\ 2 \leq \ell \leq k, \end{aligned} \tag{3.1}$$

in which, $\oplus$ is the addiction mod 2. Then, by considering $m = \psi_{10}(\mathbf{u})$ and $\alpha_m = \psi_{10}(\tilde{\mathbf{u}})$, it has $s_m = (-1)^{u_0}(2\alpha_m + 1) \in \mathcal{C}_k$, then it gets the one-dimensional Gray function

$$\gamma(\mathbf{u}) = s_m \tag{3.2}$$

which maps the binary block $\mathbf{u}$ to the signal $s_m \in \mathcal{L}_{k-1}$ (it is said that s_m is labelled by the binary block $\mathbf{u}$).

It should be emphasized that the signal s_m has assigned to it the label (block of bits) $\mathbf{u}$. The block $\tilde{\mathbf{u}} = (\widetilde{u}_{k-1}, \widetilde{u}_{k-2}, \ldots, \widetilde{u}_0)$ corresponds to the magnitude of s_m and u_0 to

[1] i.e., no signal $\mathbf{s}_{m''}$ is closer to $\mathbf{s}_m$ than $\mathbf{s}_{m'}$.

its sign bit. Also, notice that the inverse mapping, $\mathbf{u}^{-} = \beta^{-1}(\tilde{\mathbf{u}})$, is simply obtained by making $u_{k-1} = \tilde{u}_{k-2}$, and, for $2 \leq \ell \leq k$, $u_{k-\ell} = \tilde{u}_{k-\ell} \oplus \tilde{u}_{k-\ell-1}$.

The proposed hardware implementation of a two-dimensional modulating function, is based on two one-dimensional modulating Gray functions, namely γ_R and γ_I. Next theorem introduces a two-dimensional Gray function obtained by coupling γ_R and γ_I.

Theorem 2 (Gray mapping function)**.** Take any pair $(\gamma_R,\ \gamma_I)$ of one-dimensional Gray functions. Take any pair of binary blocks $\mathbf{u}_R = (u_{k_R-1}, \ldots, u_1, u_0)$ and $\mathbf{u}_I = (u_{k_I-1}, \ldots, u_3, u_1)$ with $k = k_R + k_I$. The two-dimensional modulating function $f = (\gamma_R, \gamma_I)$ that maps a binary block $\mathbf{u} = (u_{k-1}, \ldots, u_1, u_0)$ to a signal $\mathbf{s} = (s_{m_R}, s_{m_I}) = f(\mathbf{u}) = (\gamma_R(\mathbf{u}_R), \gamma_I(\mathbf{u}_I))$, is also a Gray mapping function.

The theorem inspires a hardware implementation for an adaptive two dimensional modulating/demodulating scheme in which the Gray Mapping Rule applies and also applies for arbitrary dimension modulating signals (other than two), as it will be explained later.

To build a rectangular, two-dimensional, M_k-QAM signal constellation simply take $f = (\gamma_R,\ \gamma_I)$ in which $(\gamma_R,\ \gamma_I)$ are one-dimensional Gray mapping modulation functions[2].

The diagram in Figure 12 illustrates several M_k-QAM modulation constellations with cardinality $M_k = 2^k$ of interest in this thesis. Square constellations (black lines) as well as the so-called cross constellations (blue lines) are illustrated in this figure, as suggested in [54]. It has then, for any point $\mathbf{s}_\ell$, of a given M-QAM constellation, that $\mathbf{s}_\ell = (s_{\ell,R}, s_{\ell,I}) \in \mathcal{L}_k^2$, in which $\mathcal{L}_k = \{\pm 1,\ \pm 3,\ \ldots\ \pm 2^k - 1\}$ Let us say that $\mathcal{C}_k$, the M_k - ary constellation under use (chosen by the adaptation rule), is a square constellation, suitable for transmitting blocks $\mathbf{u} = [u_{k-1} \ldots\ u_1\ u_0]$ of size $k = \log_2(M_k)$ bits.

Aiming to describe the proposed technique, it is necessary to introduce an easily implementable function f that Gray maps $\mathbf{u} = (u_{k-1}, \ldots, u_1, u_0) \in \mathbb{B}^k$, a variable size block $(2 \leq k \leq K)$ into a signal $\mathbf{s}_m$ belonging to $\mathcal{L}_{k/2 \times k/2}$, a two-dimensional, rectangular, variable size constellation.

Having implementation in mind and considering that the digital circuit has to cover the highest M-QAM order, the transmitter hardware needs registers sufficiently large, say $\mathbf{r}_R = \left[r_{K/2-1,R} \cdots\ r_{1,R}\ r_{0,R}\right]$ and $\mathbf{r}_I = \left[r_{K/2-1,I} \cdots\ r_{1,I}\ r_{0,I}\right]$ to store blocks of up to K bits. To send $k = 2, 3, \ldots, K$ bits (even or odd) of the variable size block $\mathbf{u}$ (message $m = \psi_{10}(\mathbf{u})$), one choice is to load these registers, $\mathbf{r}_R$ and $\mathbf{r}_I$, by setting $r_{K/2-1,R} = u_0$, $r_{K/2-1,I} = u_1$,

[2] For square constellations use $\gamma_R = \gamma_I = \gamma_{\frac{k}{2}}$.

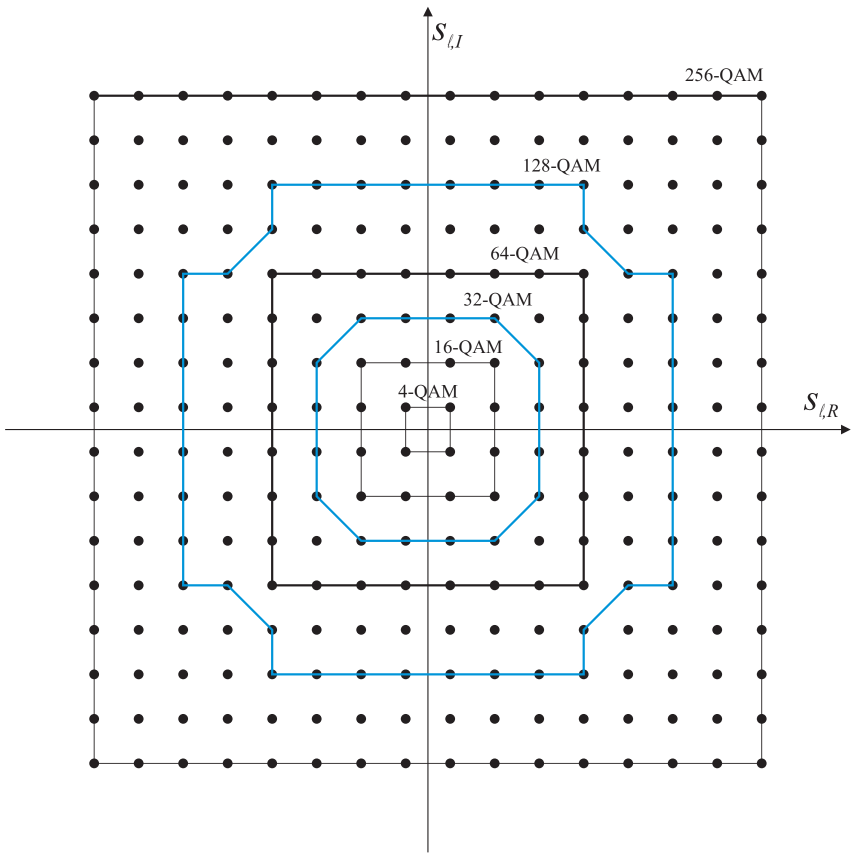


Figure 12: M-QAM modulation with M_k belonging to the set $\{4, 16, 32, 64, 128, 256\}$.

$$r_{\ell-1,R} = u_{2\ell}, \text{ for } 1 \leq \ell \leq k_R - 1, \tag{3.3}$$

and

$$r_{\ell-1,I} = u_{2\ell+1}, \text{ for } 1 \leq \ell \leq k_I - 1. \tag{3.4}$$

All remaining positions, not specified in (3.3) and (3.4), are set to zero.

The specified loading rule is arbitrarily chosen, rules other than the one in (3.3) and (3.4) fit as well. However, the next step is mandatory to achieve a Gray mapping constellation. Based on Theorem 1 and Equation (3.1), both Gray vectors $\mathbf{r}_R$ and $\mathbf{r}_I$ are converted to a binary sequence, then, by considering that $m = \psi_{10}(\mathbf{u})$ and, $s_m = (-1)^{u_0}(2m+1) \in \mathcal{C}_k$, one have that two-dimensional function $\mathbf{s}_{m_i} = f(\mathbf{u}_i) = (s_{m_{i,R}}, s_{m_{i,I}}) \in \mathcal{C}_k \subset \mathbb{R}^2$ for square constellations. To proceed with cross constellations, it is mandatory to change the position of m, so Algorithm 1 describes the modulation procedure for Tech #1.

It is seen that, in practice the transmission of blocks $\mathbf{u} \in \mathbb{B}^k$, when the values of k is even uses a **square** constellation, with $k_R = k_I = k/2$. To apply Tech #1 to cross constellations, or, considering odd values of k, a rectangular grid is used to get the signal coordinates but, in order to balance the mean power of the in-phase and quadrature

Algorithm 1: Generic Modulating Procedure

```
initialization:
 Set the constellation size to $k$.
begin
    • Step #1: Set sign bit
    $r_{K/2-1,R} = u_0$;
    $r_{K/2-1,I} = u_1$;
    • Step #2: Amplitude Coding
    for $1 \leq \ell \leq k/2 - 1$ do
        $r_{\ell-1,R} = u_{2\ell}$;
        $r_{\ell-1,I} = u_{2\ell+1}$;
    end
    • Step #3: Gray to Binary Conversion
    $m_R = \psi_{10}(\gamma_R(\mathbf{r}_R))$;
    $m_I = \psi_{10}(\gamma_I(\mathbf{r}_I))$;
    • Step #4: Done ($m$ is the modulated signal).
end
```

components [23,24], a further step will re-map the input block $\mathbf{u}$ into a signal $\mathbf{s}_m$ belonging to a two-dimensional **cross** constellation (say $\mathcal{C}_k^\dagger$). The mapping so obtained is no longer a Gray mapping and will have some neighbouring symbols labelled by binary blocks no longer at a unitary Hamming distance [17,20], designated by *pseudo-Gray mapping.* In [18], a pseudo-Gray mapping for cross QAM constellation is presented. The transformation that will map points falling in a rectangular constellation $\mathcal{C}_k$ into points falling in a cross-constellation $\mathcal{C}_k^\dagger$ will be next described.

To understand the idea behind this transformation consider a 128-QAM constellation. By inspecting the coordinates of the signal belonging to the rectangular constellation it is possible to obtain the new coordinates, rearranging thus the points into a cross-constellation. The diagram in Figure 13, from [18], illustrates which points are to be moved and the final arrangement with points in the new positions. The general procedure is described as follows: let us say that k is odd. In this case $k_I = \left\lfloor \frac{k}{2} \right\rfloor < k_R = \left\lceil \frac{k}{2} \right\rceil$ and the signal with coordinates (s'_{m_R}, s'_{m_I}), indexed by (m_R, m_I), falls in a rectangular constellation. The following mathematical manipulation can map the coordinates (s'_{m_R}, s'_{m_I}) into the new coordinates $(s_{m_R}, s_{m_I}) \in \mathcal{C}_k^\dagger$. Only those points lying in the two outer columns of the square constellation are moved to their new positions on the cross-constellation.

The rectangular to cross transformation is described in Algorithm 2, with (s_{m_R}, s_{m_i}) a point belonging to a cross-constellation. Notice that for even k, $m_R \geq 2^{(k_R-2)} + 2^{(k_I-2)} = 2^{(k_R-1)}$ is always true.

The demodulation process, as initially suggested in [19], starts by considering that the received signal is

$$\mathbf{y} = \mathbf{s} + \mathbf{v}, \tag{3.5}$$

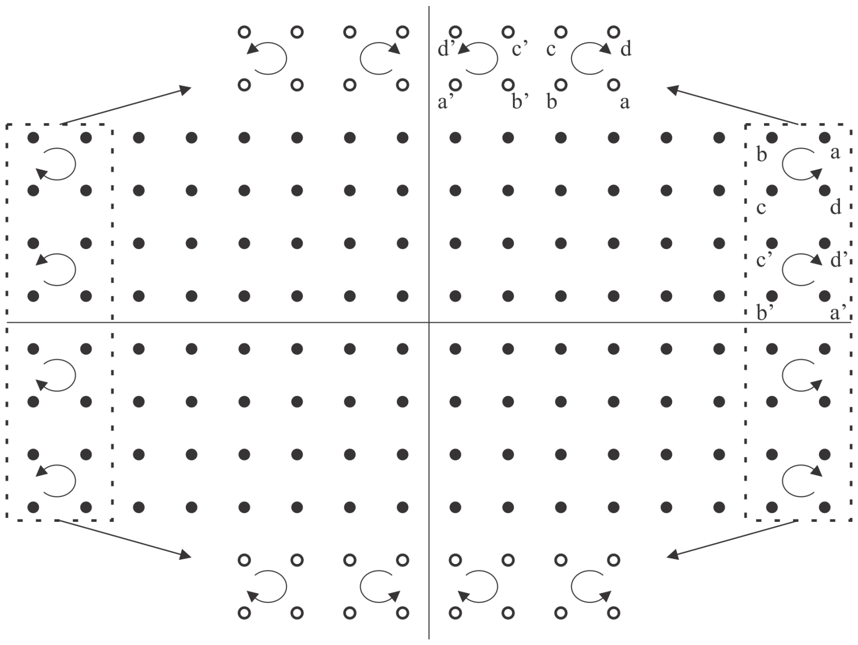


Figure 13: The translation from rectangular to cross constellation of a 128-QAM.

Algorithm 2: Rectangular to Cross QAM Translation Algorithm

```
begin
    if $m_R \geq 2^{(k_R-2)} + 2^{(k_I-2)}$ then
        if $m_I \geq 2^{(k_I-2)}$ then
            $\alpha_{m_R} = m_R - 2^{(k_R-2)}$;
            $\alpha_{m_I} = 2^{k_I} - m_I - 1$;
        else
            $\alpha_{m_R} = 2^{(k_R-1)} - m_R - 1$;
            $\alpha_{m_I} = m_I + 2^{(k_I-1)}$;
        end
    else
        (The point belong to the square nucleus of the cross)
        $\alpha_{m_R} = m_R$;
        $\alpha_{m_I} = m_I$;
    end
    $s_{m_R} = (-1)^{u_0}(2\alpha_{m_R} + 1)$;
    $s_{m_I} = (-1)^{u_1}(2\alpha_{m_I} + 1)$;
end
```

in which the index i was removed for the sake of simplicity. In this thesis, it is at the output of the analog to digital converter (ADC), when the input is the received (analog) vector $\mathbf{y} = (y_R, y_I)$, the digital signal with a binary representation is given by[3]

$$\mathbf{y}_R = \left[y_{K/2-1,R}, \ldots, \ y_{0,R}, \ y_{-1,R}, \ldots, y_{-L/2,R}\right], \tag{3.6}$$

[3] Only if soft-decision is an option the registers positions $[y_{-1,R}, \ y_{-2,R}, \ldots, y_{-L,R}]$ and $[y_{-1,R}, \ y_{-2,R}, \ldots, y_{-L,R}]$ are used.

and

$$\mathbf{y}_I = \left[y_{K/2-1,I}, \ldots,\ y_{0,I},\ y_{-1,I}, \ldots, y_{-L/2,I}\right]. \tag{3.7}$$

This binary representation uses bits $y_{K/2-1,R}$ and $y_{K/2-1,I}$ to store the sign of the signal orthogonal components, i.e., $y_{K/2-1,R} = 0$ if decimal representation of $\mathbf{y}_R$ is a positive number and is equal to 1 otherwise. Similarly, for the orthogonal component, $y_{K/2-1,I} = 0$ if decimal representation of $\mathbf{y}_I$ is a positive number. The bits $y_{\ell,R}$, for $0 \leq \ell \leq k/2 - 1$ store the integer part of $\mathbf{y}_R$ while the bits for $-L/2 \leq \ell \leq -1$ store its decimal part. Whenever allowed it will not be explicitly mention that the same procedure would apply to the orthogonal component $y_{\ell,I}$.

Let us consider that proper automatic gain control (AGC) and ADC conversion have taken place. To better understand the meaning of "proper" it is worth mentioning, when $\mathbf{u}$ has being sent and $\mathbf{v} = 0$, that $\mathbf{y} = \mathbf{s} \in \mathcal{C}_K$, $\widehat{\mathbf{m}}_R = \mathbf{m}_R$ and $\widehat{\mathbf{m}}_I = \mathbf{m}_I$ and the estimated bits would be, of course, $\hat{\mathbf{u}} = \mathbf{u}$. When noise is present then the nearest neighbour rule (2.9) should be applied. A receiver can operate in what is called a hard-decision rule or, a soft-decision rule—but, in this thesis, the description will be restricted to the hard-decision rule, because the purpose is to come up with a simple and low cost M-QAM technique. Algorithm 3 performs the Combined Saturator, a detection algorithm for square and cross constellations, to ensure that only valid constellation points will be demodulated.

Let us consider now that a 2^k points constellation is under use and one of the bits $y_{\ell,R}$, $k/2 \leq \ell \leq K/2 - 2$ of the received value is equal to 1 (an overflow condition has been observed). In this case, the mapping of the received vector $\mathbf{y}$ to its nearest neighbour $\mathbf{s}_{\widehat{m}_i}$ can be achieved by an *overflow correction*, which will set to one all the register cells $y_{\ell,R}$ for $0 \leq \ell \leq k/2 - 1$ while keeping $y_{\ell,R} = 0$ for $k/2 \leq \ell \leq K/2 - 1$. If an overflow condition has not been observed, the value stored in register $\mathbf{y}_R$ is an integer which is either odd ($y_{0,R} = 1$) or even ($y_{0,R} = 0$). If the integer is even, i.e., $y_{0,R} = 0$, the nearest neighbor vector is simply the received signal corresponding to the received signal binary representation that can be obtained by simply flipping the value of the bit in cell $y_{0,R}$, i.e. setting $y_{0,R} = 1$. If the integer is odd, i.e., $y_{0,R} = 1$, then the obtained bits corresponds to the received signal binary representation.

Let us also consider that the channel output (demodulator input), vector $\mathbf{y} \in \mathbb{R}^2$, is subject to proper AGC and, after the ADC, it gets $y_{_R}$ and $y_{_I}$. The receiver starts by examining the received signal amplitude to check if the point belongs to a valid cross constellation point. And, next, the saturation condition is applied independently for the in-phase and quadrature components, being represented by the equation $\lfloor x \rfloor = \max(m \in \mathbb{Z} | m \leq x)$, which is the floor function that map a number to the largest previous integer. Algorithm 3 performs this task.

Algorithm 3: Combined Saturator Algorithm

```
initialization:
 Set:
 $\hat{\alpha}'_{m_R} = |y_R|$;
 $\hat{\alpha}'_{m_I} = |y_I|$;
begin
    if $\hat{\alpha}'_{m_R} \geq 2^{(k_R-1)} + 2^{(k_I-1)}$ and $\hat{\alpha}'_{m_R} \geq \hat{\alpha}'_{m_I}$ then
        $\hat{\alpha}_{m_R} = \left\lfloor 2^{(k_R-2)} + 2^{(k_I-2)} - 1 \right\rfloor$;
    else if $\hat{\alpha}'_{m_R} \geq 2^{k_I}$ and $\hat{\alpha}'_{m_I} \geq 2^{k_I}$ and $\hat{\alpha}'_{m_R} < \hat{\alpha}'_{m_I}$ then
        $\hat{\alpha}_{m_R} = \left\lfloor 2^{(k_I-1)} - 1 \right\rfloor$;
    else
        $\hat{\alpha}_{m_R} = \left\lfloor \hat{\alpha}'_{m_R}/2 \right\rfloor$;
    end
    if $\hat{\alpha}'_{m_R} \geq 2^{(k_R-1)} + 2^{(k_I-1)}$ and $\hat{\alpha}'_{m_R} < \hat{\alpha}'_{m_I}$ then
        $\hat{\alpha}_{m_I} = \left\lfloor 2^{(k_R-2)} + 2^{(k_I-2)} - 1 \right\rfloor$;
    else if $\hat{\alpha}'_{m_R} \geq 2^{k_I}$ and $\hat{\alpha}'_{m_I} \geq 2^{k_I}$ and $\hat{\alpha}'_{m_R} \geq \hat{\alpha}'_{m_I}$ then
        $\hat{\alpha}_{m_I} = \left\lfloor 2^{(k_I-1)} - 1 \right\rfloor$;
    else
        $\hat{\alpha}_{m_I} = \left\lfloor \hat{\alpha}'_{m_I}/2 \right\rfloor$;
    end
end
```

So the next step is to translate the points belonging to the top side to the rectangular grids (periphery regions). The proposed algorithm to perform this task is Algorithm 4. Notice that it performs the reverse task performed by Algorithm 2.

Algorithm 4: Cross to Rectangular Translation Algorithm

```
begin
    if $\hat{\alpha}_{m_I} \geq 2^{(k_I-1)}$ then
        if $\hat{\alpha}_{m_R} \geq 2^{(k_R-3)}$ then
            $\hat{m}_R = \hat{\alpha}_{m_R} + 2^{(k_R-2)}$;
            $\hat{m}_I = 2^{(k_R-1)} - \hat{\alpha}_{m_R} - 1$;
        else
            $\hat{m}_R = 2^{k_I} - \hat{\alpha}_{m_I} - 1$;
            $\hat{m}_I = \hat{\alpha}_{m_I} - 2^{(k_I-1)}$;
        end
    else
        (The point belong to the square nucleus)
        $\hat{m}_R = \hat{\alpha}_{m_R}$;
        $\hat{m}_I = \hat{\alpha}_{m_I}$;
    end
end
```

As a result, the demodulating procedure is implemented by an algorithm which finds $\hat{\mathbf{u}}$, an estimate of the k bits of the variable size transmitted block $\mathbf{u}$ (not always it gets $\widehat{\mathbf{u}} = \mathbf{u}$), using the Decision Rule as explicited in (2.9). In order to reduce the computational complexity, the in-phase and quadrature components are demodulated independently [25].

The specified loading rule in (3.3) and (3.4) has to be recovered to get the block of estimated bits ($\widehat{\mathbf{u}}$). Also the digital circuit has to attend the highest M-QAM order, the receiver hardware needs registers sufficiently large, say $\widehat{\mathbf{u}}_R = \left[\widehat{u}_{K/2-1,R} \ldots\ \widehat{u}_{1,R}\ \widehat{u}_{0,R}\right]$ and $\widehat{\mathbf{u}}_I = \left[\widehat{u}_{K/2-1,I} \ldots\ \widehat{u}_{1,I}\ \widehat{u}_{0,I}\right]$ to store blocks of up to K bits. To receive k (even or odd) bits of the variable size block $\mathbf{u}$ (message $m = \psi_{10}(\mathbf{u})$), set $\widehat{u}_0 = y_{K/2-1,R}$, $\widehat{u}_1 = y_{K/2-1,I}$,

$$\hat{u}_{2\ell} = \hat{u}_{R,\ell}, \text{ for } 0 \leq \ell \leq k_R - 1, \tag{3.8}$$

and

$$\hat{u}_{2\ell+1} = \hat{u}_{I,\ell}, \text{ for } 0 \leq \ell \leq k_I - 1. \tag{3.9}$$

The next step is to recover the Gray mapping constellation. The reverse operation of Theorem 1 and equation (3.1) should be applied, both vectors $\widehat{\mathbf{u}}_R$ and $\widehat{\mathbf{u}}_I$ are converted to a Gray sequence. This demodulating procedure, which is based on hard-decision rule, is presented in Algorithm 5, performing the reverse task performed by Algorithm 1.

Algorithm 5: Demodulating Procedure: Hard Decision Rule Algorithm

```
initialization:
 Set the constellation size to $k$.
begin
    • Step #1: Set sign bit
    $\widehat{u}_{R,0} = y_{k/2-1,R}$;
    $\widehat{u}_{I,0} = y_{k/2-1,I}$;
    • Step #2: Binary to Gray Conversion
    $\hat{\mathbf{r}}_R = \psi_{10}^{-1}(\hat{m}_R) \Rightarrow \hat{\mathbf{u}}_R = \left(\gamma_R^{-1}(\hat{\mathbf{r}}_R), \widehat{u}_{R,0}\right)$;
    $\hat{\mathbf{r}}_I = \psi_{10}^{-1}(\hat{m}_I) \Rightarrow \hat{\mathbf{u}}_I = \left(\gamma_I^{-1}(\hat{\mathbf{r}}_I), \widehat{u}_{I,0}\right)$;
    • Step #3: Amplitude Decoding
    for $0 \leq \ell \leq k/2 - 1$ do
        $\hat{u}_{i,2\ell} = \hat{u}_{R,\ell}$;
        $\hat{u}_{i,2\ell+1} = \hat{u}_{I,\ell}$;
    end
    • Step #4: Done ($\hat{\mathbf{u}}$ is the block of estimated bits).
end
```

3.1.1 Implementation in a FPGA Device

This section describes the digital circuit associated with implementation of Tech #1 in a FPGA device. The implementation was carried out with Altera Quartus II tool

[55] which corresponds to a computer aided design (CAD) software for the synthesis and analysis of digital circuitry in programmable logic devices. This software, which makes use of the programming language Verilog or very high speed integrated circuits hardware description language (VHDL) (or even a block diagram), generates a synthesizable digital circuit that may be uploaded to a FPGA chipset defining component location in an optimal way [55].

The digital circuit implemented in FPGA device that synthesizes the Tech #1 is shown in Figure 14. This has been generated through the Register Transfer Level (RTL) Viewer, a tool from Quartus II [55] that allows to visualize block diagrams and digital circuits.

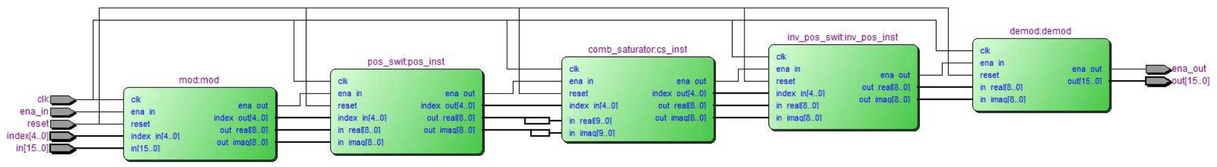


Figure 14: Block diagram of the designed digital circuit for Tech #1.

The basic components of the implemented circuit described in Section 3.1 are as follows:

- Modulation (mod): it handles the constellation mapping by direct implementation, as described in Algorithm 1.
- Position Switcher (pos_swit): The Position Switcher is responsible for activating the translation of the rectangular constellation to an cross constellation, as described in Algorithm 2.
- Combined Saturator (comb_saturator): The Combined Saturator guarantees that no constellation point falls outside the M-QAM demodulation range, as described in Algorithm 3.
- Inverse Position Switcher (inv_pos_swit): The Inverse Position Switcher is responsible for translating the cross constellation to a rectangular constellation, as described in Algorithm 4.
- Demodulation (demod): The Demodulation block handles the constellation demapping, as described in Algorithm 5.

Each block originates a digital circuit design. The modulation and demodulation components can be easily generalized and are going to be presented in Subsection 3.1.2 and Subsection 3.1.6. The other three blocks, Position Switcher, Combined Saturator and Inverse Position Switcher, can be generalized, but not in a simple way as the modulation and demodulation blocks, because it involves more complex operations. They are going to be presented in Subsections 3.1.3, 3.1.4 and 3.1.5.

3.1.2 Digital Circuit for Modulation

The modulation digital circuit is presented in Figure 15 for $K = 4$ bits and in Figure 16 for $K = 6$ bits. Each logical element group is represented by K units, therefore the number of logical elements increases as K increases. Part (a) covers algorithm inputs, which are **clk** (digital circuit clock), **reset** (synchronous digital circuit reset), **ena_in** (signaling that defines whether the input data is valid or not), **index_in** (represents k) and **in** (bit sequence to be modulated). The purpose of part (a) is to remove and store the sign bit and also to store the input bits. The flip flop is necessary in digital circuits to turn the bit sequence to a stable state, so that can be used to store the information state to the next operation.

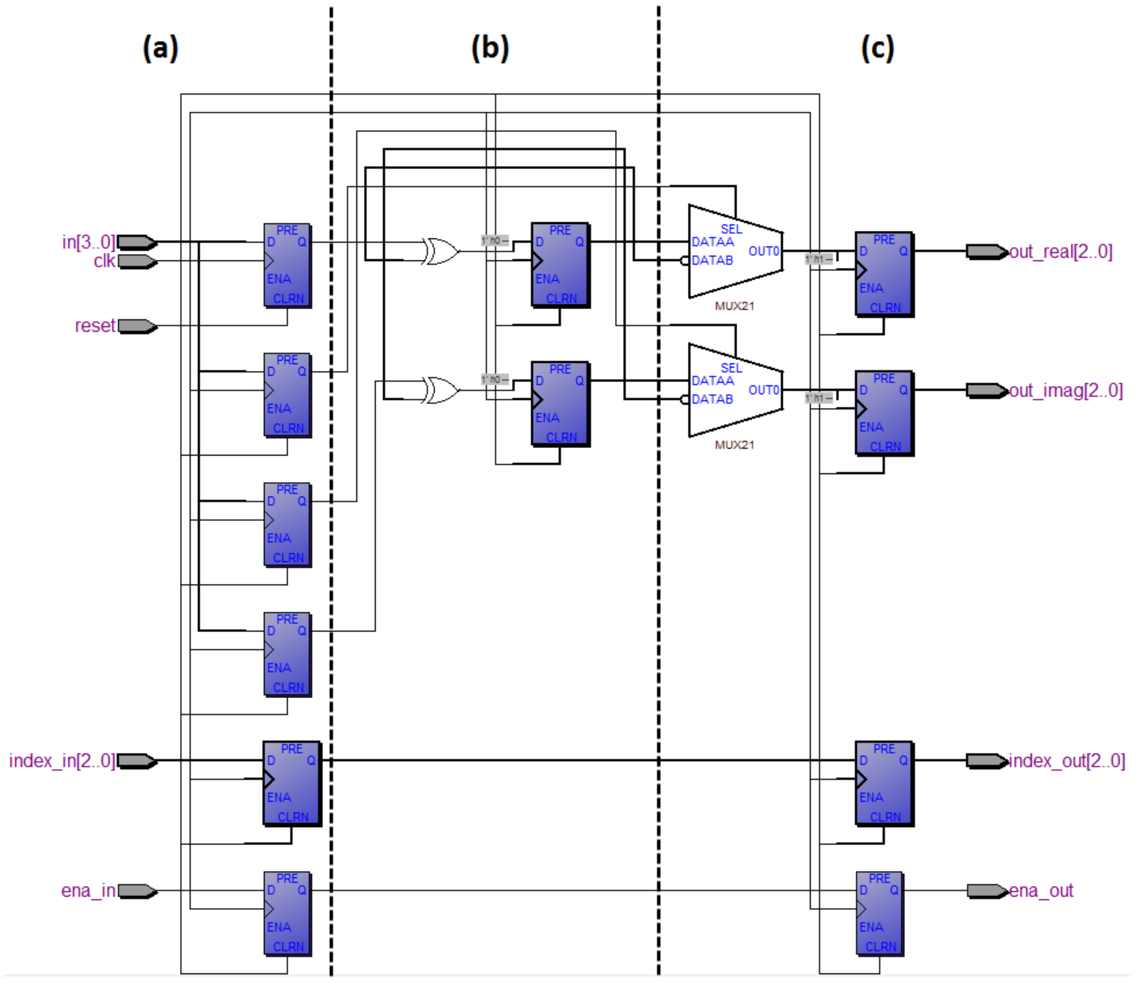


Figure 15: Digital circuit for Modulation when $K = 4$ bits.

Part (b) performs logics that convert the Gray sequence to a binary sequence, and to store this result into the flip flop chain. As it can be observed on the digital circuit of the FPGA modulation implementation in part (b), if the maximum number of bits K, which can be transmitted, switches from 4 to 6, the number of logical multiplexer (MUX) units and adders is increased by 2, and keeps increasing as K increases by 2 (only odd numbers are allowed for K, but even number of bits can be transmitted, because $k \leq K$).

