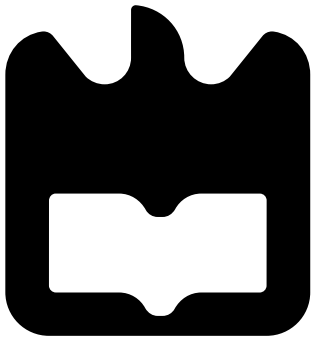
Universidade de Aveiro
Departamento de
Eletrónica, Telecomunicações e Informática
2015

Hélder Filipe
Pereira Machado

Implementação em FPGA de um Modem QPSK



Hélder Filipe Pereira Machado

Implementação em FPGA de um Modem QPSK

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Arnaldo Silva Rodrigues de Oliveira e do Professor Doutor Nuno Miguel Gonçalves Borges de Carvalho, Professores do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president	**Professor Doutor Manuel Alberto Reis de Oliveira Violas** Professor Auxiliar, Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)
vogais / examiners committee	**Doutor Nelson José Valente Silva** Investigador doutorado, PT Inovação e Sistemas (arguente)
	Professor Doutor Arnaldo Silva Rodrigues de Oliveira Professor Auxiliar, Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Queria agradecer à minha família, principalmente aos meus pais e à minha irmã por todo o apoio que sempre me deram.

Aos meus amigos que partilharam esta jornada a meu lado e cuja amizade nunca esquecerei.

Ao João Silva e a todos os que comigo partilharam o laboratório de Radio System por todo o apoio e disponibilidade, bem como todos os momentos de descontração e lazer que me proporcionaram.

Ao meu orientador, Arnaldo Oliveira, e co-orientador, Nuno Borges de Carvalho, por me terem dado a oportunidade de aprender com eles.

Ao Instituto de Telecomunicações e à Universidade de Aveiro por todas as condições oferecidas para este trabalho se tornar numa realidade.

A todos um muito obrigado!

Plavras-Chave SDR, Modem, QPSK, ZedBoard, AD-FMCOMMS1-EBZ

Resumo

Esta dissertação insere-se num conjunto de trabalhos a decorrer no Instituto de Telecomunicações de Aveiro que tem como objetivo o desenvolvimento de um sistema de comunicação para um UAV. Neste sentido, apresenta a implementação e validação de um modem em banda base aberto e flexível implementado em FPGA, baseado em abordagem SDR, com possibilidade de integração no sistema de comunicação com o UAV.

Ao longo desta dissertação implementou-se, utilizando o MATLAB, um modem de modulação adaptável, ao qual foram integrados algoritmos de sincronismo e de correção de fase. Desta forma, foi possível realizar uma análise ao modelo comportamental dos vários constituintes do modem abstraindo-se dos tempos de atraso do processamento ou da precisão da representação dos dados, e assim simplificar a sua implementação em *hardware*.

Analisado o modelo comportamental do modem desenvolvido em MATLAB realizou-se a sua implementação em *hardware* para a modulação QPSK. A sua prototipagem foi realizada, com recurso à ferramenta computacional *Vivado Design Suite 2014.2*, utilizando o kit de desenvolvimento ZedBoard e o *frontend* AD-FMCOMMS1-EBZ.

O correto funcionamento dos módulos implementados em *hardware* foi posteriormente avaliado através de uma interface entre o MATLAB e a ZedBoard, sendo que, os resultados obtidos no modelo em MATLAB serviram como termo de comparação. Através da utilização desta interface é possível validar parte do modem implementado em FPGA, mantendo o restante processamento a ser realizado em MATLAB, validando assim os módulos em FPGA de uma forma isolada.

Keywords SDR, Modem, QPSK, ZedBoard, AD-FMCOMMS1-EBZ

Abstract

This thesis is part of a series of work in progress at the Instituto de Telecomunicações of Aveiro which aims to develop a communication system for an UAV. Thereby, it presents the implementation and validation of a flexible and open baseband modem implemented in FPGA, based on SDR approach, that can be integrated in the communication system with the UAV.

Throughout this thesis was implemented an adaptive modulation modem and have been integrated timing recovery and phase correction algorithms, using the MATLAB. Thus, it is possible to perform an analysis of the behavioral of the several modem components abstracting the delay times of the processing or the precision of data representation, simplifying its hardware implementation.

Analyzed the modem behavior developed in MATLAB was performed its hardware implementation for the QPSK modulation. Its prototype was done using the computational tool *Vivado Design Suite 2014.2*, the ZedBoard development kit and the frontend AD-FMCOMMS1-EBZ.

The proper function of the modules implemented in hardware was evaluated by an interface between MATLAB and ZedBoard. This interface allows an individual validation of the implemented modem components in FPGA, keeping the remaining processing to be done in MATLAB and compared with the ones obtained in MATLAB.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

A/D	Analógico para Digital
ADC	Analog-to-Digital Converter
ASIC	Application Specific Integrated Circuits
ASK	Amplitude-Shift Keying
BER	Bit Error Ratio
BPSK	Binary Phase Shift Keying
CR	Cognitive-Radio
D/A	Digital para Analógico
DAC	Digital-to-Analog Converter
DSP	Digital Signal Processor
EVM	Error Vector Magnitude
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSK	Frequency-Shift Keying
I	In-phase
I2C	Inter-Integrated Circuit
IQ	In-phase and Quadrature
ISI	Intersymbol Interference
MAC	Multiply Accumulate
O&M	Martin Oerder e Heinrich Meyr
PSK	Phase-Shift Keying
Q	Quadrature
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RF	Radio Frequency
RRC	Root Raised Cosine
SDR	Software Defined Radio

SNR	Signal-to-Noise Ratio
SoC	System on a Chip
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
VHDL	Very high speed integrated circuits Hardware Description Language

Capítulo 1

Introdução

1.1 Enquadramento

Esta dissertação de mestrado faz parte de um conjunto de trabalhos a decorrer no Instituto de Telecomunicações de Aveiro que tem como objetivo o desenvolvimento e implementação de um sistema de comunicações para *Unmanned Arial Vehicles* (UAVs). Neste sentido, apresenta a implementação e validação de um modem flexível e aberto, que permita a sua integração neste sistema.

A primeira tentativa de utilizar um avião não tripulado surgiu durante a Primeira Guerra Mundial com o intuito de voar contra um alvo e explodir a sua carga. Desde então a utilização destes dispositivos designados por, UAVs como são vulgarmente conhecidos, sofreram um crescimento e evolução sem precedentes para as mais variadas aplicações, tanto militares como civis. Missões de vigilância, cartografia, patrulha fronteiriça, reconhecimento, busca e salvamento, deteção de incêndios e imagem agrícola são apenas alguns exemplos das suas vastas aplicações.

Os UAVs tal como o próprio nome indica são veículos aéreos não tripulados, isto é, são aeronaves que voam de forma autónoma ou remotamente controlada.

Nos últimos anos tem-se assistido a uma grande procura por este tipo de tecnologia, o que contribuiu para um forte desenvolvimento nesta área. Um dos principais desafios tem sido a melhoria da autonomia e do alcance, para que os UAVs consigam voar maiores distâncias, com maior precisão e de forma cada vez mais autónoma.

Apesar do esforço em fazer o máximo do processamento no UAV, para diminuir a transferência de dados com a base de controlo, o sistema de comunicações continua a ser uma das suas grandes limitações, nomeadamente ao nível do alcance das suas comunicações.

Os UAVs controlados remotamente são o caso mais evidente desta limitação, pois como é manobrado remotamente, o seu alcance de voo está limitado ao alcance das comunicações. Mesmo em UAVs autónomos, que não necessitam de qualquer apoio para voar, a restrição da sua cobertura, isto é, da distância a que conseguem comunicar é um dos grandes problemas que enfrentam. Pois, o aumento e a variabilidade da latência compromete a sua realização em tempo real.

Esta dissertação de mestrado apresenta o desenvolvimento de um modem que, juntamente com um conjunto de trabalhos a decorrer no Instituto de Telecomunicações de Aveiro, vem combater este problema oferecendo uma maior liberdade aos UAVs através do desenvolvimento e implementação de um sistema de comunicação baseadas em abordagens *cognitive-*

radio e *all digital transceivers* que permitem aumentar a adaptabilidade e robustez das comunicações com o UAV.

1.1.1 Sistema de Comunicação

Ao longo da nossa história, a comunicação sempre desempenhou um papel importantíssimo e permitiu a evolução da sociedade humana até ao ponto como a conhecemos hoje em dia. Contudo o acesso à informação nem sempre esteve à distancia de um *click*, foi só na última metade do século XX que os sistemas de comunicação começaram a proliferar de uma forma global.

Um sistema de comunicação tem como objetivo transmitir uma mensagem entre um emissor e um recetor através de um meio físico, denominado de canal. De forma a obter uma comunicação eficiente e viável tanto o emissor como o recetor devem de tirar o máximo partido do canal, isto é, construir cadeias de transmissão e receção que adequem o sinal a transmitir a esse canal. Este processo é ilustrado no diagrama da Figura 1.1.

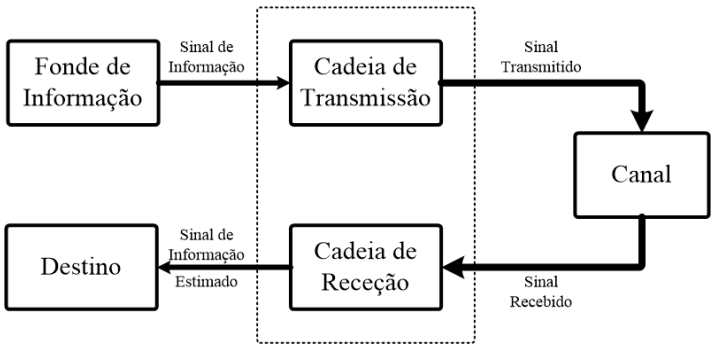


Figura 1.1: Diagrama de blocos de um sistema de comunicação

A **cadeia de transmissão** recebe o sinal com a informação proveniente da Fonte de Informação e codifica-o adequadamente para a transmissão pelo canal.

O **canal** é o meio físico por onde se vai propagar o sinal entre a fonte e o recetor, pode ser com ou sem fios. Por norma degrada o sinal transmitido adicionando-lhe atenuações, atrasos, distorções, interferências e ruído.

A **cadeia de Receção** tenta compensar a degradação de provocada pelo canal de forma a obter uma maior eficiência na recuperação da informação e reverte o processo da codificação realizado pela transmissão [Wes09] e [Hay01].

A constante necessidade de aumentar as distâncias de comunicação, aumentar as taxas de transmissão e manter a sua fiabilidade provocou uma enorme evolução na forma como as mensagens, ideias ou pensamentos são transmitidos. Muitos fatores contribuíram para essa evolução, mas o cerne foi a adoção de sistemas de comunicações digitais [NS09].

1.2 Motivação

Durante os últimos anos, os sistemas de rádio definidos por *software*, em inglês *Software Defined Radio* (SDR), proporcionaram uma enorme evolução na forma como os sistemas de rádio são implementados.

Os sistemas SDR utilizam implementações baseadas numa lógica programável e, deste modo, conseguem ajustar as suas funcionalidades com uma simples reprogramação. O SDR forum define-os da forma: "*Radio in which some or all of the physical layer functions are software defined*"[SDR]. Isto significa, rádio em que algumas, ou todas as funções da camada física são definidos por *software*, ou seja, um sistema em que o processamento é definido por *software* (*hardware* digital reconfigurável).

Na Figura 1.2 é representado o diagrama de blocos de um sistema SDR ideal, onde todo o processamento dos dados é realizado digitalmente, controlado por *software*. Apenas a filtragem e amplificação continua a realizar-se analogicamente.

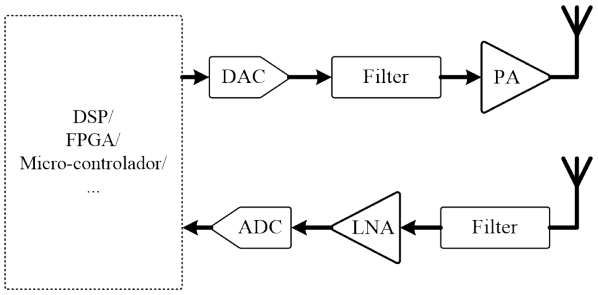


Figura 1.2: Diagrama de blocos de um sistema SDR ideal

Este tipo de sistemas permitem obter arquiteturas flexíveis e adaptáveis, possibilitando a utilização de diferentes formas de onda, larguras de banda ou até mesmo frequências de operação. Desta forma, sendo sistemas de rádio programável e de fácil atualização, permitem a aplicação do conceito de rádios cognitivos, em inglês *Cognitive-Radios* (CRs).

O SDR forum define os sistemas CR como sendo rádios que possuem conhecimento do seu ambiente e estado interno e podem tomar decisões sobre o seu comportamento operacional de rádio com base nessa informação e objetivos previamente definidos [SDR]. Ou seja, são sistemas que possuem a capacidade de avaliar a sua performance e com a devida gestão e controlo dos seus recursos, adaptarem os seus parâmetros de operação de forma a melhorar a sua eficiência.

1.3 Objetivos

Esta dissertação de mestrado tem como principal objetivo desenvolver um modem para integrar um sistema de comunicação rádio com um UAV que lhe permita um carácter aberto e flexível.

Pretende-se, em conformidade, atingir os seguintes objetivos intermédios:

- Implementação em MATLAB do dodelo comportamental de um modem adaptável a várias modulações, realizando a sua análise e respetiva validação, incluindo algoritmos de sincronismo e de correção de rotação.
- Implementação em FPGA de um modem QPSK, baseado em abordagem SDR;
- Validação e teste com *frontend* de forma a avaliar o desempenho de toda a camada física.

1.4 Organização do documento

Para além deste capítulo, onde é realizada uma breve introdução ao que se irá desenvolver no documento, este está dividido em mais 6 capítulos.

Capítulo 2: Conceitos Fundamentais
Neste capítulo é realizada uma analisa genérica aos sistemas de comunicação digitais, com uma descrição sucinta dos seus fundamentos teóricos e alguns problemas que lhes estão associados.

Capítulo 3: Modelação Comportamental e Simulação
É desenvolvido e validado um modem de modulações variáveis em *software*, utilizando o MATLAB, e realizada uma análise do seu comportamento para diferentes parâmetros.

Capítulo 4: Implementação em *Hardware*
Neste capítulo é explicado como cada módulo projetado e desenvolvido em *software* é implementado em *hardware*.

Capítulo 5: Prototipagem Laboratorial
Realizada uma breve descrição do kit de desenvolvimento utilizado e expõe-se o método empregue para a implementação do sistema neste kit.

Capítulo 6: Resultados Experimentais
Mostram-se os resultados obtidos utilizando um sistema de *lookback* e faz-se uma análise de desempenho comparando os dados enviados e recebidos através do MATLAB.

Capítulo 7: Conclusão e Trabalho Futuro
Por último, no Capítulo 7 expõem-se as conclusões finais desta dissertação e sugerem-se algumas propostas de como o trabalho pode ser otimizado.

Capítulo 2

Conceitos Fundamentais

Neste capítulo será apresentada uma breve definição de alguns conceitos básicos, mas essenciais no estudo de um sistema de comunicação.

2.1 Introdução

O objetivo de um sistema de comunicações é transmitir uma mensagem entre o emissor e o recetor através de um meio físico, denominado de canal. Este pode ser com ou sem fios e introduz distorção, ruído e interferências ao sinal transmitido, tal como demonstra o diagrama da Figura 2.1.

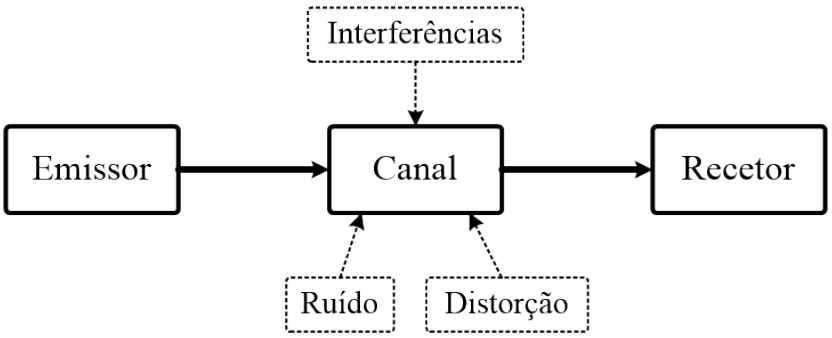


Figura 2.1: Elementos principais de um sistema de comunicação

Para uma comunicação eficiente e viável tanto o emissor como o recetor têm de tirar o máximo partido do canal e contrariar os seus efeitos.

Estes sistemas estão divididos em duas categorias principais, *Sistemas Digitais* e *Sistemas Analógicos*. A grande diferença entre eles está na forma de onda que utilizam, enquanto que os sistemas analógicos produzem ondas contínuas onde os recetores têm de lidar com infinitos conjuntos de formas de onda, os sistemas digitais produzem um número finito de formas de onda conhecidas à priori pelo recetor.

Os sistemas de comunicação digital têm uma cada vez uma maior importância e uma maior utilização. Dos vários os fatores que influenciam este crescimento destacam-se:

- Facilidade em codificar os sinais de forma a minimizar efeitos de ruído e interferências;
- Devido ao constante crescimento do processamento computacional nos mais variados dispositivos, grande parte da informação já se encontra em formato digital;
- Existência de técnicas de processamento digital já bem desenvolvidas. Um bom exemplo é a encriptação de mensagens, em comunicações facilmente intersectáveis como é o caso das comunicações sem fios a segurança é uma questão importantíssima, sendo o sinal de informação digital, podem-se utilizar os poderosos algoritmos de encriptação já existentes.

Para além claro, da sua performance, pois uma vez que o recetor tem que lidar com um número finito de formas de ondas, que conhece à priori, permite obter performances bastante sólidas para níveis de potência transmitida muito inferiores, comparando com os sistemas analógicos [Ric09].

Porém, possui também algumas desvantagens relativamente aos sistemas analógicos, entre elas o sincronismo, pois este é um pouco mais complexo e quando a relação sinal ruído, em inglês *Signal-to-Noise Ratio* (SNR), baixa um certo limite a qualidade do sinal passa de muito boa para muito má, enquanto que, nos sistemas analógicos esta transição é mais suave [Skl01].

2.2 Largura de Banda e Interferência Entre Símbolos

Nos sistemas de comunicações é vital que o sinal a transmitir ocupe uma largura de banda limitada de modo a evitar interferências nos canais adjacentes e conseguir uma utilização eficiente do espetro. Aplicando um filtro ao sinal antes da sua transmissão, consegue-se restringir a sua largura de banda, contudo a filtragem causa um aumento na resposta temporal, pode gerar sobreposições e consequentemente interferências entre símbolos, em inglês *Intersymbol Interference* (ISI), como se ilustra na Figura 2.2.

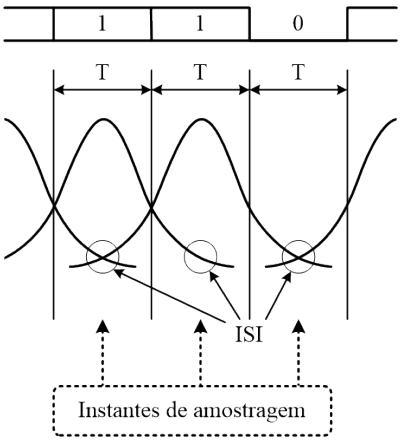


Figura 2.2: Interferência entre símbolos

Uma forma de limitar a largura de banda do sinal mantendo uma ISI nula é utilizando um filtro que respeite as características do critério de Nyquist. Este critério obriga a que as

respostas dos filtros $[h(t)]$ tenham um ISI nula, ou seja

$$h(t) = \begin{cases} 1, & t = 0 \\ 0, & t = \pm nT \end{cases} \tag{2.1}$$

Onde $T = 1/r$ e r o ritmo de transmissão.

Como Nyquist demonstrou, se a resposta do meio for igual à de um filtro passa baixo ideal, existirá um instante de tempo no intervalo do *bit* (no centro) em que a sua resposta é máxima, enquanto que, a resposta dos outros símbolos é nula. Tal como é ilustrado na Figura 2.3 através da resposta da função *sinc*.

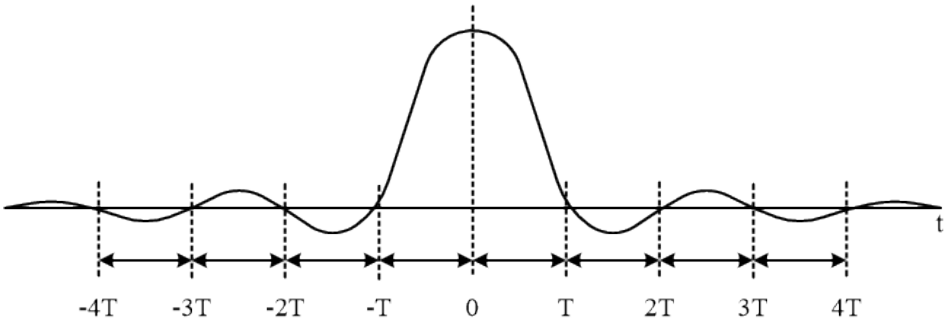


Figura 2.3: Função *sinc*

Um filtro que obedece a estes critérios e bastante utilizado em sistemas de comunicação é o cosseno elevado. A sua resposta impulsional e em frequência é expressa matematicamente pelas fórmulas (2.2) e (2.3), respectivamente.

$$s(t) = sinc(\pi t/T)\frac{\pi\alpha t/T}{1-(2\alpha t/T)^2} \tag{2.2}$$

$$S(f) = \begin{cases} T & |f| \leq \dfrac{1-\alpha}{2T} \\ \dfrac{T}{2}\left[1 + cos\left(\dfrac{\pi T}{\alpha}\left(|f| - \dfrac{1-\alpha}{2T}\right)\right)\right] & \dfrac{1-\alpha}{2T} \leq |f| \leq \dfrac{1+\alpha}{2T} \\ 0 & |f| \geq \dfrac{1+\alpha}{2T} \end{cases} \tag{2.3}$$

Onde T representa o período de símbolo e α o chamado fator de *roll-off*. Este fator varia entre $0 \leq \alpha \leq 1$ e determina a largura de banda do filtro. A sua resposta em frequência e impulsional é ilustrada na Figura 2.4 para vários valores de *roll-off*.

O filtro no emissor, como já foi referido, é essencial para restringir a largura de banda do sinal. Contudo, como também já foi referido, o canal de comunicações adiciona uma grande quantidade de ruído e distorção ao sinal, pelo que a implementação de um filtro no recetor

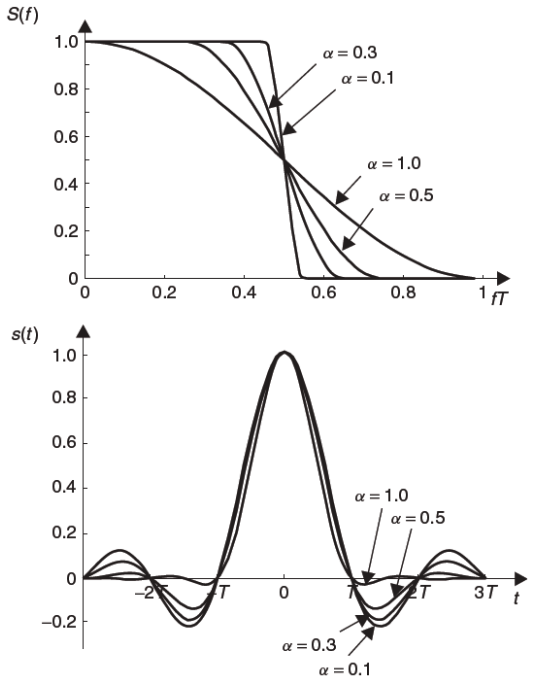


Figura 2.4: Resposta em frequência e impulsional do filtro cosseno elevado para diferentes valores de *roll-off* [Wes09]

---

para reduzir essas interferências e ruído também seria aconselhável. Posto isto, a solução ideal será a divisão do filtro de Nyquist entre o emissor e o recetor, normalmente utiliza-se o filtro raiz de cosseno elevado, em inglês *Root Raised Cosine* (RRC), e assim consegue-se restringir a largura de banda e filtrar as interferências introduzidas pelo canal mantendo uma ISI reduzida [Bat99].

2.3 Modulações Digitais

Nos sistema de comunicação digitais a informação é transferida na forma de impulsos, ou por uma sequências deles para ser mais especifico, designados por sinais em *banda base*, porém estes não podem ser transmitidos diretamente para o canal de comunicação. A sua frequência tem de ser elevada para a fazer coincidir com a gama de frequências adequadas ao canal.

Esta modulação obtém-se fazendo variar as características de uma onda sinusoidal, consoante a informação digital a transmitir. Desta forma, através da utilização de uma onda sinusoidal de alta frequência, designada por portadora, faz-se coincidir o sinal em *banda base* com a gama de frequências pretendidas.