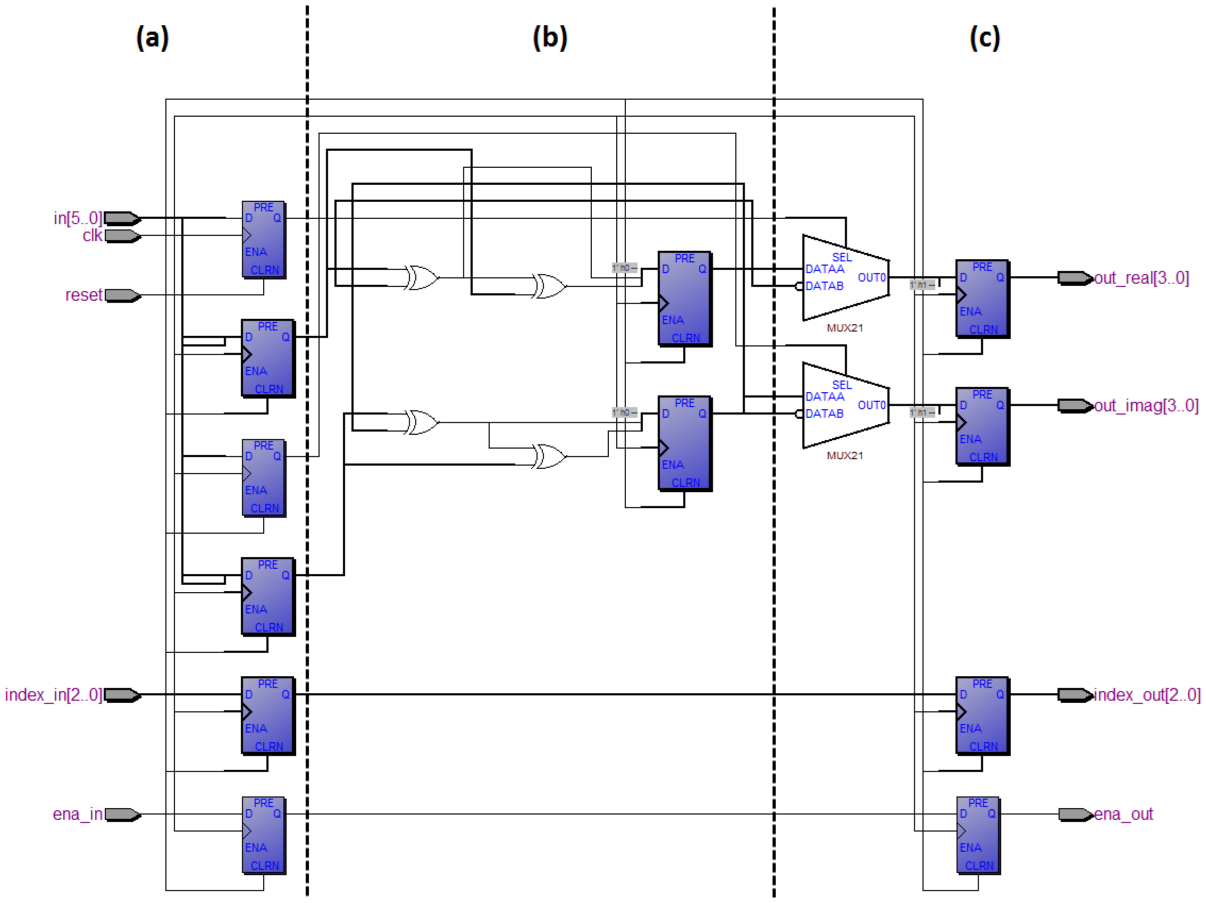


Figure 16: Digital circuit for Modulation when $K = 6$ bits.

Finally, part (c) is responsible for presenting the digital circuit outputs, which are: **ena_out** (signaling that defines whether the output data is valid or not), **index_out** (represents k, the order of the M-QAM), **out_real** (in-phase component of the modulated signal) and **out_imag** (quadrature component of the modulated signal). Also, part (c) adds back the positive and/or negative sign at the output and stores the information state to the next block.

3.1.3 Digital Circuit for Position Switcher

The Position Switcher digital circuit is presented in Figure 17 for $K = 4$ bits. Each logical element group is represented by K units, in this case the cloud representation was used to represent a high number of logical components deployed. Therefore the number of logical elements increases as K increases. Part (a) shows the digital circuit inputs, which are **clk**, **reset**, **ena_in**, **index_in** (represents k) and **in_real** (in-phase component of the modulated signal) and **in_imag** (quadrature component of the modulated signal).

Also, part (a) removes the sign bit, makes the first exponential operation in base 2 to start the translation operation and to store the information state.

Part (b) processes the logic to convert a rectangular constellation (for odd k values) to a cross constellations by translation of the corner points, and also store the information state. If the maximum number of bits K that can be transmitted increases, the number of logical MUX units and adders also increases.

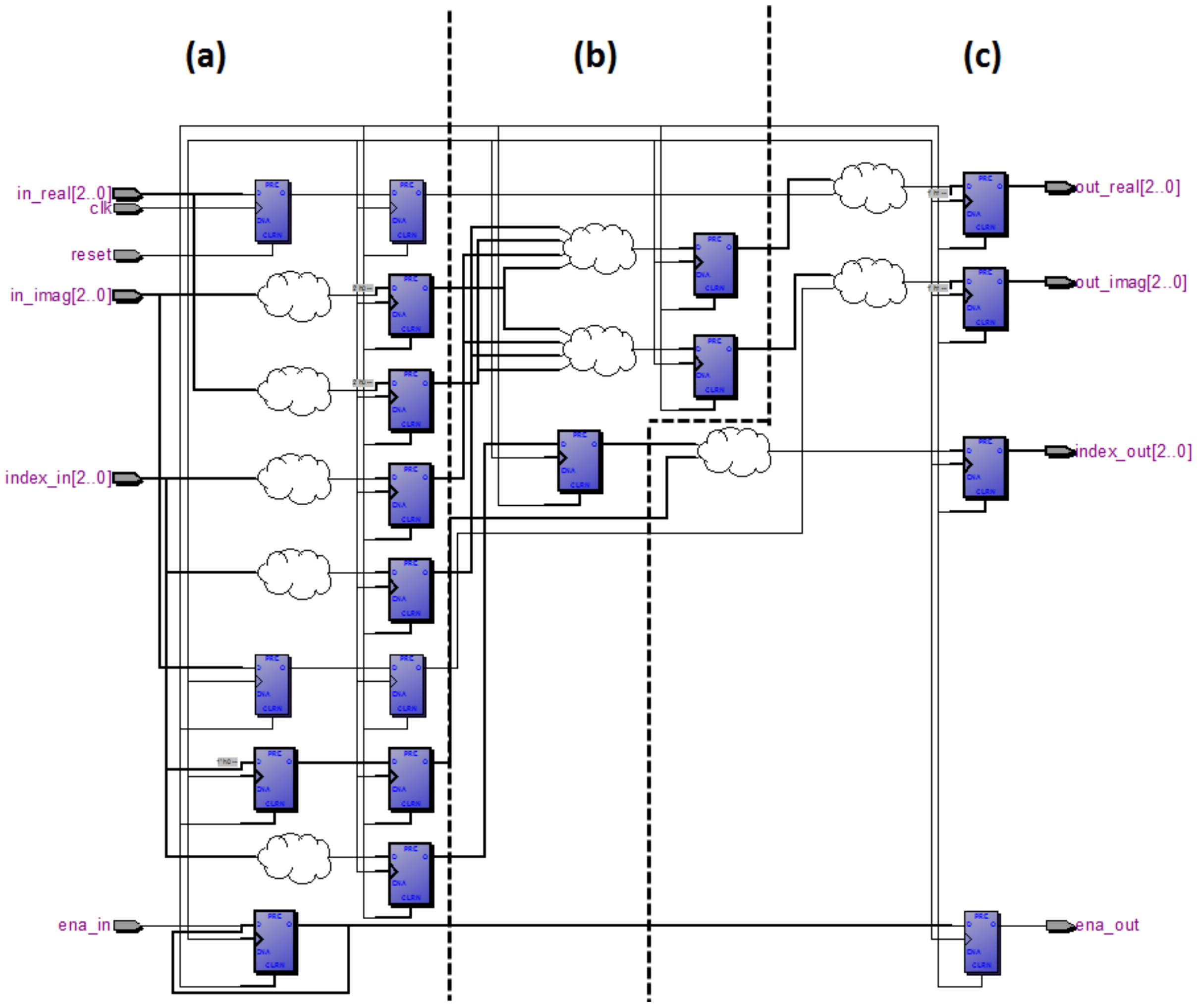


Figure 17: Digital circuit for Position Switcher when $K = 6$ bits.

Finally, part (c) shows the digital circuit outputs, which are: **ena_out**, **index_out**, **out_real** (in-phase component of the modulated signal) and **out_imag** (quadrature component of the modulated signal). Also, part (c) adds back the positive and/or negative sign at the output and executes the last flip flop chain, to store the information state.

3.1.4 **Digital Circuit for Combined Saturator**

The Combined Saturator digital circuit is presented in Figure 18 for $K = 4$ bits. Each logical element group is represented by K units and the same cloud notation is employed. Therefore the number of logical elements increases as K increases. Part (a) shows the digital circuit inputs, which are **clk**, **reset**, **ena_in**, **index_in** and **in_real** (in-phase component of the modulated signal) and **in_imag** (quadrature component of the modulated signal). Also, part (a) removes the sign bit, makes the first exponential operation in base 2 to start the Combined Saturator operation and performs to store the information state.

Part (b) processes the logic that changes the received signal position to a valid cross/square M-QAM constellation depending on the value of the transmitted k. It first analyses the edge (for odd k values) and then the corners, and also stores the information state twice for two different operations. If the maximum number of bits K that can be transmitted increases, the number of logical adders units and adders is increases.

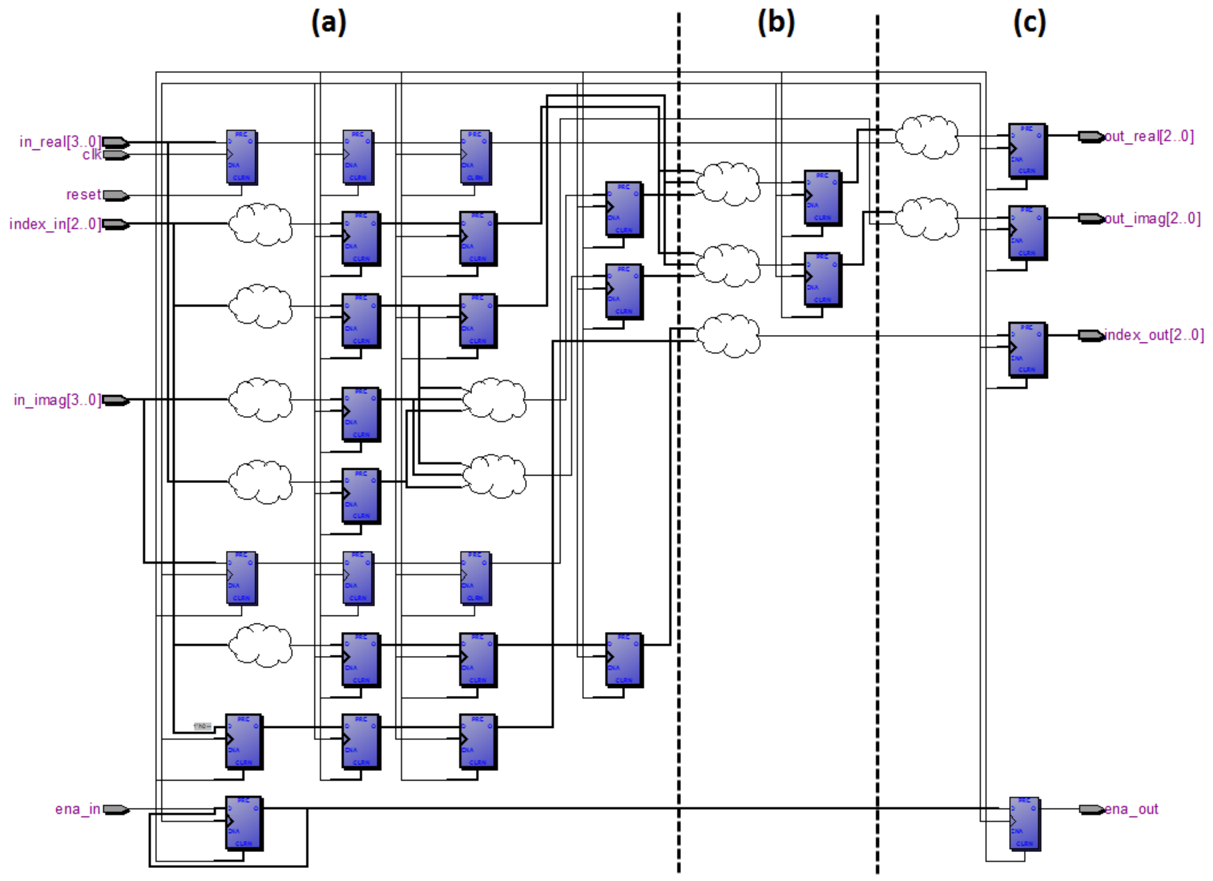


Figure 18: Digital circuit for Combined Saturator when $K = 6$ bits.

Finally, part (c) shows the digital circuit outputs, which are: **ena_out**, **index_out**, **out_real** (in-phase component of the modulated signal) and **out_imag** (quadrature component of the modulated signal). Also, part (c) adds back the positive and/or negative sign at the output and store the information state at the end of the operation.

3.1.5 Digital Circuit for Inverse Position Switcher

The Inverse Position Switcher digital circuit is presented in Figure 19 for $K = 4$ bits. Each logical element group is represented by K units using the same cloud notation as previously. Therefore the number of logical elements increases as K increases. Part (a) shows the digital circuit inputs, which are **clk**, **reset**, **ena_in**, **index_in** and **in_real** (in-phase component of the modulated signal) and **in_imag** (quadrature component of the modulated signal). Also, part (a) removes the sign bit, makes the first exponential operation in base 2 to start the translation operation and to store the information state.

Part (b) processes the logic that converts an cross constellation (for odd k values) to a rectangular constellations by translation of the corner points, and also store the information state. If the maximum number of bits K that can be transmitted switches, the number of logical MUX units and adders are increased, the difference depends on the value of K.

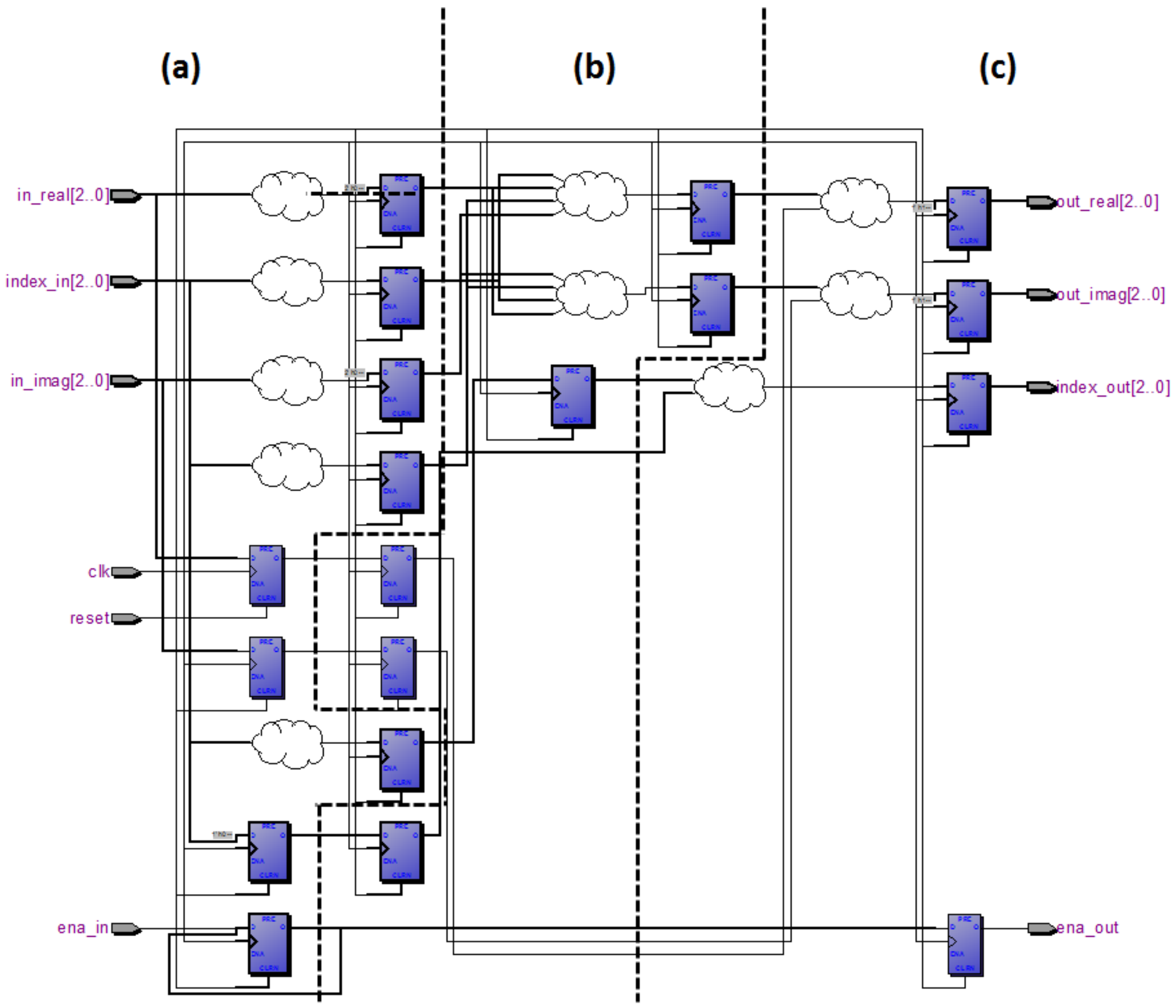


Figure 19: Digital circuit for Inverse Position Switcher when $K = 6$ bits.

Finally, part (c) shows the digital circuit outputs, which are: **ena_out**, **index_out**, **out_real** (in-phase component of the modulated signal) and **out_imag** (quadrature component of the modulated signal). Also, part (c) adds back the positive and/or negative sign at the output and store the information state of the resulting operation.

3.1.6 Digital Circuit for Demodulation

The demodulation digital circuit is presented in Figure 20 for $K = 4$ bits and in Figure 21 for $K = 6$ bits. Each logical element group is represented by K units. Therefore the number of logical elements increases as K increases. Part (a) shows the digital circuit inputs, which are **clk**, **reset**, **ena_in** and **in_real** (in-phase component of the modulated signal) and **in_imag** (quadrature component of the modulated signal). Also, part (a) removes the sign bit and to store the information state.

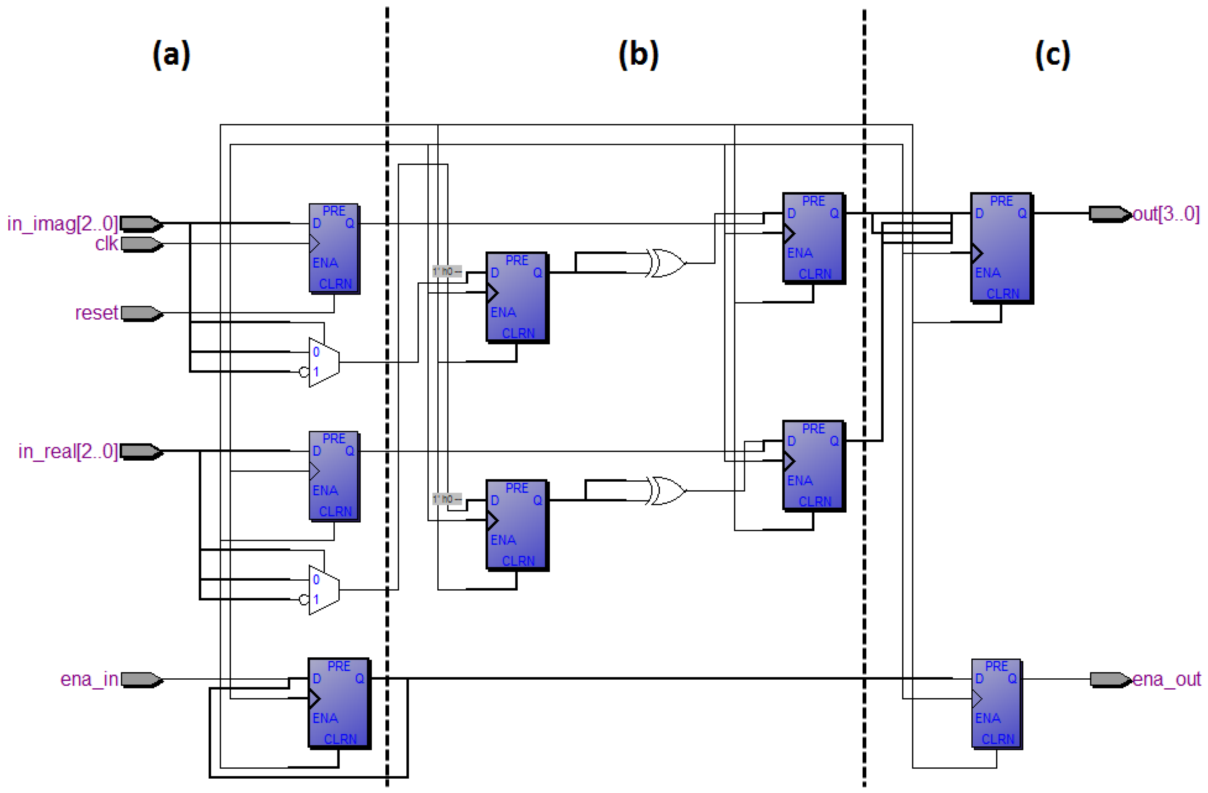


Figure 20: Digital circuit for Demodulation when $K = 4$ bits.

Part (b) processes the logic that converts the binary sequence to a Gray sequence, doing the binary to Gray transformation, and also store the information state. As it can be observed in part (b), if the maximum number of bits K that can be transmitted switches from 4 to 6, the number of logical MUX units and adders are increased by 2, and keeps increasing as K increases by 2, as well.

Finally, part (c) shows the digital circuit outputs, which are: **ena_out**, **out** (resulting modulated bit sequence). Also, part (c) adds back the positive and/or negative sign at the output and store the information state that is the result of the demodulation block.

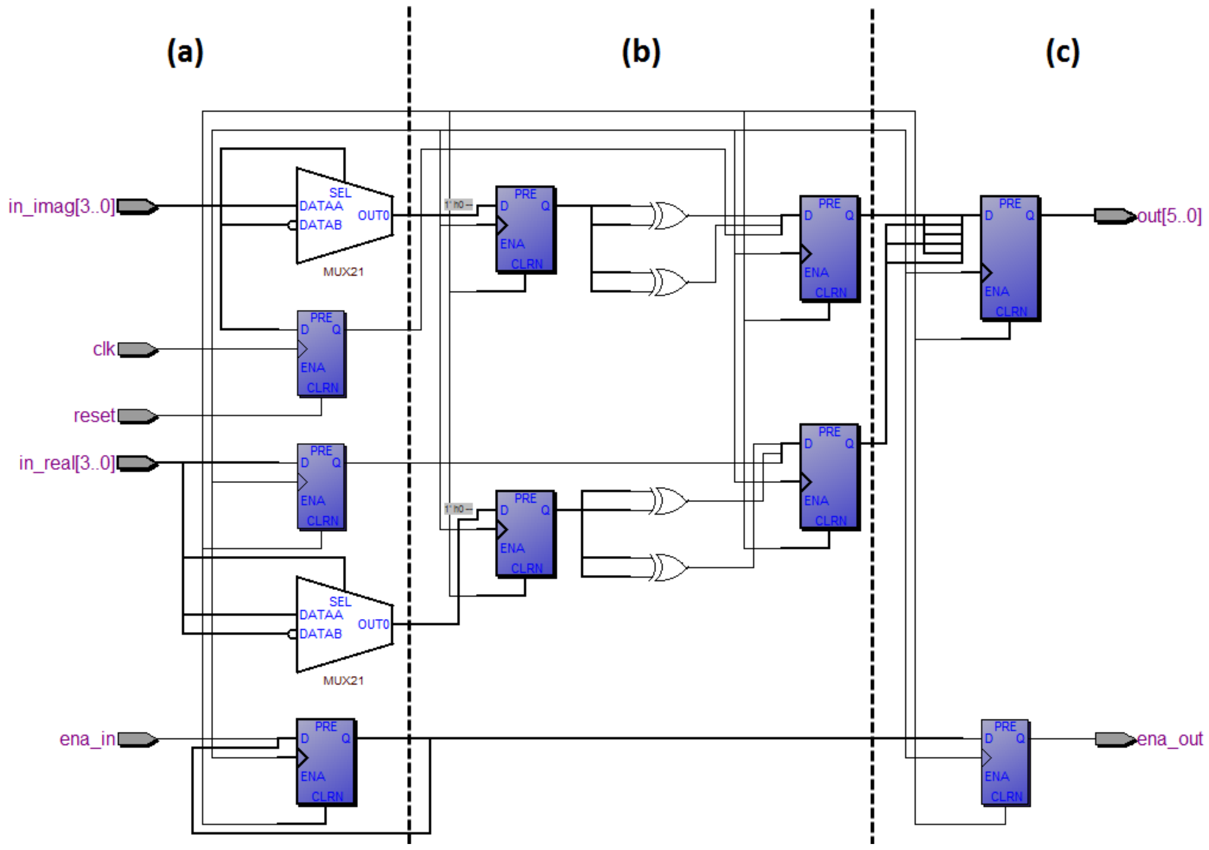


Figure 21: Digital circuit for Demodulation when $K = 6$ bits.

For comparison purpose, the Position Switcher, Combined Saturator and Inverse Position Switcher blocks use more logical elements than the Modulation and Demodulation blocks, as it can be observed in Table 1 for $K = 6$ bits and in Table 2 for $K = 16$. In these tables, LC Combinationals represents the simple Logical Element that can be a MUX or adders, and LC Registers represents one LC Combinational together with a register block. In this case, the three blocks are occupying 83.38% of the designed digital circuit for $K = 6$ bits and 83.96% of the designed digital circuit for $K = 16$ bits. If only square modulation is considered, the Position Switcher would not be necessary, as well as the Inverse Position Switcher, and, also, the Combined Saturator could be simplified to a simple saturator by analysing the in-phase and quadrature components individually, reducing the computational complexity significantly. The digital circuit can be analysed in Verilog codification and is shown in Annex B.

3.2 Technique #2 - Enhanced HDR for M-QAM

This section introduces an improved version of the heuristic decision region (HDR) technique - enhanced HDR technique, which was proposed in [23, 24] to modulate and demodulate/detect square and cross M-QAM symbols. The HDR technique makes use of the Gray mapping modulating function f to generate square M-QAM symbols by using a LUT which stores the pairs of values $(s_{m,R}, s_{m,I})$, which correspond to all possible values

Table 1: Logic elements allocation considering $K = 6$ bits.

Digital Circuit	**Logic Elements**	**Memory Bits**
Modulation	30	0
Position Switcher	85	0
Combined Saturator	137	0
Inverse Position Switcher	64	0
Demodulation	27	0
Total	**343**	**0**

Table 2: Logic elements allocation considering $K = 16$ bits.

Digital Circuit	**Logic Elements**	**Memory Bits**
Modulation	84	0
Position Switcher	272	0
Combined Saturator	364	0
Inverse Position Switcher	207	0
Demodulation	77	0
Total	**1004**	**0**

that $\mathbf{u}_i$ can assume. The way that HDR technique works for square constellation can be easily understood by using the following example: Let the one-dimensional Gray mapping modulating function γ specified in Table 3. Notice that γ is a function that maps a block $\mathbf{u}^{[1]} \in \mathbb{B}^3$ into points $s_{m,R}$ belonging to the signal constellation $\mathcal{C}_1 = \{s_i : i \in \mathbb{Z}_7\}$[4] in which, if $|j - i| = 1$, s_i and s_j, are neighbours. In this case $m = \psi_{\mathbf{10}}(\mathbf{u})$ and, from (2.9), $s_{m,R} = \gamma(\mathbf{u})$. Let us say that the binary input block $\mathbf{u} = 111$ has to be transmitted. The value $m = \psi_{\mathbf{10}}(111) = 7$ could represent the memory address, so the value $\mathbf{s}_7 = 3$ would be retrieved and provisions would be taken to transmit the corresponding pulse. Of course the values in Table 3 has to be stored, then a LUT applies. The demodulation g is performed by using a HDR rule, for that case, it is needed one HDR rule (or a SM), for each M-QAM constellation. Figure 22 presents the state machine that describes the 16-QAM detection based on HDR rule. As it can be observed, the detection is simplified to a binary tree, with a few comparisons between multiple decision regions in a SM. The SM stays on the idle state until it receives valid data input. Considering that X represents $s_{m,R}$ and/or $s_{m,I}$, the data input is compared to the decision region borders, which are are -2,0 and 2 for 16-QAM. In this case 7 possible states are enough, but considering that the number of possible states is $n_k = 2^{(k/2+1)} - 1$, for high-order M-QAM constellations this number increases significantly, for example, for $k = 16$, there are 511 possible states. Note that for $k = 2, ..., 15$ each constellation has a dedicate state machine associated, so the number of logic operations increases exponentially while increasing M.

The Enhanced HDR technique for M-QAM constellations, named Tech #2 in this thesis, applies Gray mapping modulating function f by using a single LUT that stores

[4] It will be used the notation $\mathbb{Z}_{2^K}$ to represent the set of integers k such that $0 \leq k \leq K - 1$.

Table 3: Gray mapping: $m = \psi_{\mathbf{10}}(\mathbf{m})$, $s_m = \gamma(\mathbf{u})$.

$\mathbf{u}$	000	001	011	010	110	111	101	100
m	0	1	3	2	5	7	5	4
s_m	-7	-5	-3	-1	1	3	5	7

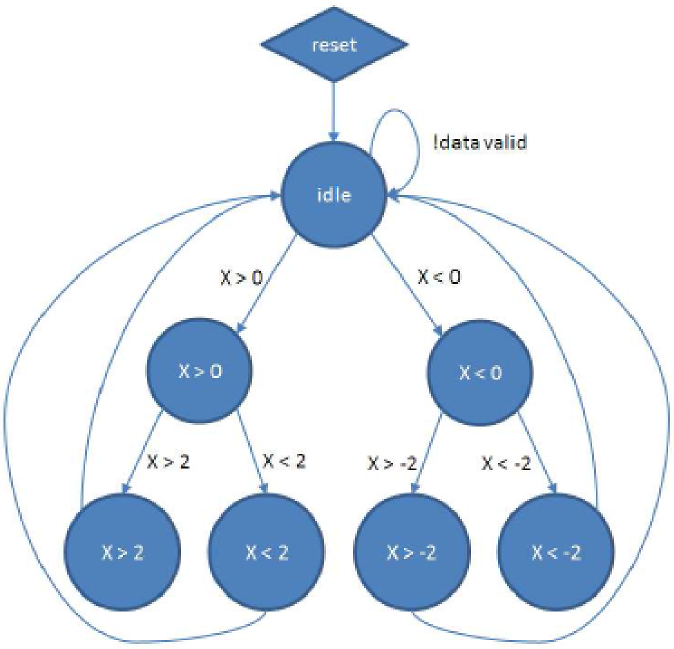


Figure 22: State machine for 16-QAM detection.

all pairs of values $(s_{m,R}, s_{m,I})$ and because of that make uses of a single HDR rule, which is implemented by a state machine, for de-mapping/detection. Tech #2 which is an improved version of HDR technique [23, 24], covers even and odd number of input bits $k = 2, 3, \ldots, K$, satisfying the Gray mapping rule to represent M-QAM symbols of any order, just changing the value of k, from 2 to 16. The main innovations of this technique are: i) obtaining low-order M-QAM constellation points from higher ones, that reduces the need for hardware resource utilization and computational complexity; ii) introduction of cross to rectangular and rectangular to cross translations procedures, which was proposed in Tech #1, to reduce computational complexity; iii) capacity to handle cross and square M-QAM symbols, similar to Tech #1; and iv) the use of only one LUT and only one SM to perform the modulation and demodulation for all M-QAM constellations, $M = 2^K$, in which K is the largest number of bits that can be transmitted, such that $k = 2, 3, \ldots, K$. Overall, Tech #2 converts Gray coded symbols to generate the constellations indexes in the transmitter that are stored in a LUT for the highest M-QAM constellation order (K), and to recover the constellation point by using only one SM in the receiver. It means that, low-order M-QAM constellation points can be obtained from high-order ones.

For square and rectangular M-QAM constellations, the derivation of lower-order constellations from higher ones is a simple task to be accomplished because the decimal representation of the input bits $\mathbf{u}$ shall receive the same index, independently of the M-QAM constellation order as previously presented for Tech #1. So the same example applies, the bit sequence $\mathbf{u} = 0011$ represents the decimal number 3, and, as a consequence, receives the index constellation point $s_m = (-1, -1)$ if 16-QAM applies; therefore, $\mathbf{u} = 000011$ represents the decimal number 3 and receives the index constellation point $s_m = (-1, -1)$ if 64-QAM applies, and so on. As a result, all points of square or rectangular M-QAM constellation with order lower than $M = 2^K$ can be obtained from square or rectangular 2^K-QAM constellations. Different from Tech #1 there is no intermediate variable, it is a direct consult to the LUT, and the same procedure is executed to rectangular M-QAM.

The proposed Tech #2 makes use of rectangular to cross and cross to rectangular translations to cover cross M-QAM constellations as presented in Tech #1. Then, Tech #2 can also represent any Gray coded M-QAM symbols (odd or even k).

Figure 23 shows the flowchart of the Enhanced HDR technique. As it can be observed, the Position Switcher, Combined Saturator and Inverse Position Switcher are only necessary when k is odd (cross constellation). The difference between Tech #1 and Tech #2 relies in the adopted modulation and demodulation procedures. Tech #2 makes use of a pre fixed LUT to store each point of the highest M-QAM constellation (storage demand exponentially increases as K increases). This procedure has been employed to attend international standards [1, 3, 4], to do so one rule should be defined to fulfil the LUT, but it is highly recommended to attend the Gray mapping constellation rule and to attend different M-QAM order with the same M-QAM constellation in order to reduce it is computational complexity. Therefore, the demodulation process is based on HDR, simplifying the demodulation process in finding the right decision rule that contains the constellation point to be demodulated into only one SM. The procedure starts by dividing the decision region in two sub-regions, then the subregion that contains the constellations point is chosen, then if there is a unique constellation point, the output bit sequence $\hat{\mathbf{u}}$ is obtained, but if there are more than one possible constellation point, the subregion is once again divided into two new subregion, until there is a unique constellation point. This procedure has to be done for both in-phase and quadrature components, independently.

The idea behind of Tech #2 is to use the same $\mathbf{u}_i$ index to any M-QAM constellation order. Notice that these procedures can only be applied to square M-QAM constellation, so that different cross M-QAM constellations, will still require different LUT and SM. To handle this, rectangular constellations can be considered and then rectangular to cross translation (usage of Algorithm 2) are applied on the transmitter (TX) as introduced in Tech #1, see Section 3.2. The reverse process guarantees that the demodulation will get the right index points, as described for the Combined Saturator (Algorithm 3), and for the cross to rectangular translation (Algorithm 4). Section 3.2.1 describes the implementation

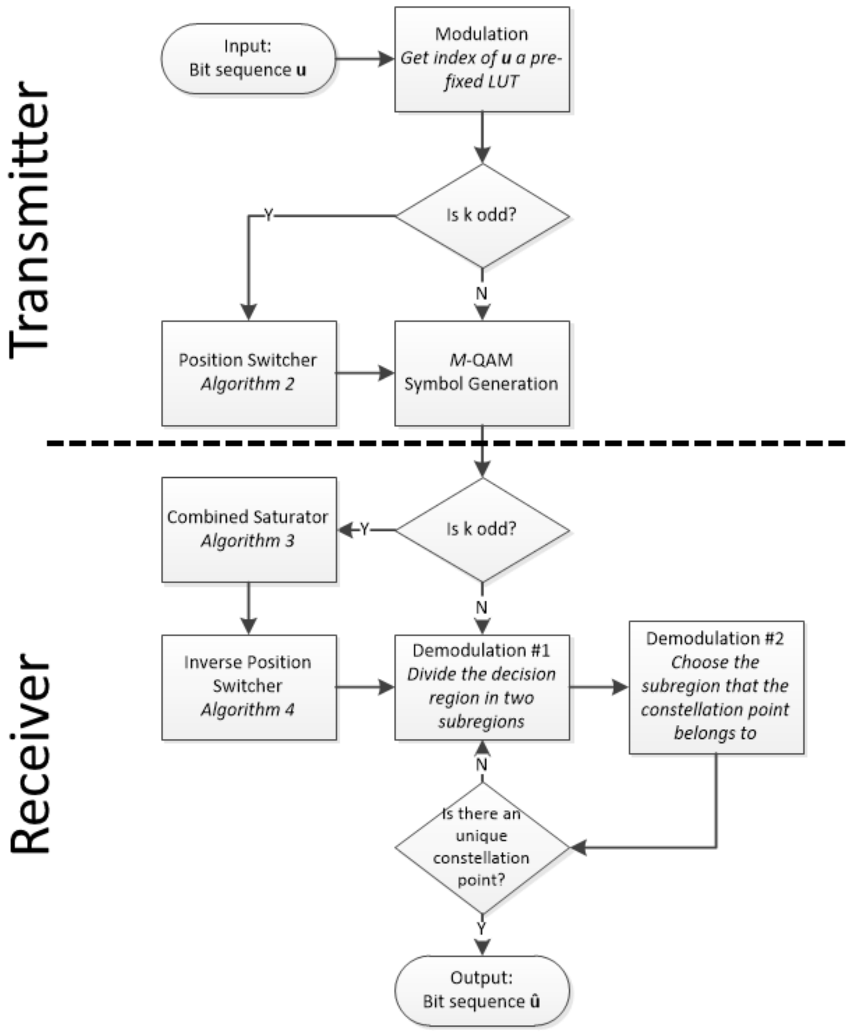


Figure 23: Flowchart for Tech #2.

in a FPGA device.

3.2.1 Implementation in a FPGA Device

This section describes the digital circuit associated with implementation of Tech #2 in a FPGA device. The digital circuit implemented in FPGA device that synthesizes the Tech #2, is shown in Figure 24. This has been generated through the Register Transfer Level (RTL) Viewer, a tool from Quartus II [55] that allows to visualize block diagrams and digital circuits.

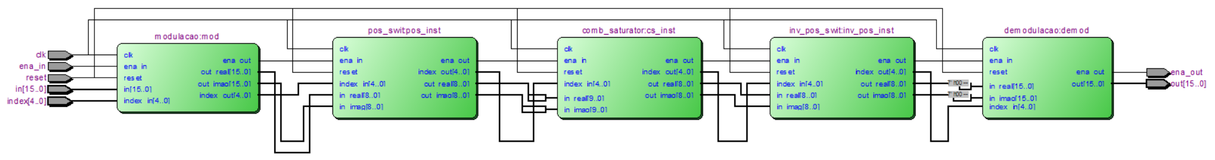


Figure 24: Block diagram of the designed digital circuit for Tech #2.

The basic components of the implemented circuit described in Section 3.2 are as follows:

- Modulation (modulacao): it handles the constellation mapping by using a single LUT, as described in Section 3.2.
- Position Switcher (pos_swit): responsible for activating the translation of rectangular constellation to cross constellation, as described in Algorithm 2 and Section 3.1.1.
- Combined Saturator (comb_saturator): guarantees that any constellation point falls outside the M-QAM demodulation range, as described in Algorithm 3 and Section 3.1.1.
- Inverse Position Switcher (inv_pos_swit): responsible for translating the cross constellation to rectangular constellation, as described in Algorithm 4 and Section 3.1.1.
- Demodulation (demodulacao): The Demodulation block handles the constellation de-mapping using a single SM, as described in Section 3.2.

Each block originates a digital circuit design and has its own characteristics. The modulation and demodulation components are going to be present in Subsection 3.2.2 and Subsection 3.2.3. The other three blocks, Position Switcher, Combined Saturator and Inverse Position Switcher were previously presented in Subsections 3.1.3, 3.1.4 and 3.1.5, respectively.

3.2.2 Digital Circuit for Modulation

The modulation digital circuit is presented in Figure 25 for $K = 4$ bits and Figure 26 for $K = 6$ bits. The algorithm inputs are **clk** (digital circuit clock), **reset** (synchronous digital circuit reset), **ena_in** (signaling that defines whether the input data is valid or not), **index_in** (represents k) and **in** (bit sequence to be modulated). As it can be observed in the digital circuit of the modulation implemented in a FPGA device, if the maximum number of bits K, which can be transmitted, switches from 4 to 6, the modulation table (green border) switches from two logic table using MUX to two random access memory (RAM). This is due to the fact that the size of the LUT increases considerably, and for $K \geq 6$ the Quartus toll optimizes to the memory usage. Flip flops at the output are used to synchronize the data output with control signals. Finally, the digital circuit outputs are: **ena_out** (signaling that defines whether the output data is valid or not), **index_out** (represents k, the order of the M-QAM), **out_real** (in-phase component of the modulated signal) and **out_imag** (quadrature component of the modulated signal).

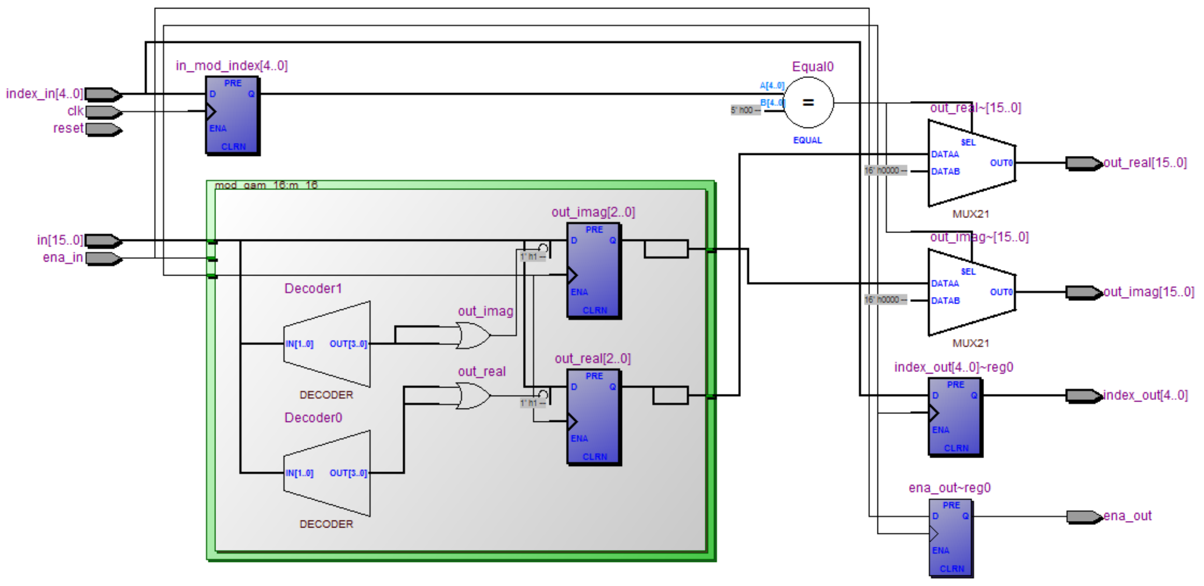


Figure 25: Digital circuit for Modulation of Tech #2 when $K = 4$ bits.

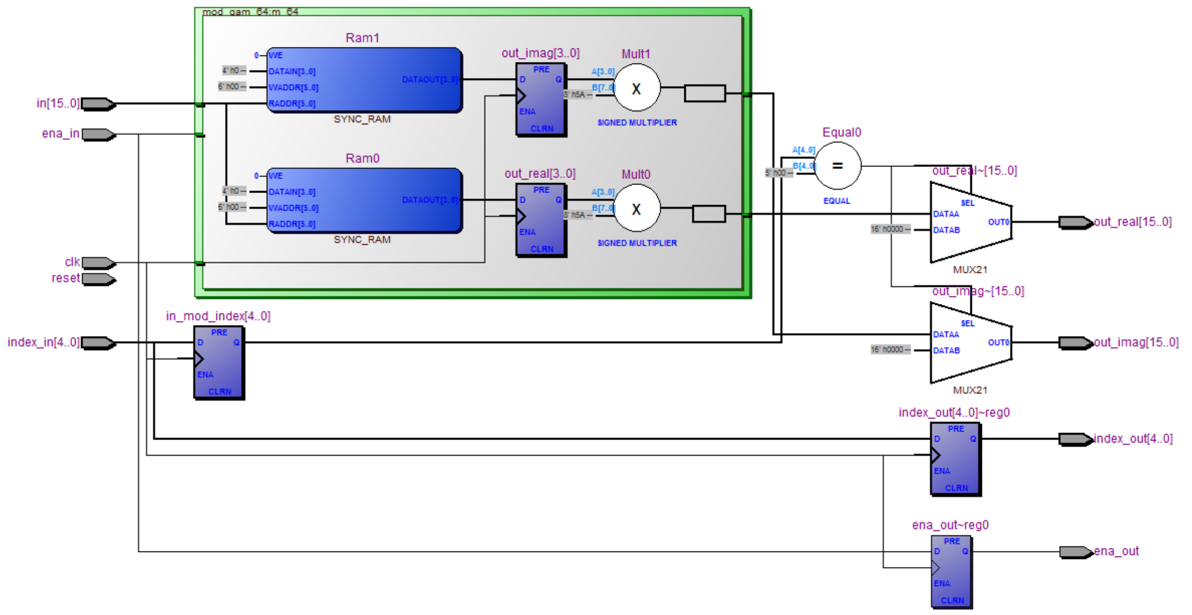


Figure 26: Digital circuit for Modulation of Tech #2 when $K = 6$ bits.

3.2.3 Digital Circuit for Demodulation

The demodulation digital circuit is presented in Figure 27 for $K = 4$ bits and Figure 28 for $K = 6$ bits. The algorithm inputs are **clk**, **reset**, **ena_in**, **index_in** and **in_real** (in-phase component of the modulated signal) and **in_imag** (quadrature component of the modulated signal).

As it can be observed on the digital circuit of the FPGA demodulation implementation, if the maximum number of bits K, which can be transmitted, switches from 4 to 6, the demodulation table (green border) as well as the number of possible states (yellow blocks) increases enormously, due to the size of the state machine. The number of

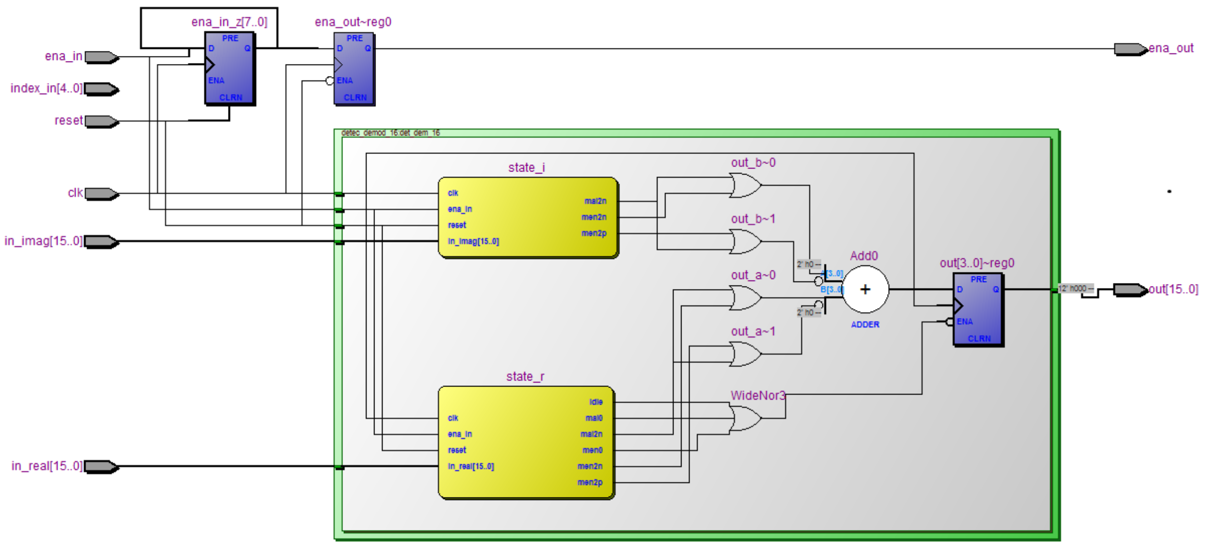


Figure 27: Digital circuit for Demodulation of Tech #2 when $K = 4$ bits.

possible states n in the SM is calculated by the number of K bits that can be transmitted, $n = 2^{(K/2+1)} - 1$, so for $K = 4$ bits (Figure 27), there are 7 possible states, and for $K = 6$ bits (Figure 28), there are 15 possible states. The other flip flops are used to synchronize the data output with the control signals. Finally, the digital circuit outputs are: **ena_out** and **out** (resulting modulated bit sequence).

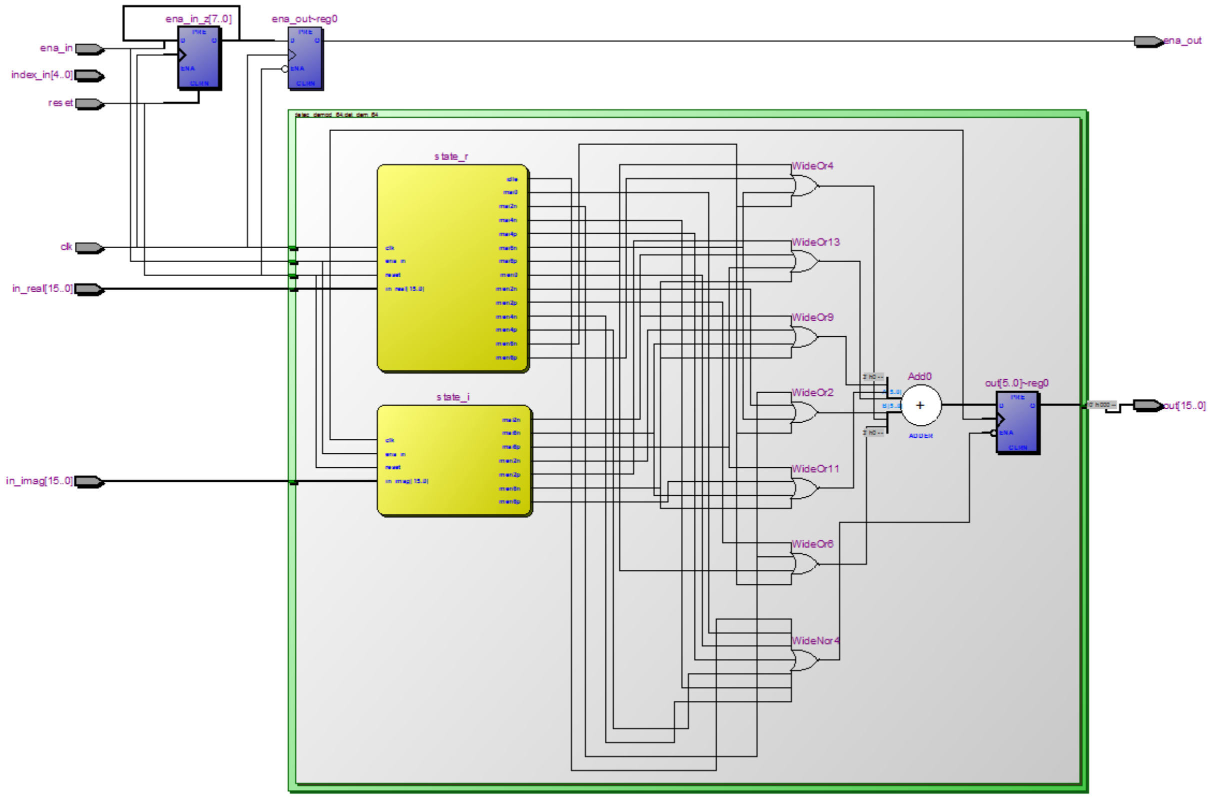


Figure 28: Digital circuit for Demodulation of Tech #2 when $K = 6$ bits.

4 NUMERICAL RESULTS

This section addresses performance, in terms of BER, and computational complexity analyzes, in terms of hardware resource utilization in a FPGA device (maximum operating frequency, energy consumption, delay, latency, memory usage and logic elements usage), for the technique in [23, 24], and for the proposed ones (Tech #1 and Tech #2). For the sake of simplicity, the technique introduced in [23, 24] will be named Tech #3.

To carry out simulations to verify the performance of the three techniques, the parameter $E_b/\mathcal{N}_0$ is the ratio between the average energy of the transmitted bits (E_b) and the power spectral density of the background noise, $\mathbf{v}_{bkgr}$, for AWGN or AIGN channel models ($\mathcal{N}_0$); $E_b = 0$ dB, $\mathcal{N}_0{=}\sigma^2_{bkgr}$ and $\mathcal{N}_1{=}G\mathcal{N}_0$ ($G{=}10$), where $\mathcal{N}_1$ denotes the variance of the impulsive noise and $E_b/\mathcal{N}_0$ values range from 0 dB to the point where BER reaches the value of 10^{-5}. The impulsive noise assumes to be zero-mean Gaussian random process. For each point in the plane BER $\times$ $E_b/\mathcal{N}_0$, it is established a value of at least 150 errors to stop the simulation and also a minimum of 10^6 transmitted symbols.

The computational complexity analyses are based on the Stratix III, a family of devices with FPGA technology developed by Altera Corporation. The Stratix III offers devices with densities up to 337,500 logic elements, memory blocks that can be configured as read only memory (ROM), RAM, first in, first out (FIFO) buffers, maximum of 1.120 inputs/outputs and 12 PLL inmates with up to 5 outputs. It also include high-speed external interface for double data rating (DDR), synchronous dynamic random access memory (SDRAM) and quad data rate (QDRII) static random access memory (SRAM) access at frequencies up to 200 MHz [56]. The FPGA chipset EP3SL150F1152C2 was used in this implementation. The main characteristics of this device are shown in Table 4.

Table 4: Resources of the chipset Stratix III

Resources	**EP3SL150F1152C2**
Logic elements (LE)	142,500
Memory (Kbits)	5,499
DSP Block 18x18	384
PLL	8
MAX IO	744

This chapter is organized as follows: Section 4.1 and 4.2 present analyses related to square and cross M-QAM constellations, respectively, which includes the BER analysis, maximum operating frequency, energy consumption, delay, latency, memory usage and logic elements usage. Section 4.3 addresses analyses related to the combination of square and cross M-QAM constellations into the same implementation.

4.1 Square QAM Constellations

In this section, the numerical results of the square M-QAM are presented, considering $K = 2, 4, \ldots, 16$, that is, $M = 4, 16, \ldots, 65536$. In this case, the functionalities covered by Algorithms 2, 3 and 4 are suppressed for Tech #1 and Tech #2, because the Position Switcher and the Inverse Position Switcher are only necessary for cross M-QAM and the Combined Saturator can be substituted to a bit relocation as shown in Chapter 3, see Equations (3.6) and (3.7).

4.1.1 **BER Analysis**

BER performance comparisons among Tech #1, Tech #2 and Tech #3 are shown in Figures 29-36 for AWGN and in Figures 37-44 for AIGN. The results shown that the performance loss introduced by the proposed techniques are, on average, less than 1 dB when theoretical results are used for comparison. For AIGN, Tech #1, Tech #2 and Tech #3 shows almost the same performance.

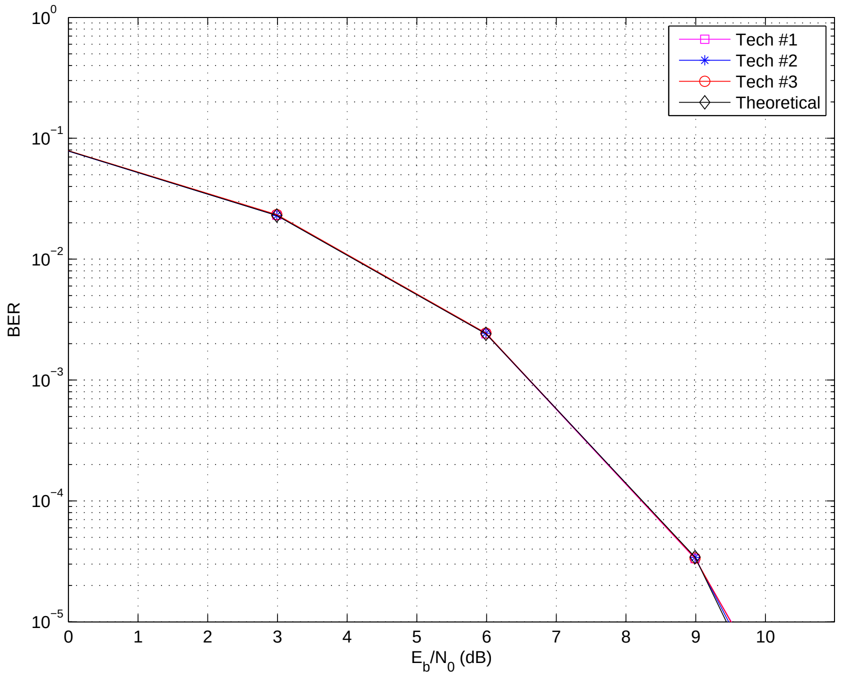


Figure 29: BER $\times$ $E_b/\mathcal{N}_0$ for 4-QAM corrupted by AWGN.

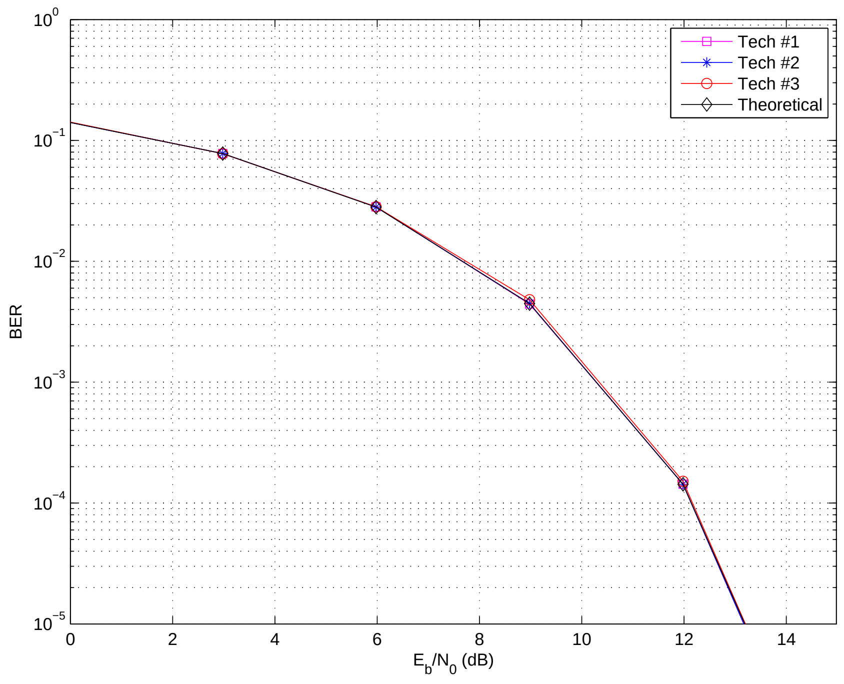


Figure 30: BER $\times$ $E_b/\mathcal{N}_0$ for 16-QAM corrupted by AWGN.

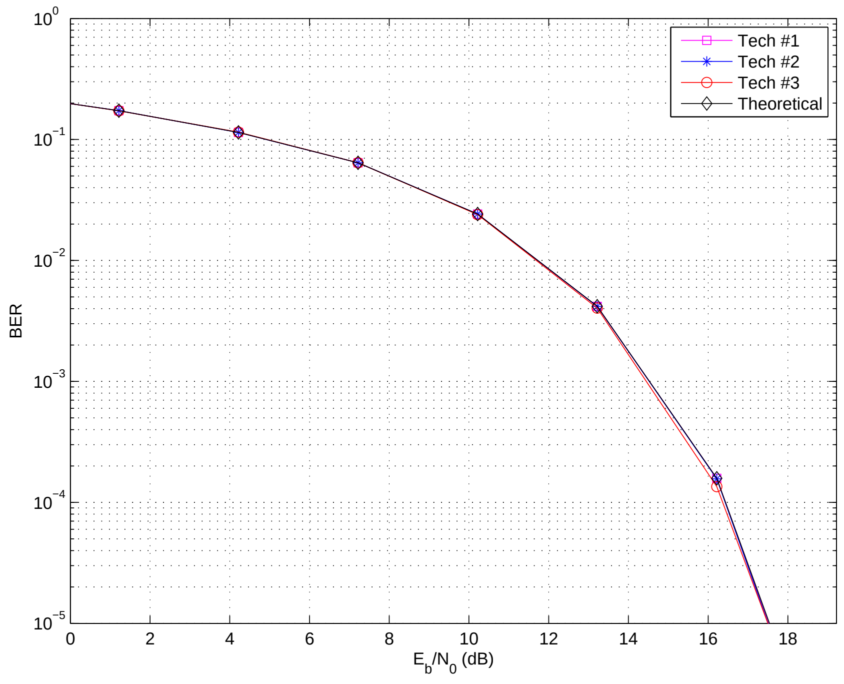


Figure 31: BER $\times$ $E_b/\mathcal{N}_0$ for 64-QAM corrupted by AWGN.

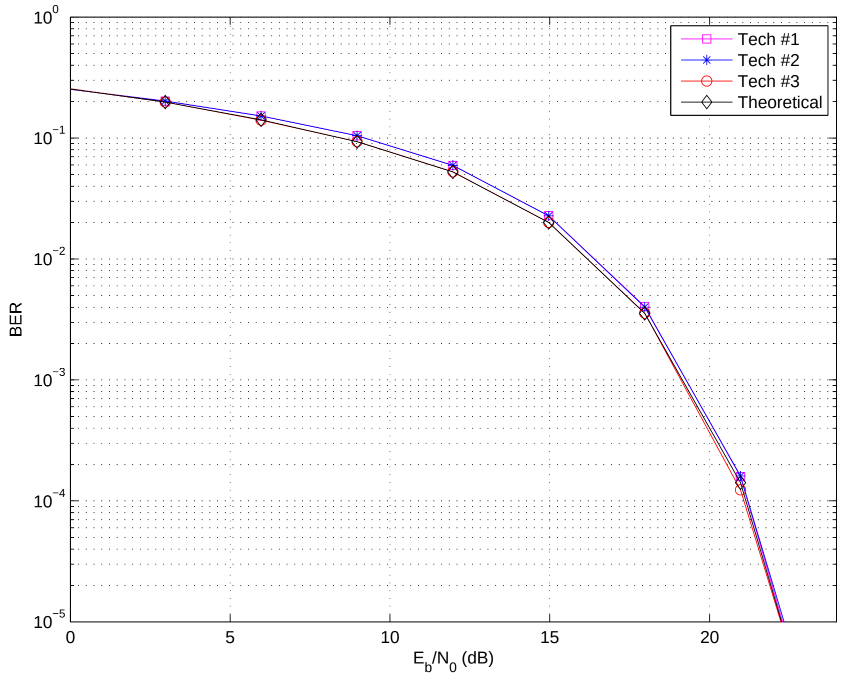


Figure 32: BER $\times$ $E_b/\mathcal{N}_0$ for 256-QAM corrupted by AWGN.

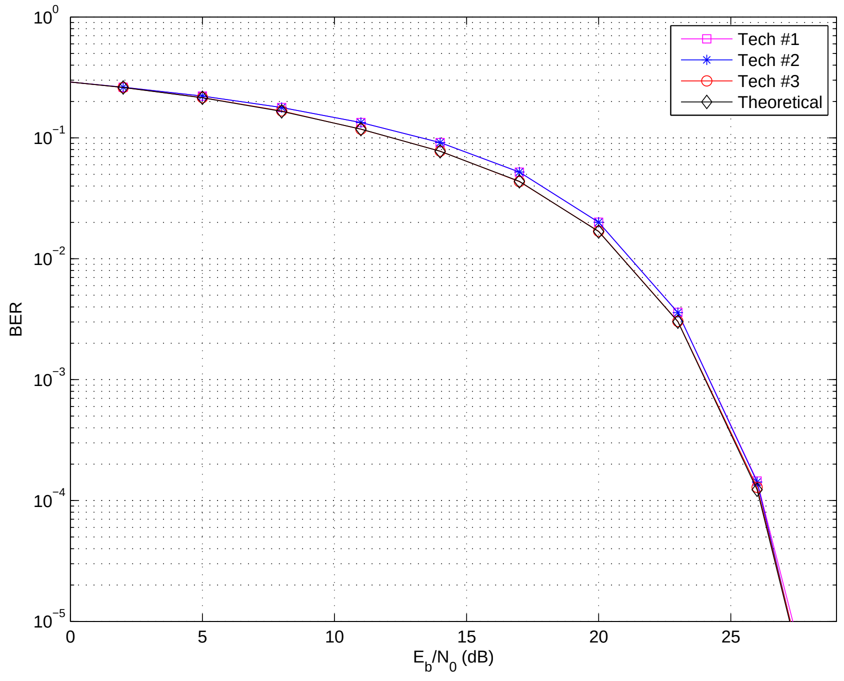


Figure 33: BER $\times$ $E_b/\mathcal{N}_0$ for 1024-QAM corrupted by AWGN.

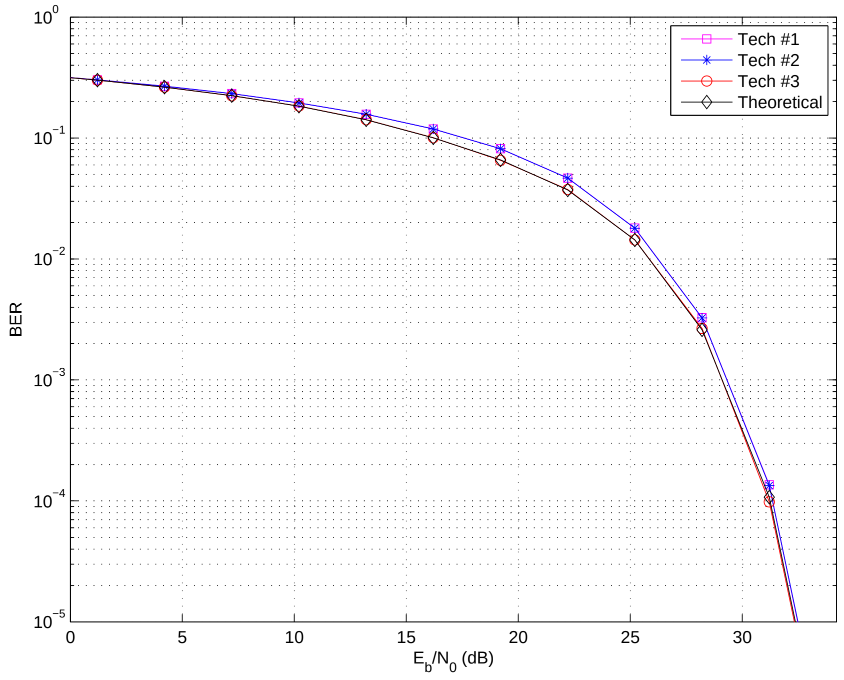


Figure 34: BER $\times$ $E_b/\mathcal{N}_0$ for 4096-QAM corrupted by AWGN.

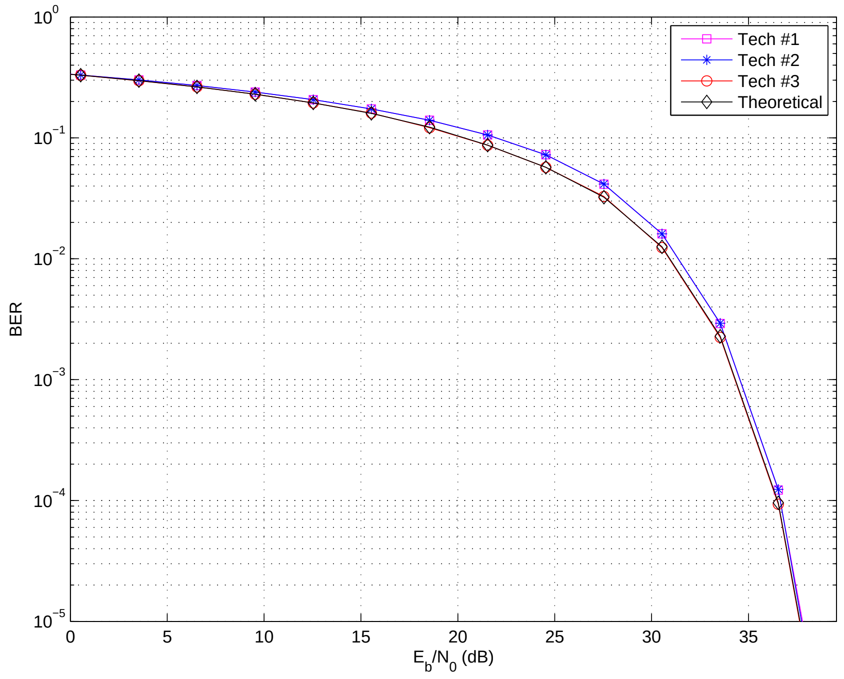


Figure 35: BER $\times$ $E_b/\mathcal{N}_0$ for 16384-QAM corrupted by AWGN.