Assim sendo, a informação pode ser transmitida na onda portadora $x_p(t)$, variando a amplitude A_p, frequência ω_p ou fase θ_p, tal como indica a fórmula (2.4).

$$x_p(t) = A_p cos(\omega_p t + \theta_p) \tag{2.4}$$

2.3.1 ASK

Na modelação por amplitude, designada por *Amplitude-Shift Keying* (ASK), tal como o próprio nome indica, a onda portadora é modulada alterando a sua amplitude consoante o sinal a modular, tal como demonstra a fórmula

$$x_p(t) = x(t)\ A_p\ cos(\omega_p t) \tag{2.5}$$

Onde, $x_p(t)$ é a onda portadora modulada, $A_p cos(\omega_p t)$ é a sinusoide portadora e $x(t)$ é o sinal a modular.

Na Figura 2.5 é ilustrado um exemplo de uma portadora ASK para uma dada sequência de entrada.

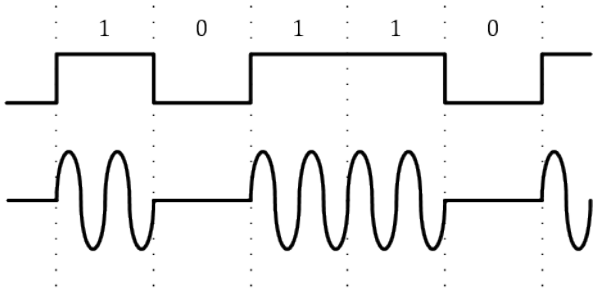


Figura 2.5: Portadora *ASK*

2.3.2 FSK

Na modulação *Frequency-Shift Keying* (FSK) a informação a transmitir é modulada na frequência da onda portadora, isto é, para cada valor da informação, 0's ou 1's, faz-se corresponder uma frequência diferente.

Desta forma, a onda portadora assume valores de frequências diferentes dependendo da informação do sinal a modular, tal como expressa a fórmula

$$\begin{array}{ll} x(t) = 0 & \longrightarrow \quad x_p(t) = cos(\omega_0 t) \\ x(t) = 1 & \longrightarrow \quad x_p(t) = cos(\omega_1 t) \end{array} \tag{2.6}$$

A Figura 2.6 apresenta um exemplo de uma portadora FSK para uma dada sequência de '0's e '1's.

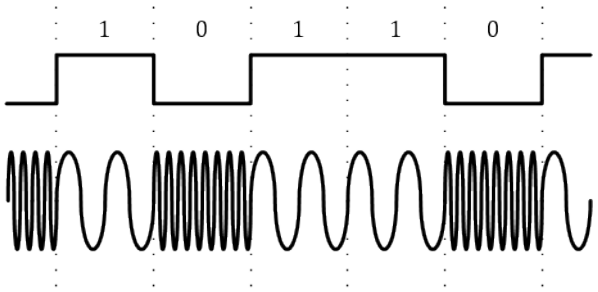


Figura 2.6: Portadora *FSK*

2.3.3 PSK

No caso da modulação *Phase-Shift Keying* (PSK) a informação é transmitida através da fase da onda portadora. Ou seja, diferentes valores do sinal a modular originam mudanças de fase na portadora, tal como se pode verificar pela fórmula

$$\begin{array}{ll} x(t) = 0 & \longrightarrow \quad x_p(t) = cos(\omega_p t) \\ x(t) = 1 & \longrightarrow \quad x_p(t) = -cos(\omega_p t) \end{array} \tag{2.7}$$

A Figura 2.7 apresenta um exemplo de portadora modulada com este tipo de modulação, para uma dada sequência de entrada.

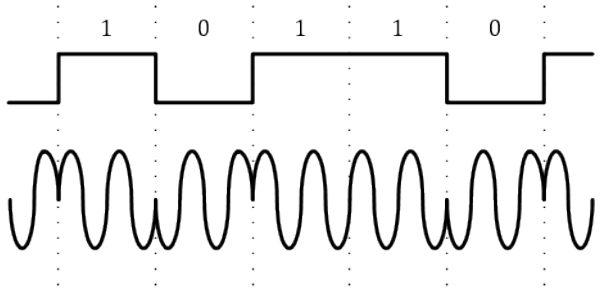


Figura 2.7: Portadora *PSK*

2.3.4 QPSK

A modulação *Quadrature Phase-Shift Keying* (QQPSK) é uma variante da modulação PSK. Explora a ortogonalidade do cosseno e seno para combinar a informação de dois *bits* do sinal e colocá-los na mesma portadora, desta forma, cada combinação de dois *bits* de entrada dá origem a uma forma de onda distinta.

Nesta modulação, a onda modulada é adquirida através da soma de duas portadoras com a mesma frequência mas fases desfasadas, tal como se pode verificar na fórmula (2.8), onde a cada portadora se faz corresponder a informação de um *bit*, ou através do atraso de fase de uma portadora, tal como é expresso pela fórmula matemática (2.9).

$$x_p(t) = a_i cos(\omega_p t) + b_i sen(\omega_p t) \qquad i = 1, 2, 3, 4 \tag{2.8}$$

Ou alternativamente por

$$x_p(t) = cos(\omega_p t + \theta_i) \qquad i = 1, 2, 3, 4 \tag{2.9}$$

Nas Tabelas 2.1 e 2.2 são apresentados os valores de θ_i e (a_i, b_i) para dois exemplo de modulação QPSK e na Figura 2.8 os seus respetivos diagramas de constelação, onde o eixo horizontal representa a componente em fase (I), do inglês *In-phase*, correspondente ao termo $cos(\omega_p t)$ e o eixo vertical representa a componente em quadratura (Q), do inglês *Quadrature*, correspondente ao termo $sen(\omega_p t)$.

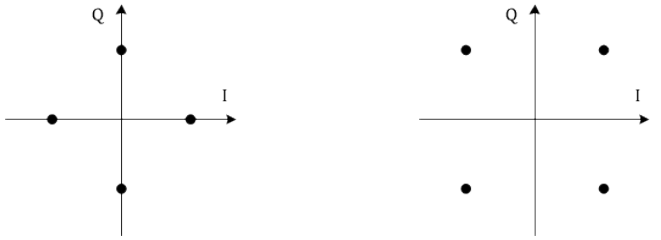
Na Figura 2.9 é apresentado um exemplo da sua forma de onda modulada para uma dada sequência de informação binária e as distribuições de fases da Tabela 2.1.

$\boldsymbol{\theta_i}$	0	$\frac{\pi}{2}$	π	$-\frac{\pi}{2}$
$\boldsymbol{(a_i, b_i)}$	$(1, 0)$	$(0, -1)$	$(-1, 0)$	$(0, -1)$

Tabela 2.1: Distribuição de fase da modulação QPSK

$\boldsymbol{\theta_i}$	$\frac{\pi}{4}$	$\frac{3\pi}{4}$	$-\frac{3\pi}{4}$	$-\frac{\pi}{4}$
$\boldsymbol{(\sqrt{2}a_1, \sqrt{2}b_1)}$	$(1, -1)$	$(-1, -1)$	$(-1, 1)$	$(1, 1)$

Tabela 2.2: Distribuição de fase da modulação QPSK



(a) Tabela 2.1 (b) Tabela 2.2

Figura 2.8: Diagrama de constelação QPSK

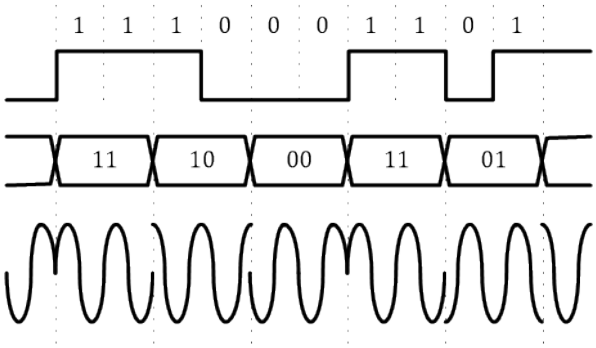


Figura 2.9: Portadora *QPSK*

Desta forma, são modulados dois *bits* por cada forma de onda, enquanto que nas modulações anteriores, cada forma de onda apenas representava um único *bit*. Assim sendo, esta modulação consegue duplicar o seu ritmo de transmissão mantendo a mesma largura de banda ou reduzir a sua largura de banda para metade mantendo o mesmo ritmo de transmissão, relativamente às modulações anteriormente apresentadas.

2.3.5 QAM

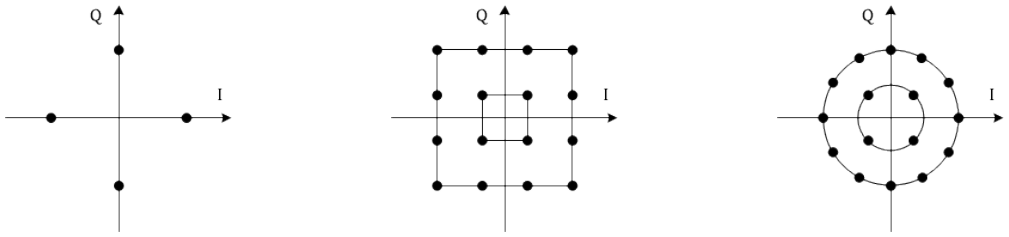
Na modulação *Quadrature Amplitude Modulation* (QAM) a informação digital a ser transmitida atua sobre as componentes em fase e quadratura das portadoras, tal como na modulação QPSK e atua também nas suas amplitudes, pelo que pode ser vista como uma combinação das modelações ASK e PSK.

Nesta modulação, as diferentes formas de onda necessárias para modular cada símbolos são definidas pela expressão

$$s_i(t) = a_i cos(\omega_p t) + b_i sen(\omega_p t) \qquad i = 1, 2, .., M \tag{2.10}$$

em que $M = 2^n$, n o número de *bits* de cada palavra, a_i e b_i são diferentes níveis de tensão gerados de acordo com a informação a modular.

Num modulador QAM, para modular uma palavra com 2 *bits* são gerados quatro pontos na constelação, correspondentes ao número de formas de onda distintas, por isso é chamado de 4-QAM. De igual modo, um modulador QAM que module palavras de 4 *bits* dará origem a uma constelação de 16 símbolos e diz-se um modulador 16-QAM, e assim sucessivamente. De notar que existem várias formas de se implementar este tipo de constelação, pelo que as suas constelações podem assumir vários formatos. Na Figura 2.10 estão representados alguns exemplos.



(a) 4-QAM (b) 16-QAM (c) 16-QAM

Figura 2.10: Diagrama de constelação QAM

O exemplo de uma forma de onda de uma portadora 16-QAM pode ser visualizada na Figura 2.11.

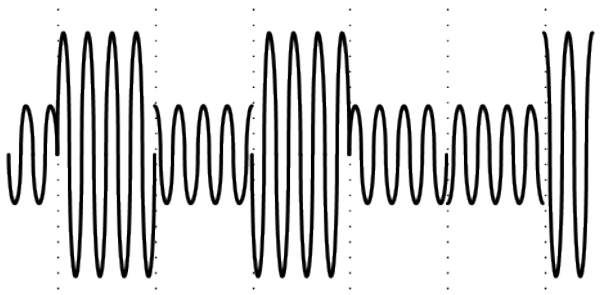


Figura 2.11: Portadora 16-QAM.

Da mesma forma que se advertiu na modulação anterior, aumentando do número de *bits* por forma de onda, consegue-se aumentar o ritmo de transmissão mantendo a mesma largura de banda, pois cada forma de onda contem a informação de mais *bits*.

Contudo, embora se consiga aumentar os ritmos de transmissão aumentando a ordem da modulação, se mantivermos a energia da constelação, os seus pontos ficam mais próximos e desta forma mais suscetíveis a originar erros quando afetados por ruídos.

Assim sendo, é necessário escolher entre o melhor compromisso de obter ritmos elevados e manter uma imunidade ao ruído aceitável.

2.4 Análise de Desempenho

Uma forma comum de avaliar o desempenho dos sistemas de comunicação é através da taxa de *bits* errados, ou em inglês *Bit Error Ratio* (BER), que é uma relação entre o número de *bits* errados (N_{err}) e o número total de *bits* (N_{total}) e calcula-se através da fórmula

$$BER = 100\ \frac{N_{err}}{N_{toltal}}\ (\%) \tag{2.11}$$

Assim sendo um sistema pode ser avaliado pelo BER recebido no recetor ou através do alcance máximo para um determinado BER.

Devido aos ruídos ou interferências, na receção os símbolos surgem deslocados da sua posição ideal do diagrama de constelação, contudo utilizando o BER, estas variações apenas seriam detetadas, quando estas fossem suficientes para provocar um erro de *bit*. Desta forma, avalia-se também o desempenho do sistema através do cálculo do erro de amplitude vetorial, vulgarmente conhecido pela sua sigla em Inglês, *Error Vector Magnitude* (EVM).

O EVM expressa a diferença entre a posição dos símbolos recebidos e a sua posição ideal, tal como ilustra a Figura 2.12 e é calculado através da fórmula:

$$\text{EVM} = 100\sqrt{\frac{\rho_{error}}{\rho_{ref}}}\ (\%) \tag{2.12}$$

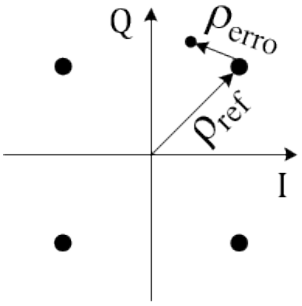


Figura 2.12: Definição do EVM numa constelação QPSK

2.5 Sincronismo

A sincronização é um dos ponto críticos de qualquer sistema de comunicação, uma vez que a sua falha pode ter efeitos catastróficos na sua performance [MD13].

Neste tipo de sistemas é essencial que o recetor esteja sincronizado com o sinal recebido. Assim sendo, o recetor precisa de determinar os instantes de início e fim de cada símbolo, a este processo dá-se o nome de *recuperação temporal* (*timing recovery*), bem como recuperar o atraso e variações de fase provocadas pelo canal e osciladores locais, a este processo dá-se o nome de *correção de fase* ou *correção de rotação*.

2.5.1 Recuperação Temporal

Para uma receção robusta e fiável o recetor tem de saber o exato momento em que cada impulso começa e acaba. Isto é, o relógio do recetor tem de ser continuamente ajustado para otimizar os instantes de amostragem do sinal e compensar as suas variações.

Tal como vimos no início deste capítulo, utilizando filtros RRC na transmissão e receção consegue-se eliminar a ISI. No entanto, esta situação apenas se verifica se a amostragem for realizada nos instantes corretos.

A amostragem do sinal é realizada durante o período de cada símbolo T, no entanto, o instante ideal deve ser o pico da sua resposta impulsiva, ou seja, onde o pulso tem as propriedades de um filtro de Nyquist. À diferença entre o instante de amostragem real e seu valor ótimo denomina-se desvio de sincronismo temporal (τ) e é ilustrado na Figura 2.13.

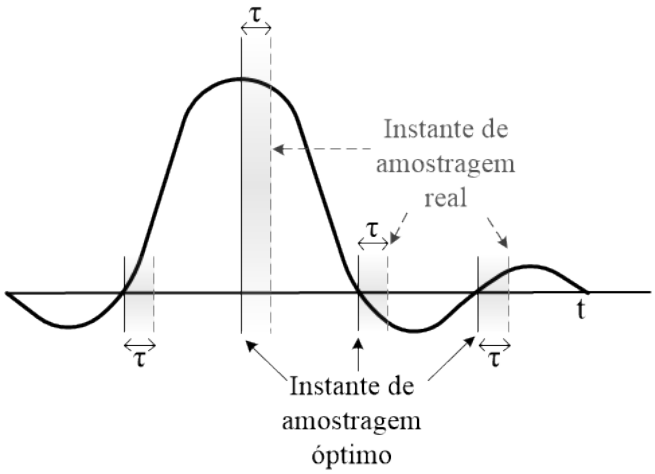


Figura 2.13: Instante de amostragem

Existem duas formas de lidar digitalmente com a questão do desvio temporal.

- ***Amostragem síncrona*** — consiste no ajuste do instante de amostragem de modo a que este coincida com o τ. Ou seja, o instante em que o sinal é amostrado de analógico para digital é controlado por um circuito de recuperação temporal;
- ***Amostragem assíncrona*** — a frequência de amostragem é constante e o valor amostrado é corrigido posteriormente por um interpolador digital.

A amostragem assíncrona depende somente de *software* e será a implementação utilizada nesta dissertação.

2.5.2 Correção de Fase

São vários os fatores que podem induzir em erros de fase no sinal recebido, entre eles, os osciladores locais, o ruído térmico e o próprio movimento relativo entre o emissor e o recetor (efeito de doppler) são os mais relevantes.

Assumindo que a fase da portadora é θ_p e a fase do oscilador local da receção é θ_o tem-se que o atraso de fase é dado por:

$$\theta = \theta_p - \theta_o \tag{2.13}$$

Este atraso de fase é responsável por uma rotação no diagrama de constelação dos símbolos recebidos. Na Figura 2.14 pode-se observar este efeito de rotação numa constelação QPSK, em que θ é o atraso de fase.

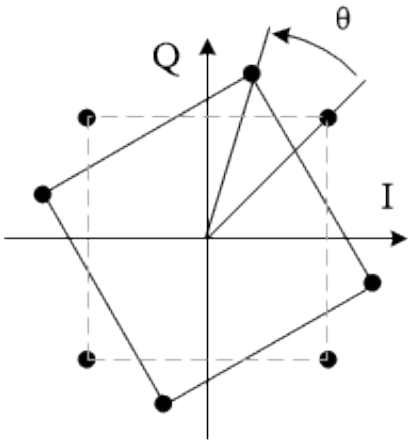


Figura 2.14: Rotação de fase numa constelação QPSK

A forma de implementar este tipo de algoritmos pode ser dividida em dois grupos principais:

- ***Feedback*** — compensa o desvio de sincronismo utilizando interações para determinar os instantes de amostragens;
- ***Feed-forward*** — compensa o instante de amostragem tendo apenas em conta as amostras atuais e o conhecimento das amostras anteriores.

Uma descrição detalhada sobre estes algoritmos pode ser encontrada em [MD13] e [PS15].

Ao longo deste capítulo foram analisados os fundamentos teóricos necessários à implementação de um modem digital, que se irá realizar no próximo capítulo.

Capítulo 3

Modelação Comportamental e Simulação

Neste capítulo é desenvolvido, um modem adaptável a várias modulações em MATLAB, ao qual se integram algoritmos de recuperação temporal e de correção de rotação. Realiza-se ainda uma análise do comportamento dos seus módulos para diferentes parâmetros.

3.1 Introdução

Expondo de uma forma muito sucinta, o sistema deve receber uma mensagem em formato digital de uma fonte de informação, processá-la através da cadeia de transmissão e enviá-lo para um canal de comunicação. Por sua vez, a cadeia de receção colhe o sinal do canal e devolve-o ao destino com a maior fiabilidade possível. Um esquemático desta visão geral está representado na Figura 3.1.

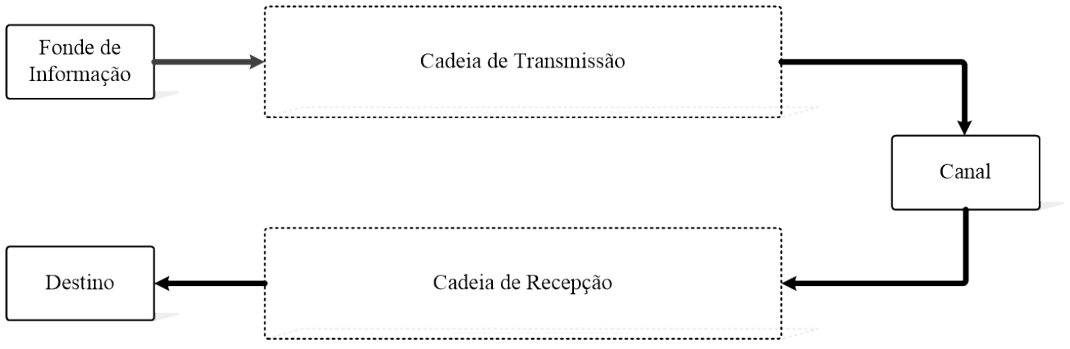


Figura 3.1: Visão geral do sistema

Para efeitos de desenvolvimento e respetivos testes, as cadeias de transmissão e receção estão no mesmo sistema, interligadas diretamente entre si. Isto significa que o sinal enviado pela cadeia de transmissão é recebido diretamente na cadeia de receção, este tipo de configuração é conhecido como *lookback*.

Desta forma, é possível testar e validar o correto funcionamento das várias cadeias utilizando para isso apenas um sistema físico.

3.2 Cadeia de Transmissão

A cadeia de transmissão recebe um sinal binário da fonte, e tem como função transmitir a informação nele contida de uma forma robusta, que atravesse eficientemente o canal e permita a sua leitura na cadeia de receção como menor número de perdas possíveis. O diagrama com os bloco projetados para a cadeia de transmissão está representado na Figura 3.2.

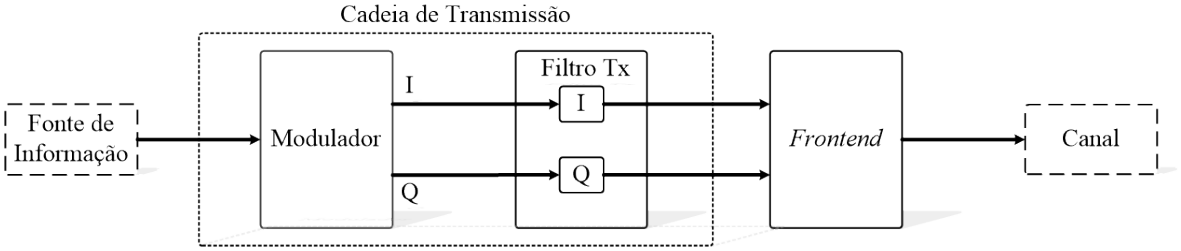


Figura 3.2: Cadeia de transmissão implementada em MATLAB

3.2.1 Modulador

Como foi referido na secção 2.3, a modulação do sinal numa portadora é vital para um bom desempenho da sua comunicação.

A modulação QPSK, para uma mesma largura de banda, permite transmitir ao dobro do ritmo, comparando com uma simples modulação PSK, vulgarmente conhecida como BPSK. Isto é, para uma mesma taxa de transmissão de símbolo (*symbol rate*) é enviada o dobro da informação, pois enquanto que, na modulação QPSK cada símbolo contem a informação de 2 *bits*, na modulação BPSK contem apenas a informação de um único *bit*.

Na Figura 3.3 são apresentados os diagramas de constelação de ambas as modulações onde se pode observar uma possível representação entre a informação a ser enviada e os seus símbolos.

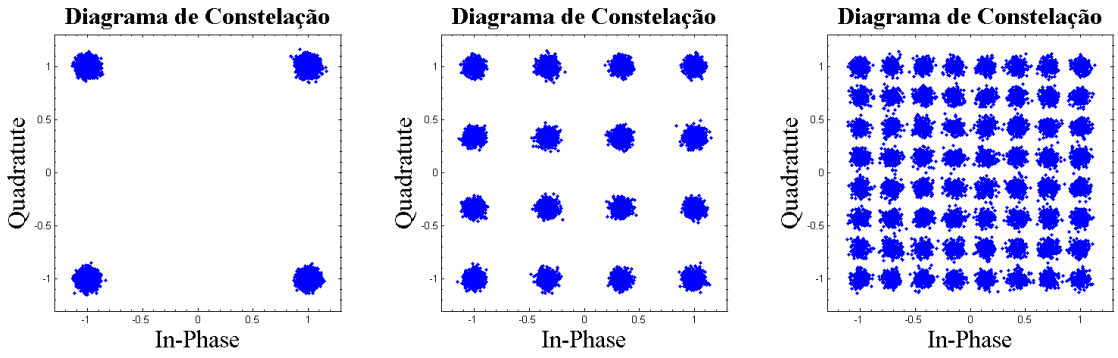
(a) BPSK (b) QPSK

Figura 3.3: Diagramas de constelação

Com as modulações 16-QAM ou 64-QAM ainda se conseguem ritmos de transmissão mais elevados, visto que a cada símbolo é atribuído a informação de 4 e 6 *bits*, respetivamente. Contudo, a sua imunidade ao ruído é bastante degradada, pois para a mesma energia da

constelação, os seus pontos ficam mais próximos entre si, o que torna mais complicada a tarefa do desmodulador em identificar qual a informação que o símbolo transporta.

Na Figura 3.4 podem ser observadas as constelação de modulações QPSK, 16-QAM e 64-QAM sobre as quais foi adicionadas o mesmo nível de ruído e podemos verificar que quanto mais elevada é a ordem da modulação, isto é, quantos mais pontos tem o seu diagrama de constelações, mais próximos esses pontos ficam entre si, e consequentemente maior é a probabilidade do desmodulador decidir pela informação que cada símbolo representa de forma errada.