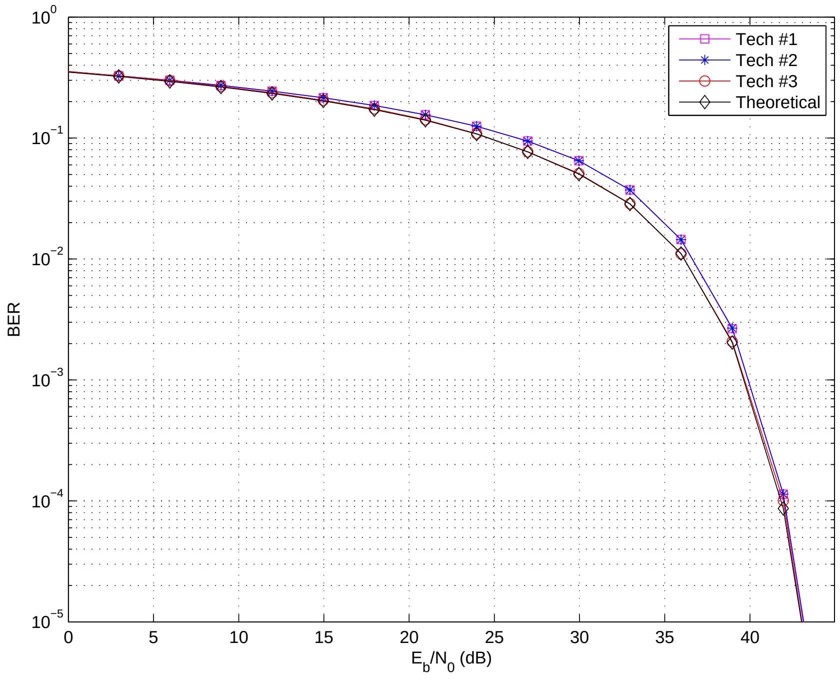


Figure 36: BER × $E_b/\mathcal{N}_0$ for 65536-QAM corrupted by AWGN.

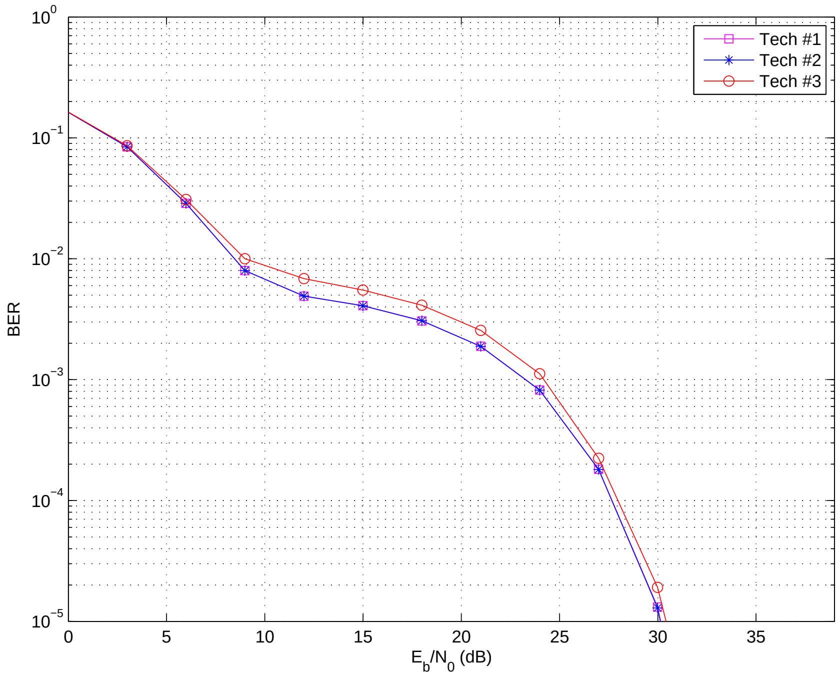


Figure 37: BER × $E_b/\mathcal{N}_0$ for 4-QAM corrupted by AIGN.

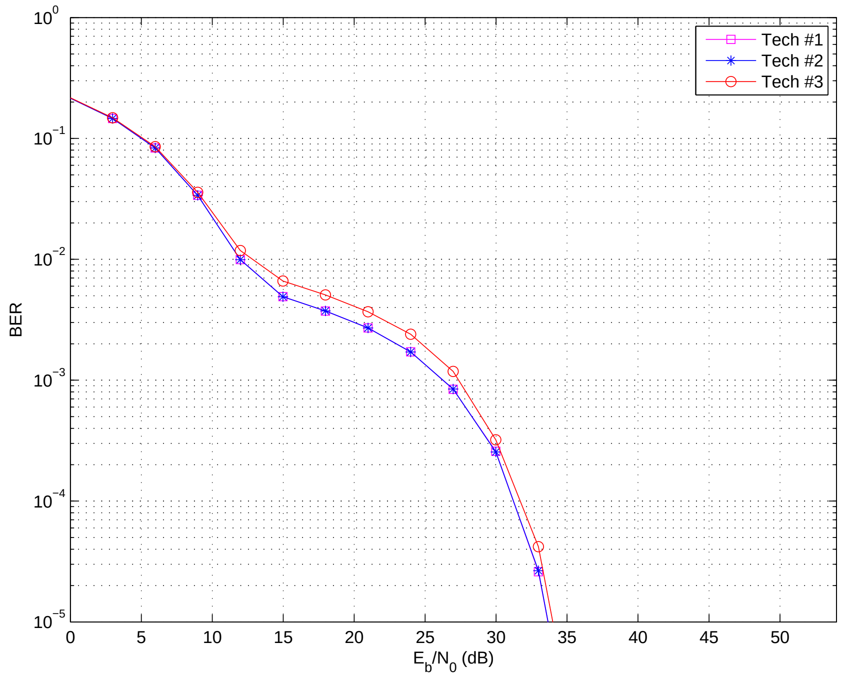


Figure 38: BER × $E_b/\mathcal{N}_0$ for 16-QAM corrupted by AIGN.

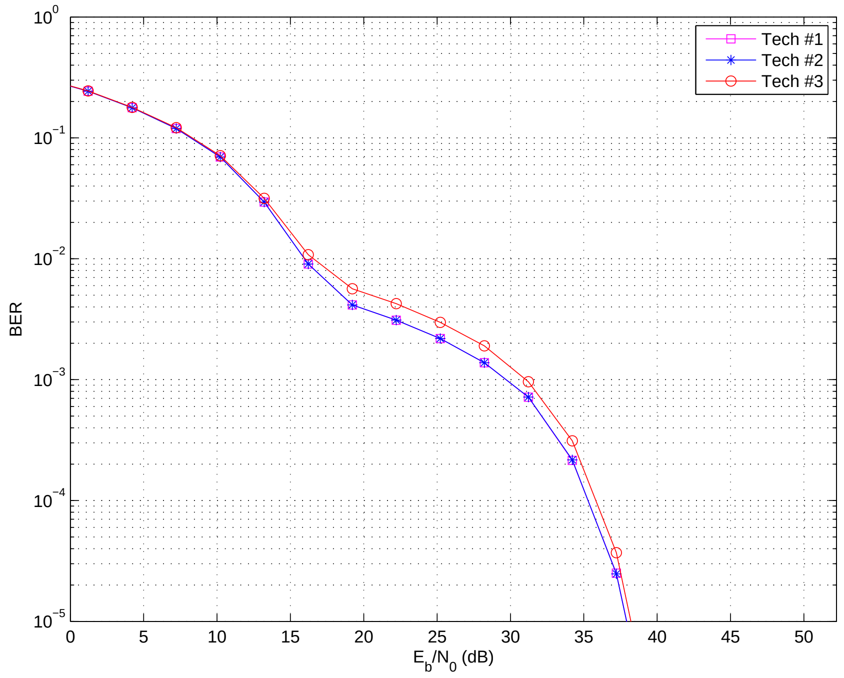


Figure 39: BER × $E_b/\mathcal{N}_0$ for 64-QAM corrupted by AIGN.

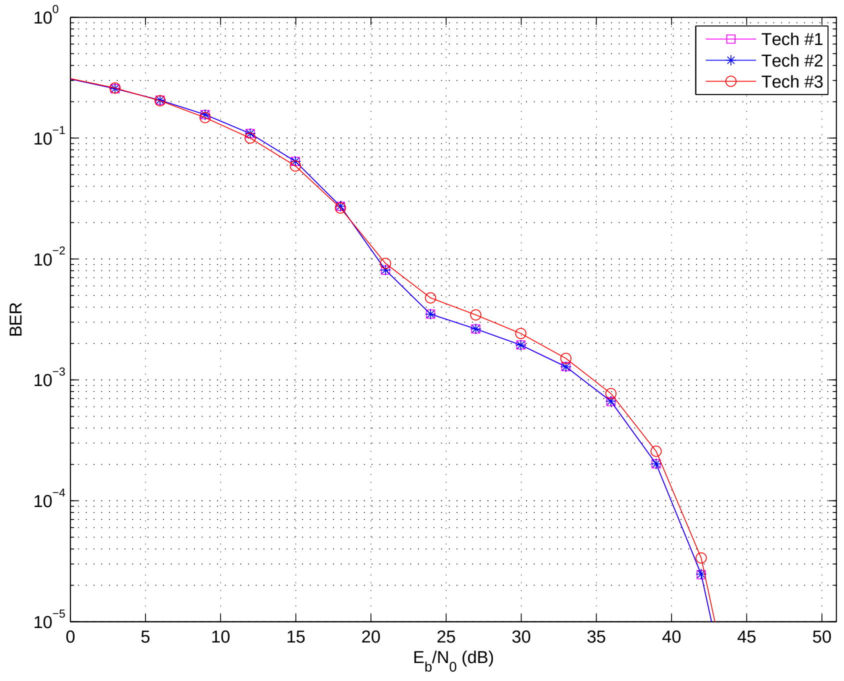


Figure 40: BER $\times$ $E_b/\mathcal{N}_0$ for 256-QAM corrupted by AIGN.

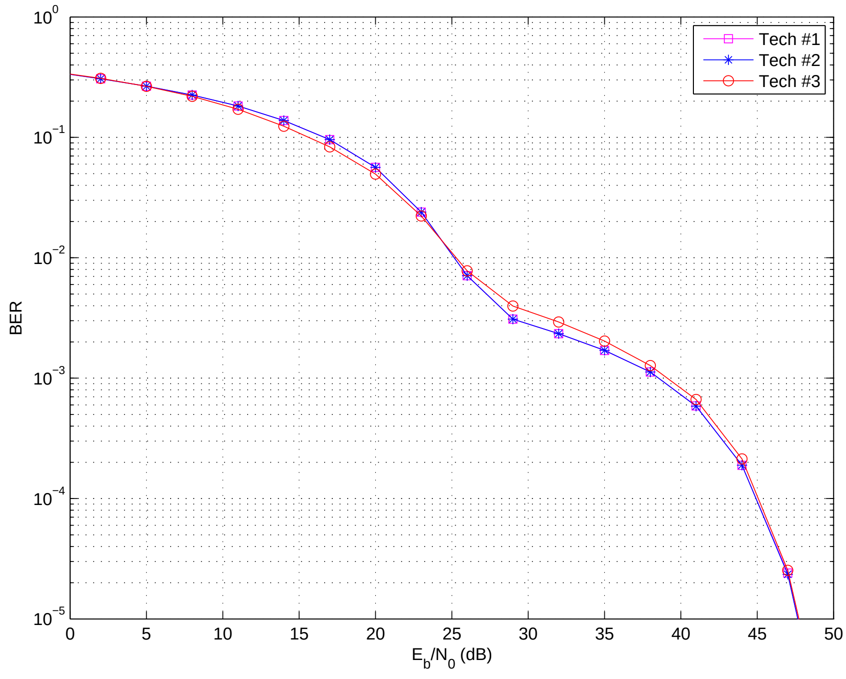


Figure 41: BER $\times$ $E_b/\mathcal{N}_0$ for 1024-QAM corrupted by AIGN.

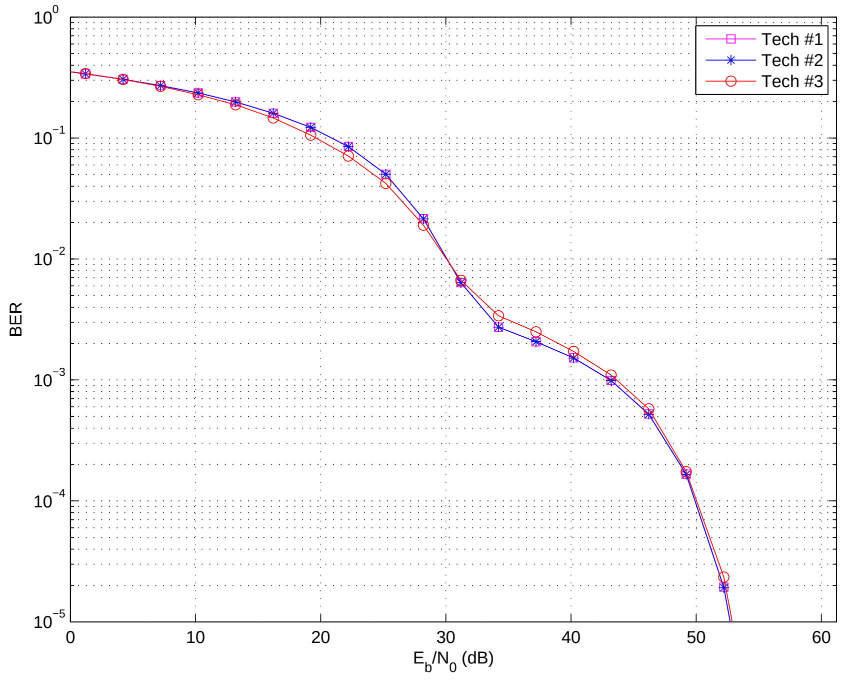


Figure 42: BER × $E_b/\mathcal{N}_0$ for 4096-QAM corrupted by AIGN.

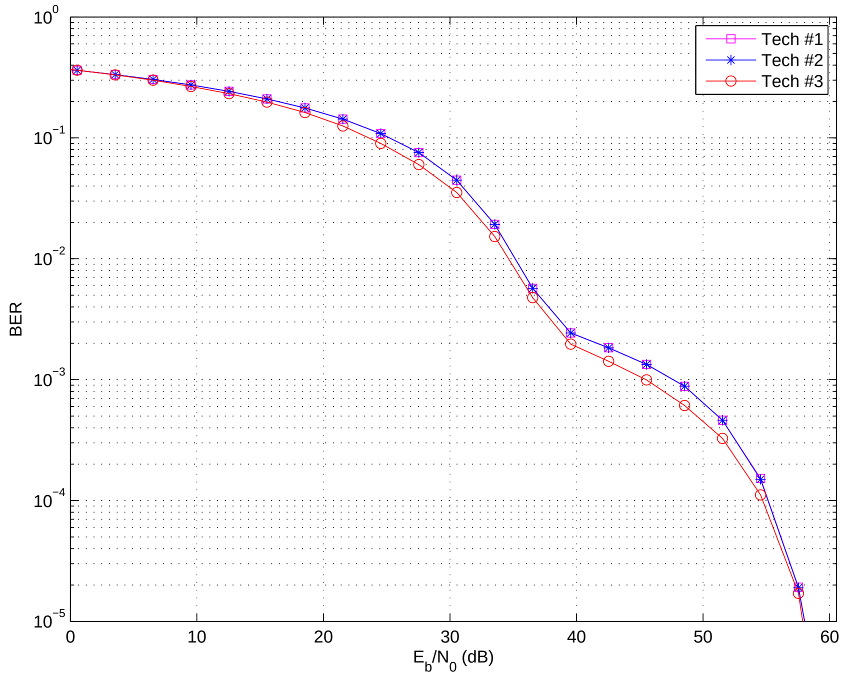


Figure 43: BER × $E_b/\mathcal{N}_0$ for 16384-QAM corrupted by AIGN.

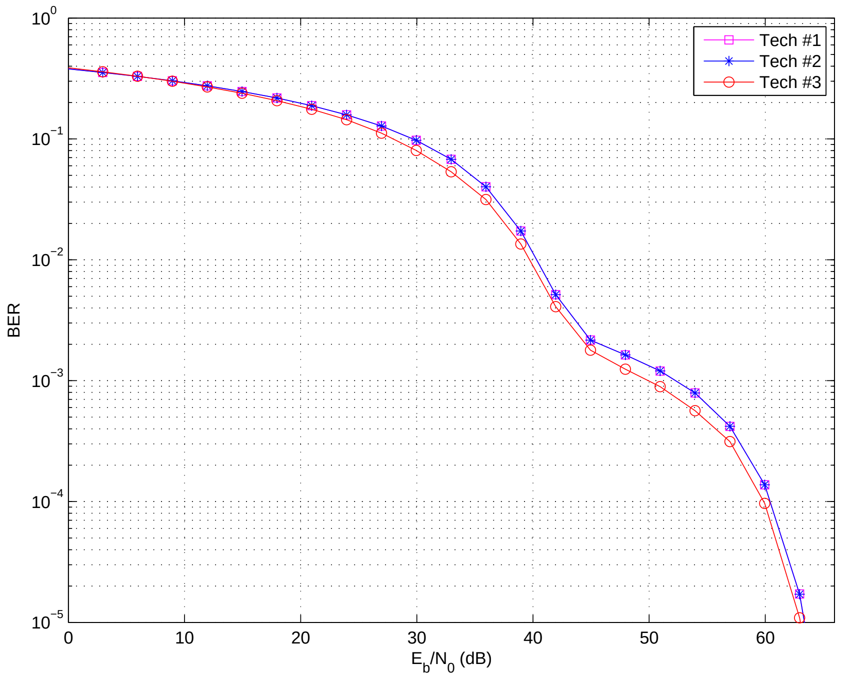


Figure 44: BER $\times$ $E_b/\mathcal{N}_0$ for 65536-QAM corrupted by AIGN.

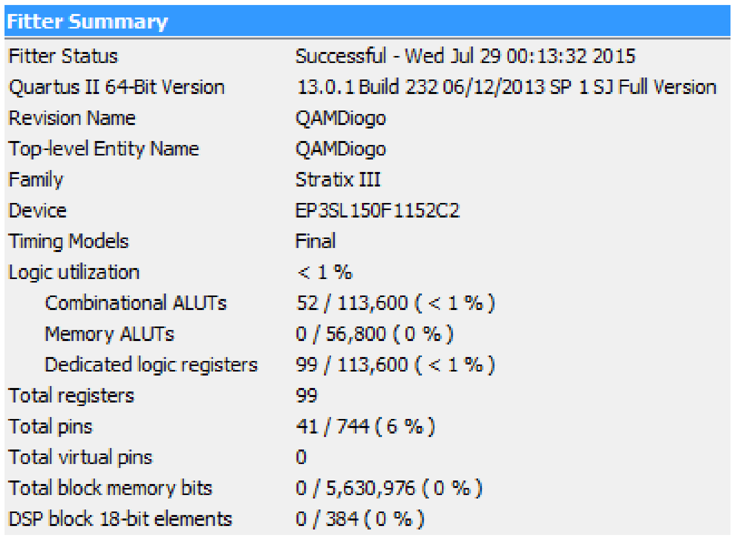
A small gain in favour of Tech #3 is noticed when $M \geq 256$ for AWGN, and $M \geq 16384$ for AIGN. Overall, performances differences among Tech #1, Tech #2 and Tech #3 totally disappear when $E_b/\mathcal{N}_0 << 1$ or $E_b/\mathcal{N}_0 >> 1$. The reason for the performance differences in intermediate values of $E_b/\mathcal{N}_0$ is the fact that Tech #1 and Tech #2 makes use of finite precision. Usually the word length of them is $K + 1$.

4.1.2 **Hardware Resource Utilization Analysis**

All techniques (#1, #2 and#3) were implemented in Verilog by using fixed point logic. Information data generated by the compilation performed by Quartus II 13.0 SP1 report called Fitter Summary [55] over Verilog codes, show that the implementation of the proposed M-QAM algorithm, for $K = 16$ bits, in the chosen Stratix III chipset, as shown in Figure 45, require few resources of logic element (less than 1% of the total available), any memory bit and any multiplier as well.

A comparison in terms of LE and memory, for $K = 16$ bits is shown in Table 5. As it can be observed, the Generic M-QAM technique - Tech #1, requires much less logic elements than the others and any memory resources. Also, it is clear that Tech #2 shows lower hardware resource utilization than Tech #3 due to the usage of only one LUT and only one SM for $K = 16$ bits. In the transmitter, Tech #3 has more LE's than Tech #2 because its implementation requires an approach to switch to the right M-QAM table. The same occurs in the receiver to choose the right SM.

The obtained results are shown graphically in Figure 46 for memory utilization comparison, in Figure 47 for LE utilization comparison and in Figure 48 for memory and



Fitter Summary	
Fitter Status	Successful - Wed Jul 29 00:13:32 2015
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	QAMDiogo
Top-level Entity Name	QAMDiogo
Family	Stratix III
Device	EP3SL150F1152C2
Timing Models	Final
Logic utilization	< 1 %
Combinational ALUTs	52 / 113,600 (< 1 %)
Memory ALUTs	0 / 56,800 (0 %)
Dedicated logic registers	99 / 113,600 (< 1 %)
Total registers	99
Total pins	41 / 744 (6 %)
Total virtual pins	0
Total block memory bits	0 / 5,630,976 (0 %)
DSP block 18-bit elements	0 / 384 (0 %)

Figure 45: Hardware resource utilization of Tech #1 for square QAM in a FPGA device.

Table 5: Resources comparison between implementations for square QAM constellation

	Modulator		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE's	77	8	61
Memory (Bits)	0	1179648	1572840
	Demodulator		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE's	74	2994	6454
Memory (Bits)	0	0	0
	Total		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE	151	3002	6515
Memory	0	1179648	1572840

LE utilization comparison. The graphical analyses are made with K varying from 2 to 16, considering

$$\beta_\alpha = \frac{C_\alpha}{C_{ref}}, \tag{4.1}$$

the resource utilization ratio, in which C_α is the hardware resource utilization for the Tech #1 or Tech #2 and C_{ref} refers to the hardware resource utilization associated to the Tech #3. Note that $\alpha \in \{\text{MEM}, \text{LE}\}$, in which MEM and LE refer to the number of used bits in the memory and the number of logic elements demanded by each technique, respectively. In Figure 48, C_α and C_{ref} were normalized by the number of available hardware resource components in the chosen FPGA device, chipset Stratix EP3SL150F1152C2, that is

$$\overline{C}_\alpha = \frac{1}{2}\frac{C_{MEM}}{C_{total_{MEM}}} + \frac{1}{2}\frac{C_{LE}}{C_{total_{LE}}}, \tag{4.2}$$

and

$$\overline{C}_{ref} = \frac{1}{2}\frac{C_{ref_{MEM}}}{C_{total_{MEM}}} + \frac{1}{2}\frac{C_{ref_{LE}}}{C_{total_{LE}}}, \tag{4.3}$$

in which $C_{total_{MEM}}$ is the total number of available memory bits in the FPGA device and $C_{total_{LE}}$ is the total of available logic elements in the FPGA device, so

$$\overline{\beta}_{\alpha} = \frac{\overline{C}_{\alpha}}{\overline{C}_{ref}}, \tag{4.4}$$

denotes a kind of parameter that we may use to comparatively quantify the use of hardware resource between two techniques, where the same weight is given to both kinds of hardware resource. This weight could be defined by the impact of each resource in the final chipset cost, and/or, in the total thermal power dissipation, depending of the final application.

In Figure 46, as it might be observed, Tech #1 does not requires any memory resources, while Tech #2 presents lower memory resource utilization than Tech #3. Due to the usage of the largest LUT table, that is $K = 16$ bits which stores 65536 possible square constellations points, Tech #2 memory requirement converges to 0.75 in relationship of Tech #3. In Figure 47, for $K \geq 4$ bits, Tech #1 uses less logic elements than the other techniques, while Tech #2 presents lower LE resource utilization than Tech #3. The combined resource utilization, in Figure 48 shown that for $K \geq 4$ bits, Tech #1 uses less hardware resource than the other techniques, while Tech #2 presents lower hardware resource utilization than Tech #3. It is important to emphasize that for $K \geq 10$, $\overline{\beta}_{\alpha}$ increases for Tech #2, but will never be greater than β_{MEM} convergence, that is 0.75. Moreover, Tech #2 and Tech #3 demand the same resource utilization only for $K = 2$. finally, but not the least, for $K \geq 4$, both Tech #1 and Tech #2 presented lower hardware resource utilization than Tech #3.

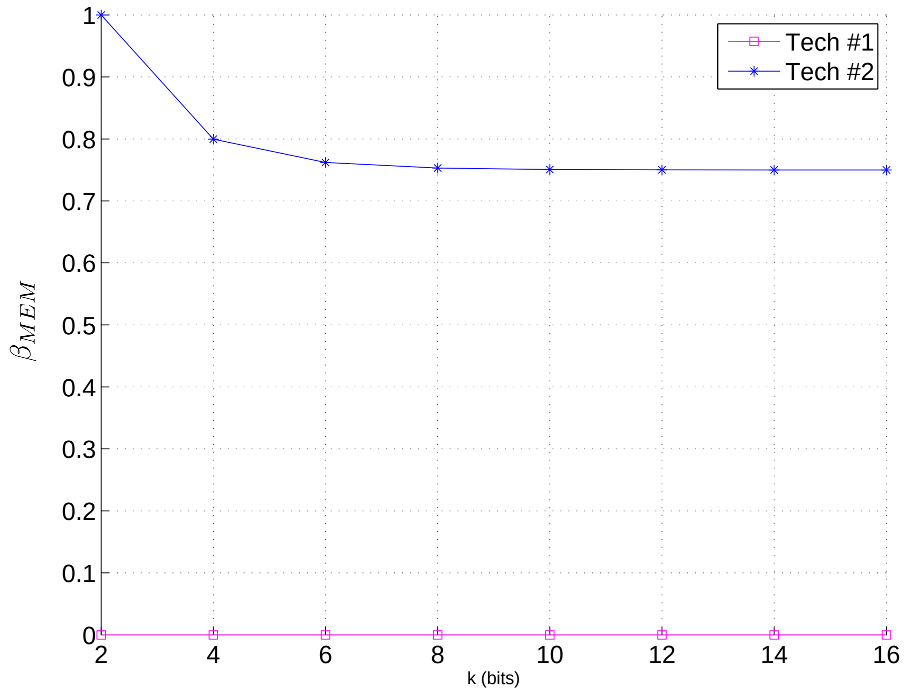


Figure 46: The memory utilization comparison for square QAM in FPGA device when the constellation cardinality changes.

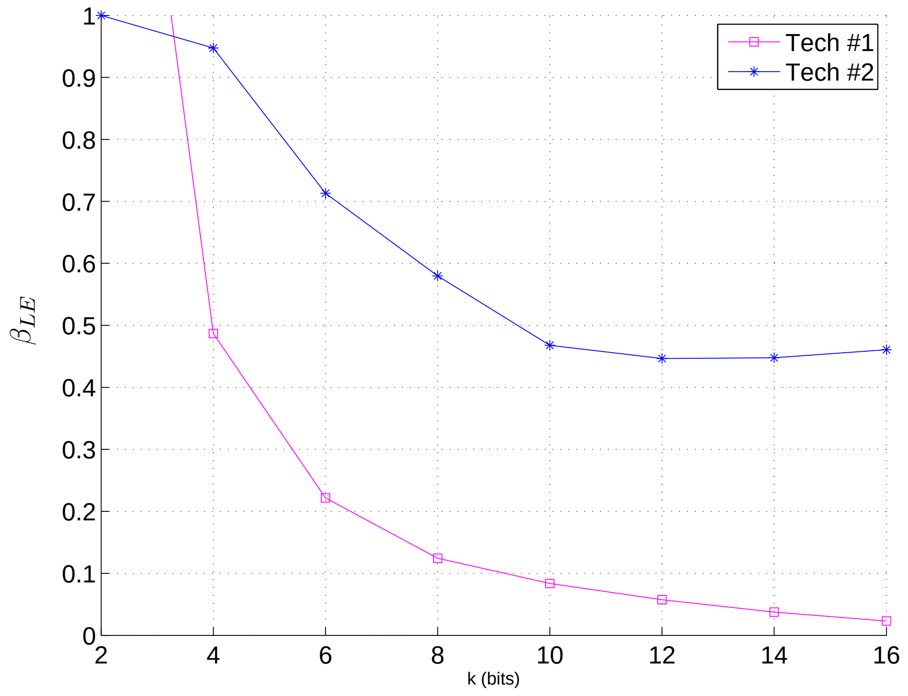


Figure 47: The LE utilization comparison for square QAM in FPGA device when the constellation cardinality changes.

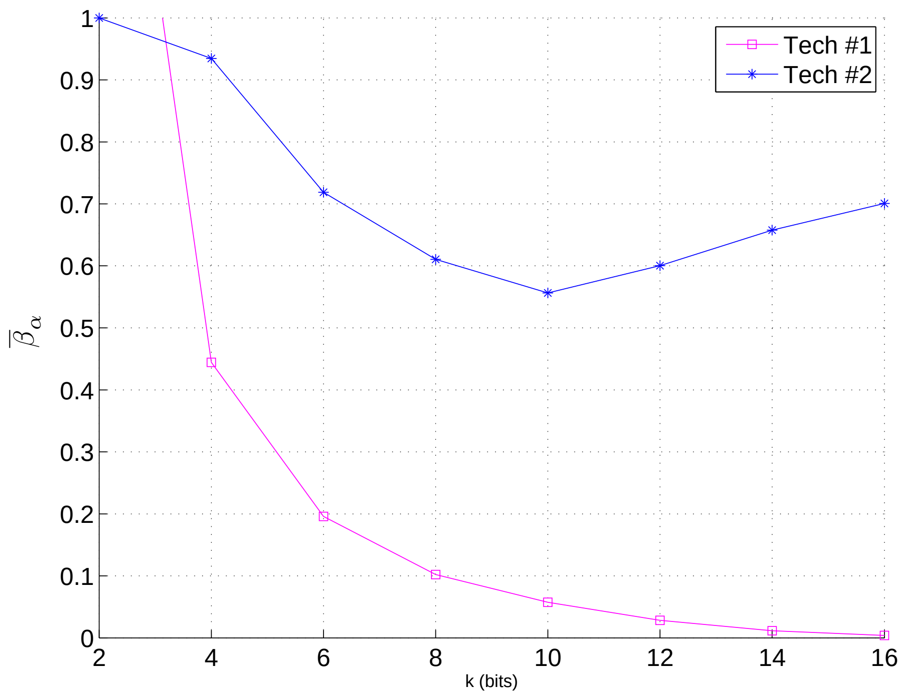


Figure 48: The LE and memory utilization comparison for square QAM in FPGA device when the constellation cardinality changes.

4.1.2.1 *Maximum Frequency Analysis*

The maximum frequency is an important indicator for the evaluation of the results. It informs the highest maximum frequency that the clock/oscillator can operate in order that the digital circuit works properly. To achieve the maximum frequency value its mandatory to associate the I/O in the FPGA chipset, so the **clock** and **reset** inputs were defined, respectively, as **PIN_T33** and **PIN_B19** in Stratix III EP3SL150F1152C2. Then, to obtain this result, this simulation considered the digital circuit in the worst situation, working at 85 Celsius degrees, the TimeQuest Timing Analyzer, a tool from Quartus II Altera, estimates maximum frequencies that Tech #1, #2 and #3 can operate. The attained results are shown in Table 6.

Table 6: Timing Analyzer for square 2^{16}-QAM implementation

Technique	**Fmax**
Tech #1	743,49 MHz
Tech #2	197,39 MHz
Tech #3	140,79 MHz

As it can be observed, Tech #1 uses few logic elements and, as a consequence, achieves the maximum frequency, because it facilitate the routing of digital circuit into the FPGA device. For instance following [57], a broadband PLC requires 50 MHz bandwidth. In this case, the communication bandwidth is 50 MHz, so it is necessary at least 120 MHz of sampling frequency to satisfy such communication system. All of the three techniques would fulfil this requirement, but the maximum frequency could decrease when new additional items (channel coding, frequency estimator, etc.) are implemented into FPGA devices.

4.1.2.2 *Power Consumption Analysis*

Another important topic to be analized is power consumption by the FPGA device. To provide analysis about the power consumption, the PowerPlay Power Analysis, a tool from Altera applies. Table 7 presents the estimated thermal power, that is the power dissipated in the device, by the three techniques.

Table 7: Power Analysis for square 2^{16}-QAM implementation

Technique	**Tech #1**	**Tech #2**	**Tech #3**
Core Dynamic Thermal Power Dissipation	8,68 mW	53,77 mW	166,78 mW
Core Static Thermal Power Dissipation	570,04 mW	574,85 mW	590,33 mW
I/O Thermal Power Dissipation	52,33 mW	54,84 mW	50,10 mW
Total Thermal Power Dissipation	**631,05 mW**	**683,46 mW**	**807,20 mW**

The Core Dynamic Thermal Power Dissipation represents the thermal power dissipated by the functional blocks when the clock process is being executed, with the exception of the I/O. Therefore, the Core Static Thermal Power Dissipation is the thermal power dissipated on the FPGA device, independent of the clock which includes the leakage power from all FPGA functional blocks, with the exception of the I/O. The static power is the only thermal power component which varies with junction temperature, selected device, and power characteristics [58]. The Total Thermal Power Dissipation is a sum of the thermal power of all the resources used in the device, including the maximum power from standby, dynamic power and I/O. This thermal power dissipation analysis does not include the external power dissipation, such as these ones associated with voltage-referenced termination resistors.

As it can be observed, Tech #1 is the one which requires less thermal power dissipation due to the low usage of logic elements and the avoidance of memory usage. Tech #3 is the one with highest thermal power dissipation because it requires high quantity of memory resources and logic elements, and Tech #2 attains intermediate values.