(a) QPSK (b) 16-QAM (c) 64-QAM

Figura 3.4: Diagramas de constelação

Assim sendo, é necessário haver um compromisso entre o ritmo de transmissão e a imunidade ao ruído. Deste modo, a solução ideal seria um sistema que se adaptasse às condições do meio. Isto é, se numa comunicação o sinal recebido apresentar uma SNR relativamente baixa, a ordem da modulação utilizada pode ser aumentada e assim aumentar o ritmo de transmissão. O mesmo quando se verificar o oposto, ou seja, quando o sinal recebido apresentar um SNR elevada, voltar a baixar a ordem da constelação e assim aumentar a tolerância ao ruído. Desta forma, consegue-se aumentar significativamente a eficiência da modulação, pois transmite-se ao ritmo máximo que o canal permite, mantendo a mesma robustez.

O modulador implementado recebe o sinal da fonte de informação num formato binário e devolve a componente IQ correspondente à modulação selecionada.

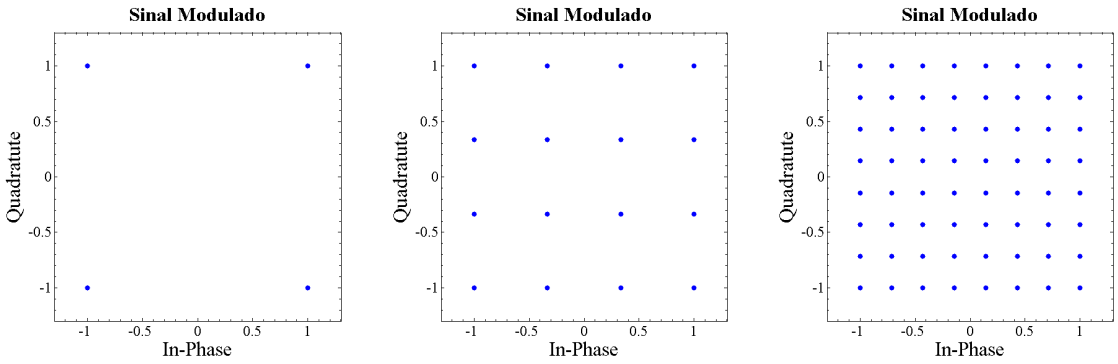
Este bloco foi implementado em MATLAB utilizando a função *qammod*. A Figura 3.5 ilustra uma sequencia aleatória de *bits* modulados, utilizando a codificação Gray, para as modulações QPSK, 16-QAM e 64-QAM.

3.2.2 Filtro Tx

Tal como foi referido na secção 2.2, utilizou-se o filtro RRC para realizar a modulação de pulso (*pulse shaping*) e restringir a largura de banda do sinal a transmitir. Este, juntamente com o filtro do recetor, formam um filtro de Nyquist que garante um baixo valor de ISI.

A largura de banda pode ser estimada através da fórmula

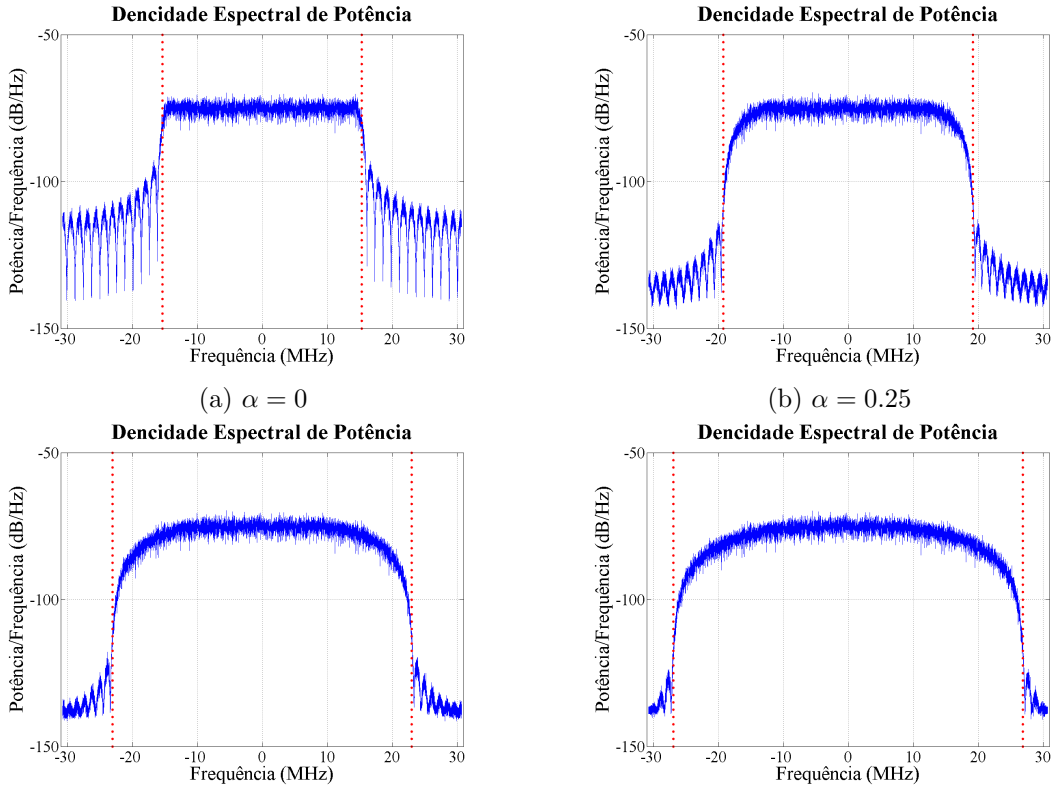
$$LB = r \cdot (1 + \alpha) \tag{3.1}$$



(a) Constelação QPSK (b) Constelação 16-QAM (c) Constelação 64-QAM

Figura 3.5: Diagrama de constelação QPSK

Onde α é o fator de *roll off* do filtro e r o seu ritmo de transmissão (*symbol rate*). Como se verifica na expressão matemática (3.1), a largura de banda é diretamente proporcional ao fator de *roll-off*. Na Figura 3.6 é ilustrado o seu efeito na largura de banda para um $r = 30.72M$ $(simbolos/seg)$.



(a) $\alpha = 0$ (b) $\alpha = 0.25$

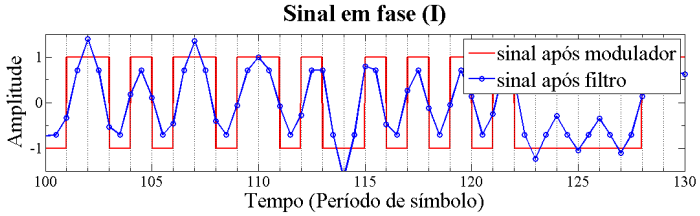
(c) $\alpha = 0.5$ (d) $\alpha = 0.75$

Figura 3.6: Efeito do fator de *roll-off* na largura de banda

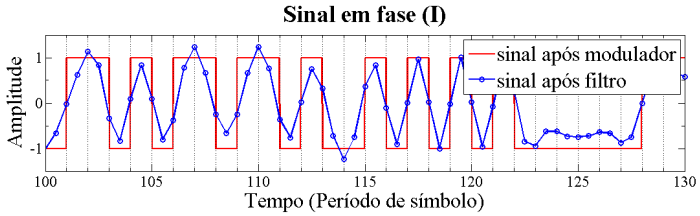
Os gráficos da Figura 3.6 foram obtidos através da função *pwelch* do MATLAB, para valores de *roll-off* de 0, 0.25, 0.5 e 0.75, em que o valor da largura de banda visualizado corres-

ponde ao estimado através da fórmula (3.1) e obtiveram-se larguras de banda de $30.72MHz$, $38.4MHz$, $46.08MHz$ e $53.76MHz$, respetivamente.

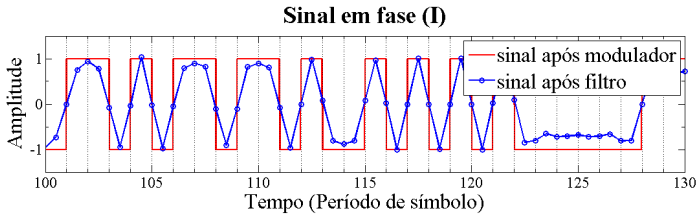
O efeito do *roll-off* na resposta temporal do filtro pode também ser observado na Figura 3.7. Nesta figura, utilizando um fator de sobre-amostragem igual a 2, pode-se visualizar o *pulse shaping* da componente em fase de uma modulação QPSK após a aplicação do filtro, comparada com a sua entrada no filtro, com a devida compensação de atraso do filtro.



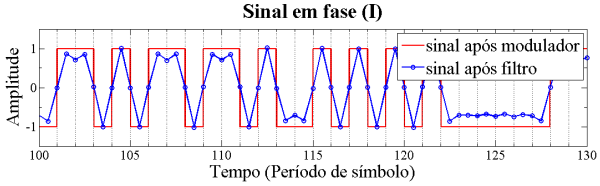
(a) $\alpha = 0$



(b) $\alpha = 0.25$



(c) $\alpha = 0.5$



(d) $\alpha = 0.75$

Figura 3.7: Resposta temporal do filtro

3.2.3 *Frontend*

O *frontend* inclui funcionalidades responsáveis pela conversão do sinal em banda base para RF. Isto é, coloca o sinal resultante da cadeia de transmissão à frequência da onda portadora.

Nesta dissertação, optou-se pela utilização de um sistema comercial para realizar esta operação, pelo que não é realizada nenhuma simulação para este circuito, contudo este tema será alvo de discussão na sessão 5.2.

3.3 Cadeia de Receção

O objetivo da cadeia de receção é recuperar a informação enviada pela fonte com a maior fidelidade possível. Desta forma, para além do filtro já referido, é também necessário compensar algumas distorções provocadas pelo canal. Para tal, implementa também alguns módulos adicionais para fazer essas compensações. Um diagrama da cadeia de receção projetada está ilustrado na Figura 3.8.

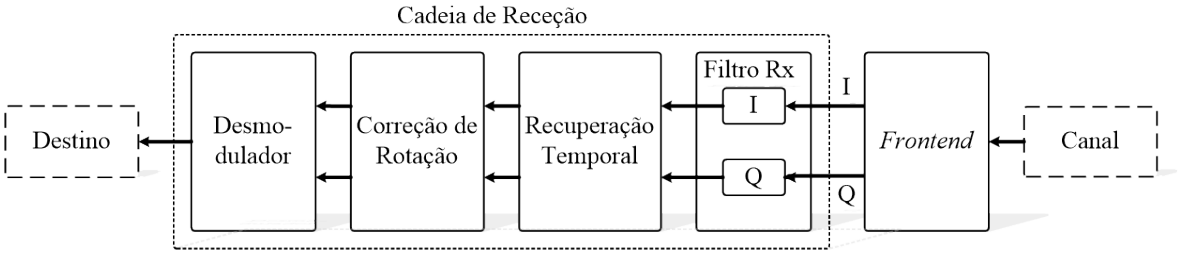
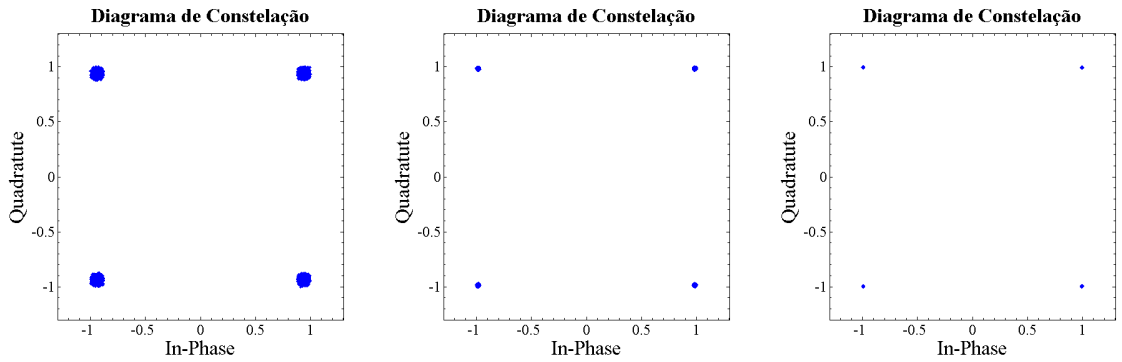


Figura 3.8: Cadeia de receção implementada em MATLAB

3.3.1 Filtro Rx

O ruído produzido pelo canal de comunicação é um dos grandes fatores que degrada substancialmente a performance dos sistemas de comunicação. Desta forma, torna-se essencial a utilização de um filtro que minimize este problema. Tal como mencionado nas secções anteriores, o filtro utilizado é um RRC, que juntamente com o filtro da transmissão formam um filtro de Nyquist. Para esta condição se verificar ambos os filtros devem ser projetados com os mesmos parâmetros.

Os filtros são projetados com 48 coeficientes, pois este valor permite um rigor razoável e é próximo do que é possível implementar em *hardware*. O fator de *roll-off* é obtido de forma iterativa, de modo a obter um valor de EVM aceitável, para uma largura de banda o mais estreita possível. A Figura 3.9 ilustra o resultado de algumas das iterações para uma modulação QPSK. Em que se obteve valores de EVM de 2.06%, 0.55% e 0.16% para fatores de *roll-off* de 0.1, 0.2 e 0.3, respetivamente.



(a) $\alpha = 0.1$; EVM = 2.06% (b) $\alpha = 0.2$; EVM = 0.55% (c) $\alpha = 0.3$; EVM = 0.16%

Figura 3.9: Efeito do fator de *roll-off* em constelações QPSK

Os filtros da transmissão e receção foram projetado em MATLAB utilizando a função *rcosdesign* e implementado com a função *upfirdn*. As especificações são praticamente as mesmas, à exceção do número de amostras por símbolo, como descrevem as Tabela 3.1 e 3.2.

rcosdesign			***upfirdn***		
Roll-off (beta)	:	0.22	*upsample*	:	2
Número de símbolos (*span*)	:	24	*downsample*	:	1
Amostras por símbolo (*sps*)	:	2			
Forma (*shape*)	:	'sqrt'			

Tabela 3.1: Especificações do filtro RRC da transmissão implementado em MATLAB

rcosdesign			***upfirdn***		
Roll-off (beta)	:	0.22	*upsample*	:	2
Número de símbolos (*span*)	:	24	*downsample*	:	1
Amostras por símbolo (*sps*)	:	4			
Forma (*shape*)	:	'sqrt'			

Tabela 3.2: Especificações do filtro RRC da receção implementado em MATLAB

3.3.2 Recuperação Temporal

De forma a combater o problema de sincronismo é necessário um circuito de recuperação temporal para estimar e compensar o instante de atraso (τ).

O algoritmo escolhido para implementar este circuito é um algoritmo *feed-forward* e foi proposto por Martin Oerder e Heinrich Meyr (O&M) em [OM88]. A sua expressão matemática é dada pela fórmula (3.2).

$$\tau = -\frac{T}{2\pi} arg \left\{ \sum_{k=0}^{NL_0-1} \mid x(kT_s) \mid^2 e^{-j2\pi k/N} \right\} \tag{3.2}$$

Em que L_0 representa a janela de observação utilizada, T o período de símbolo, T_s o período de amostragem e N o fator de sobre-amostragem (T/T_s). Este algoritmo necessita de 4 amostras por símbolo pelo que $N = 4$. O seu diagrama de blocos pode ser observado na Figura 3.10.

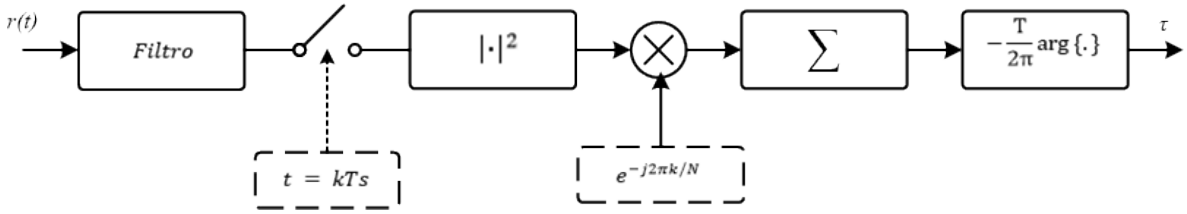


Figura 3.10: Diagrama de blocos do recuperador temporal O&M (imagem editada de [MD13])

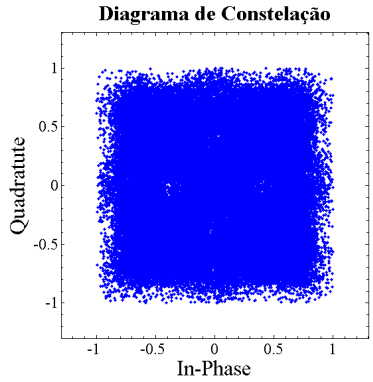
Para implementação do circuito de recuperação foi criada a rotina "*oerderMeyr(signal, L, T, T_s)*" em MATLAB que implementa o algoritmo proposto por O&M. Os parâmetros de entrada e saída desta rotina são apresentados na Tabela 3.3.

Entradas			Saídas		
signal	:	sinal a recuperar	***timming_rec***	:	sinal recuperado
L	:	janela de observação (L_0)			
T	:	período de simbolo			
$\boldsymbol{T_s}$	:	período de amostragem			

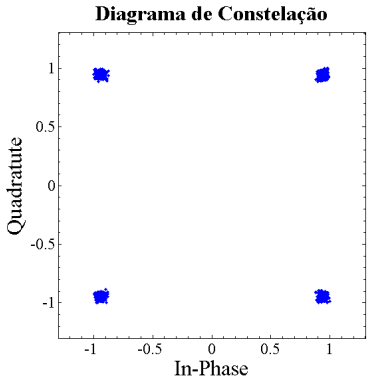
Tabela 3.3: Parâmetros da rotina *oerderMeyr* implementada em MATLAB

Esta rotina recebe como entrada o sinal a recuperar com uma sobre-amostragem de 4 amostras por símbolo, calcula o τ e através de um interpolador, estima qual a amostra correta.

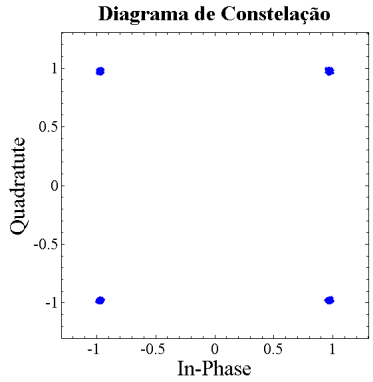
Analisando a Figura 3.11, pode-se observar em 3.11a) a constelação do sinal à entrada da rotina, gerado pelos filtros de transmissão e receção e portanto com quatro amostras por símbolo, e em (3.11b), (3.11c), e (3.11d) o sinal recuperado pela rotina, para várias janelas de observação. Obtiveram-se valores de EVM de 0.96%, 0.75% e 0.64% para $L = 128$, 256 e 512. respetivamente.



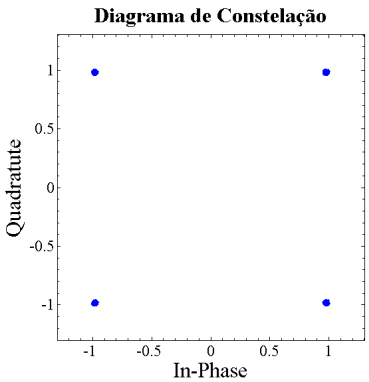
(a) Constelação antes da recuperação temporal



(b) Constelação para um $L = 128$, EVM = 0.96%



(c) Constelação para um $L = 256$, EVM = 0.75%

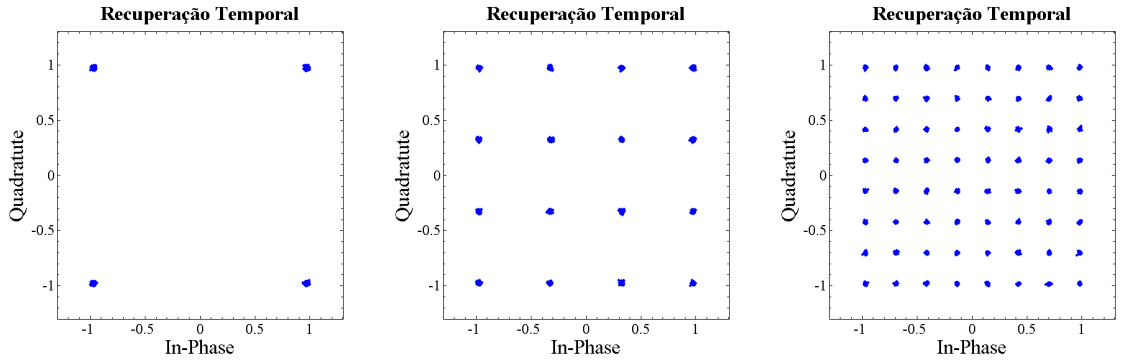


(d) Diagrama de Constelação para um $L = 512$, EVM = 0.64%

Figura 3.11: Diagramas de constelação QPSK antes e após a recuperação temporal

Na Figura 3.12 pode ser verificado o correto funcionamento do circuito de recuperação temporal para diferentes modulações IQ e um $L = 256$. Simulou-se a rotina fornecendo-lhe

um sinal modulado com várias modulações, e filtrado pelos filtros descrito na secção anterior. Obtendo-se os valores de EVM de 0.76%, 1.88% e 2.73%, paras as modulações QPSK, 16-QAM e 64-QAM, respetivamente.



(a) QPSK: EVM = 0.76% (b) 16-QAM: EVM = 1.88% (c) 64-QAM: EVM = 2.73%

Figura 3.12: Diagramas de constelação do recuperador temporal

3.3.3 Correção de Rotação

O algoritmo utilizado para realizar a correção de fase foi proposto por Andrew J. Viterbi em [Vit83]. É um algoritmo que eleva os símbolos $[x(t)]$ à sua quarta potência para retirar a fase da modulação, tal como demonstra a fórmula (3.3) [MD13].

$$\theta = \frac{1}{4} arg \left\{ \sum_{k=0}^{L_0-1} x^4(k) \right\} \tag{3.3}$$

E cujo o diagrama de blocos pode ser observado na Figura 3.13.

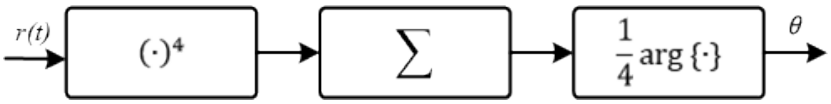


Figura 3.13: Diagrama de blocos do corretor de rotação (imagem editada de [MD13])

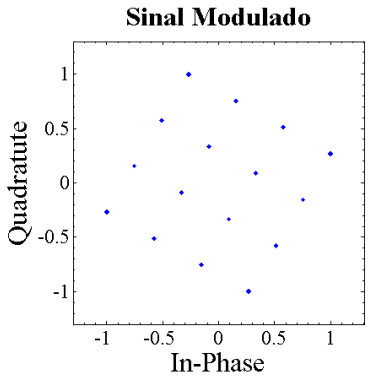
Para a sua implementação em MATLAB foi criada uma rotina, "*viterbi_stage(signal, L)*" com os parâmetros de entrada e saída descritos na Tabela 3.4.

Entradas			**Saídas**		
signal	:	sinal a recuperar	***rot_rec***	:	sinal recuperado
L	:	janela utilizada(L_0)			

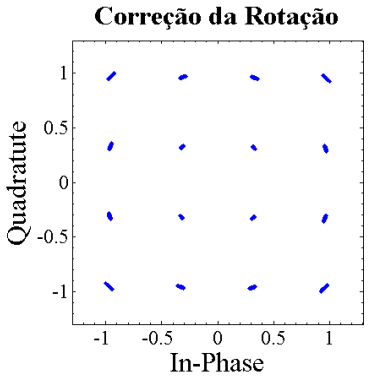
Tabela 3.4: Parâmetros da rotina *viterbi_stage* implementada em MATLAB

Esta rotina para além de calcular o ângulo de rotação faz a sua compensação e devolve o sinal já com a rotação da constelação corrigida. Na Figura 3.14 pode analisar-se em (3.14a)

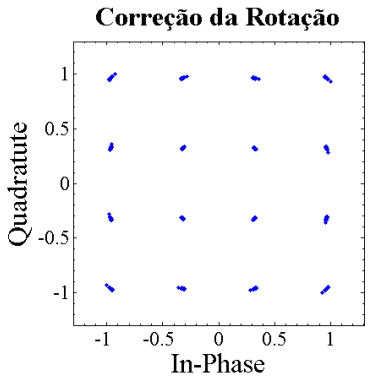
uma constelação 16-QAM à qual é provocado um atrase de fase de 30º, em (3.14b), (3.14c) e em (3.14d) o sinal recuperado pela rotina *viterbi_stage*, para várias janelas de observação. Obtiveram-se valores de EVM de 1.66%, 1.38% e 0.80% para $L = 256$, 512 e 1024, respetivamente.



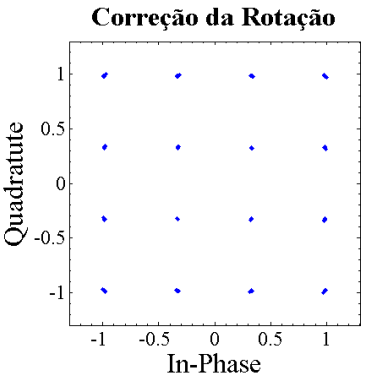
(a) Constelação antes da correção



(b) Constelação para um $L = 256$, EVM = 1.66%



(c) Constelação para um $L = 512$, EVM = 1.38%



(d) Constelação para um $L = 1024$, EVM = 0.80%

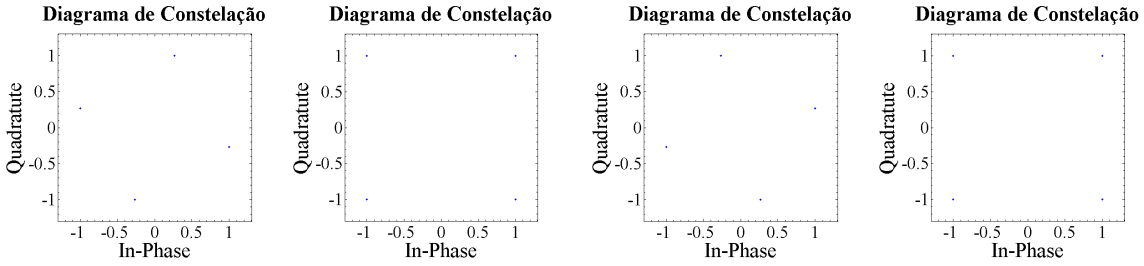
Figura 3.14: Diagramas de Constelação 16-QAM antes e após a correção de rotação de um atraso de fase 30º

Na Figura 3.15 podem-se observar alguns exemplos do funcionamento da rotina, à qual foram aplicado modelações QPSK, 16-QAM e 64-QAM com várias atrasos de fase fixos, tendo-se recuperado os sinais com EVM de 0% para as duas modulação QPSK (3.15a e 3.15b), 1.66% para a modulação 16-QAM (3.15c) e 2.56% para a modulação 64-QAM (3.15d), utilizando uma janela (L) de 256 amostras e atrasos de fase de 30º, 60º, 45º e 30º, respetivamente.

3.3.4 Desmodulador

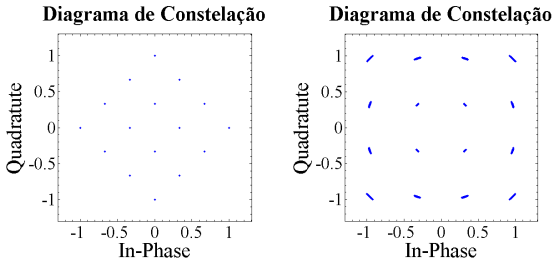
Uma vez terminadas as compensações necessárias para corrigir as distorções e interferências provocadas pelo canal, é necessário inverter o processo realizado pelo modulador. Desta forma, um circuito de decisão analisa o sinal recebido, determina qual a informação binária modulada nas suas componentes IQ e devolver a informação ao destino em formato binário.

Para realizar esta operação utilizou-se a função *qamdemod* do MATLAB.

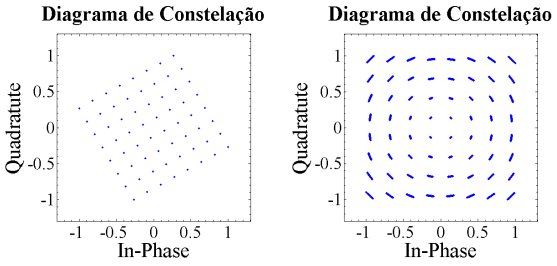


(a) QPSK com rotação 30º

(b) QPSK com rotação 60º



(c) 16-QAM com rotação 45º



(d) 64-QAM com rotação 30º

Figura 3.15: Diagramas de constelação do circuito de correção de rotação

3.4 Validação da Implementação em MATLAB

O correto funcionamento do sistema implementado em MATLAB para uma modulação QPSK pode ser verificado na Figura 3.16. Nesta figura, pode-se observar, através de diagramas de constelação, o sinal em vários pontos do sistema.

Para este exemplo foi fornecido um sinal aleatório com aproximadamente 100 mil *bits* e multiplicou-se o sinal entre a cadeia de transmissão e receção por $[e^{j\pi/12}]$ para simular um atraso de fase de 15º. As características dos filtros utilizados estão representadas nas Tabelas 3.1 e 3.1 e as janelas de observação (L) das rotinas de recuperação temporal e de e correção de fase foi de 1024 amostras.

O sinal desmodulado foi comparado com o sinal fornecido obtendo um BER de 0% e um EVM de 0.6%.

Da mesma forma, verificou-se o funcionamento para outras modulações como 16-QAM, 64-QAM e 256-QAM, ilustradas nas Figuras 3.17, 3.18 e 3.19, respetivamente.

À modulação 16-QAM, imputou-se um atraso de fase de -30^{o} e comparou-se com o sinal fornecido com o recebido obtendo um BER de 0% e um EVM de 1.03%. À modulação 64-QAM, atribuiu-se um atraso de fase de -45^{o} e obteve-se um BER de 0% e um EVM de 1.38%. Na modulação 256-QAM, simulou-se um atraso de fase de -15^{o} e obteve-se um BER de 0% e um EVM de 1.4%.

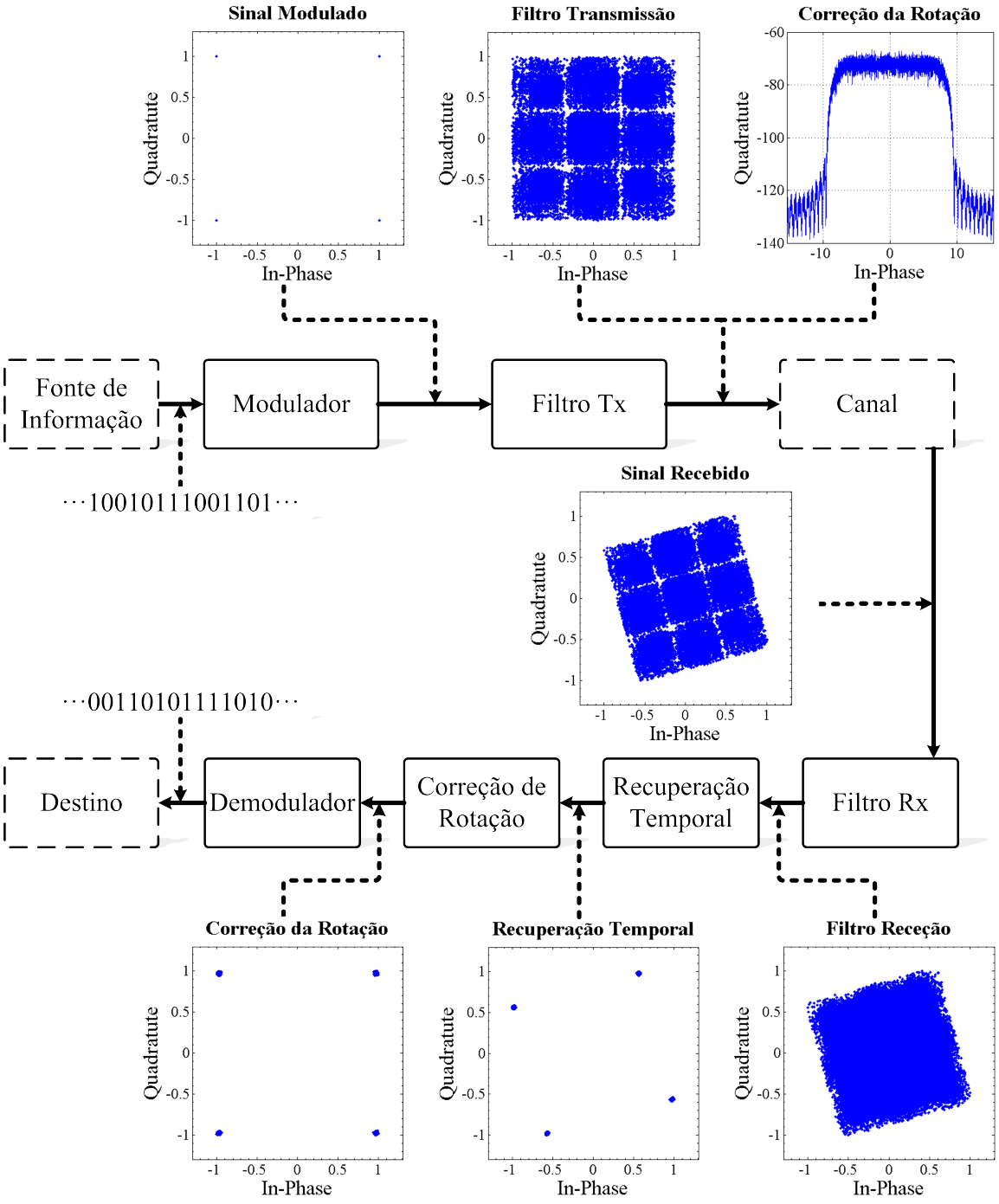


Figura 3.16: Validação de Modem QPSK em MATLAB

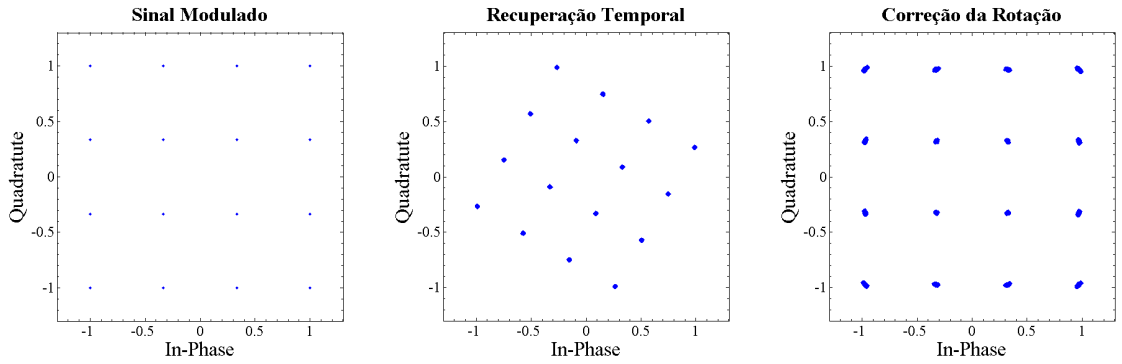


Figura 3.17: Validação de Modem 16-QAM em MATLAB

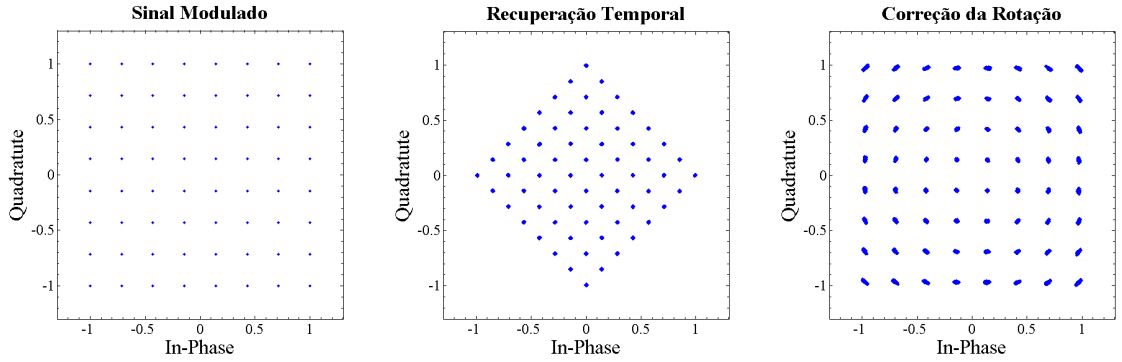


Figura 3.18: Validação de Modem 64-QAM em MATLAB

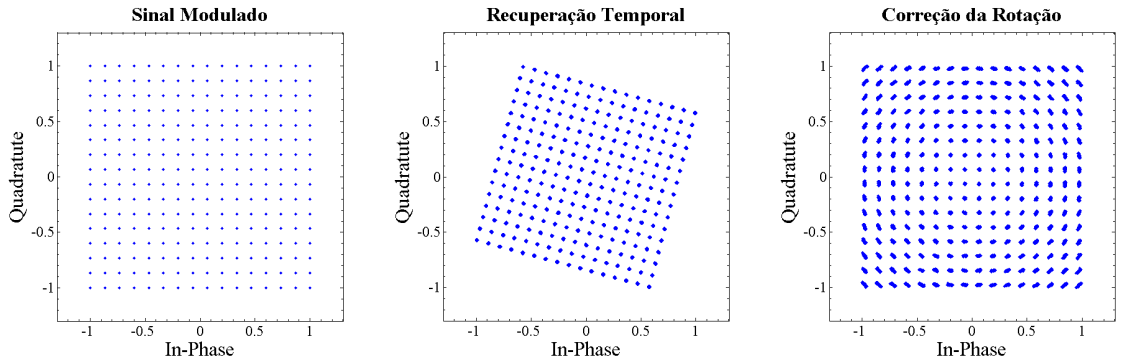


Figura 3.19: Validação de Modem 256-QAM em MATLAB

Verificado o correto funcionamento do modem em MATLAB é chegada a altura de idealizar a sua implementação em *hardware*, essa análise é realizado no próximo capítulo.

Capítulo 4

Implementação em *Hardware*

Após no capítulo anterior se ter validado e analisado o comportamento dos vários módulos do modem implementado em MATLAB, neste capítulo será demonstrado como os mesmo podem ser implementados em *hardware*.

4.1 Introdução

Como foi referido no capítulo introdutório, os sistemas SDR são sistemas em que as suas funções da camada física são definidas por *software*.

No entanto, a configuração do sistema SDR ideal ilustrada na Figura 1.2, onde todo o tratamento dos dados, desde o sinal em banda base até ao seu correspondente sinal RF era realizado digitalmente, não é possível de implementar para todas as gamas de frequências. Apesar da existência de conversores analógico para digital, em inglês *Digital-to-Analog Converters* (DACs), e conversores digital para analógico, *Analog-to-Digital Converters*(ADCs), extremamente rápidos, estes continuam limitados a algumas giga amostras por segundo.

Porém, mesmo sem um sistema totalmente digital, existem arquiteturas, tais como *RF Frontends*, que realizam a conversão das bandas de frequências intermédias e RF de uma forma controlável por *software*, permitindo assim manter a flexibilidade dos sistemas.

4.1.1 *RF Frontend*

Os *RF Frontends* são circuitos de transmissão/receção responsáveis pela translação do sinal em banda base para a gama frequências RF (*up-conversion*), ou vice versa (*down-conversion*).

Este tipo circuitos está dividido em duas arquiteturas principais:

- **Conversão direta ou homodina:** o sinal em banda base é diretamente convertido para RF, ou vice versa. Um exemplo de uma cadeia de transmissão utilizando este tipo de arquitetura é ilustrado na Figura 4.1.

- **Conversão de frequência intermédia ou heterodina:** o sinal em banda base é convertido primeiramente para uma frequência intermédia antes de sofrer uma *up-convertion* para RF, ou vice versa. A Figura 4.2 apresenta um diagrama de blocos de uma cadeia de transmissão que aplica esta arquitetura.

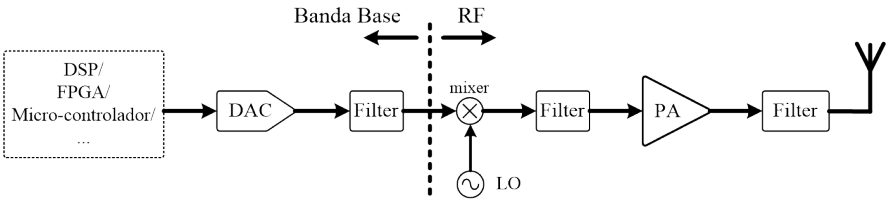


Figura 4.1: Cadeia de transmissão de uma arquitetura homodina

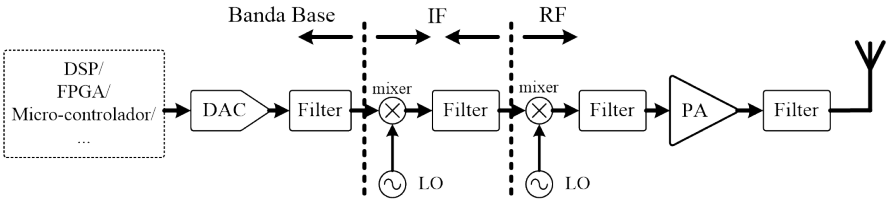


Figura 4.2: Cadeia de transmissão de uma arquitetura heterodina

---

A arquitetura homodina possui uma implementação mais simples e com menos custos, comparativamente com a arquitetura heterodina, pois não requer filtros nem etapas de amplificação à frequência intermédia.

Contudo, apresenta algumas desvantagens, tais como, possui uma menor imunidade às fugas do oscilador e, para modulação em fase, o *mixer* tem de possuir uma precisão suficiente para controlar os atrasos de fase.

4.1.2 FPGA

As FPGAs têm se revelado soluções bastante atraentes em termos de desempenho e reconfigurabilidade nos sistemas SDR. São constituídos por uma grande quantidade de portas lógicas que podem ser configuradas e conectadas das mais diversas formas. Esta liberdade de configuração permite gerar desde circuitos básicos como simples somadores a circuitos complexos como filtros ou micro-controladores.

Desta forma, as FPGAs oferecem um *hardware* bastante flexível e reconfigurável capaz de suportar algoritmos bastante complexos e computacionalmente intensivos.

Na continuação deste capítulo serão abordadas as técnicas utilizadas para a implementação em *hardware* do modem desenvolvido em MATLAB.

4.2 Cadeia de Transmissão

O modem desenvolvido em MATLAB tem a capacidade de utilizar qualquer modulação que represente os símbolos pelas componentes IQ de sinais ortogonais. No entanto, para a implementação do sistema em hardware considerou-se apenas a modulação QPSK, contudo qualquer modulação que utilize as variações das componentes IQ para modular o sinal pode ser facilmente integrada neste sistema.

4.2.1 Modulador

No modulador, ilustrado na Figura 4.3 os símbolos IQ são mapeados numa tabela de acordo com a combinação de bits que representam. Ou seja, a cada palavra da modulação é atribuída a componente IQ correspondente e esta é devidamente mapeada numa tabela.

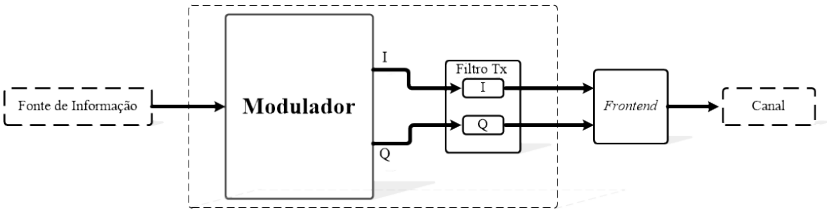


Figura 4.3: Cadeia de transmissão a implementar em *hardware*: Modulador

Um exemplo desse mapeamento pode ser observado na Tabela 4.1, onde é utilizada a codificação Gray de uma modulação QPSK.

bits de entrada	I	Q
00	-1	-1
01	-1	1
10	1	1
11	1	-1

Tabela 4.1: Mapa da modulação QPSK

Assim sendo, as componentes IQ são mapeadas na forma de uma *look up table* e o modulador, consoante a palavra que pretende modular, seleciona a componente IQ correspondente.

Desta forma consegue-se modular qualquer tipo de modulação ortogonal, sendo apenas necessário mapear a tabela de forma correta, quer se utilize uma modulação QPSK, 16-QAM, 64-QAM, etc..

4.2.2 Filtro Tx

Após a modulação, o sinal é representado por pelas suas componentes IQ, pelo que o módulo responsável pela filtragem, representado na Figura 4.4 implementa um filtro para cada componente.

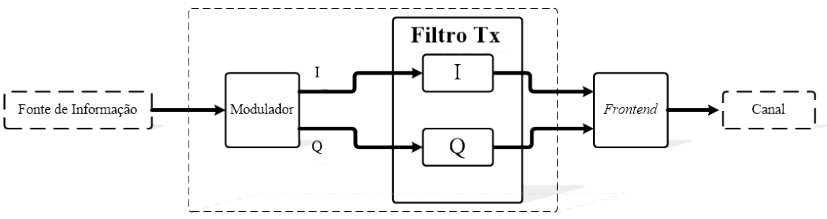


Figura 4.4: Cadeia de transmissão a implementar em *hardware*: Filtro Tx

A implementação de filtros digitais, pode ser realizada em *software*, através de processadores, tais como DSPs ou micro-controladores ou através de *hardware*, utilizando circuitos

integrados de aplicação especifica (ASICs), do inglês *Application Specific Integrated Circuits*, ou de lógica programável, tal como FPGAs.

Uma vez que o filtro RRC permite uma reduzida ISI para realizar a modelação de pulsos (*pulse shaping*) e limitar a frequência do sinal transmitido, é o selecionado para esta dissertação e é implementado numa estrutura *Finite Impulse Response* (FIR), que apresenta a seguinte função de transferência:

$$H(z) = \sum_{n=0}^{N} h[n] z^{-n} \tag{4.1}$$

Em que N é a ordem do filtro e $h[n]$ é o sinal discreto que representa a resposta ao impulso do filtro. A sua resposta a uma entrada x[n] é dada pela fórmula (4.2) e pode ser representada pelo diagrama de blocos ilustrado na Figura 4.5, em que $a_0, a_1, a_2, ...a_N$ são os coeficientes do filtro e z^{-1} blocos de atraso.

$$y[n] = \sum_{k=0}^{N} h[k] x[n-k] \tag{4.2}$$

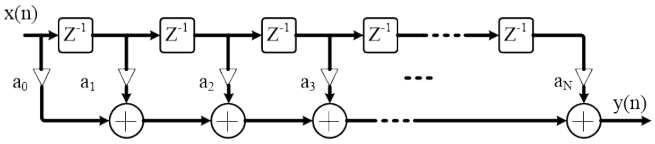


Figura 4.5: Diagrama de blocos de um filtro FIR

A implementação de filtros digitais em FPGA é realizada utilizando multiplicadores com acumuladores (MACs), um por cada coeficiente e blocos de memória (blocos RAM).

4.3 Cadeia de Receção

O sinal proveniente do canal de comunicação vem fortemente afetado com ruído e interferências, pelo que, o desempenho dos sistemas de comunicações depende bastante da eficiência de como esse sinal é recuperado.

4.3.1 Filtro Rx

A cadeia de receção utiliza os filtros ilustrados na Figura 4.6 para minimizar o ruído inserido no sinal durante a sua transmissão.

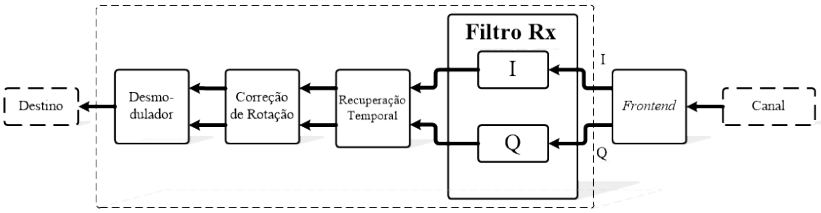


Figura 4.6: Cadeia de receção a implementar em *hardware*: Filtro Rx

Estes filtros são projetados de forma semelhante aos filtros da cadeia de transmissão para, desta forma, a sua junção respeitar as características dos filtros de Nyquist.

4.3.2 Recuperação temporal

De forma a alcançar uma desmodulação robusta e eficiente a cadeia de receção recupera o sincronismo do sinal transmitido através circuito de recuperação temporal representado na Figura 4.7

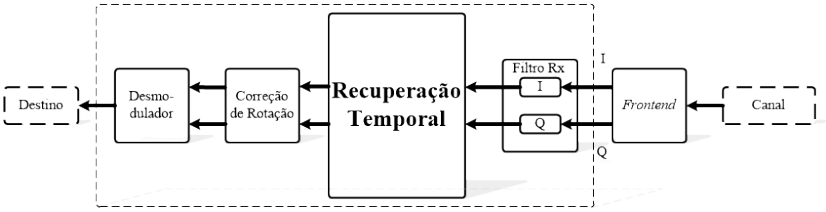


Figura 4.7: Cadeia de receção a implementar em *hardware*: Recuperação Temporal

O algoritmo escolhido para recuperar o sincronismo e estimar o tempo de atraso (τ) foi desenvolvido por O&M. Recordando a fórmula (3.2), o diagrama da Figura 3.10 e calculando a expressão $[e^{-j2\pi k/N}]$ através da formula de Euler, sabendo que $N = 4$ e k é um número inteiro, atingem-se os resultados apresentados na Tabela 4.2.

$\boldsymbol{k}$	0	1	2	3	4	5	6	7	8	...
$\boldsymbol{e^{-j2\pi k/N}}$	1	$-j$	-1	j	1	$-j$	-1	j	1	...

Tabela 4.2: Lista de valores de $[e^{-j2\pi k/N}]$ para vários k

Como se pode verificar, a expressão $[e^{-j2\pi k/N}]$ assume os valores de '1', '$-j$', '-1' e 'j' de uma forma cíclica, isto é, de 4 em 4 amostras.