4.1.2.3 *Latency and Delay Analysis*

Another analysis is associated with the number of clock cycles required to perform each technique. The results are obtained through a simulation tool called ModelSim Altera. The data collected are presented in Table 8. To calculate the required number of clock cycles to yield the first symbol, just add the latency clock cycles to the delay clock cycles. For the Tech #1, operating in continuous symbol generation, it would take seven clock cycles to yield the first symbol and one clock cycle for the following ones due to parallelism of operations (parallel circuits operation) in a FPGA device, that is represented by delay presented in Table 9. In the case of Tech #2 and Tech #3, it would takes 17 clock cycles for the first symbol and more 8 clock cycles for each new symbol due to the dynamic of the SM.

Table 8: Speed comparison for a single symbol for square QAM.

Technique	**Latency**	**Delay**
Tech #1	6	1
Tech #2	9	8
Tech #3	9	8

Finally, Table 9 shows the required number of clock cycles to perform each technique when the number of symbols belongs to the set $N_s \in \{1, 2, 4, 16, 64, 256, 1024\}$. As it can be observed, independently of the number of consecutive symbols which were previously modulated on the same constellation cardinality, Tech #1 improves the performance in terms of speed. If 1024 symbols with the same constellation are consecutively generated, then the required number of clock cycles is reduced to almost 8 times in comparison with other techniques.

Table 9: Number of clocks for generating consecutive symbol for square QAM.

# of Symbols	**1**	**2**	**4**	**16**	**64**	**256**	**1024**
Tech #1	7	8	10	22	70	262	1030
Tech #2	17	25	41	137	521	2057	8201
Tech #3	17	25	41	137	521	2057	8201

4.2 Cross QAM Constellation

In this section, the numerical results of the cross M-QAM are presented, considering $K = 3, 5, \ldots, 16$, that is, $M = 8, 32, \ldots, 32768$. Different from Section 4.1, Tech #1 and Tech #2 requires the usage of the Position Switcher, the Inverse Position Switcher and the Combined Saturator because it is necessary to perform the cross constellation by employing Algorithms 2, 3 and 4, as shown in Chapter 3.

4.2.1 **BER Analysis**

BER performance comparisons among Tech #1, Tech #2 and Tech #3 are shown in Figures 49-55 for AWGN and in Figures 56-62 for AIGN. The theoretical results were not employed for comparison in this simulation due to the usage of pseudo-Gray coded cross M-QAM into the three techniques, which make the results to be different.

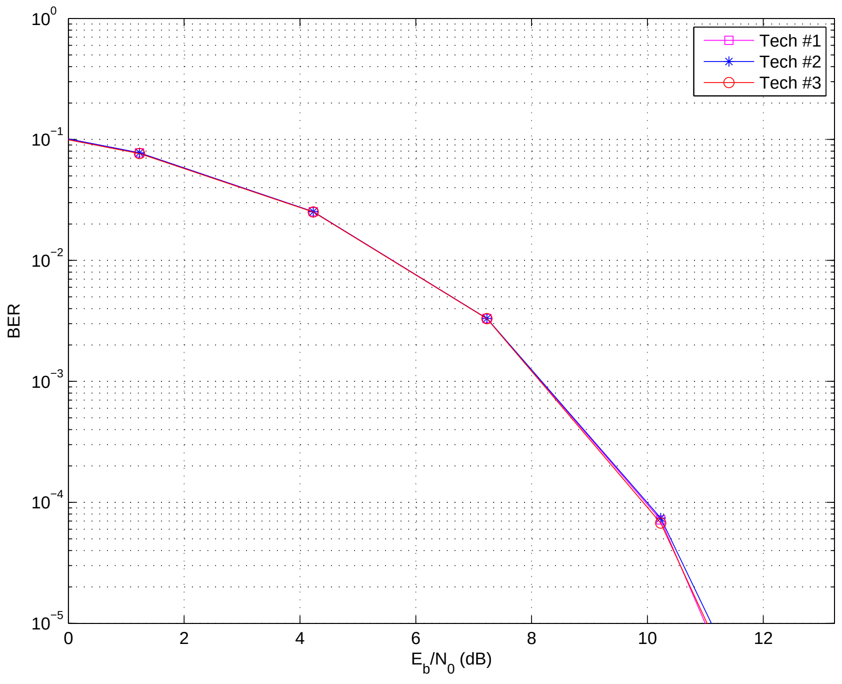


Figure 49: BER × $E_b/\mathcal{N}_0$ for 8-QAM corrupted by AWGN.

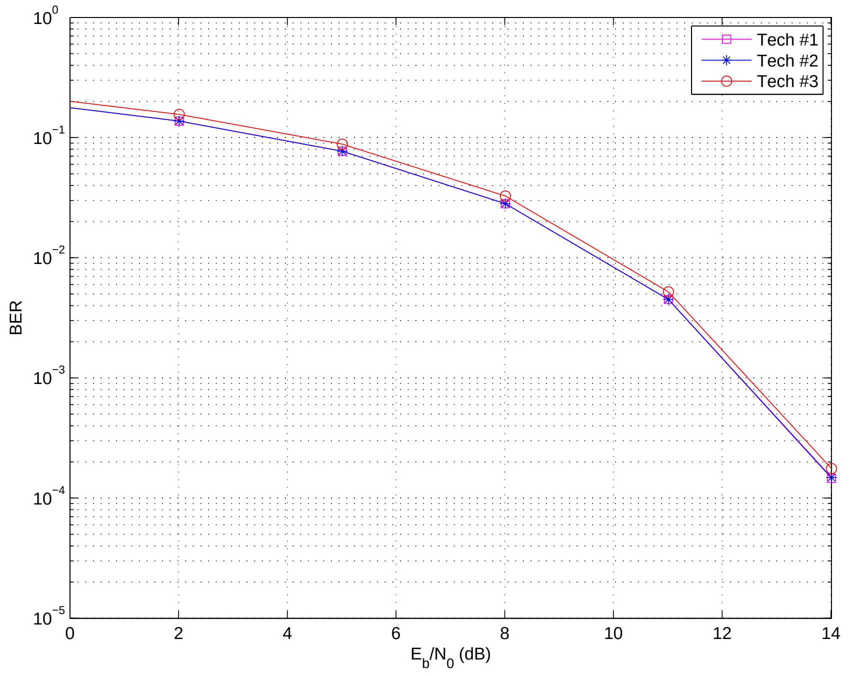


Figure 50: BER × $E_b/\mathcal{N}_0$ for 32-QAM corrupted by AWGN.

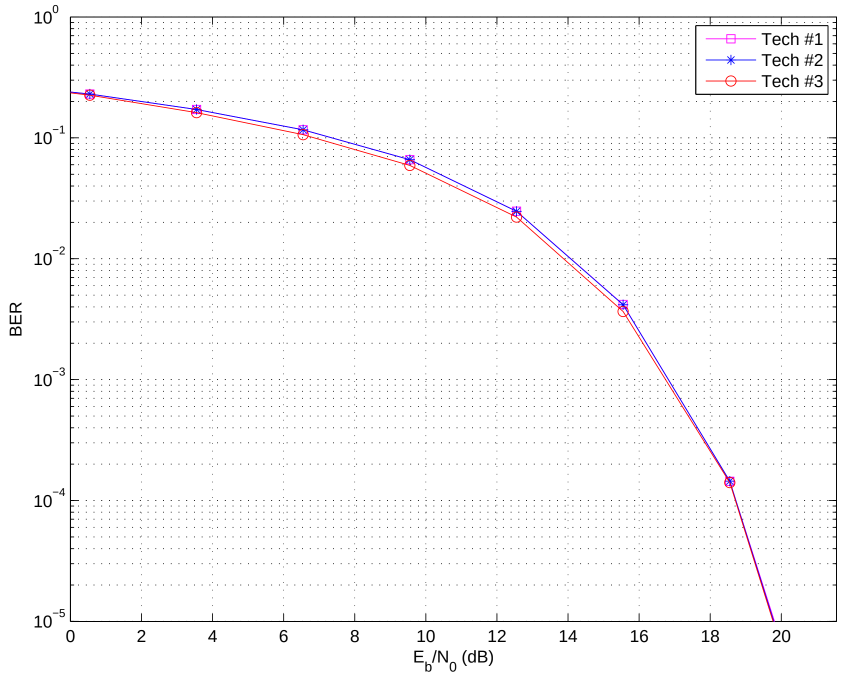


Figure 51: BER × $E_b/\mathcal{N}_0$ for 128-QAM corrupted by AWGN.

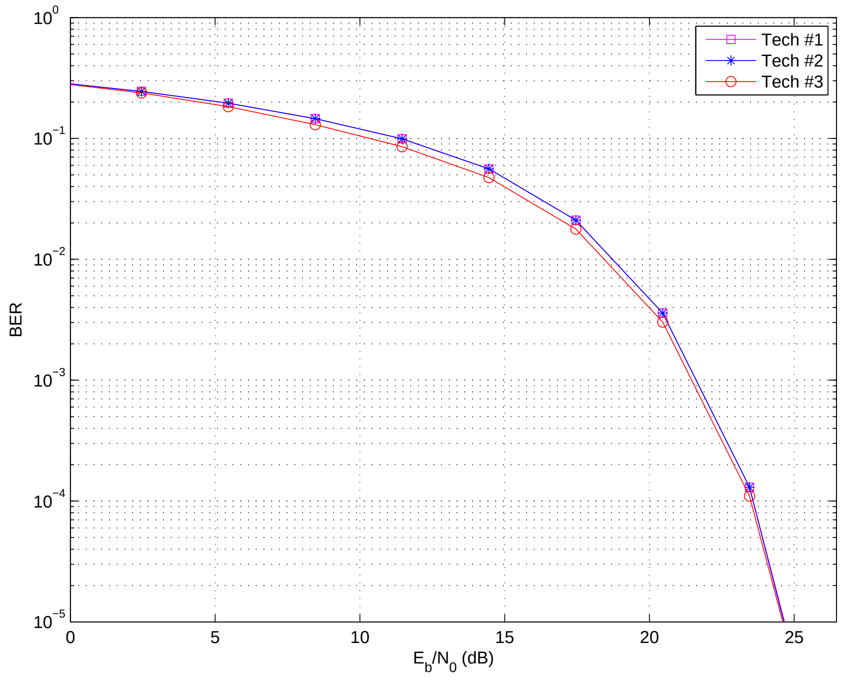


Figure 52: BER × $E_b/\mathcal{N}_0$ for 512-QAM corrupted by AWGN.

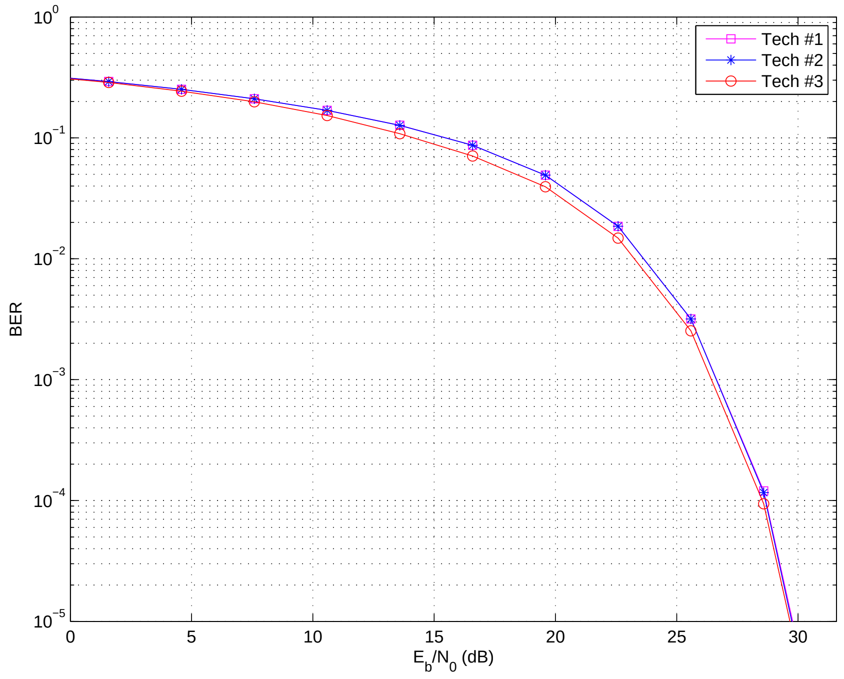


Figure 53: BER × $E_b/\mathcal{N}_0$ for 2048-QAM corrupted by AWGN.

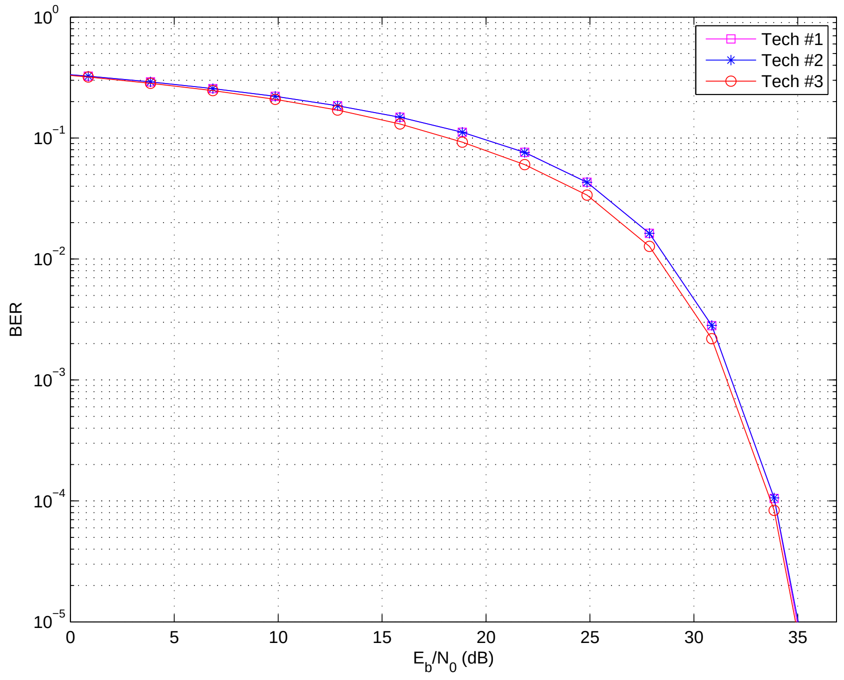


Figure 54: BER × $E_b/\mathcal{N}_0$ for 8192-QAM corrupted by AWGN.

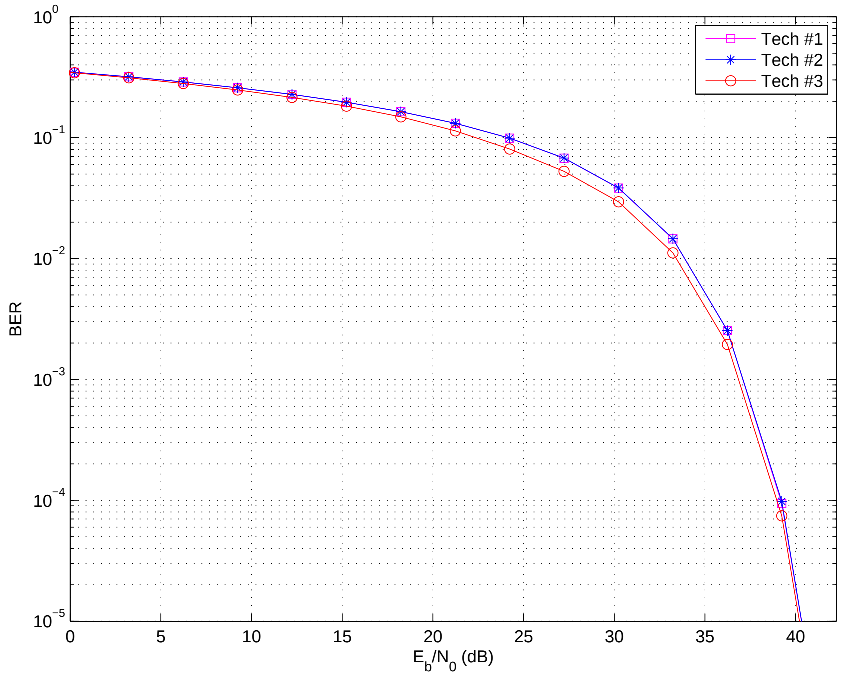


Figure 55: BER × $E_b/\mathcal{N}_0$ for 32768-QAM corrupted by AWGN.

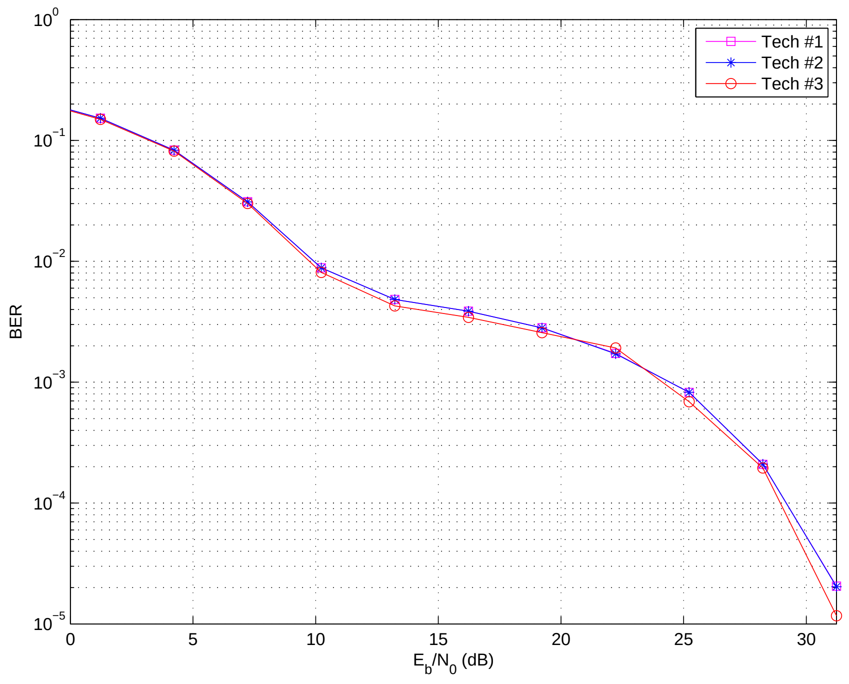


Figure 56: BER $\times$ $E_b/\mathcal{N}_0$ for 8-QAM corrupted by AIGN.

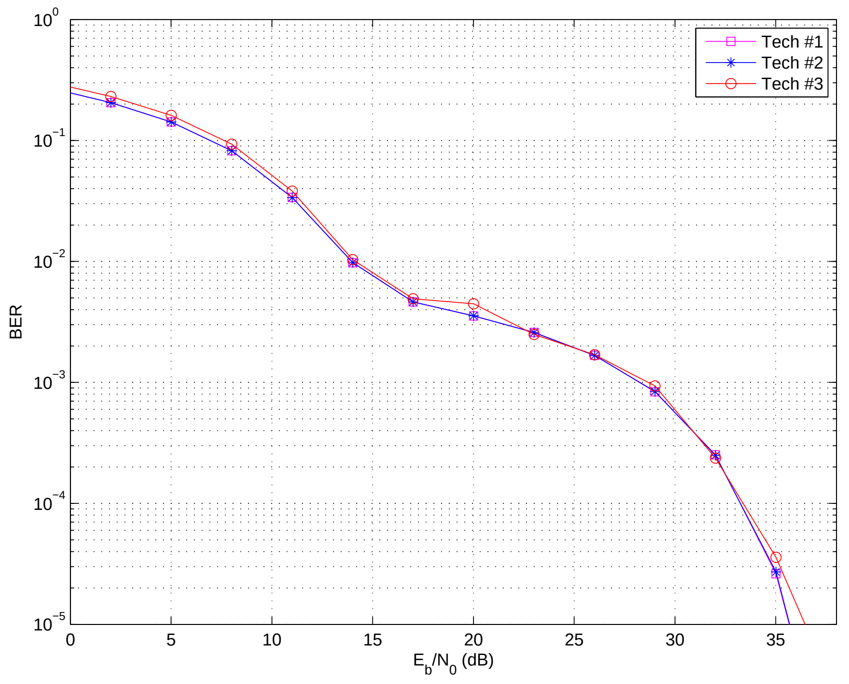


Figure 57: BER $\times$ $E_b/\mathcal{N}_0$ for 32-QAM corrupted by AIGN.

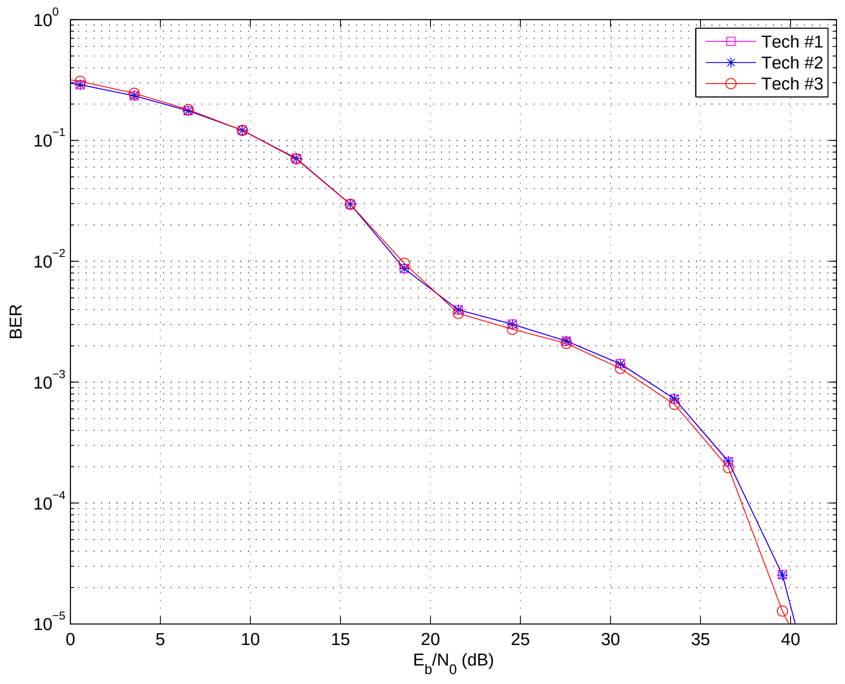


Figure 58: BER × $E_b/\mathcal{N}_0$ for 128-QAM corrupted by AIGN.

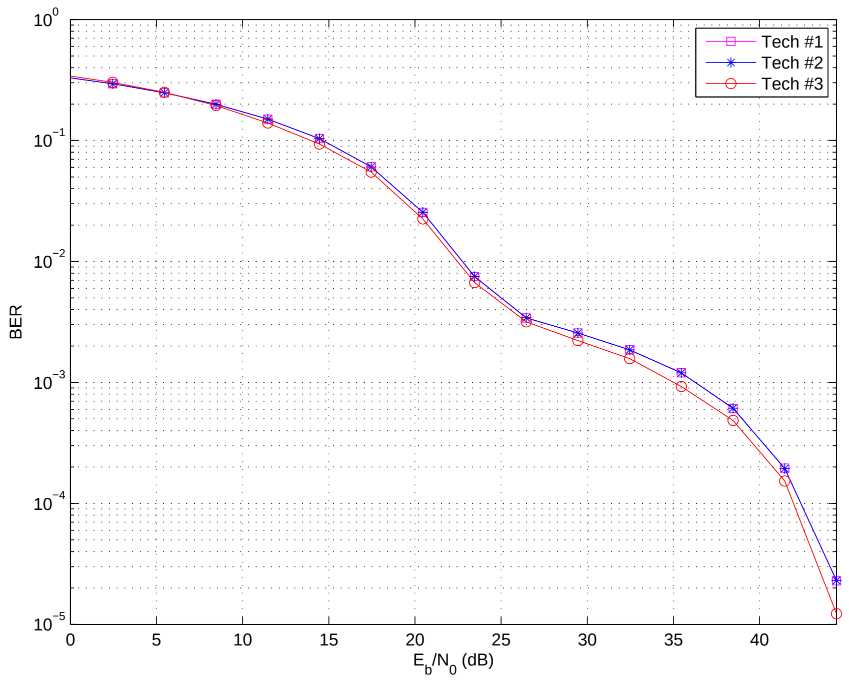


Figure 59: BER × $E_b/\mathcal{N}_0$ for 512-QAM corrupted by AIGN.

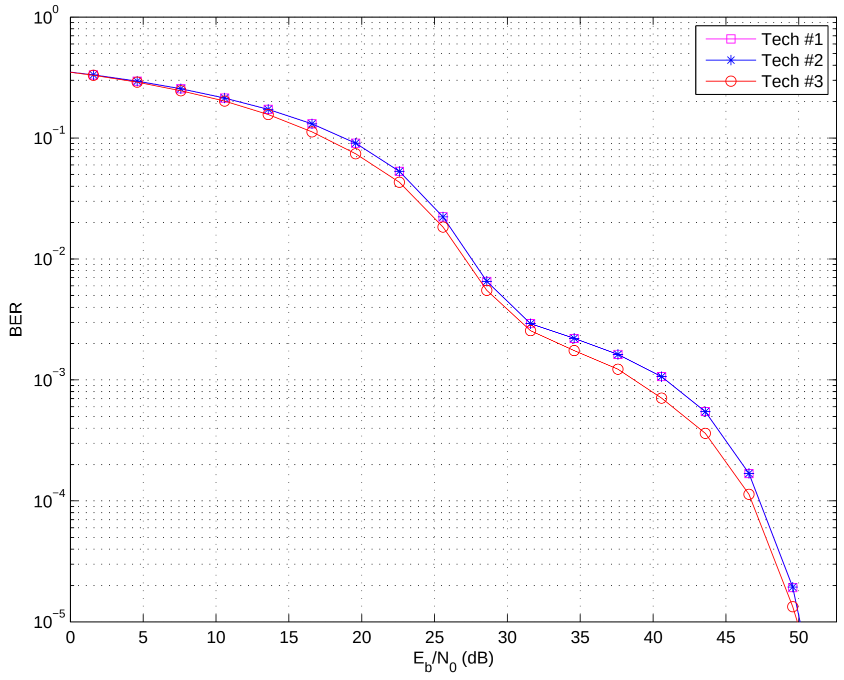


Figure 60: BER $\times$ $E_b/\mathcal{N}_0$ for 2048-QAM corrupted by AIGN.

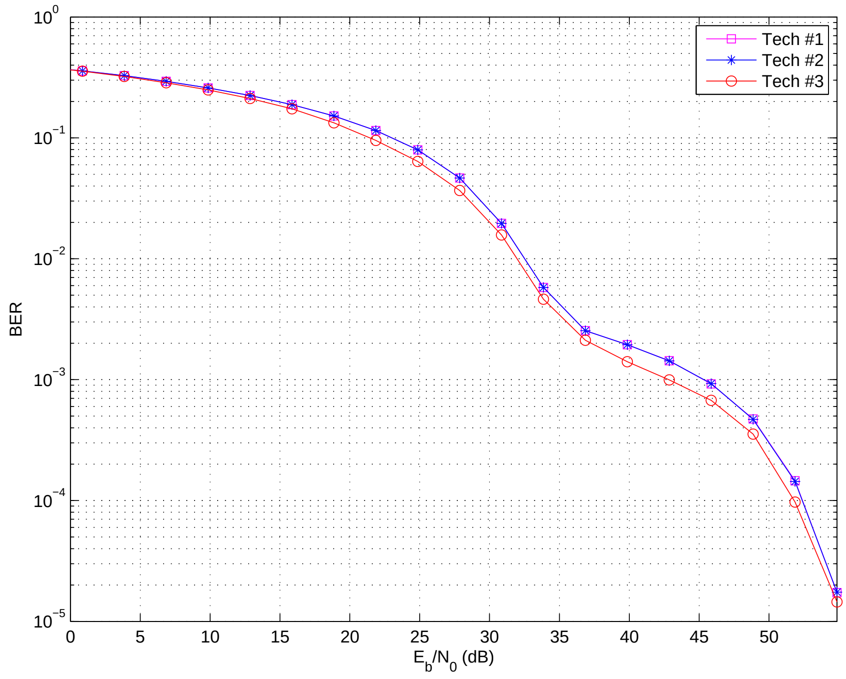


Figure 61: BER $\times$ $E_b/\mathcal{N}_0$ for 8192-QAM corrupted by AIGN.

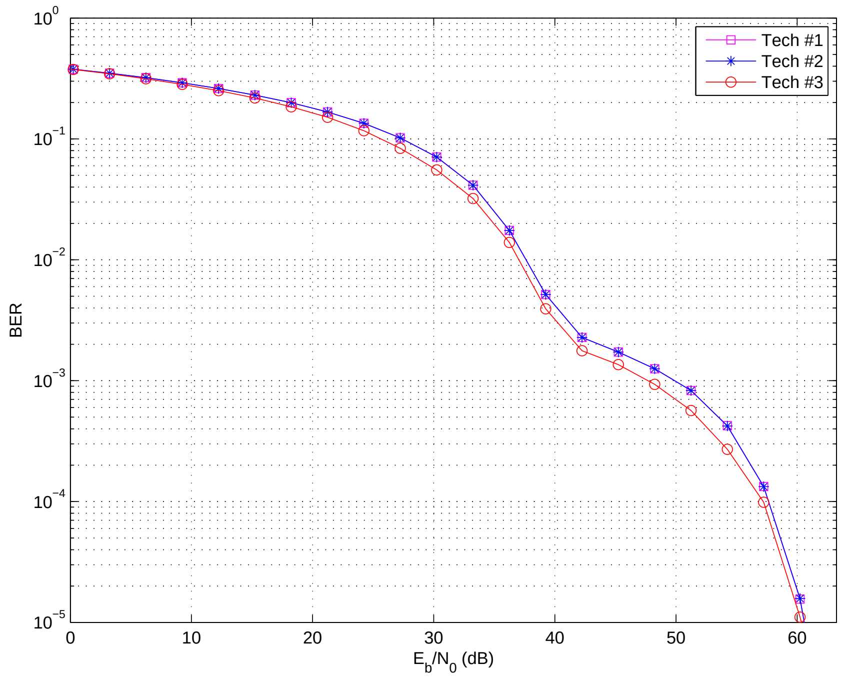


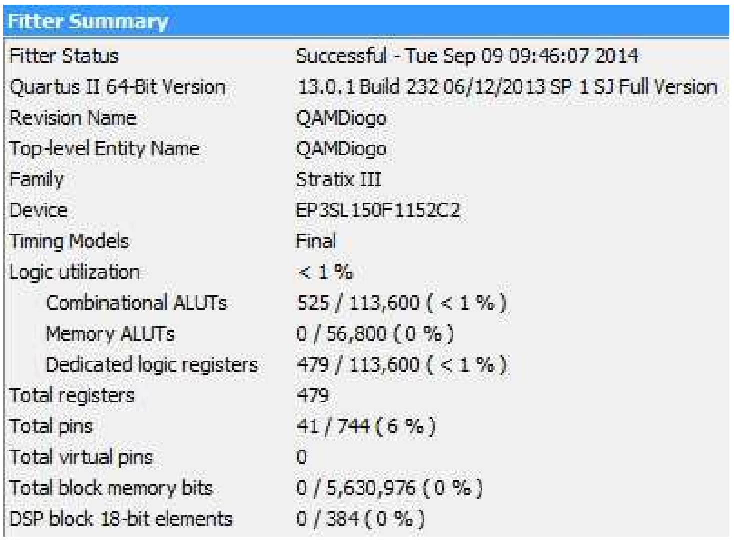
Figure 62: BER $\times$ $E_b/\mathcal{N}_0$ for 32768-QAM corrupted by AIGN.

The results shows a small performance loss introduced by the proposed techniques that, on average, are less than 1 dB when comparing to Tech #3, considering $M \geq 128$ for AWGN, and $M \geq 512$ for AIGN, as previously described in Subsection 4.1.1, due to the usage of finite precision, with word length of $K + 1$, performance differences exists in intermediate values of $E_b/\mathcal{N}_0$, but for $E_b/\mathcal{N}_0 << 1$ or $E_b/\mathcal{N}_0 >> 1$ they totally disappear.

4.2.2 **Hardware Resource Utilization Analysis**

As previously presented in Subsection 4.1.2, all techniques (#1, #2 and#3) were implemented in Verilog by using fixed point logic. Information data generated by the compilation performed by Quartus II 13.0 SP1 report called Fitter Summary [55] over Verilog codes, show that the implementation of the proposed M-QAM algorithm, for $K = 15$ bits (for Tech #1, only odd numbers are allowed for K, as explained in Chapter 3, so $K = 16$ bits), in the chosen Stratix III chipset, as shown in Figure 63, require few resources of logic element (almost 1% of the total available), any memory bit and any multiplier as well.

A comparison in terms of LE and memory, for $K = 15$ bits is shown in Table 10. As it can be observed, Tech #1 requires less logic elements than the others and any memory resources and it is clear that Tech #2 shows lower hardware resource utilization than Tech #3 due to the usage of only one LUT and only one SM for $K = 15$ bits. Also, it can be noticed that the hardware resource usage of Tech #1 and Tech #2 gets closer to Tech #3 and is higher than the ones presented in the square M-QAM, see Subsection 4.1.2, due to the employment of Algorithms 2, 3 and 4. In the transmitter, Tech #3 has



Fitter Summary	
Fitter Status	Successful - Tue Sep 09 09:46:07 2014
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	QAMDiogo
Top-level Entity Name	QAMDiogo
Family	Stratix III
Device	EP3SL150F1152C2
Timing Models	Final
Logic utilization	< 1 %
Combinational ALUTs	525 / 113,600 (< 1 %)
Memory ALUTs	0 / 56,800 (0 %)
Dedicated logic registers	479 / 113,600 (< 1 %)
Total registers	479
Total pins	41 / 744 (6 %)
Total virtual pins	0
Total block memory bits	0 / 5,630,976 (0 %)
DSP block 18-bit elements	0 / 384 (0 %)

Figure 63: Hardware resource utilization of Tech #1 for cross QAM in a FPGA device.

LE's because its implementation requires an approach to switch to the right M-QAM table and to choose the right SM as presented in Subsection 4.1.2.

Table 10: Resources comparison between implementations for cross QAM

	Modulator		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE's	356	280	56
Memory (Bits)	0	589824	786864
	Demodulator		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE's	648	2363	4509
Memory (Bits)	0	0	0
	Total		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE	1004	2643	4565
Memory	0	589824	786864

The graphical analyses of the obtained results are made with K varying from 3 to 15, considering Equation (4.1), the hardware resource utilization ratio, in Figure 64 for memory utilization comparison and in Figure 65 for LE utilization comparison, and Equation (4.4), the normalized hardware resource utilization ratio, in Figure 66 for memory and LE utilization comparison, where, as well as Subsection 4.1.2, the same weight is given to both kind of resource.

In Figure 64, as it might be observed, Tech #1 does not requires any memory resources, while Tech #2 presents lower memory resource utilization than Tech #3, with β_{MEM} converging to 0.75 while K increases. As it can be seen, for $K = 7$, β_{MEM} stop decreasing and starts increasing, this is due to the FPGA optimization made by Quartus II Tool, to synthesize only logic elements instead of memory resources when small RAM components are employed. As presented in Subsection 4.1.2, Tech #2 memory requirement also converges to 0.75 in relationship of Tech #3. In Figure 65, for $K \geq 9$ bits, Tech #1 uses less logic elements than the other techniques, while Tech #2 presents lower LE resource utilization than Tech #3 for $K \geq 11$. The combined resource utilization, in Figure 66 shown that for $K \geq 11$ bits, Tech #1 uses less hardware resource than the other techniques, almost half of the hardware resources of Tech #3, while Tech #2 presents lower hardware resource utilization than Tech #3, for $K \geq 11$. Moreover, Tech #2 and Tech #3 demand the same memory resource utilization only for $K = 3$.

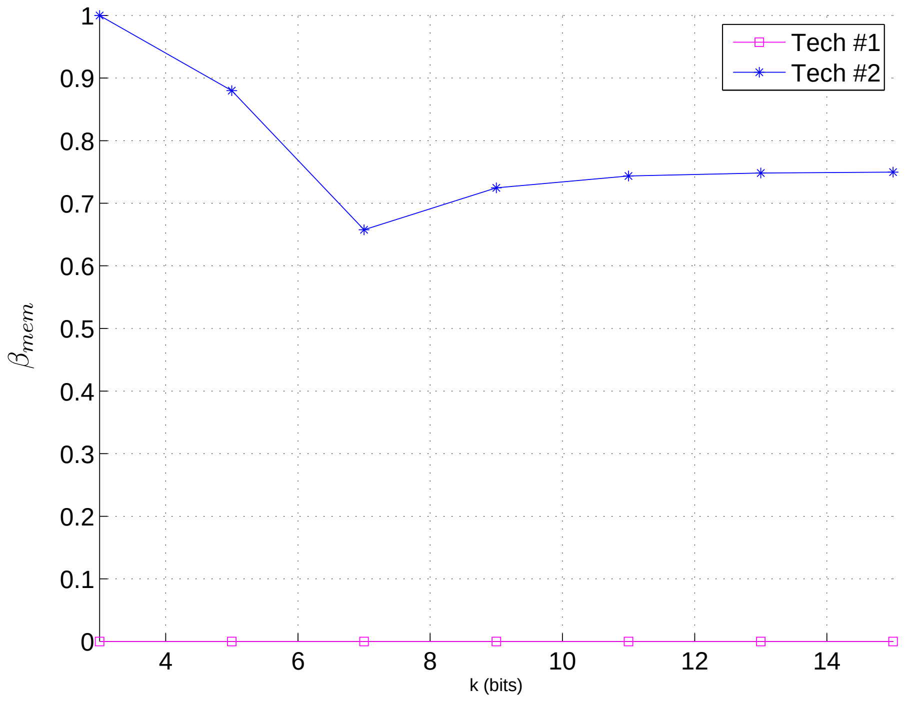


Figure 64: The memory utilization comparison for cross QAM in FPGA device when the constellation cardinality changes.

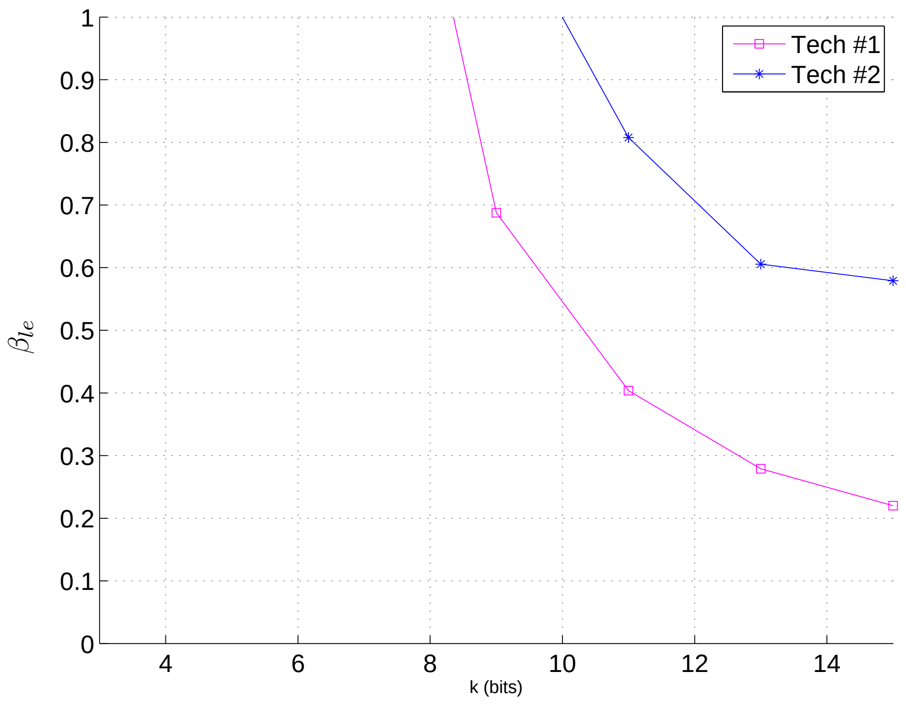


Figure 65: The LE utilization comparison for cross QAM in FPGA device when the constellation cardinality changes.

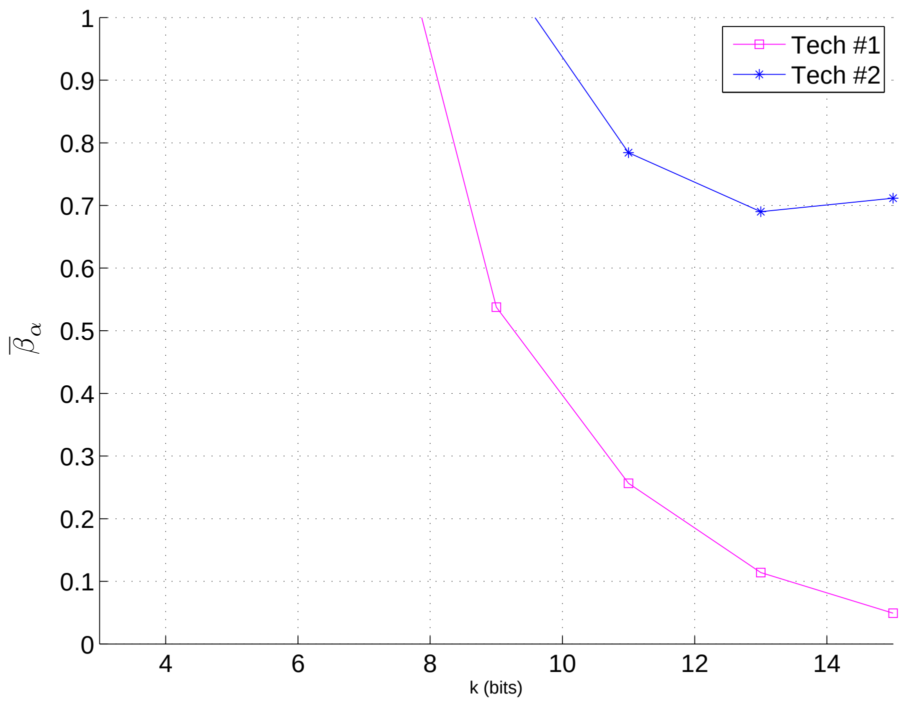


Figure 66: The LE and memory utilization comparison for cross QAM in FPGA device when the constellation cardinality changes.

4.2.2.1 *Maximum Frequency Analysis*

As previously introduced in Subsection 4.1.2.1, the maximum frequency is an important indicator for the evaluation of the results and correct configurations should be applied to calculate it. The results of the digital circuit in the worst situation, working at 85 Celsius degrees, are presented in Table 11.

Table 11: Timing Analyzer for cross 2^{15}-QAM implementation

Technique	**Fmax**
Tech #1	234,08 MHz
Tech #2	229,46 MHz
Tech #3	149,63 MHz

As it can be observed, Tech #1 and Tech #2 uses few logic elements and memory resources and, as a consequence, achieves the higher maximum frequency. If the same necessary sampling frequency of 120 MHz [57] from Subsection 4.1.2.1 is considered, all of the three techniques would fulfil this requirement, but the maximum frequency could decrease when new additional items are implemented into FPGA devices.

4.2.2.2 *Power Consumption Analysis*

To provide analysis about the power consumption, the PowerPlay Power Analysis, a tool from Altera applies. Table 7 presents the estimated thermal power, that is the power dissipated in the device, by the three techniques.

Table 12: Power Analysis for cross 2^{15}-QAM implementation

Technique	**Tech #1**	**Tech #2**	**Tech #3**
Core Dynamic Thermal Power Dissipation	29,02 mW	51,76 mW	122,32 mW
Core Static Thermal Power Dissipation	570,26 mW	572,80 mW	580,28 mW
I/O Thermal Power Dissipation	45,64 mW	54,90 mW	50,07 mW
Total Thermal Power Dissipation	**644,92 mW**	**679,46 mW**	**752,67 mW**

The explanation of each power dissipation was given by Subsection 4.1.2.2 and presented in [58]. The Total Thermal Power Dissipation is used for analysis. As it can be observed, Tech #1 is the one which requires less thermal power dissipation due to the low usage of logic elements and the avoidance of memory usage. Tech #2 power dissipation gets close to Tech #1 because uses less hardware resources than Tech #3, and Tech #2 has balanced LE and memory resources. Then, Tech #3 is the one with highest thermal power dissipation because it requires high quantity of memory resources and logic elements.

4.2.2.3 *Latency and Delay Analysis*

The analysis of the number of clock cycles required to perform each technique is presented in Table 13 through a simulation tool called ModelSim Altera. The explanation of how to calculate the required number of clock cycles to generate continuous symbols was previously presented in Subsection 4.1.2.3. For Tech #1, it would take seventeen clock cycles to yield the first symbol and one clock cycle for the following ones. In the case of Tech #2 and Tech #3, it would takes 25 clock cycles and 15 clock cycles, respectively, for the first symbol, and more 7 clock cycles for each new symbol due to the dynamic of the SM. Tech #2 requires 10 clock cycles more than the Tech #3 due to the implementation of Algorithms 2, 3 and 4.

Table 13: Speed comparison for a single symbol for cross QAM.

Technique	**Latency**	**Delay**
Tech #1	16	1
Tech #2	18	7
Tech #3	8	7

Finally, Table 14 shows the required number of clock cycles to generate continuous symbols of each technique. As it can be observed, Tech #1 also improves the performance in terms of speed. Different from Subsection 4.1.2.3, the delay of Tech #2 and Tech #3 are seven clock cycles, so if 1024 symbols with the same constellation are consecutively generated, then the required number of clock cycles is reduced to almost 7 times in comparison with other techniques.

Table 14: Number of clocks for generating consecutive symbol for cross QAM.

# of Symbols	**1**	**2**	**4**	**16**	**64**	**256**	**1024**
Tech #1	17	18	20	32	80	272	1040
Tech #2	25	32	46	130	466	1810	7186
Tech #3	15	22	36	120	456	1800	7176

4.3 Square and Cross QAM Constellations

In this section, the numerical results of the combined square and cross M-QAM are presented, considering $K = 2, 3, 4, \ldots, 16$, that is, $M = 4, 8, 16, \ldots, 65536$. In this case, as shown in Subsection 4.2, the functionalities of the Algorithms 2, 3 and 4 are considered in Tech #1 and Tech #2. Tech #1 provides the same results presented in Section 4.2 because it is generic, to cover both cross and square M-QAM constellations. Additionally, it is assumed that Tech #3 includes all the LUT's and SM's needed to represent all M-QAM constellations ($k = 2, 3, 4, \ldots, K$).

4.3.1 **BER Analysis**

The BER performances of the three techniques do not change when square and cross M-QAM constellations are considered. So, for the sake of simplicity, the comparison among the three techniques for 4, 512 and 65536-QAM constellations corrupted by AWGN and AIGN are presented in Figures 67 and 68, respectively. As it can be noted, these techniques achieves almost the same performance, disregarding the type of adopted noise. The results shown that the performance loss of the proposed techniques, as discussed before in Subsections 4.1.1 and 4.2.1, are, on average, less than 1 dB. As previously described in Subsections 4.1.1 and 4.2.1, due to the usage of finite precision, with word length of $K+1$, performance differences exists in intermediate values of $E_b/\mathcal{N}_0$, but for $E_b/\mathcal{N}_0 << 1$ or $E_b/\mathcal{N}_0 >> 1$ they totally disappear.

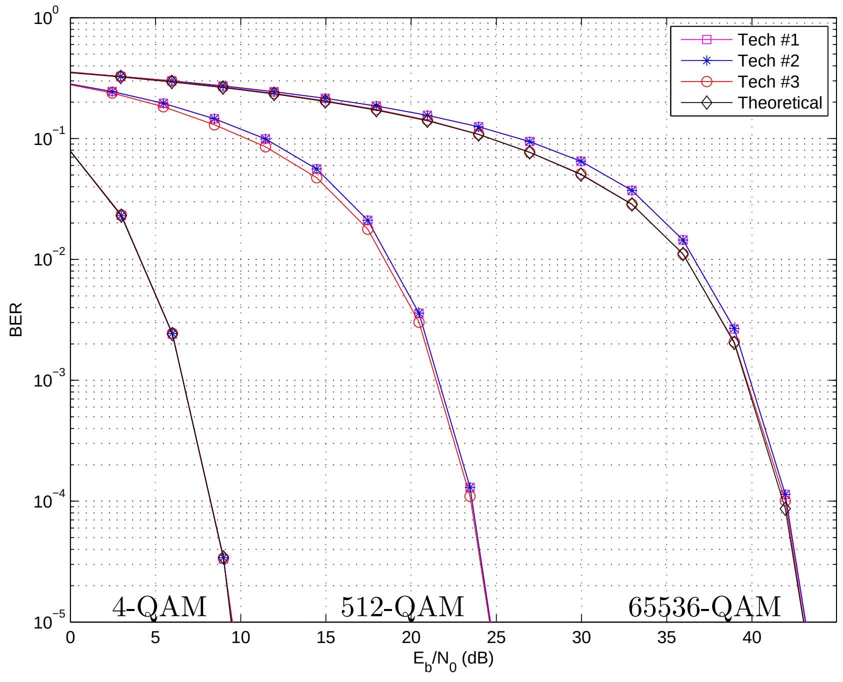


Figure 67: Performance comparison for the models - 4, 512, 65536-QAM constellation corrupted with AWGN.

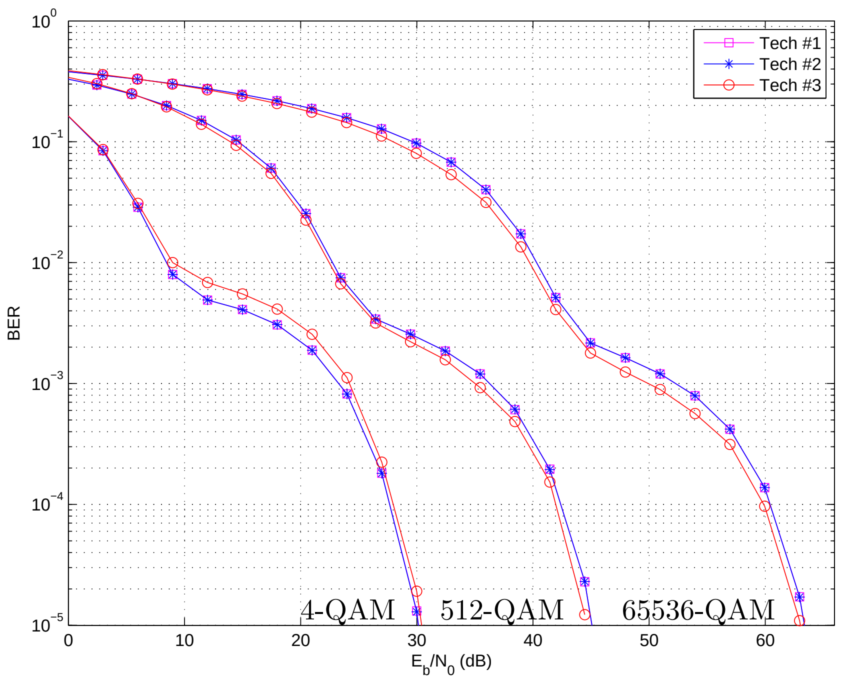


Figure 68: Performance comparison for the models - 4, 512, 65536-QAM constellation corrupted with AIGN.

4.3.2 Hardware Resource Utilization Analysis

A comparison for $K = 16$ bits is shown in Table 15. As it was discussed in previous Subsections 4.1.2 and 4.2.2, Tech #1 requires much less logic elements (almost 1% of the total available in Stratix III chipset) than the other techniques and any memory resources, it is important to emphasize that the hardware resource usage of Tech #1 is the same of the cross M-QAM implementation in Subsection 4.2.2 because it is a Generic Technique. It is also clear that Tech #2 shows lower hardware resource utilization than Tech #3 due to the usage of only one LUT and only one SM for $K = 16$ bits.

Table 15: Resources comparison between implementations for square and cross QAM

	Modulator		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE's	356	280	93
Memory (Bits)	0	1179648	2359704
	Demodulator		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE's	648	3565	10939
Memory (Bits)	0	0	0
	Total		
Type	**Tech #1**	**Tech #2**	**Tech #3**
LE	1004	3845	11032
Memory	0	1179648	2359704

Similarly to previous subsections, the obtained results made with K varying from 2 to 16 are shown graphically in Figure 69 for memory utilization comparison, Figure 70 for LE utilization comparison and in Figure 71 for memory and LE utilization comparison. The hardware resource utilization ratio, Equation (4.1), is employed in Figures 69 and 70, and the normalized hardware resource utilization ratio, Equation (4.4), is employed in Figure 71, where, as well as Subsections 4.1.2 and 4.2.2, the same weight is given to both kinds of hardware resource.

In Figure 69, in accordance with Subsections 4.1.2 and 4.2.2, Tech #1 does not requires any memory resources, while Tech #2 presents lower memory resource utilization than Tech #3, converging to 0.5, half of the memory resources employed in Tech #3. In Figure 70, for $K \geq 7$ bits, Tech #1 uses less logical elements than the other techniques, and also Tech #2 presents lower LE resource utilization than Tech #3 for $K \geq 8$. The combined resource utilization, in Figure 71 shown that for $K \geq 6$ bits, Tech #1 has less hardware resource utilization than the other techniques, and also Tech #2 presents lower hardware resource utilization than Tech #3 for $K \geq 8$. In addition, Tech #2 and Tech #3 presented the same resource utilization only for $K = 2$ into Figures 69-71, the same result presented for only square M-QAM constellations in Subsection 4.1.2 because there is no external logic element or memory resource employed to execute other M-QAM constellations different from 4-QAM.

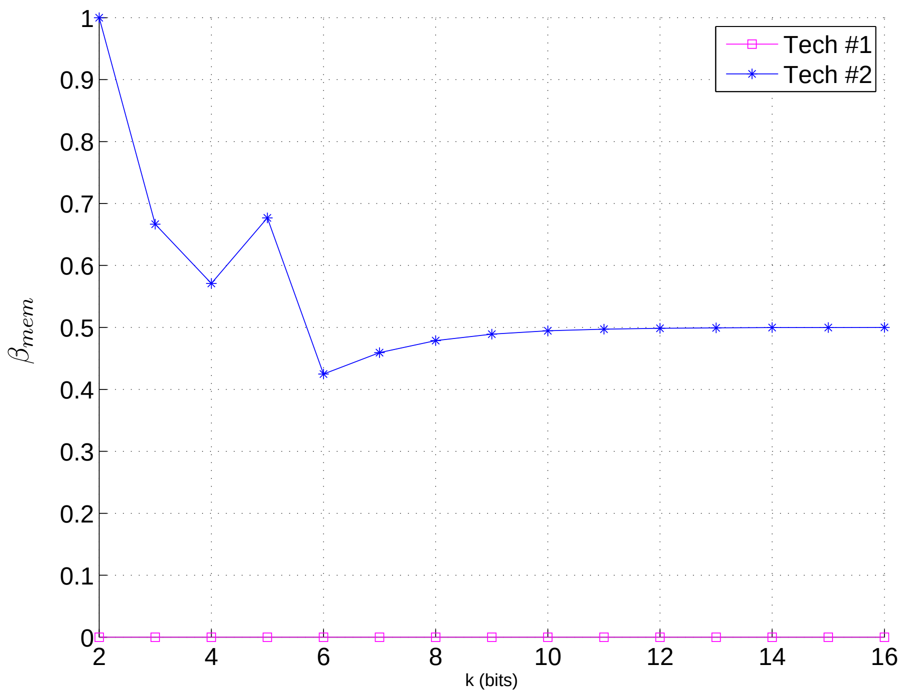


Figure 69: The memory utilization comparison for square and cross QAM in FPGA device when the constellation cardinality changes.

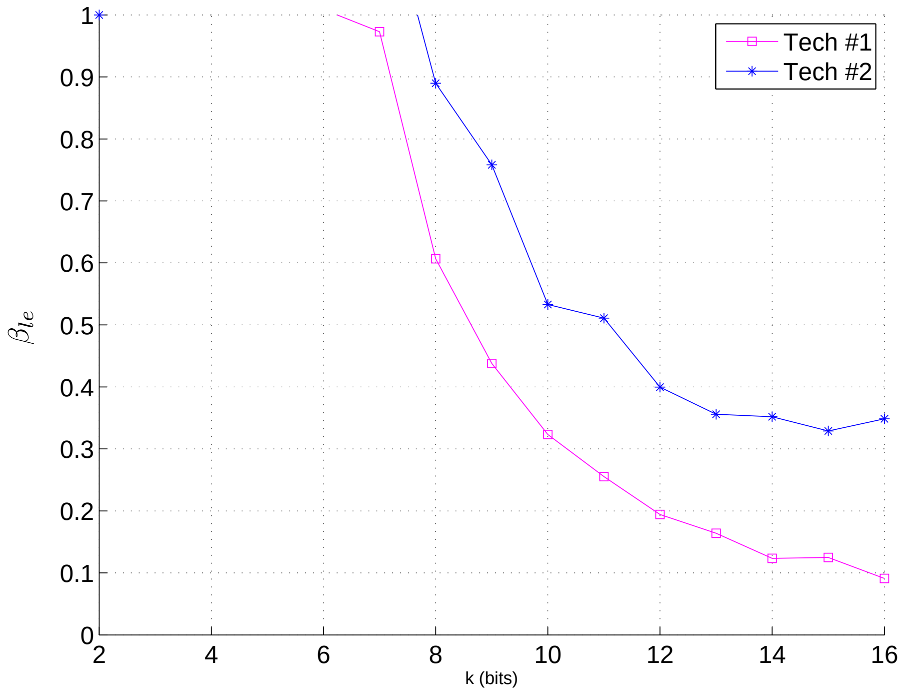


Figure 70: The LE utilization comparison for square and cross QAM in FPGA device when the constellation cardinality changes.

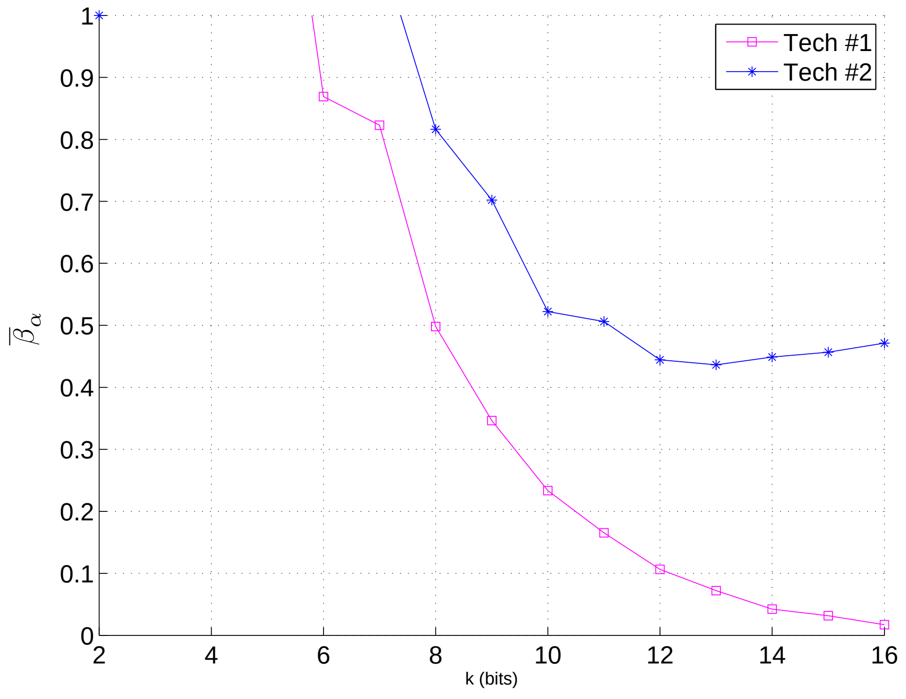


Figure 71: The LE and memory utilization comparison for square and cross QAM in FPGA device when the constellation cardinality changes.

As previously presented in Subsections 4.1.2.1 and 4.2.2.1, the maximum frequency is presented as a result and correct configurations should be applied to calculate it. The maximum frequency of the three techniques, working at 85 Celsius degrees, are presented in Table 16.

Table 16: Timing Analyzer of square and cross 2^{16}-QAM implementation

Technique	**Fmax**
Tech #1	234,08 MHz
Tech #2	207,34 MHz
Tech #3	132,21 MHz

As it can be observed, Tech #1 that uses only logical elements presents higher maximum frequency. The advantages of this relies on the design of high data-rate communication systems, as presented in Section 2.2.1 and Subsections 4.1.2.1 and 4.2.2.1 considering broadband PLC systems [57]. To provide analysis about the power consumption, Table 17 presents the estimated thermal power, that is the power dissipated in the device, of the three techniques.

Table 17: Power Analysis of square and cross 2^{16}-QAM implementation

Technique	**Tech #1**	**Tech #2**	**Tech #3**
Core Dynamic Thermal Power Dissipation	29,02 mW	60,08 mW	273,45 mW
Core Static Thermal Power Dissipation	570,26 mW	574,95 mW	601,20 mW
I/O Thermal Power Dissipation	45,64 mW	54,81 mW	51,08 mW
Total Thermal Power Dissipation	**644,92 mW**	**689,84 mW**	**925,72 mW**

As it can be observed, Tech #1 is the one which requires less thermal power dissipation due to the low usage of logical elements without the need of any memory resources. The Total Thermal Power Dissipation is used for analysis as discussed in Subsections 4.1.2.2 and 4.2.2.2. Tech #3 is the one with highest thermal power dissipation, because it requires high quantity of memory resources and logical elements, and Tech #2 is an intermediate case.

The analysis of the number of clock cycles required to perform each technique is presented in Table 18. The required number of clock cycles to generate continuous symbols was previously presented in Subsection 4.1.2.3. For the Tech #1, the operation would be faster for continuous symbol generation, the first symbol would take seventeen clock cycles and the others only 1 due to parallelism of operations, the same result of Subsection 4.2.2.3. For Tech #2 and Tech #3, it would takes 27 clock cycles and 17 clock cycles, respectively, for the first symbol and more 8 clock cycles for each new symbol due to the dynamic of the SM.

Finally, Table 19 shows the required number of clock cycles to perform each technique when the number of symbols belongs to the set $N_s \in \{1, 2, 4, \ldots, 1024\}$, as

Table 18: Speed comparison for a single symbol for square and cross QAM.

Technique	**Latency**	**Delay**
Tech #1	16	1
Tech #2	19	8
Tech #3	9	8

previously described in Subsection 4.1.2.3. It can be noticed that, for all number of consecutive symbols transmitted, Tech #1 improves the performance in terms of speed. If 1024 symbols with the same constellation are consecutively transmitted, then the same result of Subsection 4.1.2.3 is obtained, reducing the required number of clock cycles to almost 8 times less than the other techniques.

Table 19: Number of clocks for generating consecutive symbol for square and cross QAM.

# of Symbols	**1**	**2**	**4**	**16**	**64**	**256**	**1024**
Tech #1	17	18	20	32	80	272	1040
Tech #2	27	35	51	147	531	2067	8211
Tech #3	17	25	41	137	521	2057	8201

5 CONCLUSION

This thesis investigated the computational complexity reduction in high-order M-QAM constellations, considering $K = 2, 3, 4, \ldots, 16$, that is, $M = 4, 8, 16, \ldots, 65536$. Aiming at reducing the memory bits and logic elements in a FPGA device, two novel techniques were introduced (Tech #1 and Tech #2). Both proposed techniques can modulate and demodulate square and cross M-QAM symbols from constellations by using the hard decision rule. Additionally, both of them can derive low-order square or cross M-QAM constellations from the highest square M-QAM constellation.

The description of Tech #1 - Generic M-QAM technique - showed that the need for memory bit usage to store M-QAM indices in a LUT can be efficiently replaced by a simple logic approach. As a result, Tech #1 eliminates the need for LUT. Additionally, the introduction of a procedure to obtain cross constellation from a rectangular one, independent of M, and to derive a low-order M-QAM constellation from a high-order one considerably reduces the need for hardware resource utilization in a FPGA device.

By investigating the weaknesses of the proposed M-QAM technique in [23, 24], this work introduced an enhanced version of the HDR technique, Tech #2. Basically, Tech #2 improves the HDR technique by using only one LUT to store the square M-QAM constellation of the highest order, the derivation of lower-order square or cross M-QAM constellations from the square M-QAM constellation with the highest order and the use of one single SM to perform the detection of symbols in all M-QAM constellations.

In order to evaluate the effectiveness and suitability of the proposed techniques, performance analyses in terms of BER are presented and discussed when AWGN and AIGN are used to corrupt M-QAM symbols. Regarding the chosen values of k, the performance curves in terms of E_b/N_0 reveal that negligible performance degradation is inserted by the proposed technique for both square and cross M-QAM constellations, because numerical simulations were carried out with the proposed techniques implemented into the same conditions used in a FPGA device, by employing digital quantification of the received signal. Additionally, comparison results with theoretical results (AWGN) and Tech #3 (HDR technique) for $k = 2, \ldots, 16$ confirm that performance losses are, on average, less than 1 dB.

Also, this thesis described the implementation of the proposed techniques in FPGA devices and analyzed the hardware resource utilization based on memory bits, logic elements, maximum frequency of operation, energy consumption, delay and latency, by considering only square M-QAM constellation, only cross M-QAM constellation and both of them in the same FPGA device. According to the attained results, for square M-QAM constellation, Tech #1 offers the lowest hardware resource utilization if $M \geq 16$. Tech #2 showed better results than Tech #3 under the same operation speed for all values of M associated with square constellations, with the exception of $M = 4$, in which both presented the same results. For cross M-QAM constellation, Tech #1 offers the best results if $M \geq 512$. Again, Tech #2 achieve better results than Tech #3, with the same operation speed, when $M \geq 2048$. Then, Tech #3 attains the best results when $M < 512$ is taken into account. For both square and cross M-QAM, Tech #1 offers the lowest demand of hardware resource utilization compared to Tech #2 and Tech #3, when M increases ($M \geq 64$). Also, Tech #1 can modulate and demodulate M-QAM symbols up to 8 times faster than Tech #2 and Tech #3 when a large number of M-QAM symbols (e.g., 1000) are consecutively transmitted. A remarkable result is that Tech #1 completely eliminates the need for memory storage and state machines. Moreover, Tech #2 showed intermediate results, better than Tech #3, with the same operation speed, when considering $M \geq 256$. Finally, Tech #3 proved lower hardware resource utilization than Tech #1 and Tech #2 when $M < 64$. Overall, Tech #1 showed to be the most suitable M-QAM technique to deal with high and low-orders square and/or cross M-QAM constellations.

5.1 Future Works

A list of possible works that could be carried out in the future are as follows:

1. Performance comparison between the techniques in digital signal processor (DSP).
2. The extension of proposed techniques to handle three-dimensional (3D) and multidimensional M-QAM constellations as well as their use in multi-carrier schemes based on 1D and 2D discrete Fourier Transform.
3. The investigation of the gains that could be obtained using the ideas behind the proposed techniques in other digital modulation techniques.

REFERENCES

[1] *IEEE standard for broadband over power line networks: Medium access control and physical layer specifications (IEEE Std. 1901)*, IEEE Communications Society Std., Dec. 2010.

[2] V. Oksman and J. Zhang, “G.hnem: The new itu-t standard on narrowband plc technology,” *IEEE Communications Magazine*, vol. 49, no. 12, pp. 36–44, Dez 2011.

[3] ITU-T. (2011) Narrowband orthogonal frequency division multiplexing power line communication transceivers - physical layer specification. [Online]. Available: https://www.itu.int/rec/T-REC-G.9955

[4] ITU-T. (2011) Narrowband orthogonal frequency division multiplexing power line communication transceivers - data link layer specification. [Online]. Available: https://www.itu.int/rec/T-REC-G.9956

[5] T. Gea, J. Paradells, M. Lamarca, and D. Roldan, “Smart cities as an application of internet of things: Experiences and lessons learnt in barcelona,” in *Proc. 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Jul. 2013, pp. 552–557.