Assim sendo, e visto que o algoritmo necessita de quatro amostras por símbolo, a expressão (4.3) pode ser implementada em *hardware* de uma forma replicada por quatro.

$$\sum_{k=0}^{NL_0-1} \mid x(kT_s) \mid^2 e^{-j2\pi k/N} \tag{4.3}$$

Assim, são instanciados quatro módulos para realizar a implementação de $[\mid x(kT_s) \mid^2]$, e a cada módulo é multiplicado o respetivo valor da expressão $[e^{-j2\pi k/N}]$, tal como ilustra o diagrama da Figura 4.8. De notar que os valores desta expressão assumem alternadamente valores reais e imaginários, correspondentes às componentes I e Q, respetivamente.

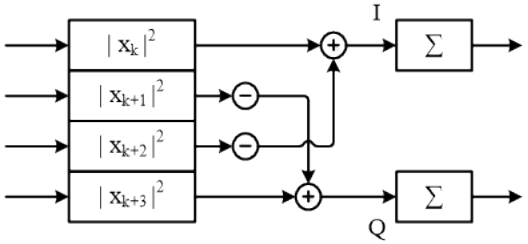


Figura 4.8: Diagrama de blocos da implementação de (4.3)

O bloco que implementa a expressão [| $x(kT_s)$ |^2] eleva o módulo de cada amostra ao quadrado, o que corresponde a somar o quadrado da sua componente I com o quadrado da componente Q, ou seja $[I^2 + Q^2]$ e é implementado com recurso a 2 multiplicadores tal como é demonstrado na Figura 4.9.

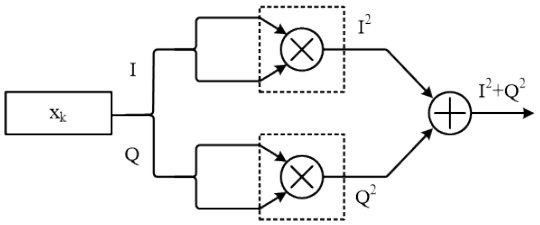


Figura 4.9: Diagrama de blocos da implementação de [| $x(kT_s)$ |^2]

Numa ótica de poupar recursos, uma vez que estes são sempre limitados neste tipo de sistemas, é possível reduzir esta replicação de *hardware* para metade, às custas de garantir uma frequência de relógio pelo menos duas vezes superior à taxa de transmissão de símbolo, o dobro do que na versão com replicação de 4 vezes.

Deste modo, instanciam-se apenas dois módulos para realizar o calculo de [| $x(kT_s)$ |^2], em vez das quatro anteriormente mencionados. O diagrama de blocos desta implementação é apresentado na Figura 4.10.

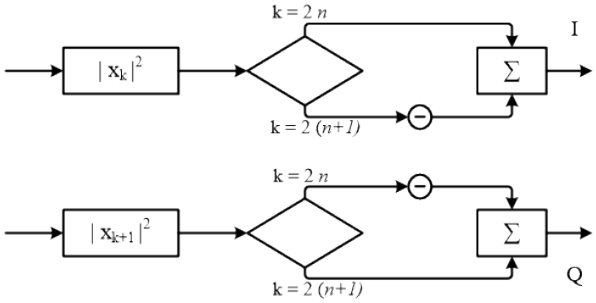


Figura 4.10: Diagrama de blocos da implementação da fórmula (4.3) com replicação de *hardware* de 2 vezes.

No entanto, esta implementação introduz uma pequena complexidade adicional, pois implica somar ou subtrair o quadrado de cada amostra de forma alternada, pois como já foi referido anteriormente e pode ser observado na Tabela 4.2, $[e^{-j2\pi k/N}]$ assume valores de '± 1' e '$\pm j$' de uma forma periódica e alternada.

A função $arg(\cdot)$ calcula o ângulo de fase da expressão (4.3), e pode ser implementada através da função arco-tangente ($arctan$), como é descrito na fórmula (4.4).

$$arg(a+jb) = \begin{cases} \arctan\left(\frac{b}{a}\right) & a > 0 \\ \arctan\left(\frac{b}{a}\right) + \pi & a < 0,\ b \geq 0 \\ \arctan\left(\frac{\pi}{2}\right) - \pi & a < 0,\ b < 0 \\ \frac{\pi}{2} & a = 0,\ b > 0 \\ -\frac{\pi}{2} & a = 0,\ b < 0 \end{cases} \tag{4.4}$$

Contudo, o τ estimado necessita de ser restrito ao período de símbolo, ou seja, $-T/2 \leq \tau \leq T/2$. De outra forma, devido à ambiguidade de fase poderia decidir pelo instante de amostragem do símbolo adjacente, o que iria provocar símbolos perdidos ou duplicados. Para lidar com este problema Oerder e Meyr propuseram realizar o *unwrapped* às estimativas de τ através da fórmula (4.5). [OM88]

$$\tau_{unw}(n) = \tau_{unw}(n-1) + SAW\left(\tau(n) - \tau_{unw}(n-1)\right) \tag{4.5}$$

Em que $SAW(\alpha)$ é uma função dente de serra que limita α de $-T/2$ a $T/2$, o funcionamento desta função pode ser observado na Figura 4.11.

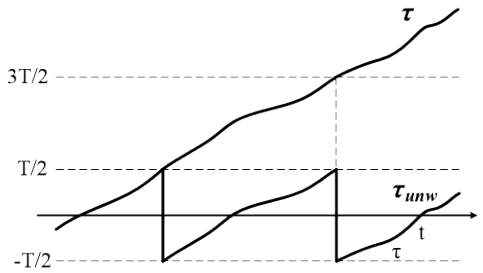


Figura 4.11: Função *unwrapped*

A escolha da melhor amostra é realizada pelo módulo de decisão, este tem a capacidade de decidir qual das 4 amostra por símbolo o melhor representa. Para tal, é necessário um módulo que atrase as amostras de entrada para compensar o tempo de cálculo do algoritmo. Este módulo é intitulado *linha de atraso* e pode ser implementado como um *shift register* de tamanho L_0.

A implementação do módulo responsável pela e recuperação temporal pode ser observado na Figura 4.12 com o *hardware* replicado por 4 e na Figura 4.13 com o *hardware* replicado por 2.

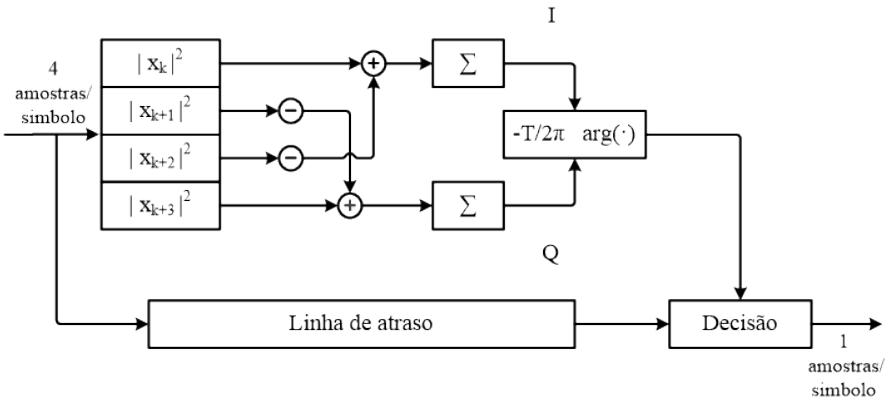


Figura 4.12: Diagrama de blocos do recuperador temporal com replicação de *hardware* de 4 vezes

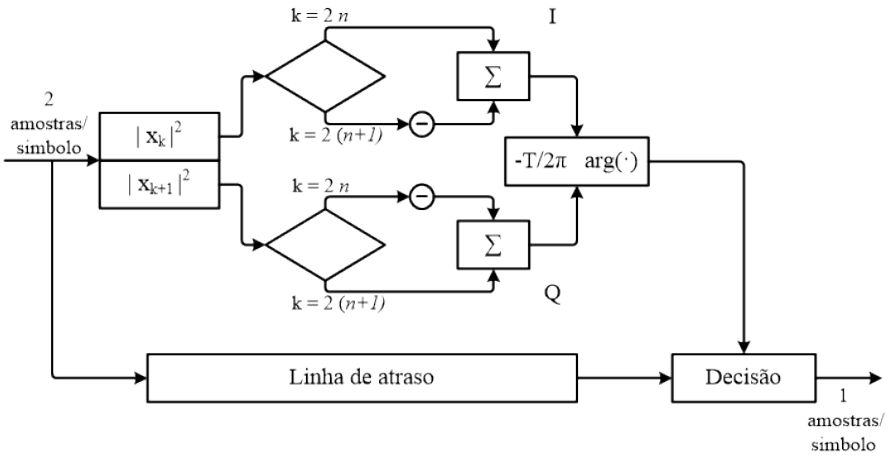


Figura 4.13: Diagrama de blocos do recuperador temporal com replicação de *hardware* de 2 vezes

4.3.3 Correção de Rotação

Tal como foi referido no capítulo anterior, o algoritmo escolhido para corrigir o erro de fase responsável pela rotação constelação foi proposto por Viterbi em [Vit83]. E a sua integração na cadeia de receção é ilustrada na Figura 4.14.

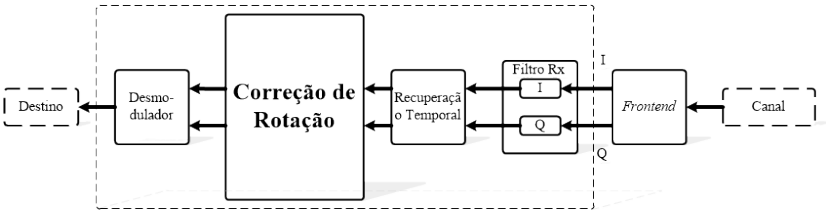


Figura 4.14: Cadeia de receção a implementar em *hardware*: Correção de Rotação

Recordando a sua expressão matemática, representada em (3.3) e o seu diagrama de blocos ilustrado na Figura 3.13, pode-se verificar que este algoritmo eleva cada amostra à sua quarta potência para calcular a fase da modulação.

Como sabemos as amostras $x(k)$ são representadas pelas suas componentes IQ, pelo que elevar cada uma à sua quarta potência representa:

$$(I + Qi)^4 \;=\; (I^4 + Q^4 - 6I^2Q^2) \;\;-\;\; 4IQ(I^2 + Q^2)i \tag{4.6}$$

E pode ser implementado com a utilização de 8 multiplicadores, como é demonstrado no diagrama da Figura 4.15.

A função $arg(\cdot)$ pode ser implementada recorrendo à *arctan*, tal como já foi referido na secção anterior e tal como no nesse caso, é necessário realizar o *unwrap* para remover a sua ambiguidade de fase.

Após calculado o erro de fase, θ, o módulo de decisão corrige a rotação da constelação multiplicando cada amostra por $[e^{-}j\theta]$.

Um diagrama completo da implementação do algoritmo ser visualizado na Figura 4.16.

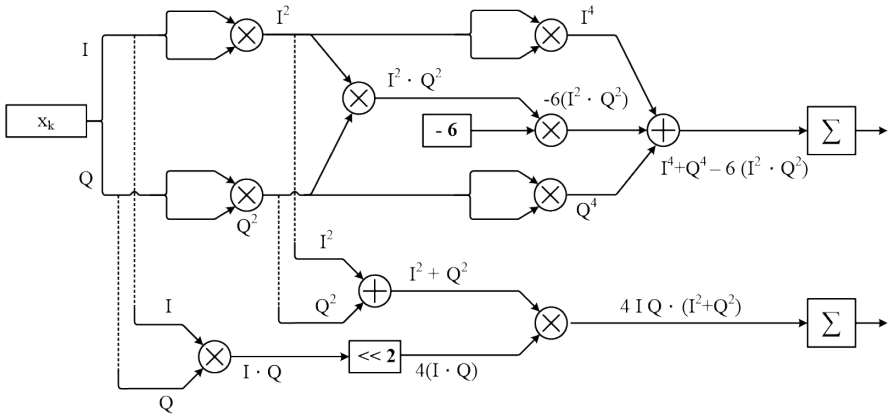


Figura 4.15: Diagrama de blocos da implementação de $[x^4(k)]$

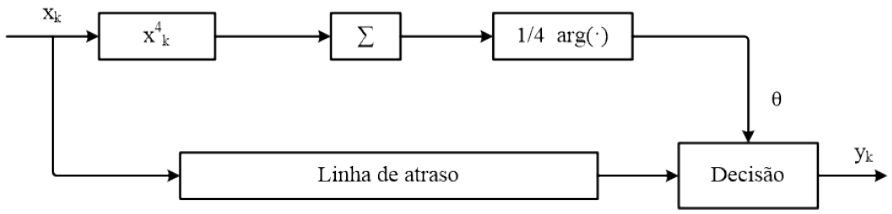


Figura 4.16: Diagrama de blocos da correção de rotação

4.3.4 Desmodulador

Recuperados os efeitos destrutivos do provocados ao sinal durante a suas transmissão, o desmodulador ilustrado na Figura 4.17 recupera a sua informação.

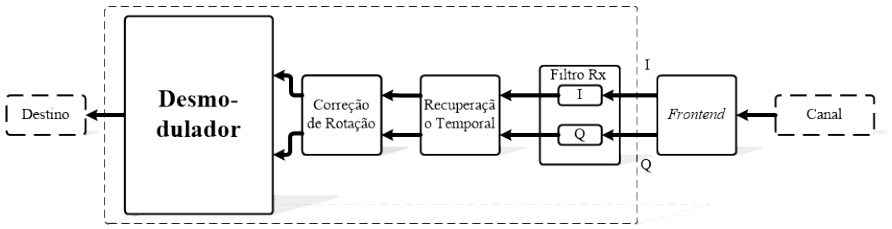


Figura 4.17: Cadeia de receção a implementar em *hardware*: Desmodulador

Tendo o desmodulador conhecimento do mapeamento utilizado no modulador, este identifica a que ponto da constelação ideal corresponde cada símbolo IQ recebido, e recorrendo à tabela de mapeamento recupera a combinação de bits correspondente. Ou seja, a cada símbolo IQ recebido é atribuído a correspondente palavra binária mapeada na sua tabela.

Para identificar qual o ponto da constelação ideal corresponde a cada símbolo IQ recebido, analisam-se as componentes I e Q de uma forma independente e identifica-se em qual quadrante está o símbolo recebido, tal como é ilustrado no diagrama de constelação da Figura 4.18.

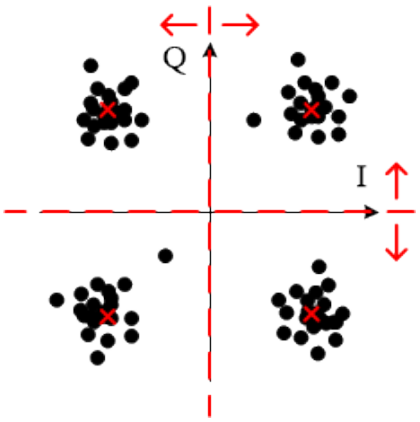


Figura 4.18: Diagrama de constelação

Ao longo deste capítulo foi analisada a forma como cada módulo do modem desenvolvido no capítulo anterior pode ser implementado em *hardware*. A sua implementação em *hardware* real é realizada no capítulo seguinte.

Capítulo 5

Prototipagem Laboratorial

Neste capítulo é apresentada a implementação fisica do modem, bem como uma breve descrição de como o kit de desenvolvimento escolhido foi utilizado.

5.1 Introdução

Nesta dissertação é implementado um sistema de comunicação QPSK. Este sistema tira proveito dos sistemas SDR, realizando o processamento do sinal digitalmente, desde modulação, filtros a circuitos de correção. Sendo que a parte de *ud/down-conversion* de RF é efetuada analogicamente por um *frontend* comercial, tal como demonstrada a Figura 5.1.

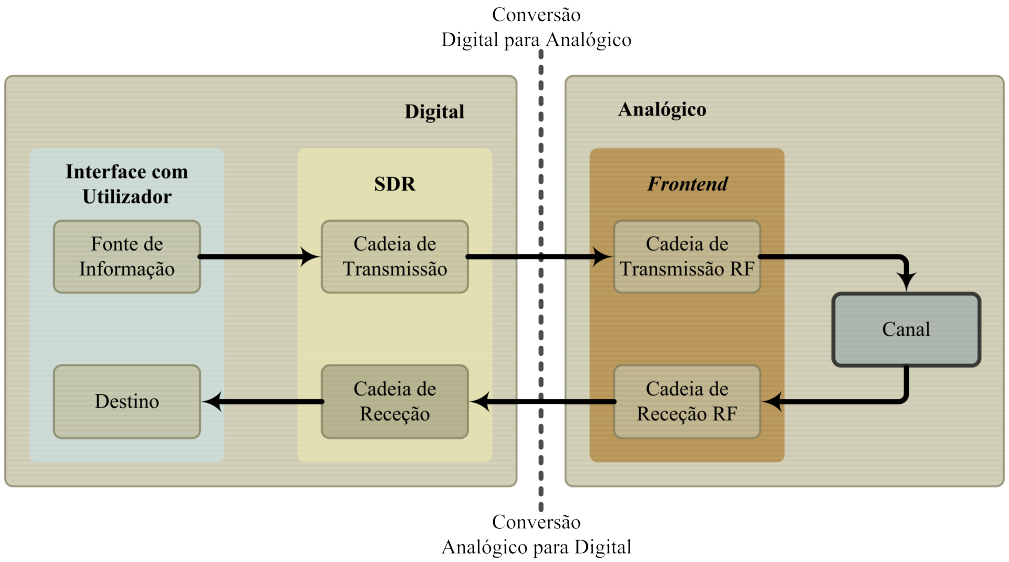


Figura 5.1: Diagrama de blocos do sistema de comunicação

5.2 Domínio Analógico - *Frontend*

O domínio analógico é o responsável pelas conversões Analógico para Digital (A/D) e Digital para Analógico (D/A) do sinal, bem como o seu *up-conversion e down-conversion*.

O *frontend* RF utilizado é o *AD-FMCOMMS1-EBZ* desenvolvido pela Analog Devices para aplicações de rádio com integração em FPGA. A Analog Devices disponibiliza uma implementação de referência (*reference design*) que utiliza uma combinação de *hardware* (*IP-cores*) com interface entre FPGA, DACs e ADC, e *software* compatível com processadores ARM que permitem controlar os vários componentes internos da placa *AD-FMCOMMS1-EBZ*.

Esta interface oferecida é sem duvida uma mais valia neste tipo de aplicações, juntamente com uma largura de banda bastante abrangente (até 200MHz) e a uma vasta gama de frequências de operação RF(400MHz - 4GHz) tornam o *AD-FMCOMMS1-EBZ*, ilustrado na Figura 5.2, a escolha ideal para este sistema.

Figura 5.2: *Frontend AD-FMCOMMS1-EBZ* [Ele]

O seu diagrama de blocos é apresentado na Figura 5.3 e irá ser descrito mais detalhadamente no decorrer desta secção.

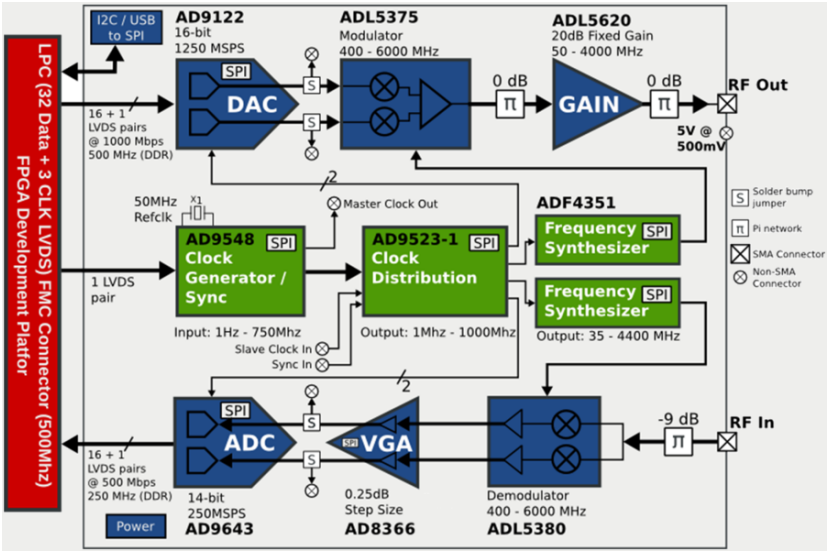


Figura 5.3: Diagrama de blocos do *frontend AD-FMCOMMS1-EBZ* [Dev]

5.2.1 Cadeia de Transmissão RF

A cadeia de transmissão converte as componentes IQ do sinal em banda base no seu respetivo sinal RF. Para tal, é constituída por uma DAC, um modulador, filtros e um amplificador de potência, tal como se pode observar no diagrama da Figura 5.4, onde está representado um circuito de transmissão semelhante ao utilizado pelo *frontend AD-FMCOMMS1-EBZ*.

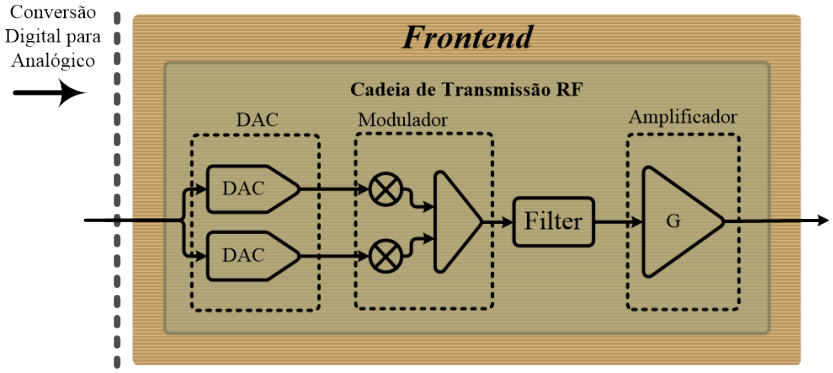


Figura 5.4: Diagrama da transmissão RF

DAC

O sinal é fornecido pela FPGA em formato digital, pelo que é necessário um circuito que realize a sua devida conversão para sinal analógico. Neste *frontend* o circuito responsável por esta operação é o AD9122 [Dev09c].

Este circuito é internamente constituído por duas DACs de 16 *bits*, uma para cada componente IQ e possui filtros interpoladores para atenuar as suas imagens criadas.

Modulador

Por sua vez, o modulador tem como função realizar o *up-conversion* do sinal em banda base para a frequência RF desejada e somar as suas componentes IQ. A este tipo de modulador denomina-se modulador em quadratura e é realizado pelo circuito ADL5375 [Dev07].

O modulador converte a informação das componentes IQ provenientes da DAC num sinal à frequência desejada fornecida pelo sintetizador ADF4351 [Dev12].

Na Figura 5.5 pode ser visualizado o efeito do *up-conversion* no espetro de frequências.

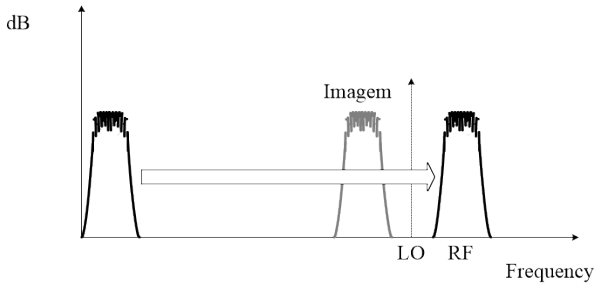


Figura 5.5: Efeito do *up-conversion* no espetro de frequências

Amplificador

Ao sinal modulado é aplicado um filtro para rejeição da imagem da componente RF, visível na Figura 5.5 e posteriormente amplificado pelo amplificador ADL5602 [Dev09b], que fornece um ganho de até 20dB.

5.2.2 Cadeia de Receção RF

A cadeia de receção tem como objetivo reverter o processo da transmissão, ou seja, converter a informação de um sinal em RF para banda base, contudo o princípio de funcionamento é bastante semelhante, mas de forma inversa.

O recetor recebe o sinal RF do canal e converte-o num sinal IQ em banda base. O diagrama de blocos utilizado pelo *frontend AD-FMCOMMS1-EBZ* pode ser observado na Figura 5.6.

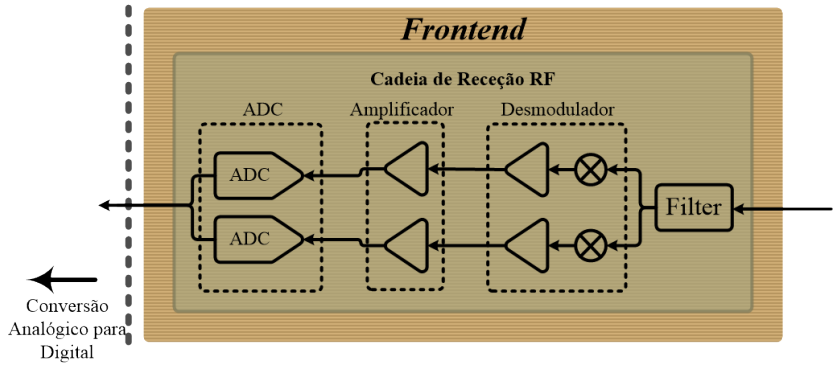


Figura 5.6: Diagrama da receção RF

Desmodulador

O desmodulador recebe um sinal RF proveniente do canal e divide-o nas suas componentes em fase e em quadratura e realiza ainda o seu o seu *down-convertion* para banda base. Este circuito é realizado pelo ADL5380 [Dev09a].