[6] Y. Y. Nasrallah, I. Al-Anbagi, and H. Mouftah, “A quality of service model for ieee 802.11p communication protocol in a smart city,” in *Proc. Global Information Infrastructure and Networking Symposium*, Sep. 2014, pp. 1–3.

[7] J. N. Murdock and T. S. Rappaport, “Consumption factor and power- efficiency factor: A theory for evaluating the energy efficiency of cascaded communication systems,” *IEEE Journal on Selected Areas Communications*, vol. 32, no. 12, pp. 1–16, Dez 2014.

[8] F. L. FIGUEIREDO, *Fundamentos da Tecnologia Wimax.* Centro de Pesquisa e Desenvolvimento em Telecomunicações - CPqD, 2009.

[9] F. P. V. D. Campos, “Multicarrier modulation transceivers and resource allocation for power line communication,” Ph.D. dissertation, Federal University of Rio de Janeiro, 2013.

[10] S. K. K. Mitra, *Digital Signal Processing: A Computer-Based Approach.* McGraw-Hill Higher Education, 2000.

[11] A. Z. A. M. Tonello, P. Siohan, and X. Mongaboure, “Challenges for 1 gbps power line communications in home networks,” in *Proc. IEEE Personal Indoor Mobile Radio Communications Symposium*, Sep. 2008, pp. 1–6.

[12] C. Cahn, “Combined digital phase and amplitude modulation communication system,” *IRE Transactions on Communications Systems*, vol. 8, no. 3, pp. 150–155, Sep. 1960.

[13] C. Campopiano and B. Glazer, “A coherent digital amplitude and phase modulation scheme,” *IRE Transactions on Communications Systems*, vol. 10, no. 1, pp. 90–95, Mar. 1962.

[14] J. C. Hancock and R. W. Lucky, “Performance of combined amplitude and phase modulated communications system,” *IRE Transactions on Communications Systems*, vol. 8, no. 4, pp. 232–237, Dec. 1960.

[15] W. T. Bisignant, G. P. Richards, and J. W. Whelan, “The improved gray scale and the coarse-fine pcm systems, two new digital tv bandwidth reduction techniques,” *Proceedings of the IEEE*, vol. 54, no. 3, pp. 376–390, Mar. 1966.

[16] J. R. Bitner, G. Ehrlich, and E. M. Reingold, “Efficient generation of the binary reflected gray code and its aplications,” *Communications of the ACM*, vol. 19, no. 9, pp. 517–521, Sep. 1976.

[17] S. Okamoto, K. Toyoda, T. Omiya, K. Kasai, M. Yoshida, and M. Nakazawa, “512 qam (54 gbit/s) coherent optical transmission over 150 km with an optical bandwidth of 4.1 ghz,” in *Proc. European Conference on Optical Communication*, Sep. 2010, pp. 1–3.

[18] P. K. Vitthaladevuni, M. S. Alouini, and J. C. Kieffer, “Exact ber computations of cross qam constellations,” *IEEE Transactions Wireless Communications*, vol. 4, no. 6, pp. 3039–3050, Nov. 2005.

[19] J. Smith, “Odd-bit quadrature amplitude-shift keying,” *IEEE Transactions on Communications*, vol. 23, no. 3, pp. 385–389, Mar. 1975.

[20] Y. Wu, Y. Zhao, and H. Li, “Constellation design for odd-bit quadrature amplitude modulation,” in *Proc. IEEE Wireless Communications and Networking Conference Workshops*, Apr. 2010, pp. 1–4.

[21] R. Swain and A. K. Panda, “Design of 16-qam transmitter and receiver: Review of methods of implementation in fpga,” *International Journal of Engineering and Science*, vol. 1, no. 9, pp. 23–27, Nov. 2012.

[22] X. T. Vu, N. A. Duc, and T. A. Vu, “16-qam transmitter and receiver design based on fpga,” in *Proc. 5th IEEE International Symposium on Electronic Design, Test and Application*, Jan. 2010, pp. 95–98.

[23] G. F. C. Lemos, M. V. S. Oliveira, L. M. A. Filho, F. P. V. De Campos, and M. V. Ribeiro, “A low-cost implementation of high-order square m-qam detection/demodulation in a fpga device,” in *Proc. 7th International Telecommunications Symposium*, Nov. 2010.

[24] G. F. C. Lemos, “Técnicas de detecção e implementação em fpga de modulações qam de ordem elevada,” Master’s thesis, Federal University of Juiz de Fora, 2011.

[25] K. Kin, N. Basutkar, K. Bae, P. Xue, and H. Yang, “One-dimensional soft-demapping algorithms for rotated qam and software implementation on dsp,” *IEEE Transactions on Signal Processing*, vol. 61, no. 15, pp. 3918–3930, Aug. 2013.

[26] B. I. of Geography and Statistics. (2010) Population census. [Online]. Available: http://www.censo2010.ibge.gov.br

[27] H. HRASNICA, A. HAIDINE, and R. LEHNERT, *Broadband Powerline Communications-Network Design.* John Wiley & Sons, 2004.

[28] L. T. Berger, S. A., and J. J. Escudero-Garzás, “Power line communications for smart grid applications,” *Journal of Electrical and Computer Engineering*, vol. 2013, no. 712376, pp. 1–16, Dec. 2012.

[29] K. DOSTERT, *Powerline Communications.* Prentice Hall, 2001.

[30] Qualcomm. (2015) Powerline communication. [Online]. Available: http://www.qca.qualcomm.com/networking/connected-home/powerline/

[31] Marvell. (2015) Powerline communication transceiver. [Online]. Available: http://www.marvell.com/in-home-networking/ghn/

[32] E. Liu, Y. Gao, G. Samdani, and O. Mukhtar, "Broadband powerline channel and capacity analysis," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Apr. 2005, pp. 7–11.

[33] A. A. M. Picorone, "A contribution to the plc time variants channel estimation using pilots signals," Master's thesis, Federal University of Juiz de Fora, 2009.

[34] T. R. Oliveira, "The characterization of hybrid plc-wireless and plc channels in the frequency band between 1.7 and 100 mhz for data communication," Ph.D. dissertation, Federal University of Juiz de Fora, 2015.

[35] F. J. A. Andrade, C. A. G. Marques, T. R. Oliveira, F. P. V. De Campos, D. O. E. J., and R. M. V., "Preliminary analysis of additive noise on outdoor and low voltage electric power grid in brazil," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Mar. 2013, pp. 109–113.

[36] A. A. M. Picorone, "Comunicação digital em canais plc: Técnicas de transmissão, detecção e caracterização de canais plc outdoor brasileiros," Ph.D. dissertation, Federal University of Juiz de Fora, 2014.

[37] M. Zimmermann and K. Dostert, "A multipath model for the powerline channel," *IEEE Transactions on Communications*, vol. 50, no. 4, pp. 553–559, Apr. 2002.

[38] F. J. Canete, L. Díez, J. A. Cortes, and J. T. Entrambasaguas, "Broadband modelling of indoor power-line channels," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 1, pp. 175–183, Feb. 2002.

[39] A. A. M. Picorone, L. R. Amado, and M. V. Ribeiro, "Linear and periodically time-varying plc channels estimation in the presence of impulsive noise," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Mar. 2010, pp. 255–260.

[40] A. A. M. Picorone, T. R. Oliveira, and R. M. V., "Plc channel estimation based on pilot signals for ofdm modulation: A review," *IEEE Latin America Transactions*, vol. 12, no. 4, pp. 580–589, Jun. 2014.

[41] F. J. C. Corripio, J. A. C. Arrabal, L. D. Del Rio, and J. T. E. Munoz, "Analysis of the cyclic short-term variation of indoor power line channels," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 7, pp. 1327–1338, Jul. 2006.

[42] M. Zimmermann and K. Dostert, "A multipath model for the powerline channel," *IEEE Transaction on Communications*, vol. 50, no. 4, pp. 553–559, Apr. 2002.

[43] M. Zimmermann and K. Dostert, "An analysis of the broadband noise scenario in powerline networks," in *Proc. 4th International Symposium on Power-Line Communications and its Applications*, 2000, pp. 131–138.

[44] L. Di Bert, P. Caldera, D. Schwingshackl, and A. M. Tonello, "On noise modeling for power line communications," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Apr. 2011, pp. 283–288.

[45] R. Hormis, B. I., and W. X., "A simple baseband transmission scheme for power line channels," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 7, pp. 1351–1363, Jul. 2006.

[46] A. A. M. Picorone, L. R. Amado, and M. V. Ribeiro, "Linear and periodically time-varying plc channels estimation in the presence of impulsive noise," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Mar. 2010, pp. 255–260.

[47] L. L. HANZO, T. K. S. X. NG, and W. WEB, *Quadrature Amplitude Modulation: From Basics to Adaptive TrellisCoded, Turbo-Equalised and Space-Time Coded OFDM, CDMA and MC-CDMA Systems.* Wiley-Blackwell, 2004.

[48] F. Gray. (1953) Pulse code communication. [Online]. Available: http://www.freepatentsonline.com/2632058.html

[49] A. R. S. BAHAI, B. R. SATZBERG, and M. ERGEN, *Multi-carrier digital communications; Theory and applications of OFDM.* Springer - Verlag Telos, 2004.

[50] Y. G. LI and G. L. STUBER, *Orthogonal Frequency Division Multiplexing for Wireless Communications.* Springer, 2006.

[51] A. Svensson, "An introduction to adaptive qam modulation schemes for known and predicted channels," *Proceedings of the IEEE*, vol. 95, no. 12, pp. 2322–2336, Dec. 2007.

[52] H. M. De Oliveira and G. Battail, "On generalized two-dimensional cross constellations and the opportunistic secondary channel," *Annales Des Télécommunications*, vol. 47, no. 5-6, pp. 202–213, May 1992.

[53] F. P. V. De Campos and M. V. Ribeiro, "Performance analysis of clustered ofdm system with bitloading algorithm for broadband plc," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Apr. 2008, pp. 345–350.

[54] T. Pfau, S. Hoffman, and R. Noé, "Hardware-efficient coherent digital receiver concept with feedforward carrier recovery for m-qam constellations," *Journal of Lightwave Technology*, vol. 27, no. 8, pp. 989–999, Apr. 2009.

[55] A. Corporation. (2013) Quartus ii handbook version 13.1. [Online]. Available: http://www.altera.com/en_US/pdfs/literature/hb/qts/quartusii_handbook.pdf

[56] A. Corporation. (2010) Stratix iii device handbook. [Online]. Available: http://www.altera.com/en_US/pdfs/literature/hb/stx3/stratix3_handbook.pdf

[57] M. V. Ribeiro, F. P. V. De Campos, G. R. Colen, H. V. Schettino, F. D., L. M. Sirimarco, and V. Fernandes, "A novel power line communication system for outdoor electric power grids," in *Proc. IEEE International Symposium on Power Line Communications and Its Applications*, Mar. 2015, pp. 228–233.

[58] A. Corporation. (2015) Powerplay early power estimator user guide. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_epe.pdf

Appendix A – Publications

The list of publications during the mastering period are as follows:

- D. Fernandes, V. Fernandes, G. F. C. Lemos, W. A. Finamore, M. V. Ribeiro, "Low-Cost Technique for Generic Square and Cross M-QAM Symbols," in *Proc. IEEE Transactions on Very Large Scale Integration Systems*, submittted on May 2015.
- F. A. Santos, L. M. F. da Silveira, D. Fernandes, L. M. B. A. Dib, F. P. V. de Campos, M. V. Ribeiro, "Integration of a Smart Meter with the Brazilian Broadband PLC System," in *Proc. XXXIII Simpósio Brasileiro de Telecomunicações - SBrT 2015*, Sep. 2015.
- G. R. Colen, H. V. Schettino, D. Fernandes, L. M. Sirimarco, F. P. V. Campos, W. A. Finamore, H. A. Latchman, M. V. Ribeiro, "A temporal compressive resource allocation technique for complexity reduction in PLC transceivers," *Trans. on Emerging Telecommunications Technologies*, accepted on May 2015.
- G. R. Colen, H. V. Schettino, D. Fernandes, L. M. Sirimarco, V. Fernandes, A. A. M. Picorone, F. P. V. Campos, M. V. Ribeiro, "A Novel Power Line Communication System for Outdoor Electric Power Grids," in *Proc. IEEE International Symposium on Power Line Communications and its Applications (ISPLC)*, Mar. 2015, pp. 228-233.
- F. P. V. de Campos, T. C. Paschoalin, D. Fernandes, "Dispositivo mostrador eletrônico remoto com informações audiovisuais e em braille para redes inteligentes,", BR1020150052456, Mar. 2015. 14p.

ANNEX A – Power line communication system

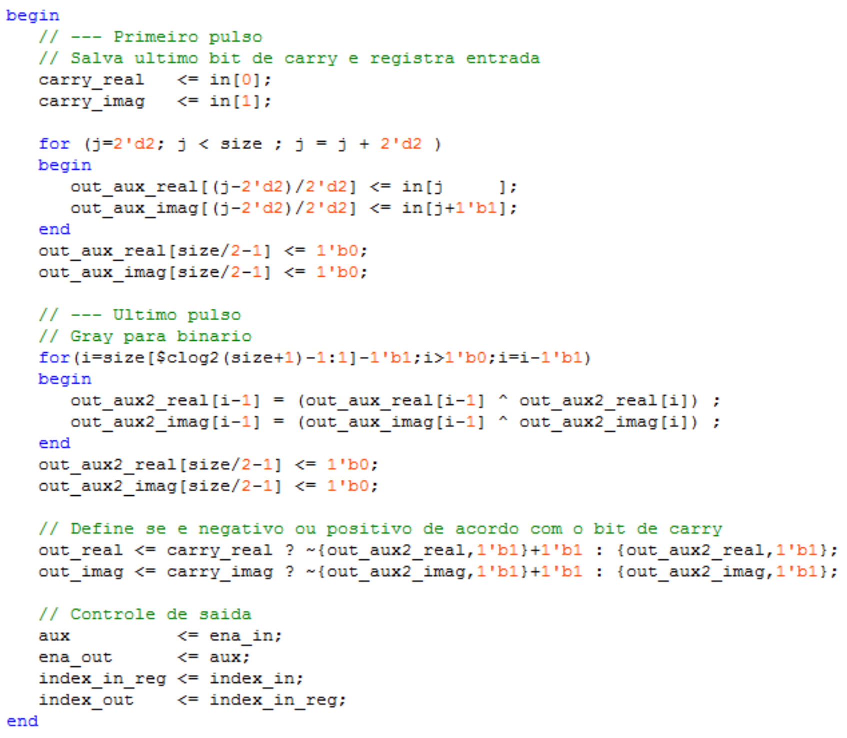
Currently, an old technology, called PLC, has gained visibility as an option for high-speed and low-speed data communication. This technology uses the already deployed electric power grid infrastructures, which were designed for energy transmission and delivery, to provide data networks. This solution is very interesting for electric utilities due to the fact that power cables capillarity reaches more than 98.9% of the households and establishments in Brazil [26]. Financial reports also suggest that the use of PLC technology can represent a cost reduction of approximately 50% in the required investment with the deployment of new data networks infrastructure [27]. Additionally, the PLC technology is being considered as the key technology to the widespread deployment of smart grids and smart cities [28].

In spite of these recent and important applications, the use of PLC is not new [29]. For example, in 1991, narrowband applications were standardized in Europe (CENELEC standard) considering data rates up to 144 kbps. In 1998, new advances have culminated in the development of broadband over power lines (BPL), with data rates up to 14 Mbps in the physical layer. In 2005, an IEEE working group started developing a standard for the link and physical layers of the BPL that culminated on the IEEE-P1901 standard [1]. This standard describes data rates higher than 100 Mbps in the physical layer. Also, G.9955 [3] and G.9956 [4], respectively, the physical layer and data link layer specifications for narrowband powerline communication transceivers, were released by the ITU-T in 2011. In the last decades, some companies offered PLC chipsets with data rates between 85 Mbps and 500 Mbps (Intellon, DS2, Panasonic and SPIDCOM). Nowadays, there are PLC chipsets that reach 1 Gbps (Qualcomm [30] and Marvel [31]), and, also, researches indicate that the new generation of PLC technologies will reach up to 2 Gbps [32].

Even with all advances on PLC technology, it is important to emphasize that electric power grids infrastructures were not designed for data communication. Therefore problems not found in other guided media of communication like telephone cables, coaxial cables and twisted pairs are relevant in power cables [9], having similar characteristics found in wireless communication. These problems can considerably reduce the performance of PLC systems. Features such as time selectivity, frequency selectivity, high power, random impulsive noise [33] and time-varying behavior is common in electric power grids. A methodology to be applied to the characterization of electric power grids for data communication purposes is presented in [34] and applied for in-home Brazilian PLC channels. Also, [34, 35] discuss a preliminary analysis of additive noises on outdoor and low voltage electric power grids in the frequency band between 1.7 and 100 MHz in Brazil. The statistical analysis in the time and frequency domains reveal the hardness of such environment for data communication and the opportunities to pursue the design of efficient and effective PLC technology for both broadband and narrowband applications. The Brazilian outdoor electric power grid of low voltage is discussed as communication medium in [36]. Several parameters were analysed, such as the average channel gain, the coherence time, the coherence bandwidth, the delay spread, the length of the channel impulse response, and the channel capacity, for the frequency bands from 1.7 to 30 MHz, from 1.7 up to 50 MHz, and from 1.7 up to 100 MHz. The connection of distinct loads and the fact that power cables are unshielded originate the uniqueness of noises in the electric power grids.

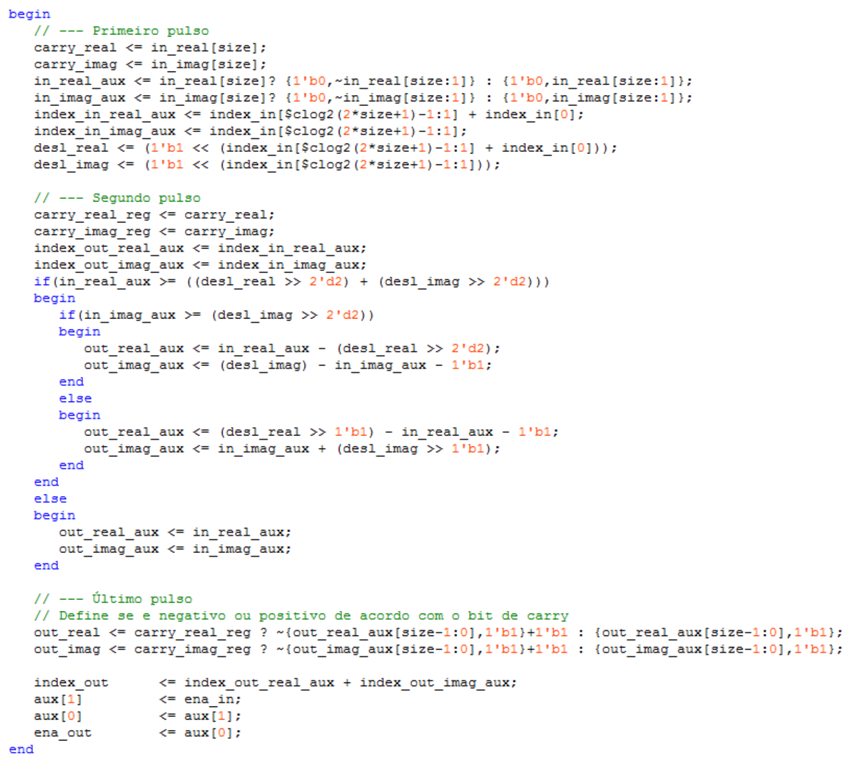
ANNEX B – Proposed Technique in Verilog

The Figures 72-81 presents the algorithms implemented on Verilog language using Quartus II Altera tool.



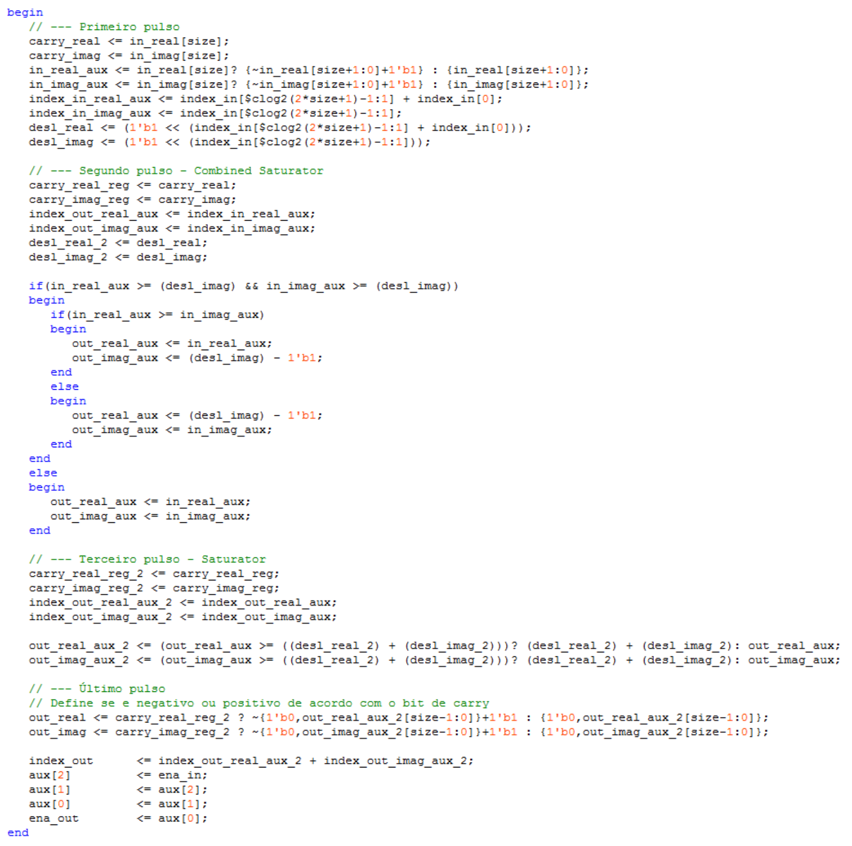
```
begin
   // --- Primeiro pulso
   // Salva ultimo bit de carry e registra entrada
   carry_real   <= in[0];
   carry_imag   <= in[1];

   for (j=2'd2; j < size ; j = j + 2'd2 )
   begin
      out_aux_real[(j-2'd2)/2'd2] <= in[j     ];
      out_aux_imag[(j-2'd2)/2'd2] <= in[j+1'b1];
   end
   out_aux_real[size/2-1] <= 1'b0;
   out_aux_imag[size/2-1] <= 1'b0;

   // --- Ultimo pulso
   // Gray para binario
   for(i=size[$clog2(size+1)-1:1]-1'b1;i>1'b0;i=i-1'b1)
   begin
      out_aux2_real[i-1] = (out_aux_real[i-1] ^ out_aux2_real[i]) ;
      out_aux2_imag[i-1] = (out_aux_imag[i-1] ^ out_aux2_imag[i]) ;
   end
   out_aux2_real[size/2-1] <= 1'b0;
   out_aux2_imag[size/2-1] <= 1'b0;

   // Define se e negativo ou positivo de acordo com o bit de carry
   out_real <= carry_real ? ~{out_aux2_real,1'b1}+1'b1 : {out_aux2_real,1'b1};
   out_imag <= carry_imag ? ~{out_aux2_imag,1'b1}+1'b1 : {out_aux2_imag,1'b1};

   // Controle de saida
   aux          <= ena_in;
   ena_out      <= aux;
   index_in_reg <= index_in;
   index_out    <= index_in_reg;
end
```

Figure 72: Modulation algorithm in Verilog.



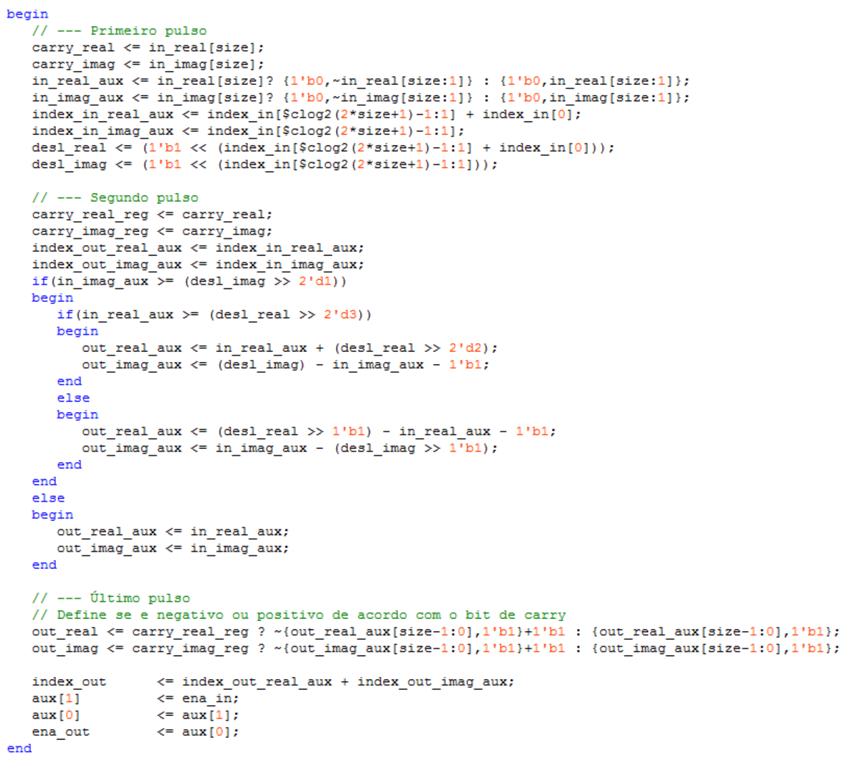
```verilog
begin
   // --- Primeiro pulso
   carry_real <= in_real[size];
   carry_imag <= in_imag[size];
   in_real_aux <= in_real[size]? {1'b0,~in_real[size:1]} : {1'b0,in_real[size:1]};
   in_imag_aux <= in_imag[size]? {1'b0,~in_imag[size:1]} : {1'b0,in_imag[size:1]};
   index_in_real_aux <= index_in[$clog2(2*size+1)-1:1] + index_in[0];
   index_in_imag_aux <= index_in[$clog2(2*size+1)-1:1];
   desl_real <= (1'b1 << (index_in[$clog2(2*size+1)-1:1] + index_in[0]));
   desl_imag <= (1'b1 << (index_in[$clog2(2*size+1)-1:1]));

   // --- Segundo pulso
   carry_real_reg <= carry_real;
   carry_imag_reg <= carry_imag;
   index_out_real_aux <= index_in_real_aux;
   index_out_imag_aux <= index_in_imag_aux;
   if(in_real_aux >= ((desl_real >> 2'd2) + (desl_imag >> 2'd2)))
   begin
      if(in_imag_aux >= (desl_imag >> 2'd2))
      begin
         out_real_aux <= in_real_aux - (desl_real >> 2'd2);
         out_imag_aux <= (desl_imag) - in_imag_aux - 1'b1;
      end
      else
      begin
         out_real_aux <= (desl_real >> 1'b1) - in_real_aux - 1'b1;
         out_imag_aux <= in_imag_aux + (desl_imag >> 1'b1);
      end
   end
   else
   begin
      out_real_aux <= in_real_aux;
      out_imag_aux <= in_imag_aux;
   end

   // --- Último pulso
   // Define se e negativo ou positivo de acordo com o bit de carry
   out_real <= carry_real_reg ? ~{out_real_aux[size-1:0],1'b1}+1'b1 : {out_real_aux[size-1:0],1'b1};
   out_imag <= carry_imag_reg ? ~{out_imag_aux[size-1:0],1'b1}+1'b1 : {out_imag_aux[size-1:0],1'b1};

   index_out      <= index_out_real_aux + index_out_imag_aux;
   aux[1]         <= ena_in;
   aux[0]         <= aux[1];
   ena_out        <= aux[0];
end
```

Figure 73: Position Switcher algorithm in Verilog.



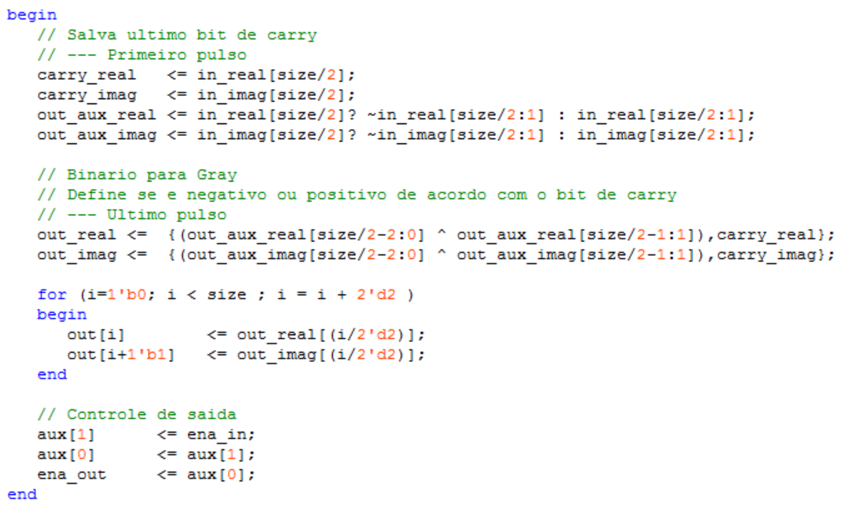
```verilog
begin
   // --- Primeiro pulso
   carry_real <= in_real[size];
   carry_imag <= in_imag[size];
   in_real_aux <= in_real[size]? {~in_real[size+1:0]+1'b1} : {in_real[size+1:0]};
   in_imag_aux <= in_imag[size]? {~in_imag[size+1:0]+1'b1} : {in_imag[size+1:0]};
   index_in_real_aux <= index_in[$clog2(2*size+1)-1:1] + index_in[0];
   index_in_imag_aux <= index_in[$clog2(2*size+1)-1:1];
   desl_real <= (1'b1 << (index_in[$clog2(2*size+1)-1:1] + index_in[0]));
   desl_imag <= (1'b1 << (index_in[$clog2(2*size+1)-1:1]));

   // --- Segundo pulso - Combined Saturator
   carry_real_reg <= carry_real;
   carry_imag_reg <= carry_imag;
   index_out_real_aux <= index_in_real_aux;
   index_out_imag_aux <= index_in_imag_aux;
   desl_real_2 <= desl_real;
   desl_imag_2 <= desl_imag;

   if(in_real_aux >= (desl_imag) && in_imag_aux >= (desl_imag))
   begin
      if(in_real_aux >= in_imag_aux)
      begin
         out_real_aux <= in_real_aux;
         out_imag_aux <= (desl_imag) - 1'b1;
      end
      else
      begin
         out_real_aux <= (desl_imag) - 1'b1;
         out_imag_aux <= in_imag_aux;
      end
   end
   else
   begin
      out_real_aux <= in_real_aux;
      out_imag_aux <= in_imag_aux;
   end

   // --- Terceiro pulso - Saturator
   carry_real_reg_2 <= carry_real_reg;
   carry_imag_reg_2 <= carry_imag_reg;
   index_out_real_aux_2 <= index_out_real_aux;
   index_out_imag_aux_2 <= index_out_imag_aux;

   out_real_aux_2 <= (out_real_aux >= ((desl_real_2) + (desl_imag_2)))? (desl_real_2) + (desl_imag_2): out_real_aux;
   out_imag_aux_2 <= (out_imag_aux >= ((desl_real_2) + (desl_imag_2)))? (desl_real_2) + (desl_imag_2): out_imag_aux;

   // --- Último pulso
   // Define se e negativo ou positivo de acordo com o bit de carry
   out_real <= carry_real_reg_2 ? ~{1'b0,out_real_aux_2[size-1:0]}+1'b1 : {1'b0,out_real_aux_2[size-1:0]};
   out_imag <= carry_imag_reg_2 ? ~{1'b0,out_imag_aux_2[size-1:0]}+1'b1 : {1'b0,out_imag_aux_2[size-1:0]};

   index_out      <= index_out_real_aux_2 + index_out_imag_aux_2;
   aux[2]         <= ena_in;
   aux[1]         <= aux[2];
   aux[0]         <= aux[1];
   ena_out        <= aux[0];
end
```

Figure 74: Combined Saturator algorithm in Verilog.