Amplificador

Após o *down-conversion* as componentes IQ são amplificadas por um amplificador de baixo ruído, presente no circuito ADL8366 [Dev10], este amplificador aplica um ganho programável de 4.5 a 20 dB.

ADC

O circuito utilizado para digitalizar o sinal em banda base para a FPGA é o AD9643 [Dev11]. Este é constituído internamente por duas ADCs de 14 *bits*.

5.2.3 Relógio e Protocolo de Comunicações

O relógio do *frontend* pode ser controlado por um cristal de 50Mhz ou a partir da FPGA, através do conector FMC.

O protocolo de comunicação utilizado pelo *AD-FMCOMMS1-EBZ* para aceder a registos e configurações é o SPI. Contudo possui um micro-controlador interno que converte as comunicações I2C em SPI.

5.3 Domínio Digital - SDR

O domínio digital é responsável por todo o processamento digital do sinal, nesta dissertação foi implementado na ZedBoard da Xilinx. Esta utiliza a arquitetura Zynq-7000 SoC que inclui um *dual Core ARM Cortex-A9*, lógica programável e possui um vasto conjunto de periféricos.

Embora, a placa de desenvolvimento ZedBoard tenha sido a escolhida nesta dissertação, existe uma vasta gama de FPGAs, tais como Artix, Kintex, Spartan, Virtex, entre muitas outras.

A ZedBoard foi a selecionada, pois para além de conter a capacidade de implementação de lógica reprogramável, inclui ainda um sistema de processamento introduzido pelo *dual Core ARM Cortex-A9*, que podem ser combinados e trabalhar em conjunto. Possui ainda uma grande variedades de interfaces e componentes adicionais, entre eles interfaces como FMC para a ligação com o *frontend*, interface USB-UART para comunicar com um computador e componentes como memórias DDR3 para armazenar informação, entre muitos outros.

A placa de desenvolvimento ZedBoard pode ser observada na Figura 5.7 e o seu diagrama de blocos é ilustrado na Figura 5.8.

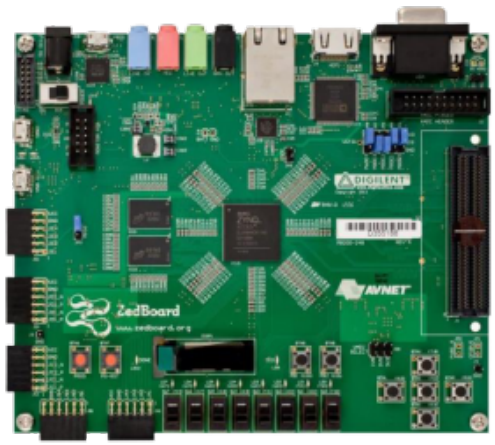


Figura 5.7: Placa de desenvolvimento ZedBoard [Zed]

Na Figura 5.9 pode ser observado o diagrama de blocos completo da placa de desenvolvimento Zedboard com o *frontend AD-FMCOMMS1-EBZ*.

Como já foi mencionado anteriormente a Analog Devices disponibiliza uma implementação de referência, onde fornece alguns *IPcores* bastantes úteis à integração da FPGA com o *frontend AD-FMCOMMS1-EBZ*, entre eles os mais significativos são os *IPcores* axi_IIC, axi_AD9122 e axi_AD9643.

- O ***IPcore* axi_IIC** é responsável pela comunicação entre o *ZYNQ* e o micro-controlador do *frontend*, que por sua vez é responsável pela configuração dos seus componentes de

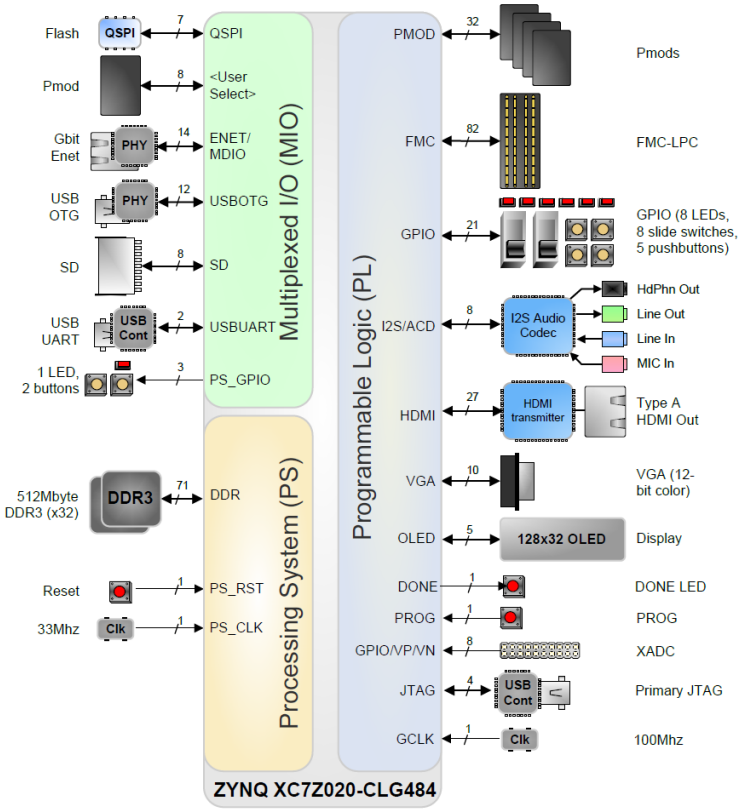


Figura 5.8: Diagrama de blocos da placa ZedBoard [AVN14]

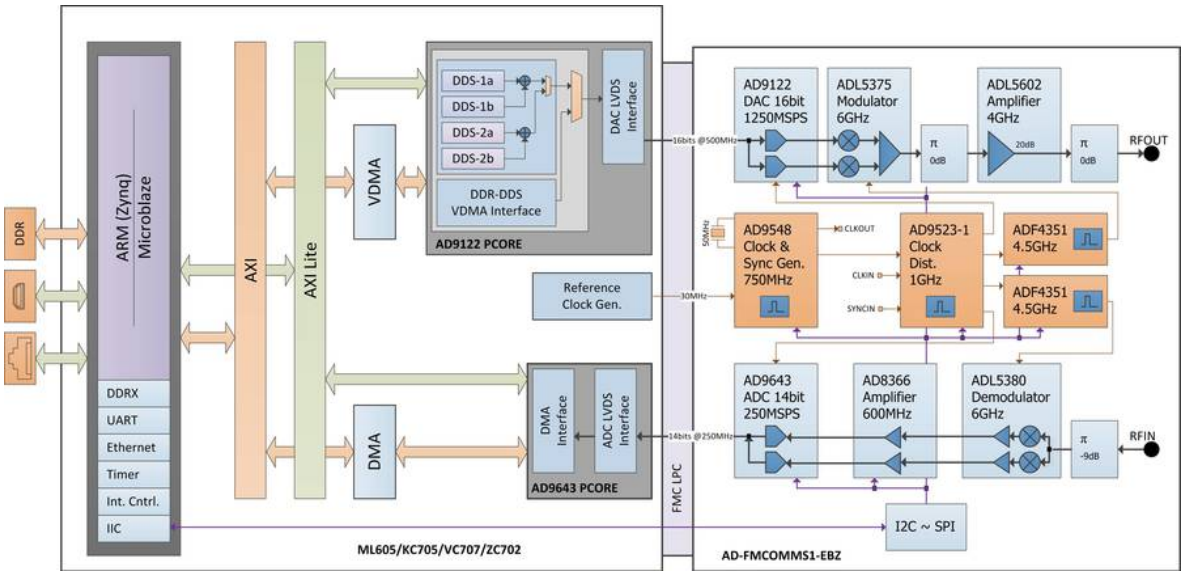


Figura 5.9: Diagrama da implementação de referência do *frontend AD-FMCOMMS1-EBZ* para a placa de desenvolvimento ZedBoard [Dev]

---

forma a obter diferentes frequências do sinal RF, frequências de amostragens da DAC

e ADC, ganho do amplificador de saída, etc..

- O ***IPcore axi_AD9122*** realiza a interligação com a DAC, fornecendo-lhe o sinal a ser transmitido e providência o relógio à cadeia de transmissão.

- O ***IPcore axi_AD9643*** gere a interface ADC-FPGA, ou seja, disponibiliza o sinal recebido do *frontend* à FPGA e fornece o relógio à cadeia de receção.

5.4 Prototipagem em FPGA

A ferramenta utilizada para a implementação da lógica programável é o *Vivado Design Suite 2014.2* e para a programação do sistema de processamento é utilizado o *Xilinx Software Development Kit (SDK)*.

A implementação de referência disponibilizada pela Analog Devices pode ser obtida em [Dev15b], onde é também apresentada uma breve descrição do procedimento necessário para a criação do seu *HDL Block Design* em *Vivado*. A Analog Devices fornece também *drivers* implementadas em *C* que podem ser utilizadas para configurar o *frontend* [Dev15a].

Os módulos fornecidos pela Analog Devices são implementados em Verilog, contudo os módulos desenvolvidos no decorrer da dissertação são implementados em *Very high speed integrated circuits Hardware Description Language* (VHDL).

O sistema deve operar a uma frequência de 2.4GHz, numa configuração de *lookback*, ou seja, o transmissor e o recetor estão conectados diretamente com um cabo coaxial. Contudo, uma vez que o sinal é amplificado utiliza-se um atenuador de 30 dB para não saturar os andares de entrada do *frontend*.

5.4.1 Cadeia de Transmissão

O diagrama com os módulos necessários à implementação da cadeia de transmissão é ilustrado na Figura 5.10 e no decorrer da secção será descrito como é realizada a sua implementação na FPGA.

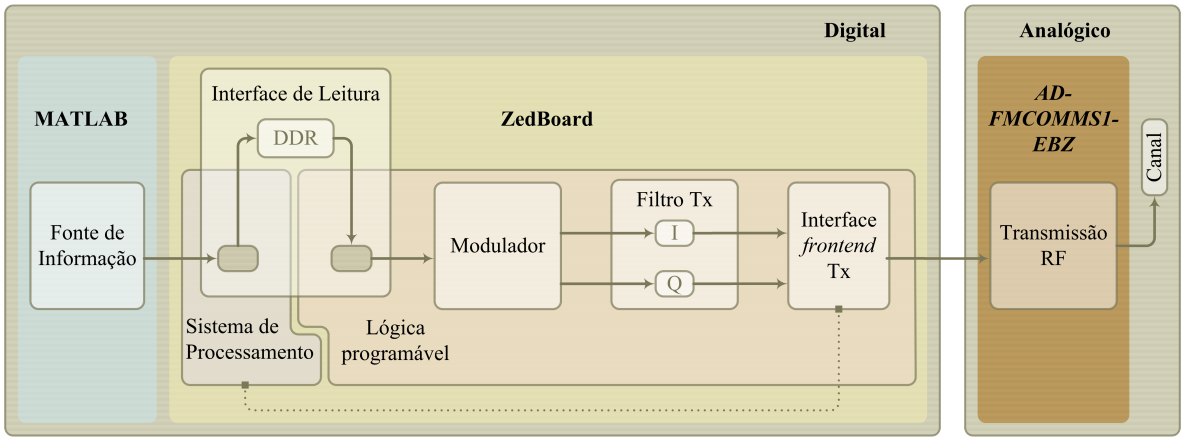


Figura 5.10: Diagrama de blocos da implementação da cadeia de transmissão

Interface de Leitura

A informação do sinal a transmitir é gerada em MATLAB de uma forma semi-aleatória e é posteriormente copiada para um ficheiro em *C*, onde a função "*dac_dma_setup()*", uma função adaptada das *drivers* fornecidas, a guarda na posição "DDRDAC_BASEADDR" da memória.

Na camada física, o *IPcore axi_adD9122_dma*, ilustrado na Figura 5.11, acede a essa mesma posição da memória e comunica a informação a ser transmitida ao modulador, através da interface *m_axis*.

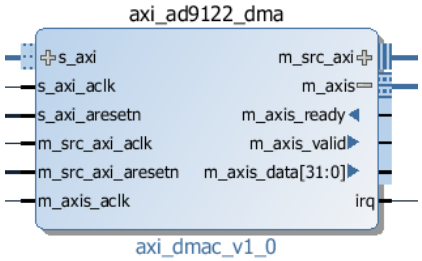


Figura 5.11: *IPcore axi_adD9122_dma*

Modulador

Este módulo foi implementado em VHDL e tal como descrito na secção 4.2.1 recebe uma palavra com dados binários e coloca à sua saída as componentes IQ correspondentes. À semelhança de todos os módulos desenvolvidos nesta dissertação, obedece ao protocolo *AXI4 Stream* [Sui15].

Na Figura 5.12 é apresentado o módulo implementado e as suas interfaces.

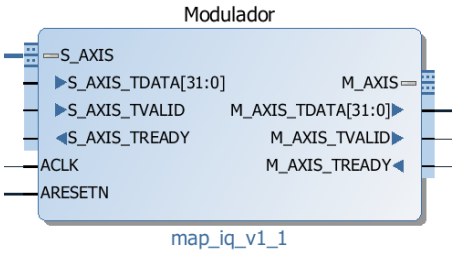


Figura 5.12: *IPcore map_iq*

À sua entrada aceita uma palavra com 32 *bits* contendo a informação binária a transmitir, esta deve ser ordenada no sentido dos *bits* mais significativos para os menos significativos e a cada transição ascendente do relógio coloca um símbolo na sua saída com as respetivas componentes IQ, sendo que este é representado em palavras de 32 *bits*, onde os 16 *bits* mais significativos corresponde à componente I e restantes 16 à componente Q, ambas em complemento para 2.

Filtro Tx

Após a modulação, é aplicado um filtro RRC às componentes I e Q. Os coeficientes do filtro utilizado são calculados através das simulações em MATLABs e gravados em formato *coe* [XIL]. Este filtro é instanciado duas vezes, um para cada componente, e é implementado através do *FIR Compiler* disponibilizado pela Xilinx [XIL15b].

Os filtros são instanciados utilizando o *customize IP* do *IP Catalog* no Vivado com as especificações descritas na Tabela 5.1, tal como é ilustrado no Anexo A.1. O módulo de um dos filtros é apresentado na Figura 5.13.

Filter Specification	***Filter Type***	*Interpolation*
	Interpolation Value Ratio	2
	Rate Change Type	*Integer*
Hardware Oversampling	***Select Format***	*Input Sample Period*
	Sample Period	1
Data Path Options	***Input Data Type***	*Signed*
	Input Data Width	16
	Input Data Fractional Bits	0
	Output Rounding Mode	*Truncate LSBs*
	Output Width	16

Tabela 5.1: Especificações do *customize IP* do *FIR Compile* para implementação do filtro RRC

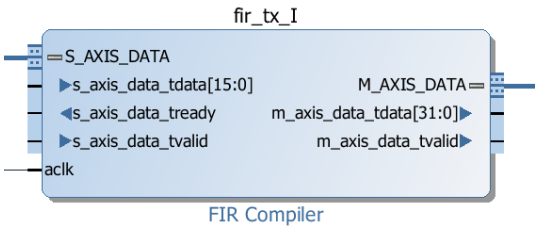


Figura 5.13: *IPcore FIR Compiler*

Interface *frontend* Tx

Concluído o processamento digital do sinal, este é enviado para o *frontend* que realiza a sua conversão D/A e o *up-conversion* para RF.

As configurações do *frontend* são controlada por *software*, através do *ZYNQ*, com recurso aos drivers disponibilizados. As configurações utilizadas podem ser observadas na Tabela 5.2.

A interface da cadeia de transmissão com o *frontend* é realizada pelo *IPcore AD9122* ilustrado na Figura 5.14, onde *dac_div_clk* é o relógio de referência que providência para toda a cadeia de transmissão. A frequência deste relógio é obtida através da divisão da frequência de amostragem da DAC por 4.

Freq. de Transmissão	2.4 GHz
Freq. de Amostragem da DAC	61.44 MHz
Freq. de Receção	2.4 GHz
Freq. de Amostragem da ADC	30.72 MHz
Ganho do Amp de Receção	4.5 dB

Tabela 5.2: Configurações do *frontend*

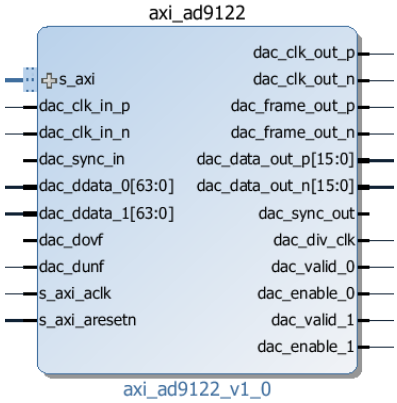


Figura 5.14: *IPcore AD9122*

5.4.2 Cadeia de Receção

A Figura 5.15 apresenta o diagrama de blocos da cadeia de receção. A implementação de cada bloco na placa de desenvolvimento ZedBoard é descrita de seguida.

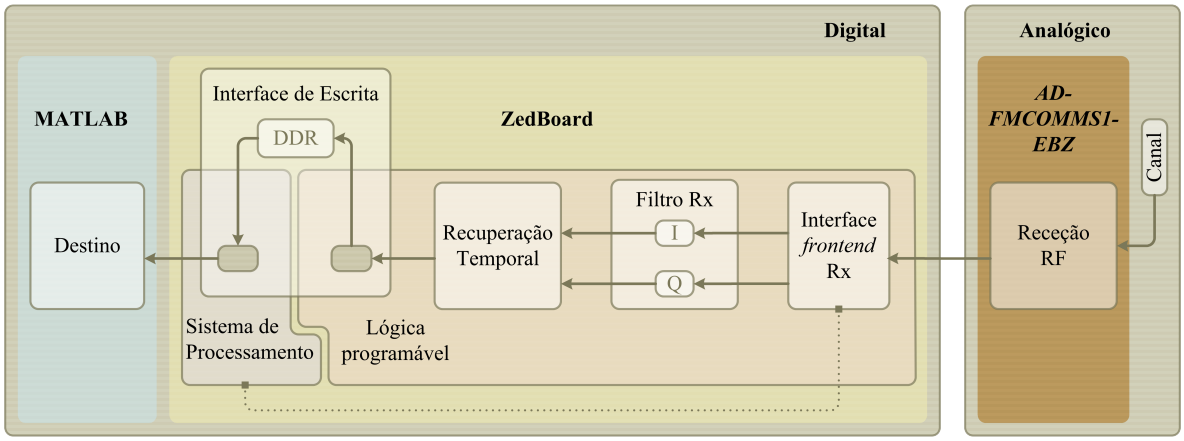


Figura 5.15: Diagrama de blocos da implementação da cadeia de receção

Interface *frontend* Rx

A cadeia de receção do *frontend* tem como função realizar o *down-conversion* do sinal RF recebido para banda base e proceder à sua conversão de A/D.

Ao *IPcore AD9643*, apresentado na Figura 5.16, cabe a responsabilidade de fornecer as componentes IQ do sinal recebido à FPGA e providenciar o relógio de referência para esta cadeia. Este relógio é disponibilizado na interface *adc_clk* e tem a mesma frequência da amostragem da ADC.

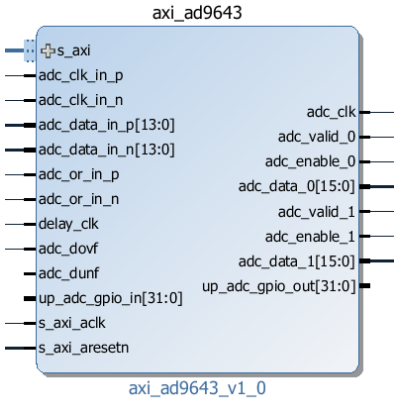


Figura 5.16: *IPcore AD9643*

Filtro Rx

À semelhança da cadeia de transmissão, os filtros da receção são implementados utilizando o *FIR Compile*. Os seus coeficientes são obtidos nas simulações em MATLAB e as especificações do *customize IP* são as mesmas, descritas na Tabela 5.1 e é o seu módulo semelhante ao apresentado na Figura 5.13.

Recuperação Temporal

Após a realização da filtragem do sinal recebido, o *IPcore timing_recovery* estima o seu atraso e recupera-o de modo a obter uma desmodulação eficiente. Para tal, implementa o algoritmo de recuperação temporal proposto por O&M, descrito na secção 4.3.2. A implementação escolhida é a de replicação de *hardware* 2 vezes representada na Figura 4.13.

O processamento do algoritmo é separado em tarefas especificas, divididas pelos vários módulos com a hierarquia demonstrada no diagrama da Figura 5.17 e seguida de uma breve descrição de individual.

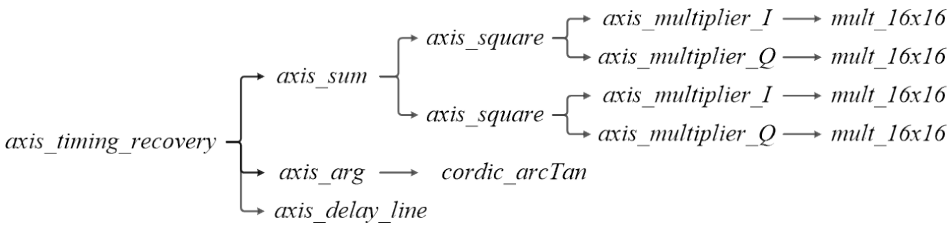


Figura 5.17: Módulos constituintes do *IPcore timing_recovery*

- ***mult_16x16:***

É implementado pelo *IPcore Multiplier(12.0)* da Xilinx e instanciado através do *IP Catalog* do Vivado, tal como é ilustrado no Anexo A.2. [XIL15c]

Aceita como entrada duas palavras de 16 *bits* e devolve o seu produto com um atraso de 4 ciclos de relógio.

- ***axis_multiplier:***

  Uma vez que o módulo anterior, *mult_16x16* não respeita o protocolo *AXI4 Stream*, o módulo *axis_multiplier* atrasa o sinal que indica se a informação à entrada é válida para indicar se o respetivo produto proveniente do multiplicador é valido.

  Este módulo aceita como entrada uma palavra de 32 *bits*, onde os 16 mais significativos correspondem a um operador da multiplicação e os restantes 16 ao segundo operador. À sua saída devolve uma palavra de 32 *bits* com o resultado de da multiplicação com um atraso de 6 ciclos de relógio.

- ***axis_square:***

  O módulo *axis_square* aceita como entrada uma palavra com 32 *bits*, correspondente a uma amostra IQ em que a componente I é representada nos 16 *bits* mais significativos e a componente Q nos restantes 16.

  Instância dois módulos *axis_multiplier*, um para calcular $[I^2]$ e outro para calcular $[Q^2]$, posteriormente realiza a soma da saída destes módulos, isto é $[I^2 + Q^2]$, e coloca-a à sua saída, numa palavra com 32 *bits* e com um atraso de 8 ciclos de relógio.

- ***axis_sum:***

  Por sua vez, o módulo *axis_sum* recebe duas palavra de 32 *bits*, correspondentes às amostras recebidas, representadas na Figura 4.13 por $[x_k^2]$ e $[x_{k+1}^2]$. As componentes I das amostras são recebidas nos 16 *bits* mais significativos de cada entrada e as componentes Q nos restantes 16.

  Instancia um módulo *axis_square* para cada amostra e na saída coloca duas palavras com 32 *bits* correspondentes ao somatório das componentes reais e imaginárias da fórmula 4.3. Este módulo coloca a sua saída valida após $12 + 2L_0$ ciclos de relógio desde a primeira amostra de cada janela de processamento.

- ***cordic_arcTan:***

  Este módulo é responsável pela execução da função arco-tangente e é implementado pelo *IPcore Cordic (6.0)* da Xilinx [XIL15a] e instanciado através do *IP Catalog* do Vivado, tal como é ilustrado no Anexo A.3.

  Recebe uma palavras de 64 *bits*, em que os 32 *bits* mais significativos representam o denominador e os restantes 32 o numerador da função arco-tangente. E apresenta o seu resultado numa interface com 32 *bits* e com um atraso de 36 ciclos de relógio.

- ***axis_arg:***

  O módulo *axis_arg* tem como entradas os dois sinais provenientes do módulo *axis_sum* e tem como função calcular o ângulo de fase de $[\sum(I) + j\sum(Q)]$. Para tal, instancia o *cordic_arcTan* para realizar a função arco-tangente e calcula o ângulo de fase através da fórmula 4.4. Este módulo apresenta um atraso de 39 ciclos de relógio.

- ***axis_delay_line:***

  Este módulo tem como função criar uma fila de espera, isto é, um FIFO que atrasa as amostras que chegam ao circuito de recuperação, e compensa o atraso de calculo do algoritmo.

- ***axis_timing_recovery***

  O *axis_timing_recovery* é o módulo principal do algoritmo, é ele que instancia os restantes módulos e realiza o *unwrapped* do τ estimado e decide qual a amostra que melhor representa cada símbolo.

  Recebe como entrada duas palavras de 32 *bits*, uma para as componentes I e outra para as componentes Q das amostras a estimar. Os 16 *bits* mais significativos correspondem às primeiras amostras, isto é $[x_k]$ e os restantes 16 à amostra seguinte $[x_{k+1}]$.

Na Figura 5.18 pode ser observado o módulo implementado e as suas interfaces.

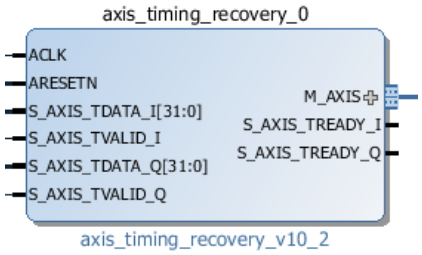


Figura 5.18: *IPcore axis_timing_recovery*

5.4.3 Interface de Escrita

Recuperado o sincronismo o sinal é colocado de novo em memória para ser avaliado utilizado o MATLAB, para tal o *IPcore axi_adD9643_dma*, ilustrado na Figura 5.19, grava-o na posição "DDR_BASEADDR"da memória.

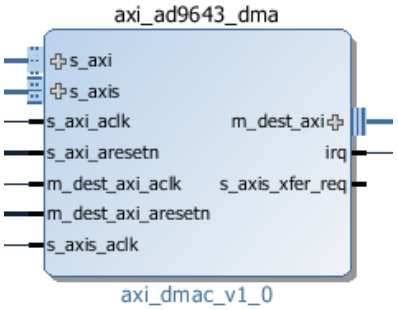


Figura 5.19: *IPcore axi_adD9643_dma*

A informação do sinal sincronizado é lido pelo ZYNQ através da função *capture()*, desenvolvida em *C* com base na função *adc_capture* das *drivers* fornecidas. E posteriormente

enviado para o MATLAB através da interface UART.

Este capítulo revelou os aspetos mais relevantes da implementação num sistema físico do modem desenvolvido. No capítulo seguinte serão apresentados um conjunto de resultados que validam o seu correto funcionamento.

Capítulo 6

Resultados Experimentais

Este capítulo tem como objetivo principal apresentar os resultados obtidos, através de uma série de medições, que permitem observar o funcionamento do trabalho desenvolvido durante a dissertação.

6.1 Introdução

A validação comportamental do sistema desenvolvido é realizada através da ferramenta computacional MATLAB, na qual foi implementado um modem de modulações variáveis. No entanto, para testar a robustez e sensibilidade que o sistema apresenta é indispensável valida-lo em ambiente real, em vez de se confiar meramente em dados teóricos.

Utilizando uma interface entre o MATLAB e o sistema implementado fisicamente, tal como a descrita no capítulo anterior, é possível realizar a devida validação do sistema desenvolvido em MATLAB utilizando dados práticos.

Esta abordagem, para além da validação e análise do comportamento dos diferentes módulos integrados em MATLAB, permite ainda realizar a sua implementação em *hardware* de uma forma progressiva. Isto é, o processamento do modem pode ser repartido entre o MATLAB e sistema implementado em FPGA.

Implementando os vários módulos em FPGA de uma forma sequencial, estes podem ser testado isoladamente, reduzindo assim a complexidade do sistema completo e tirando proveito do sistema em MATLAB para realizar validações mais precisas e completas.

6.2 Co-simulação com *frontend* RF

Antes da implementação do sistema em FPGA validou-se primeiro o seu funcionamento realizando o processamento do sinal em MATLAB e utilizando o kit de desenvolvimento ZedBoard e o *frontend AD-FMCOMMS1-EBZ* para realizar a sua transmissão e respetiva receção.

O sinal de informação a ser transmitido é processado pela cadeia de transmissão em MATLAB e posteriormente fornecido ao kit de desenvolvimento. Por sua vez, este transmite-o numa configuração de *lookback* através do *frontend* e devolve o sinal recebido ao MATLAB, onde é recuperada a informação do sinal, tal como é ilustrado na Figura 6.1.

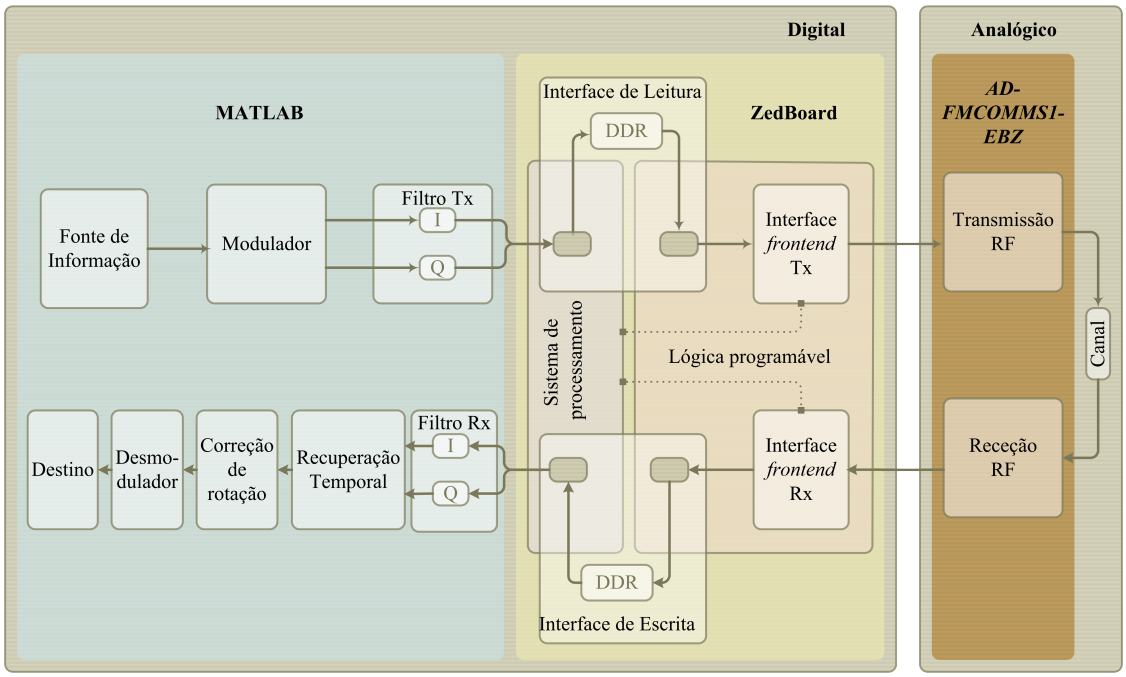


Figura 6.1: Diagrama de blocos da co-simulação do modem desenvolvido em MATLAB

O correto funcionamento do sistema implementado em MATLAB para uma modulação QPSK em ambiente real pode ser verificado na Figura 6.2.

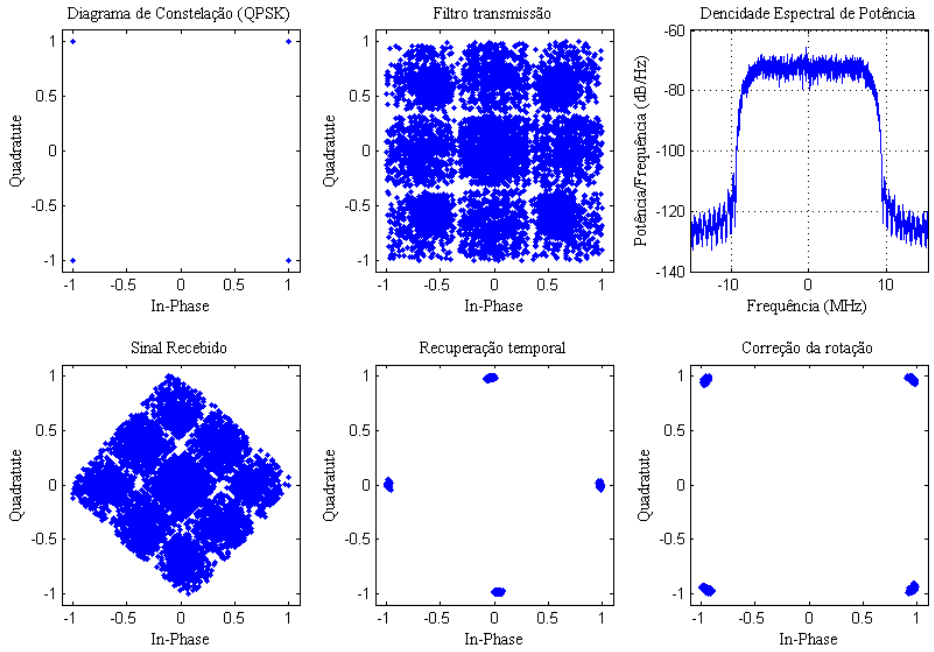


Figura 6.2: Co-simulação de Modem QPSK

Nesta figura, pode-se observar, através de diagramas de constelação, o sinal em vários pontos do sistema. O sinal de informação é gerado aleatoriamente com aproximadamente 22 mil *bits*, as configurações dos filtros utilizados estão representadas nas Tabelas 3.1 e 3.1, as especificações do *frontend* são apresentadas na Tabela 5.2 e as janelas de observação (L_0) das rotinas de recuperação temporal e de correção de fase é de 1024 amostras.

Como se pode observar, o sinal recebido vem afetado com uma rotação na sua constelação, contudo a cadeia de receção consegue recuperar o sinal com um BER de 0% e um EVM de 1.336%.

Na Figura 6.3 é apresentado um outro exemplo que utiliza as mesmas configurações do caso anterior, no entanto sofre uma rotação na sua constelação diferente, este fenómeno deve-se principalmente aos deslocamentos de fase entre os osciladores locais. Neste exemplo obteve-se um BER de 0%, e um EVM de 1.493%.

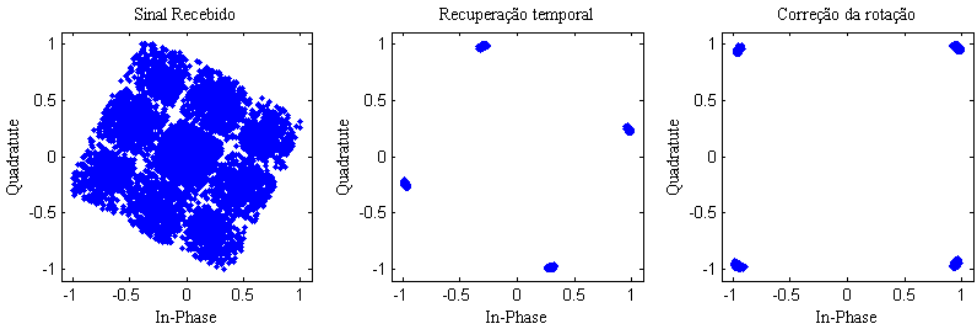


Figura 6.3: Co-simulação de Modem QPSK

6.3 Cadeia de Transmissão

Uma vez testado e verificado o correto funcionamento do modem desenvolvido em MATLAB, é implementada a cadeia de transmissão em FPGA, mantendo o resto do processamento a ser realizado em MATLAB.

Para realizar o processamento da cadeia de transmissão em FPGA são implementados os filtros da transmissão e o módulo do modulador descrito no capítulo anterior segundo a configuração ilustrada na Figura 6.4.

Os resultados obtidos com esta configuração, mantendo os mesmos parâmetros do caso anterior podem ser observados nas Figuras 6.5 e 6.6, onde os sinais após a sua transmissão foram fornecidos ao MATLAB que recuperou a sua informação com BER de 0% e EVM de 1.574% e 1.677%, respetivamente.

Com as especificações escolhidas a informação é transmitida a um ritmo de 15.36 M amostras por segundo, uma vez que o sinal é sobre-amostrado pelo filtro de transmissão equivale a 7.68M símbolos por segundo e como cada símbolo representa 2 bits, chega-se ao ritmo de transmissão de 15.36M (bits/segundo).

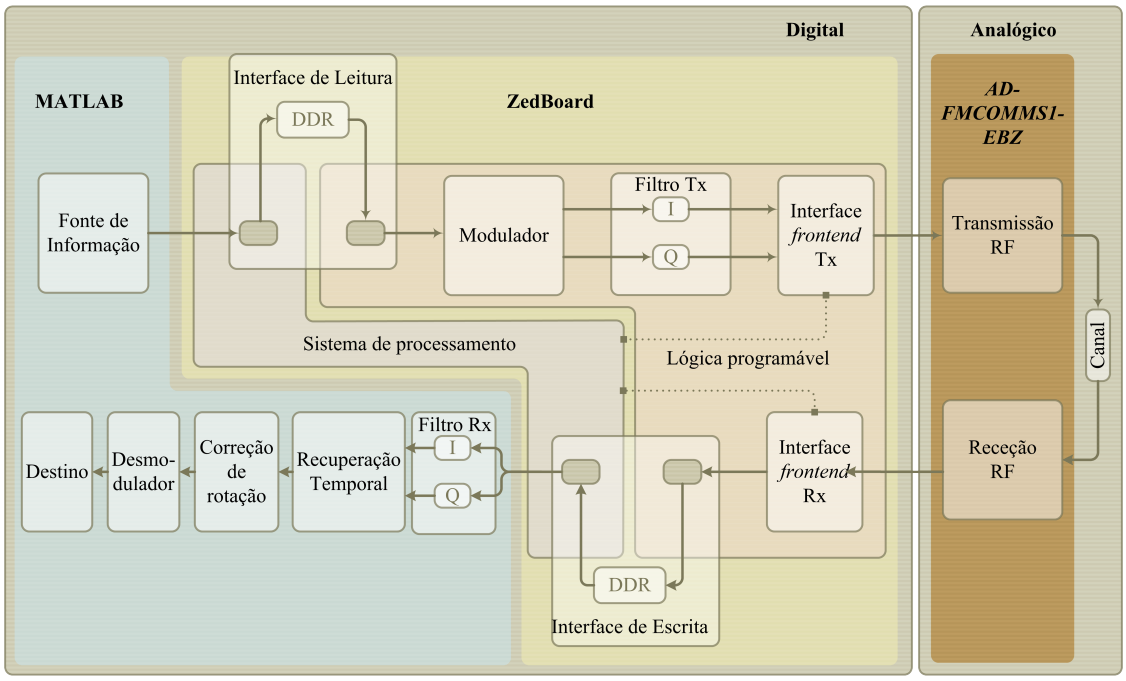


Figura 6.4: Diagrama de blocos do cenário de teste da cadeia de transmissão em FPGA

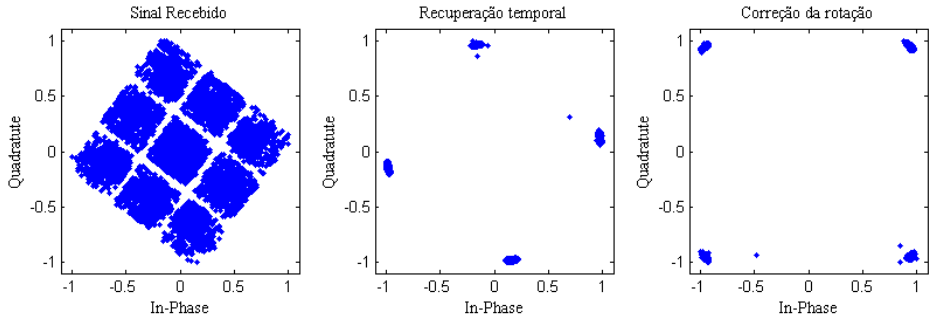


Figura 6.5: Validação da cadeia de transmissão em FPGA

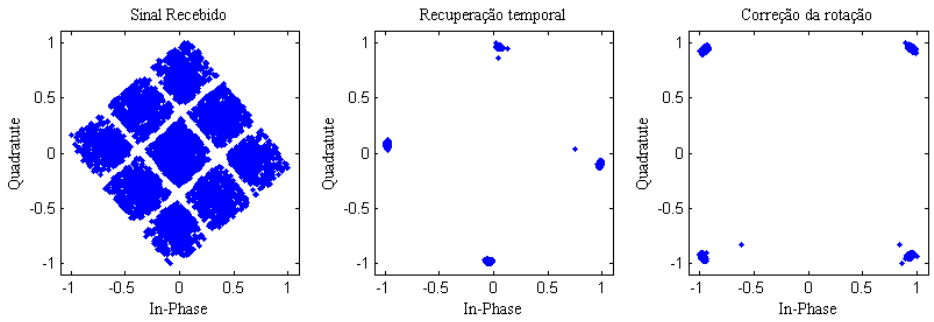


Figura 6.6: Validação da cadeia de transmissão em FPGA

A largura de banda do sinal transmitido pode ser relacionada com o ritmo de transmissão e o fator de *roll-off* do filtro, tal como indica a fórmula (3.1). Desta forma, uma vez que o fator de roll-*off* do filtro é de 0.22 a largura de banda expectável será de 18.74 MHz.

De forma a avaliar o desempenho do módulo da filtragem, conectou-se a saída do *frontend* a um analisador de espetros, o FSH8 da Roshde&Schwarz [Roh15], e observou-se o sinal transmitido no espetro de frequências. O resultado obtido pode ser visualizado na Figura 6.7.

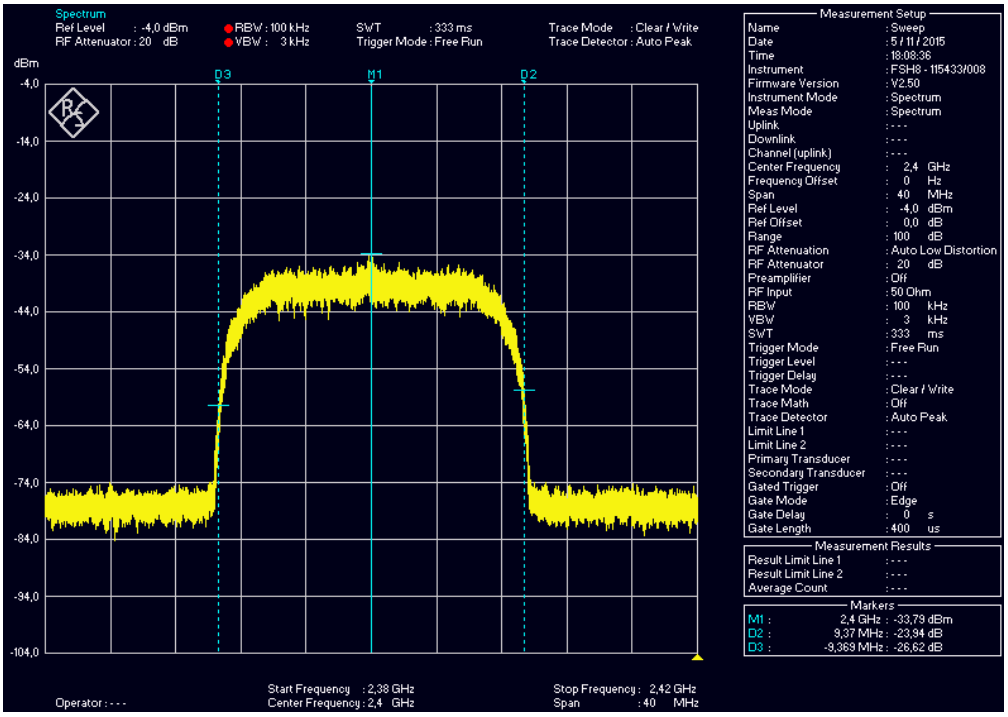


Figura 6.7: Espetro de frequências do sinal transmitido

Tal como se pode verificar a resposta em frequência do sinal foi eficientemente limitada para a largura pretendida.

6.4 Cadeia de Receção

A cadeia de receção tem como função recuperar a informação contida no sinal enviado pela cadeia de transmissão, e para tal necessita de compensar as distorções que lhe são provocadas. A performance de todo o sistema depende da eficiência como esta cadeia consegue recuperar essa informação.

6.4.1 Recuperação Temporal

Uma das compensações cruciais para um bom desempenho das comunicações é a recuperação do sincronismo e é realizada pelo *IPcore timing_recovery*.

De modo a validar o funcionamento deste módulo, utilizou-se a interface com o MATLAB para testar este módulo de uma forma individual. Assim sendo, instanciou-se este IPcore

juntamente com o filtro da receção num projeto isolado. Neste projeto, a cadeia de transmissão era processada em MATLAB e o seu resultado fornecido diretamente ao módulo sem ser transmitido pelo *frontend*, tal como demonstra a Figura 6.8.

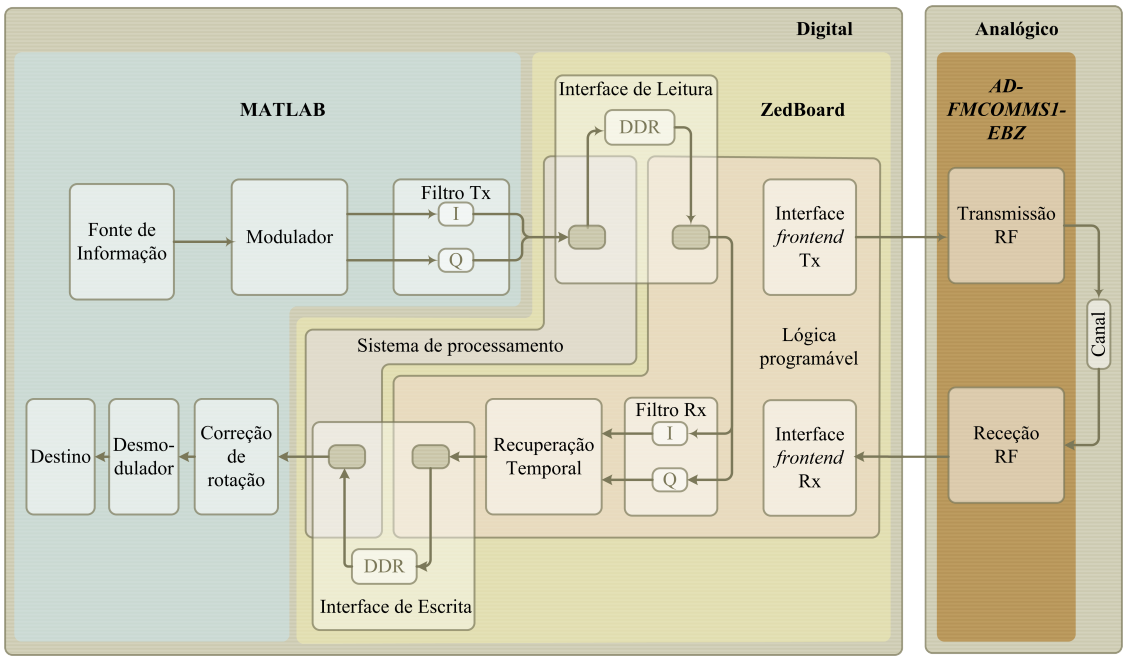


Figura 6.8: Diagrama de blocos do cenário de teste do módulo de Recuperação Temporal e do Filtro Rx

O diagrama de constelação obtido pode ser visualizado na Figura6.9. Com este projeto conseguiu-se atingir um EVM de 0.6385% em que o EVM gerado pelo mesmo circuito implementado em MATLAB foi de 0.6089%.

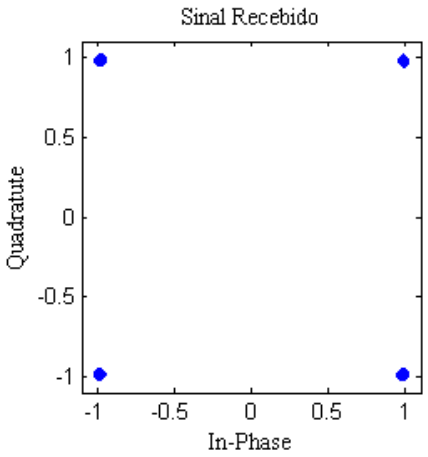


Figura 6.9: Validação do circuito de recuperação temporal em FPGA

Verificado o correto funcionamento do módulo numa situação ideal, isto é, em que o sinal não sofria as interferências da transmissão, verificou-se o seu funcionamento em ambiente real

com os módulos precedentes também implementados em FPGA. Tal como é representado na Figura 6.10.