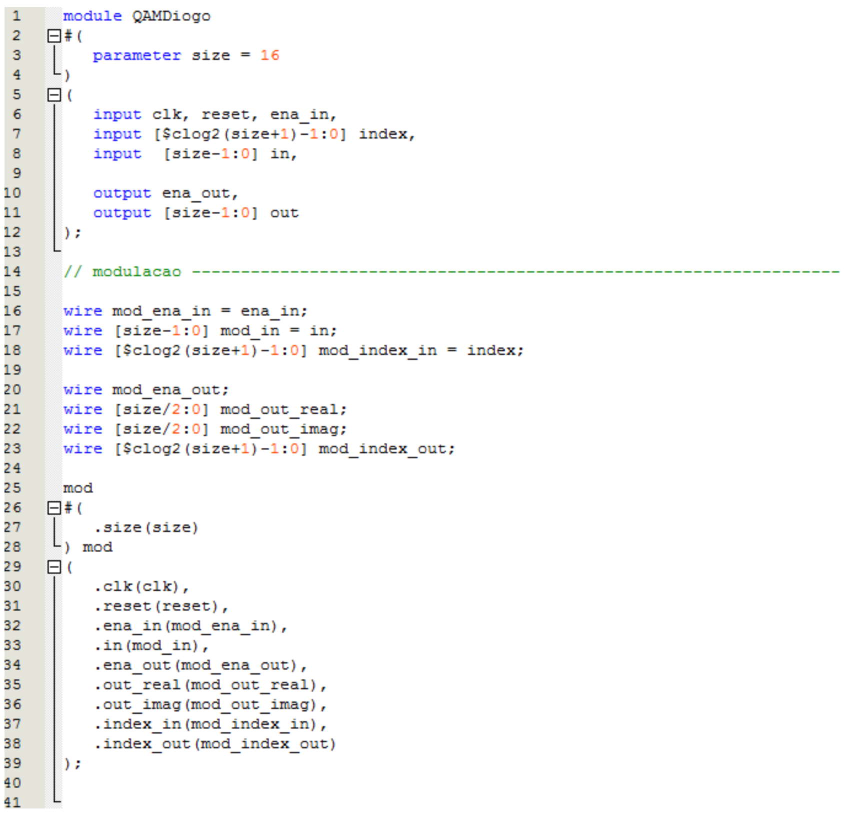
```verilog
begin
   // --- Primeiro pulso
   carry_real <= in_real[size];
   carry_imag <= in_imag[size];
   in_real_aux <= in_real[size]? {1'b0,~in_real[size:1]} : {1'b0,in_real[size:1]};
   in_imag_aux <= in_imag[size]? {1'b0,~in_imag[size:1]} : {1'b0,in_imag[size:1]};
   index_in_real_aux <= index_in[$clog2(2*size+1)-1:1] + index_in[0];
   index_in_imag_aux <= index_in[$clog2(2*size+1)-1:1];
   desl_real <= (1'b1 << (index_in[$clog2(2*size+1)-1:1] + index_in[0]));
   desl_imag <= (1'b1 << (index_in[$clog2(2*size+1)-1:1]));

   // --- Segundo pulso
   carry_real_reg <= carry_real;
   carry_imag_reg <= carry_imag;
   index_out_real_aux <= index_in_real_aux;
   index_out_imag_aux <= index_in_imag_aux;
   if(in_imag_aux >= (desl_imag >> 2'd1))
   begin
      if(in_real_aux >= (desl_real >> 2'd3))
      begin
         out_real_aux <= in_real_aux + (desl_real >> 2'd2);
         out_imag_aux <= (desl_imag) - in_imag_aux - 1'b1;
      end
      else
      begin
         out_real_aux <= (desl_real >> 1'b1) - in_real_aux - 1'b1;
         out_imag_aux <= in_imag_aux - (desl_imag >> 1'b1);
      end
   end
   else
   begin
      out_real_aux <= in_real_aux;
      out_imag_aux <= in_imag_aux;
   end

   // --- Último pulso
   // Define se e negativo ou positivo de acordo com o bit de carry
   out_real <= carry_real_reg ? ~{out_real_aux[size-1:0],1'b1}+1'b1 : {out_real_aux[size-1:0],1'b1};
   out_imag <= carry_imag_reg ? ~{out_imag_aux[size-1:0],1'b1}+1'b1 : {out_imag_aux[size-1:0],1'b1};

   index_out      <= index_out_real_aux + index_out_imag_aux;
   aux[1]         <= ena_in;
   aux[0]         <= aux[1];
   ena_out        <= aux[0];
end
```

Figure 75: Inverse Position Switcher algorithm in Verilog.



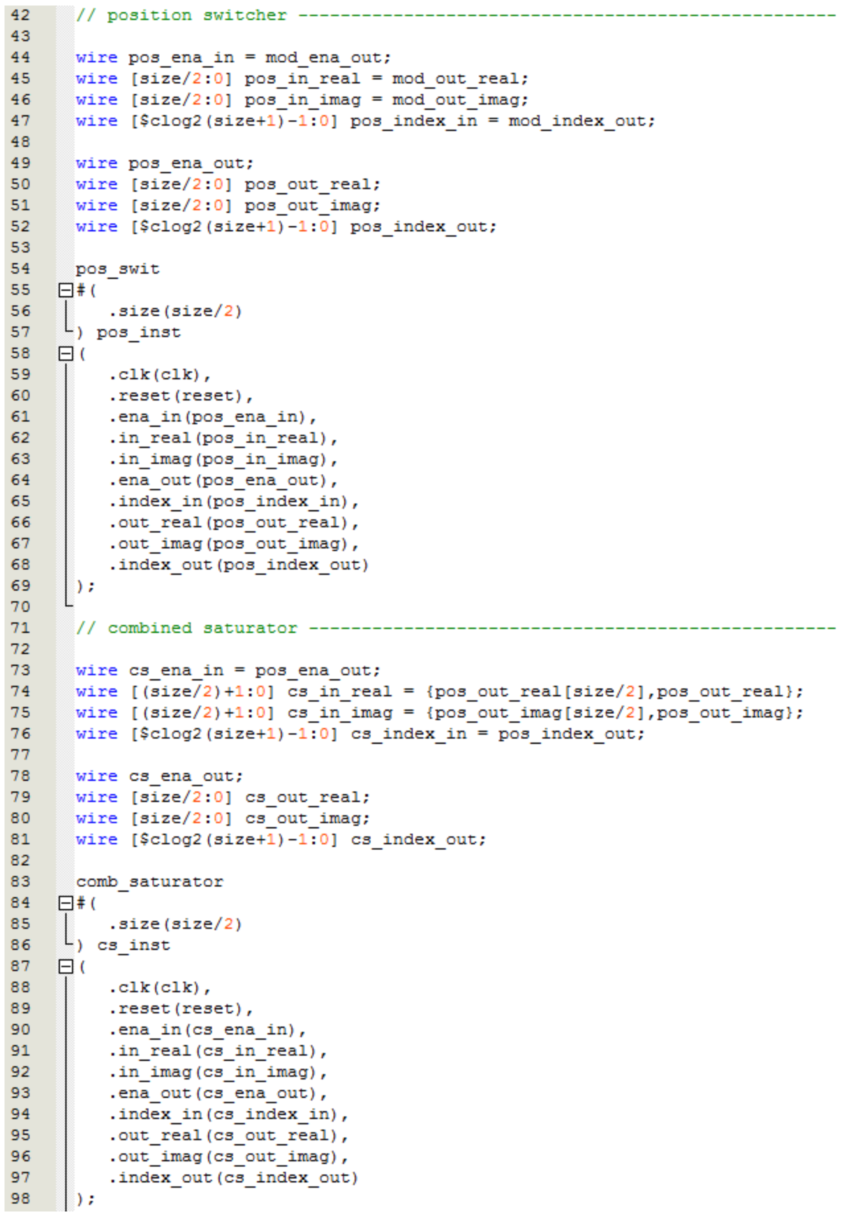
```verilog
begin
   // Salva ultimo bit de carry
   // --- Primeiro pulso
   carry_real   <= in_real[size/2];
   carry_imag   <= in_imag[size/2];
   out_aux_real <= in_real[size/2]? ~in_real[size/2:1] : in_real[size/2:1];
   out_aux_imag <= in_imag[size/2]? ~in_imag[size/2:1] : in_imag[size/2:1];

   // Binario para Gray
   // Define se e negativo ou positivo de acordo com o bit de carry
   // --- Ultimo pulso
   out_real <=  {(out_aux_real[size/2-2:0] ^ out_aux_real[size/2-1:1]),carry_real};
   out_imag <=  {(out_aux_imag[size/2-2:0] ^ out_aux_imag[size/2-1:1]),carry_imag};

   for (i=1'b0; i < size ; i = i + 2'd2 )
   begin
      out[i]        <= out_real[(i/2'd2)];
      out[i+1'b1]   <= out_imag[(i/2'd2)];
   end

   // Controle de saida
   aux[1]      <= ena_in;
   aux[0]      <= aux[1];
   ena_out     <= aux[0];
end
```

Figure 76: Demodulation algorithm in Verilog.



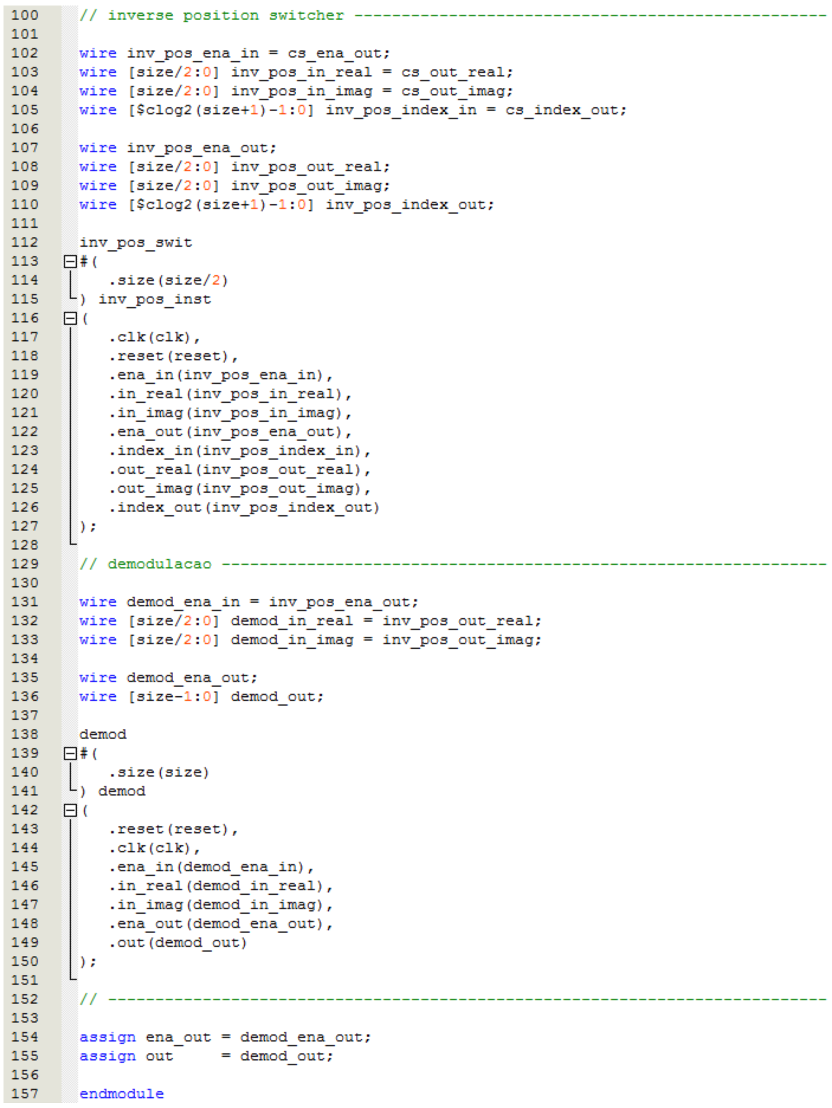
```verilog
module QAMDiogo
#(
    parameter size = 16
)
(
    input clk, reset, ena_in,
    input [$clog2(size+1)-1:0] index,
    input  [size-1:0] in,

    output ena_out,
    output [size-1:0] out
);

// modulacao ------------------------------------------------------------------

wire mod_ena_in = ena_in;
wire [size-1:0] mod_in = in;
wire [$clog2(size+1)-1:0] mod_index_in = index;

wire mod_ena_out;
wire [size/2:0] mod_out_real;
wire [size/2:0] mod_out_imag;
wire [$clog2(size+1)-1:0] mod_index_out;

mod
#(
    .size(size)
) mod
(
    .clk(clk),
    .reset(reset),
    .ena_in(mod_ena_in),
    .in(mod_in),
    .ena_out(mod_ena_out),
    .out_real(mod_out_real),
    .out_imag(mod_out_imag),
    .index_in(mod_index_in),
    .index_out(mod_index_out)
);

```

Figure 77: Top level algorithm in Verilog, part 1.



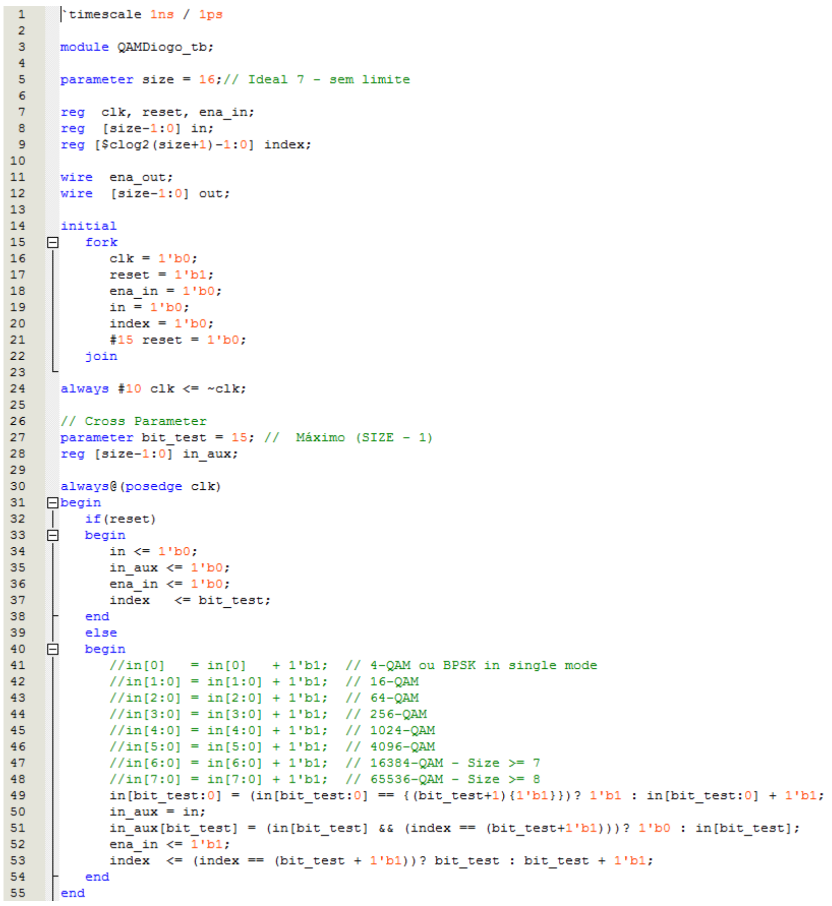
```verilog
// position switcher ---------------------------------------------------

wire pos_ena_in = mod_ena_out;
wire [size/2:0] pos_in_real = mod_out_real;
wire [size/2:0] pos_in_imag = mod_out_imag;
wire [$clog2(size+1)-1:0] pos_index_in = mod_index_out;

wire pos_ena_out;
wire [size/2:0] pos_out_real;
wire [size/2:0] pos_out_imag;
wire [$clog2(size+1)-1:0] pos_index_out;

pos_swit
#(
    .size(size/2)
) pos_inst
(
    .clk(clk),
    .reset(reset),
    .ena_in(pos_ena_in),
    .in_real(pos_in_real),
    .in_imag(pos_in_imag),
    .ena_out(pos_ena_out),
    .index_in(pos_index_in),
    .out_real(pos_out_real),
    .out_imag(pos_out_imag),
    .index_out(pos_index_out)
);

// combined saturator --------------------------------------------------

wire cs_ena_in = pos_ena_out;
wire [(size/2)+1:0] cs_in_real = {pos_out_real[size/2],pos_out_real};
wire [(size/2)+1:0] cs_in_imag = {pos_out_imag[size/2],pos_out_imag};
wire [$clog2(size+1)-1:0] cs_index_in = pos_index_out;

wire cs_ena_out;
wire [size/2:0] cs_out_real;
wire [size/2:0] cs_out_imag;
wire [$clog2(size+1)-1:0] cs_index_out;

comb_saturator
#(
    .size(size/2)
) cs_inst
(
    .clk(clk),
    .reset(reset),
    .ena_in(cs_ena_in),
    .in_real(cs_in_real),
    .in_imag(cs_in_imag),
    .ena_out(cs_ena_out),
    .index_in(cs_index_in),
    .out_real(cs_out_real),
    .out_imag(cs_out_imag),
    .index_out(cs_index_out)
);
```

Figure 78: Top level algorithm in Verilog, part 2.



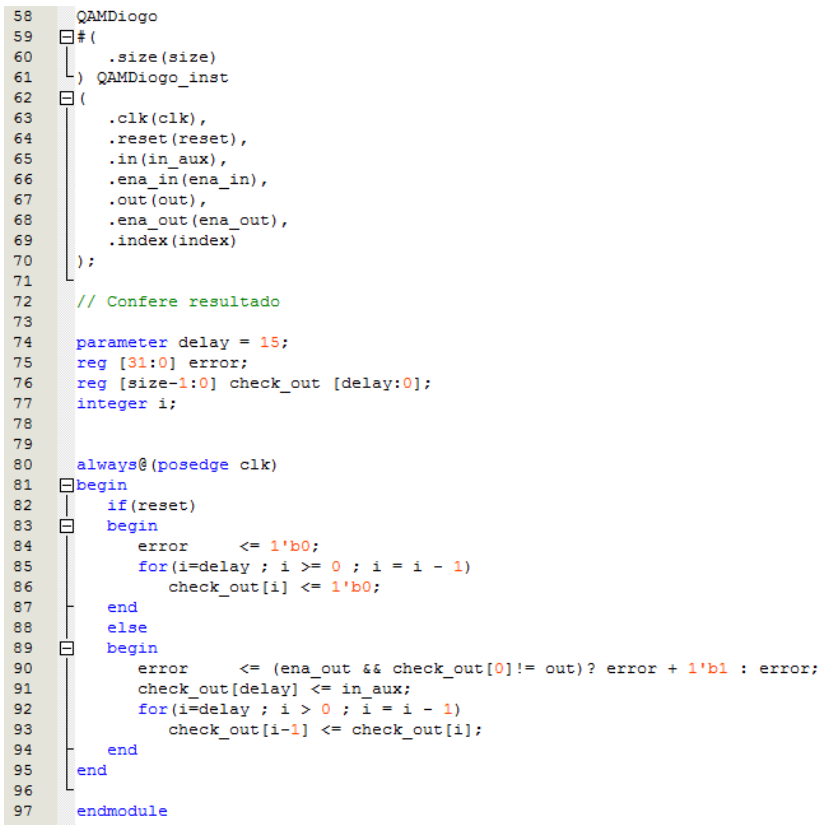
```verilog
// inverse position switcher -------------------------------------------------

wire inv_pos_ena_in = cs_ena_out;
wire [size/2:0] inv_pos_in_real = cs_out_real;
wire [size/2:0] inv_pos_in_imag = cs_out_imag;
wire [$clog2(size+1)-1:0] inv_pos_index_in = cs_index_out;

wire inv_pos_ena_out;
wire [size/2:0] inv_pos_out_real;
wire [size/2:0] inv_pos_out_imag;
wire [$clog2(size+1)-1:0] inv_pos_index_out;

inv_pos_swit
#(
    .size(size/2)
) inv_pos_inst
(
    .clk(clk),
    .reset(reset),
    .ena_in(inv_pos_ena_in),
    .in_real(inv_pos_in_real),
    .in_imag(inv_pos_in_imag),
    .ena_out(inv_pos_ena_out),
    .index_in(inv_pos_index_in),
    .out_real(inv_pos_out_real),
    .out_imag(inv_pos_out_imag),
    .index_out(inv_pos_index_out)
);

// demodulacao ----------------------------------------------------------------

wire demod_ena_in = inv_pos_ena_out;
wire [size/2:0] demod_in_real = inv_pos_out_real;
wire [size/2:0] demod_in_imag = inv_pos_out_imag;

wire demod_ena_out;
wire [size-1:0] demod_out;

demod
#(
    .size(size)
) demod
(
    .reset(reset),
    .clk(clk),
    .ena_in(demod_ena_in),
    .in_real(demod_in_real),
    .in_imag(demod_in_imag),
    .ena_out(demod_ena_out),
    .out(demod_out)
);

// ----------------------------------------------------------------------------

assign ena_out = demod_ena_out;
assign out     = demod_out;

endmodule
```

Figure 79: Top level algorithm in Verilog, part 3.



```verilog
`timescale 1ns / 1ps

module QAMDiogo_tb;

parameter size = 16;// Ideal 7 - sem limite

reg  clk, reset, ena_in;
reg  [size-1:0] in;
reg [$clog2(size+1)-1:0] index;

wire  ena_out;
wire  [size-1:0] out;

initial
   fork
      clk = 1'b0;
      reset = 1'b1;
      ena_in = 1'b0;
      in = 1'b0;
      index = 1'b0;
      #15 reset = 1'b0;
   join

always #10 clk <= ~clk;

// Cross Parameter
parameter bit_test = 15; //  Máximo (SIZE - 1)
reg [size-1:0] in_aux;

always@(posedge clk)
begin
   if(reset)
   begin
      in <= 1'b0;
      in_aux <= 1'b0;
      ena_in <= 1'b0;
      index   <= bit_test;
   end
   else
   begin
      //in[0]   = in[0]   + 1'b1;  // 4-QAM ou BPSK in single mode
      //in[1:0] = in[1:0] + 1'b1;  // 16-QAM
      //in[2:0] = in[2:0] + 1'b1;  // 64-QAM
      //in[3:0] = in[3:0] + 1'b1;  // 256-QAM
      //in[4:0] = in[4:0] + 1'b1;  // 1024-QAM
      //in[5:0] = in[5:0] + 1'b1;  // 4096-QAM
      //in[6:0] = in[6:0] + 1'b1;  // 16384-QAM - Size >= 7
      //in[7:0] = in[7:0] + 1'b1;  // 65536-QAM - Size >= 8
      in[bit_test:0] = (in[bit_test:0] == {(bit_test+1){1'b1}})? 1'b1 : in[bit_test:0] + 1'b1;
      in_aux = in;
      in_aux[bit_test] = (in[bit_test] && (index == (bit_test+1'b1)))? 1'b0 : in[bit_test];
      ena_in <= 1'b1;
      index  <= (index == (bit_test + 1'b1))? bit_test : bit_test + 1'b1;
   end
end
```

Figure 80: Testbench algorithm in Verilog, part 1.



```verilog
QAMDiogo
#(
   .size(size)
) QAMDiogo_inst
(
   .clk(clk),
   .reset(reset),
   .in(in_aux),
   .ena_in(ena_in),
   .out(out),
   .ena_out(ena_out),
   .index(index)
);

// Confere resultado

parameter delay = 15;
reg [31:0] error;
reg [size-1:0] check_out [delay:0];
integer i;


always@(posedge clk)
begin
   if(reset)
   begin
      error     <= 1'b0;
      for(i=delay ; i >= 0 ; i = i - 1)
         check_out[i] <= 1'b0;
   end
   else
   begin
      error     <= (ena_out && check_out[0]!= out)? error + 1'b1 : error;
      check_out[delay] <= in_aux;
      for(i=delay ; i > 0 ; i = i - 1)
         check_out[i-1] <= check_out[i];
   end
end

endmodule
```

Figure 81: Testbench algorithm in Verilog, part 2.

ANNEX C – Proposed Technique in MATLAB

The algorithm below was implemented on MATLAB language using MATLAB tool.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%                   Univesidade Federal de Juiz de Fora
%
%                        Faculdade de Engenharia
%
%               Programa de Pós Gradução em Engenharia Elétrica
%
%                    Mestrado em Sistemas Eletrônicos
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Data: Junho/2015
% Algoritmo: Transmissão M-QAM com mod/demod por transformação de bits
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%% Inicialização

tic
clc
clear all
close all

%% Parâmetros

K = 16;                                                % Número de bits
máximo por símbolo ( K>= 4, par )
%k = 16;                                               % Número de bits
transmitidos ( k <= K )
n = 1e5;                                               % Número mínimo de
Amostras de Entrada

qntminerros = 150;                                     % Número mínimo de
erros
int_max = 30;                                          % Número mínimo de
iterações

%% Dados de Entrada

N = 2^K;                                                % Ordem da maior
Modulação (N-QAM)
k = 2;
```

```
%BER_FINAL = zeros(K-1,32);
%EbN0_FINAL = zeros(K-1,32);
%ERROS = zeros(K-1,32);

while 1

    oo = 1;
    M = 2^k;                                                        % Ordem da 
Modulação (M-QAM)
    I = floor(k/2);
    R = k - I;

    N0_dB = 0;                                                      % Densidade 
Espectral do Ruído em dB

    Es      = 10^(0);                               % Energia do símbolo 
Unitária (em dB)
    Es_dB   = 10*log10(Es);

    if(mod(k,2) == 0)
        d = sqrt(6*Es/(M-1));                     % Fórmula da distância dos 
pontos da constelação QAM
    else
        d = sqrt(6*Es/(((31/32)*M)-1));
    end

    while 1

    N0      = 10.^(-N0_dB/10);                      % Densidade Espectral do 
Ruído
    EsN0_dB = 10.*log10(Es./N0);                    % Relação Energia do 
Símbolo pela Densidade Espectral do Ruído em dB
    EbN0    = EsN0_dB - 10*log10(k);                % Relação EbN0 para 
sinais complexos
    snr = EbN0+10*log10(k);                         % Relação Sinal-Ruído

    jj=0;

    erros_1 = 0;
    ber_2 = 0;

    while 1
        jj=jj+1;
    %% Gerando Dados de Entrada
        in     = randi([0 M-1], n, 1);            % Entrada Aleatória
        in_bin = dec2bin(in,K);

        % Divide parte real e imaginária
        indice_real_bin = in_bin(:,2:2:end);
        indice_imag_bin = in_bin(:,1:2:end);

        % Prepara entrada
        carry_real = -bin2dec(indice_real_bin(:,end));
        carry_imag = -bin2dec(indice_imag_bin(:,end));
        indice_real_bin(:,end) = 0;
        indice_imag_bin(:,end) = 0;

        % Conversão gray para binário
        indice_real_gray = bin2gray((bin2dec(indice_real_bin)), 'pam',
2^((K/2)-1));
        indice_imag_gray = bin2gray((bin2dec(indice_imag_bin)), 'pam',
2^((K/2)-1));

        if(k~=3)

            % Posições para Position Switcher QAM Cruzada
            cond_sat_x = (2^(R-2)) + (2^(I-2));
            saturated_in = find(((indice_real_gray)) >= cond_sat_x);

            cond_sat_y = (2^(I-2));
            saturated_in_reg_1  = find((indice_imag_gray) >= cond_sat_y);
            saturated_in_reg_2  = find((indice_imag_gray) <  cond_sat_y);

            indice_sat_1 = intersect(saturated_in,saturated_in_reg_1);
            indice_sat_2 = intersect(saturated_in,saturated_in_reg_2);

            % Realiza position switcher 2D

            indice_real_gray(indice_sat_1) =
(indice_real_gray(indice_sat_1) - (2^(R-2)));
            indice_imag_gray(indice_sat_1) =  ((2^I) -
indice_imag_gray(indice_sat_1) - 1);

            indice_real_gray(indice_sat_2) =  ((2^(R-1)) -
indice_real_gray(indice_sat_2) - 1);
            indice_imag_gray(indice_sat_2) =
(indice_imag_gray(indice_sat_2) + (2^(I-1)));
        end

        % Compensa o sinal
        valor_real = (((indice_real_gray .* ((2*carry_real)+1)) +
carry_real)*2) + 1;
        valor_imag = (((indice_imag_gray .* ((2*carry_imag)+1)) +
carry_imag)*2) + 1;
        %pontos_constelacao = 0:N-1;

        %% Mapeamento QAM

        out_mod = valor_real + 1j*valor_imag;
        %figure(1)
        %plot(valor_real,valor_imag,'o','color','blue');
        x_mod  = d*out_mod/2;

        % Ruído AWGN com distribuição normalizada e PSD variável
        v = normrnd(0,sqrt(N0/2),1, n) + 1j*normrnd(0,sqrt(N0/2),1, n);

        %simfs = 1.2e6;
        %Temp_ruido = 1e4/simfs;
        %v = (Ruido_AIGN(simfs, Temp_ruido, N0/2) + 1j*Ruido_AIGN(simfs,
Temp_ruido, N0/2))';

        % Saída do Canal
        y = x_mod' + v;
        y_real = real(y)*2/d;
        y_imag = imag(y)*2/d;

        y2 = (y_real) + 1j*(y_imag);

        %figure(2)
        %plot(y_real,y_imag,'o','color','blue');
        % ======================== Demodulação M-QAM
==============================

        % A demodulação analisa separadamente as partes reais e
imaginárias
        % recebidas e demapeia o sinal originalmente enviado.

       % Posições para Combined Saturator
        cond_sat = (2^(I));
        saturated_1 = find((abs(real(y2))) >= cond_sat);
        saturated_2 = find((abs(imag(y2))) >= cond_sat);
        sat_x = find(abs(real(y2)) < abs(imag(y2)));
        sat_y = find(abs(real(y2)) >= abs(imag(y2)));

        indice_sats  = intersect(saturated_1,saturated_2);
        indice_sat_x = intersect(indice_sats,sat_x);
        indice_sat_y = intersect(indice_sats,sat_y);

        % Realiza Combined Saturator
        y2(indice_sat_x) =  (min(cond_sat-1, max(-cond_sat+1,
real(y2(indice_sat_x))))) + 1j*imag(y2(indice_sat_x));
        y2(indice_sat_y) =  real(y2(indice_sat_y)) + 1j*(min(cond_sat-1,
max(-cond_sat+1, imag(y2(indice_sat_y)))));
        %figure(3)
        %plot(real(y2),imag(y2),'o','color','blue');

        % Divide, satura e trunca parte real e imaginária
        if(k==3)
            cond_sat_rp = (2^(R));
        else
            cond_sat_rp = (2^(R-1)) + (2^(I-1));
        end
        indice_real = (floor((min(cond_sat_rp-1, max(-cond_sat_rp+1,
real(y2'))))/2)*2)+1;
        if(k==3)
            cond_sat_rp = (2^(I));
        end
        indice_imag = (floor((min(cond_sat_rp-1, max(-cond_sat_rp+1,
imag(y2'))))/2)*2)+1;


        %figure(4)
        %plot(indice_real,indice_imag,'o','color','blue');
        carry_real  = -(real(y2') < 0.000);
        carry_imag  = -(imag(y2') < 0.000);

        % Inverte bits e divide por 2
        valor_real = floor(((indice_real .* ((2*carry_real)+1)) +
carry_real)./2);
        valor_imag = floor(((indice_imag .* ((2*carry_imag)+1)) +
carry_imag)./2);

        %figure(1)
        %plot(valor_real,valor_imag,'o','color','blue');
        if(k>4)
            % Posições para Inverse Position Switcher QAM Cruzada
            cond_sat_y = (2^(I-1));
            saturated_out = find(((valor_imag)) >= cond_sat_y);

            cond_sat_x = (2^(R-3));
            saturated_out_reg_1  = find((valor_real) >= cond_sat_x);
            saturated_out_reg_2  = find((valor_real) <  cond_sat_x);

            indice_sat_out_1 =
intersect(saturated_out,saturated_out_reg_1);
            indice_sat_out_2 =
intersect(saturated_out,saturated_out_reg_2);

            % Realiza inverse position switcher 2D
            valor_real(indice_sat_out_1) =  (valor_real(indice_sat_out_1)
+ (2^(R-2)));
            valor_imag(indice_sat_out_1) =  ((2^I) -
valor_imag(indice_sat_out_1) - 1);

            valor_real(indice_sat_out_2) =  ((2^(R-1)) -
valor_real(indice_sat_out_2) - 1);
            valor_imag(indice_sat_out_2) =  (valor_imag(indice_sat_out_2)
- (2^(I-1)));
        end

        %figure(2)
        %plot(valor_real,valor_imag,'o','color','blue');

        % Conversão binário para gray
        indice_real_gray = gray2bin(((valor_real)), 'pam', 2^((K/2)-1));
        indice_imag_gray = gray2bin(((valor_imag)), 'pam', 2^((K/2)-1));

        % Compensa o sinal
        valor_final_real = 2*indice_real_gray - carry_real;
        valor_final_imag = 2*indice_imag_gray - carry_imag;

        out_real_bin = dec2bin(valor_final_real',K/2);
        out_imag_bin = dec2bin(valor_final_imag',K/2);

        %out          = bin2dec([out_real_bin out_imag_bin]);
        out_aux(:,2:2:K) = out_real_bin;
        out_aux(:,1:2:K) = out_imag_bin;
        out2           = bin2dec(out_aux);
        out            = rem(out2,M);

        [nbits, ber_est] = biterr(in, out);                % Cálculo da
BER
        ber               = ber_est;                        % Valor da
BER para cada relção EbN0
        erros_int         = length(find((in-out)~=0)) ;     % Número de
simbolos errados
        erros_1           = erros_1 + erros_int;
        ber_2             =  ber_2 + ber ;                       % BER
(Iterações e variações de EbN0)

        if (erros_1 >= qntminerros && jj >= int_max)
            break
        end

        if(jj >= 300)
            break
        end

    end

    N0_dB = N0_dB + 3;
    ber_final = ber_2/jj;
    BER_FINAL(k-1,oo) = ber_final
    EbN0_FINAL(k-1,oo) = EbN0;
    %ERROS(k-1,oo) = erros_1;

    if (BER_FINAL(k-1,oo) < 10^-5)
        break
    end

    oo = oo+1


    end

    k = k + 1
    if(k > K)
        break;
    end


end
ber = zeros(size(EbN0_FINAL));
for jj = 1 : 1 : 15
    if(EbN0_FINAL(jj,end)==0)
        aux = find((EbN0_FINAL(jj,:)) == 0);
    else
        aux = size(EbN0_FINAL,2)+1;
    end
    ber(jj,1:aux(1)-1)=berawgn(EbN0_FINAL(jj,1:aux(1)-
1),'qam',2.^(jj+1));
end
```

```
%EbNoLin = 10.^(EbN0_FINAL/10);
%ber2 = qfunc(sqrt(EbNoLin/2))/k;

%% Salvando Dados

String_aux = ['dados_qam_',num2str(N),'_diogo.mat'];
save(String_aux, 'EbN0', 'EbN0_FINAL', 'ber', 'BER_FINAL');

% =================== Plotagem da Curva BER x EbN0
========================

for jj = 1 : 15
    figure(jj)
    semilogy(EbN0_FINAL(jj,:), ber(jj,:), EbN0_FINAL(jj,:),
BER_FINAL(jj,:));

    if(EbN0_FINAL(jj,end)==0)
        aux = find((EbN0_FINAL(jj,:)) == 0);
    else
        aux = size(EbN0_FINAL,2)+1;
    end

    xlim([0,EbN0_FINAL(jj,aux(1)-1)]);
    ylim([.1E-5, 1]);
    xlabel('Eb/N0');ylabel('BER');
    legend('BER teórica','BER estimada')
    grid;
end

% Tempo despendido
tempo = toc
```