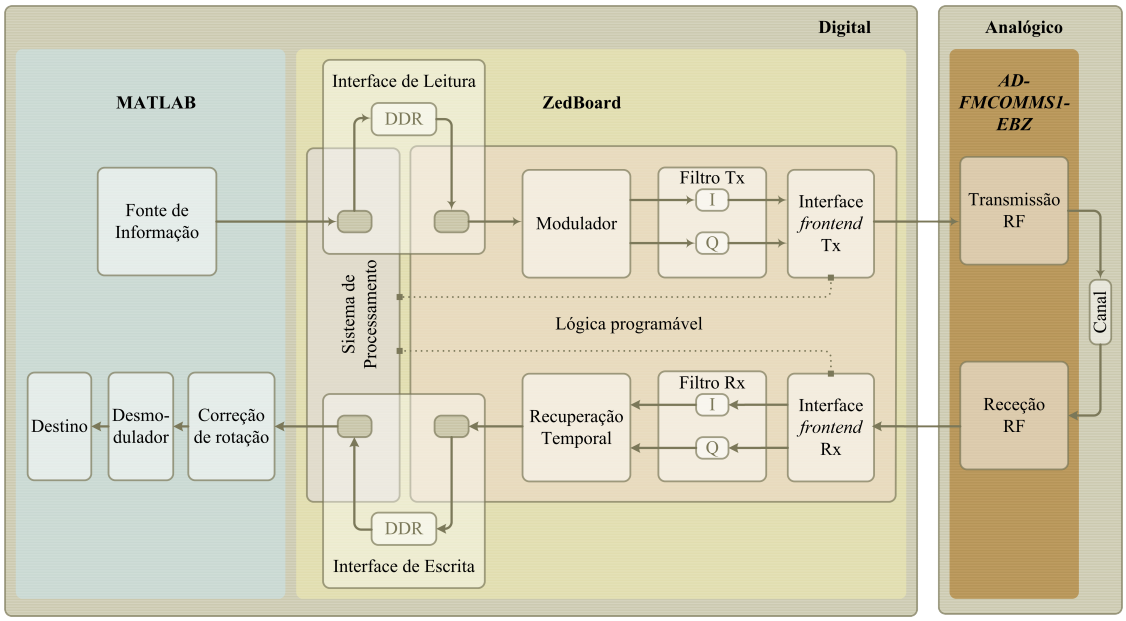


Figura 6.10: Diagrama de blocos do cenário de teste do modem implementado em FPGA até ao recuperador temporal

Os diagramas de constelação obtidos são apresentados nas Figuras 6.11 e 6.12, onde se atingiram EVMs de 1.925% e 2%, respetivamente.

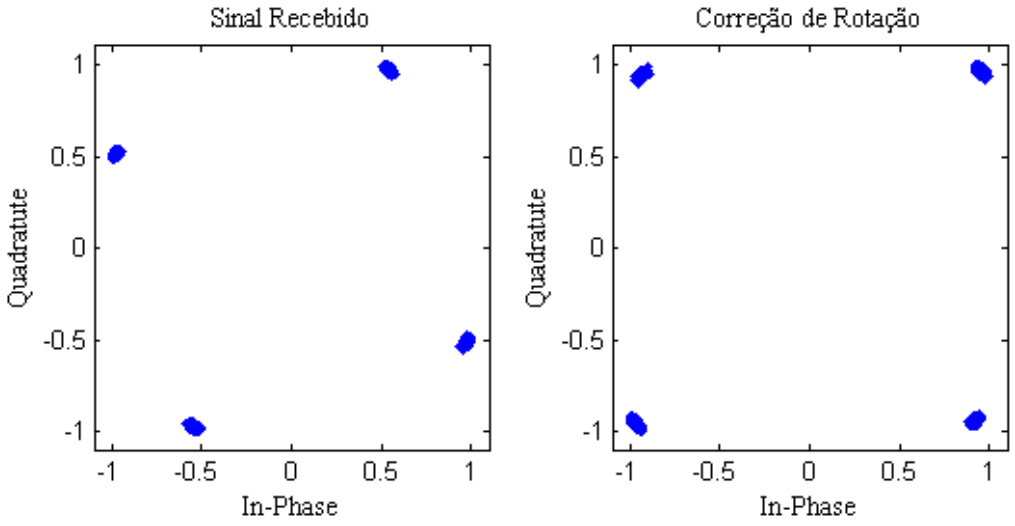


Figura 6.11: Validação do circuito de recuperação temporal com dados reais

Realizando uma média de 10 ensaios utilizando este cenário de teste obteve-se um EVM médio de 2.03%, enquanto que para as mesmas configurações, obteve-se um EVM médio de 1.41% no cenário em que o modem era integralmente implementado em MATLAB.

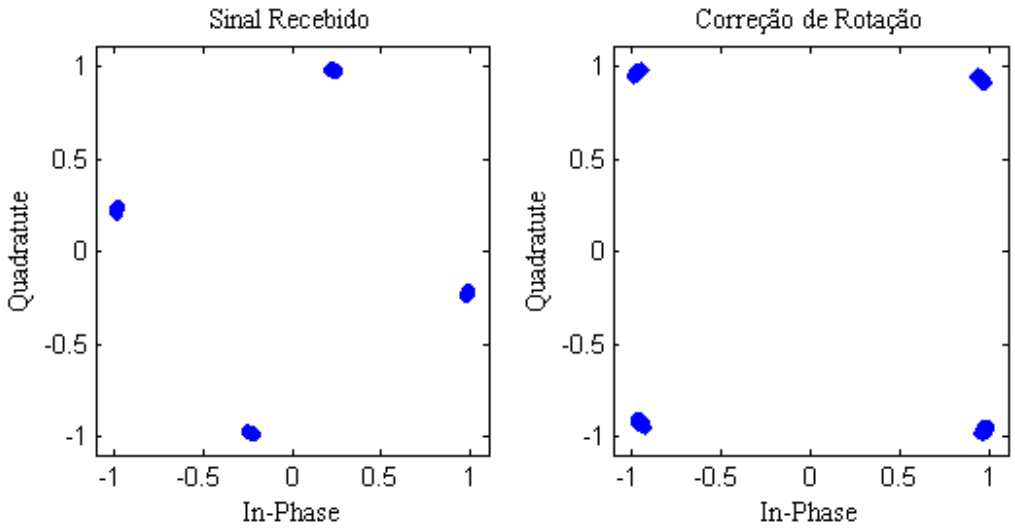


Figura 6.12: Validação do circuito de recuperação temporal com dados reais

---

Terminada a exposição dos resultados obtidos nesta dissertação de mestrado, segue o capítulo com as suas conclusões.

Capítulo 7

Conclusão e Trabalho Futuro

Neste capítulo, são apresentadas as conclusões mais importantes resultantes deste trabalho e referidas algumas possíveis alterações, de forma a ser possível melhorar o trabalho futuro.

7.1 Conclusão

Esta dissertação de mestrado teve como objetivo o desenvolvimento e implementação em FPGA de um modem baseado em abordagem SDR.

Este projeto visa a sua integração num sistema de comunicação a implementar num UAV, pelo que se pretendia um sistema de rádio de elevada robustez e que permitisse manter a ligação mesmo em cenários adversos. Deste modo, uma das grandes preocupações no seu decorrer foi o desenvolvimento e implementação de algoritmos que compensassem os efeitos destrutivos que os canais podem provocar aos sinais transmitidos.

O trabalho iniciou-se com uma análise dos fundamentos teóricos dos sistemas de comunicações digitais, mais precisamente de modems digitais. De seguida foi desenvolvido o modelo de um modem em MATLAB adaptável a qualquer modulação ortogonal, onde foram integrados e validados algoritmos de recuperação temporal e de correção de rotação implementados também em MATLAB.

Após validar o correto funcionamento do sistema em MATLAB e averiguar o comportamento dos vários módulos, realizou-se uma análise para determinar a forma como estes poderiam ser implementados em *hardware*.

Para a sua implementação foi utilizando o kit de desenvolvimento ZedBoard e o *frontend* AD-FMCOMMS1-EBZ.

A sua implementação foi realizada de uma forma progressiva e validada através da interface com o modem implementado em MATLAB. Esta interface permitiu testar várias componente do modem implementado em FPGA realizando o processamento restante do modem em MATLAB, permitindo assim validar os vários módulos individualmente.

O modelo em MATLAB é utilizado como termo de comparação uma vez que apenas é relevante o comportamento do sistema e o processamento dos dados em virgula flutuante de precisão dupla. Comparando os valores de EVM obtidos na implementação em *hardware* com os resultados do modelo em MATLAB observa-se uma ligeira degradação, explicada pela quantização dos dados em vírgula fixa.

Nos testes realizados utilizando uma configuração *lookback* com *frontend* RF, para as mesmas condições o modem implementado integralmente em MATLAB obteve um EVM

médio de 1.41%, com a implementação da cadeia de transmissão em FPGA e o restante processamento realizado em MATLAB obteve-se um EVM médio de 1.64%. Com a cadeia de transmissão, filtros de receção e recuperador temporal implementados em FPGA e a correção de rotação e desmodulador implementados em MATLAB obteve-se um EVM médio de 2.03%.

7.2 Trabalho Futuro

Este trabalho pode ser considerado como uma etapa preparatória para a construção de um modem flexível que adapte a sua modulação de forma a melhorar a sua eficiência de comunicação, contudo existem ainda alguns aspetos que precisão de ser melhorados e completados, entre eles:

- Concluir a implementação do algoritmo de correção de fase em FPGA;
- Adicionar tabelas mapeadas com novas modulações e validar o correto funcionamento do modem para essas novas modulações;
- Otimizar a utilização dos filtros da transmissão e receção em termos de eficiência e área de utilização em FPGA;
- Implementação de circuitos de correção de erros.

Anexos

Anexo A

Customização de *IPcores*

Neste Anexo é ilustrada a interface do *customize IP* do Vivado utilizada para criar os vários *IPcores* da Xilinx instanciados.

A.1 *IPcore TX_fir*

Este *IPcore* foi criado a partir do *FIR Compiler(7.2)* desenvolvido pela Xilinx e pode encontrado através do *IP Catalog* no Vivado.

Este *IPcore* é o responsável pela implementação dos filtros da cadeia de transmissão, no entanto os filtros da receção são criados do mesmo modo.

Nas Figuras A.1, A.2, A.3 e A.4 é ilustrada a interface do *customize IP* do módulo criado.

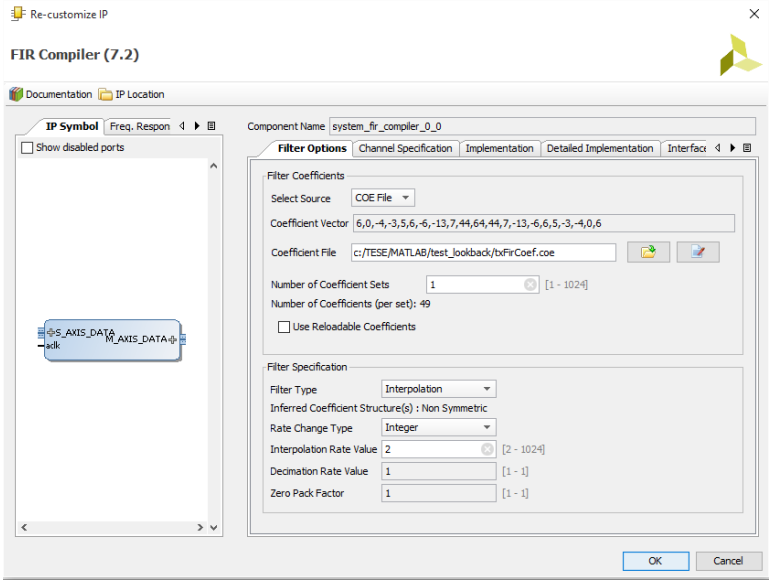


Figura A.1: Instanciação do *IPcore TX_fir*(1)

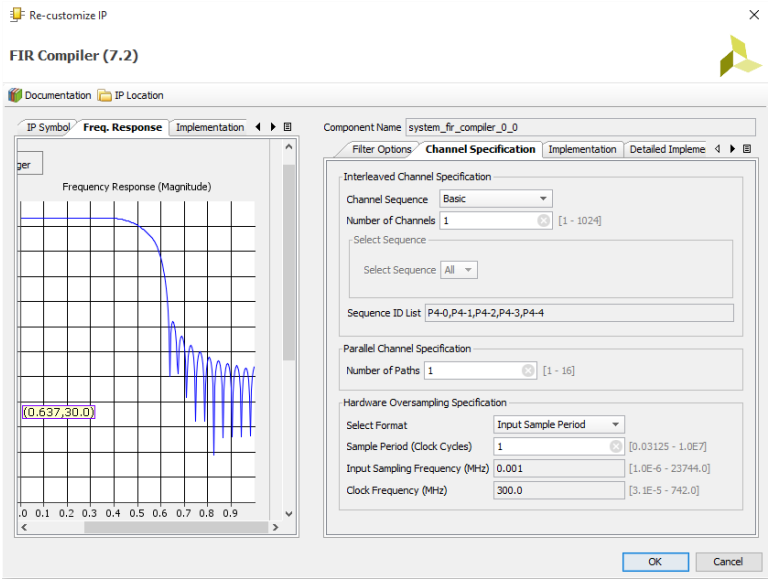


Figura A.2: Instanciação do *IPcore TX_fir*(2)

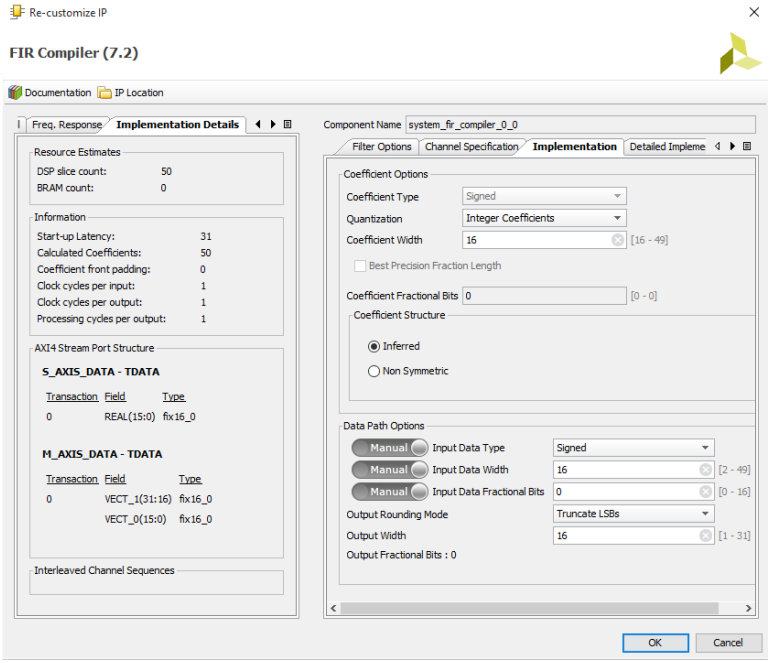


Figura A.3: Instanciação do *IPcore TX_fir*(3)

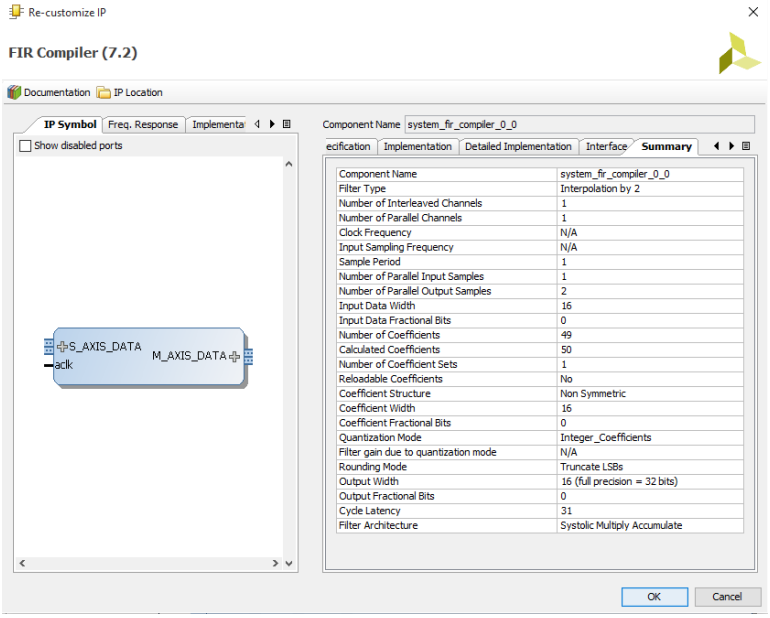


Figura A.4: Instanciação do *IPcore TX_fir*(4)

A.2 *IPcore multiplier_16x16*

Este *IPcore* foi criado a partir do *Multiplier(12.0)* desenvolvido pela Xilinx que pode ser encontrado através do *IP Catalog* no Vivado.

É o responsável pela implementação de multiplicadores com operadores de 16 *bits* e o resultado apresentado em 32 *bits*.

Nas Figuras A.5 e A.6 é ilustrada a interface do *customize IP* do *IPcore* criado.

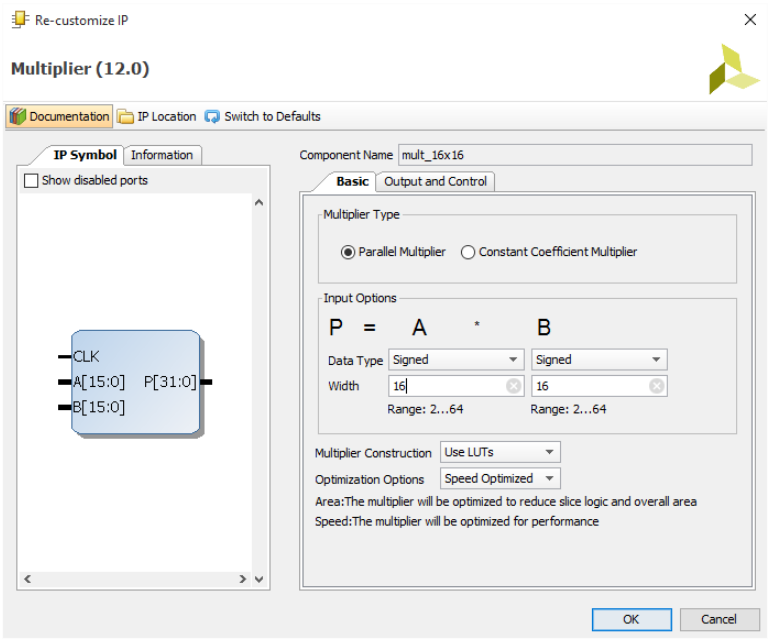


Figura A.5: Instanciação do *multiplier_16x16* (1)

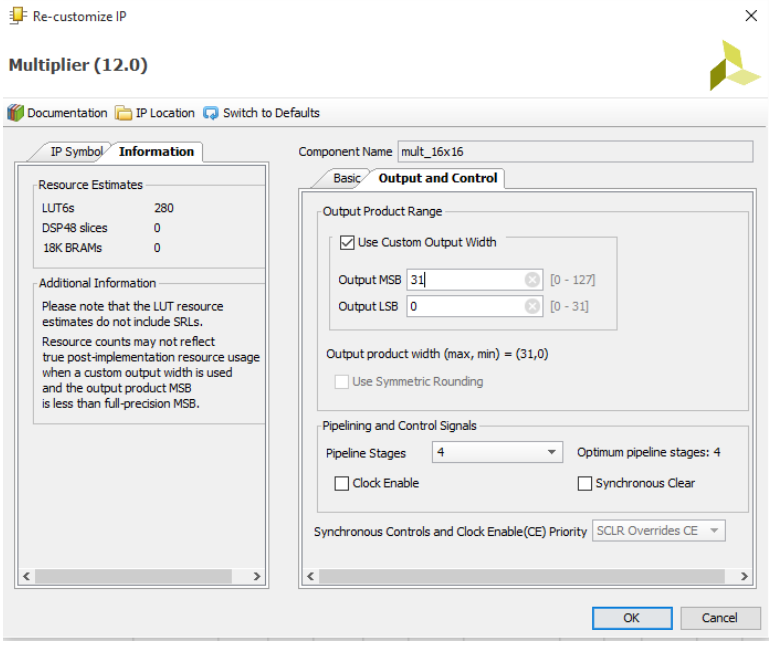


Figura A.6: Instanciação do *multiplier_16x16* (2)

A.3 *IPcore cordic_ArcTan*

Este *IPcore* foi criado a partir do *CORDIC(6.0)* desenvolvido pela Xilinx que pode ser encontrado através do *IP Catalog* no Vivado.

É utilizado na implementação da função *arg*.

Nas Figuras A.7 e A.8 é ilustrada a interface do *customize IP* do *IPcore* criado.

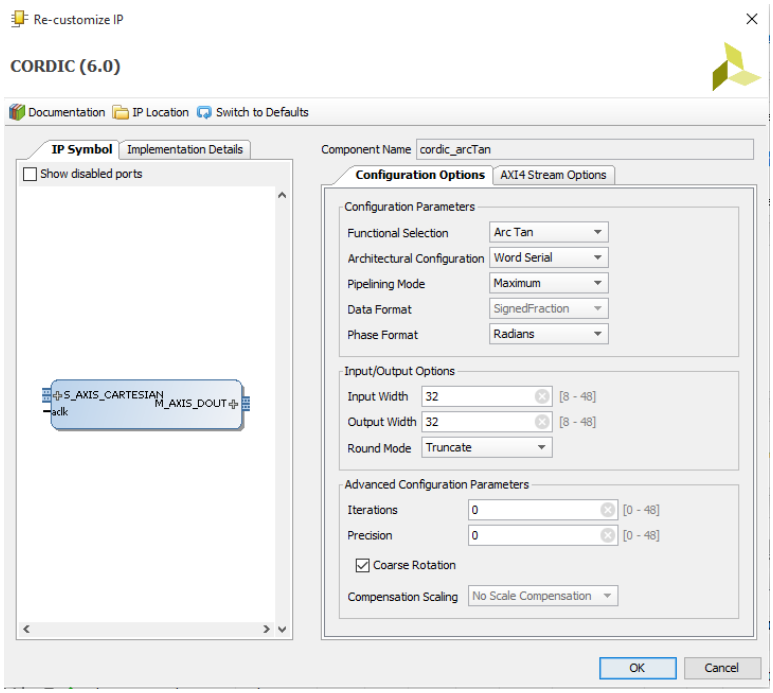


Figura A.7: Instanciação do *IPcore cordic_ArcTan* (1)

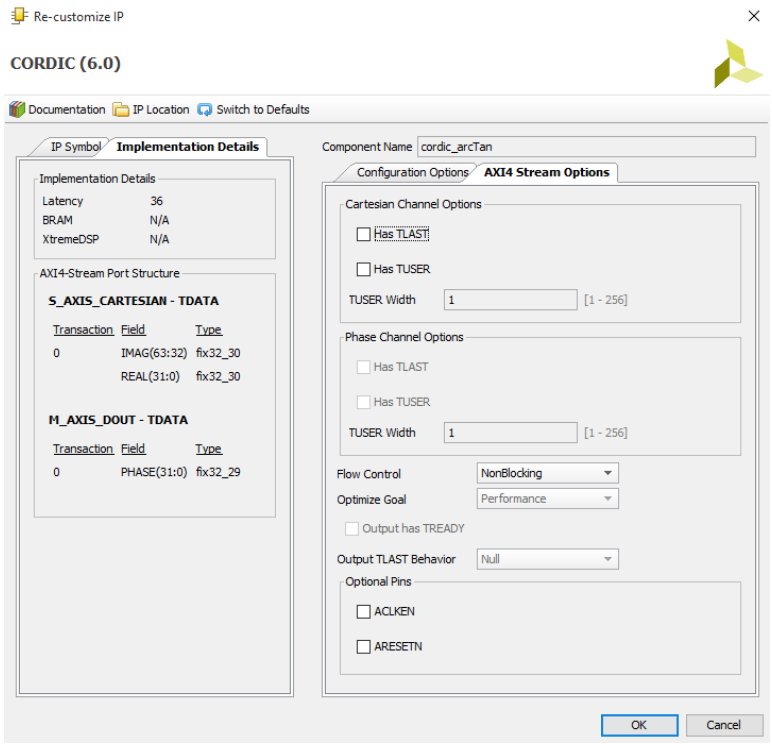


Figura A.8: Instanciação do *IPcore cordic_ArcTan* (2)

Bibliografia

[AVN14] AVNET. *ZedBoard (Zynq Evaluation and Development), Hardware User's Guide*. AVNET, http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf, 2014.

[Bat99] Andy Bateman. *Digital Communications*. Addison Wesley Longman, 1999.

[Dev] Analog Devices. *AD-FMCOMMS1-EBZ Functional Overview*. Analog Devices, https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms1-ebz/hardware/functional_overview.

[Dev07] Analog Devices. *400 MHz to 6 GHz Broadband Quadrature Modulator: ADL5375*. Analog Devices, http://www.analog.com/media/en/technical-documentation/evaluation-documentation/ADL5375.pdf, 2007.

[Dev09a] Analog Devices. *400 MHz to 6 GHz Quadrature Demodulator: ADL5380*. Analog Devices, http://www.analog.com/media/en/technical-documentation/data-sheets/ADL5380.pdf, 2009.

[Dev09b] Analog Devices. *50 MHz to 4.0 GHz RF/IF Gain Block: ADL5602*. Analog Devices, http://www.analog.com/media/en/technical-documentation/evaluation-documentation/ADL5602.pdf, 2009.

[Dev09c] Analog Devices. *Dual, 16-Bit, 1230 MSPS, TxDAC+ Digital-to-Analog Converter: AD9122*. Analog Devices, http://www.analog.com/media/en/technical-documentation/data-sheets/AD9122.pdf, 2009.

[Dev10] Analog Devices. *DC to 600 MHz, Dual-Digital Variable Gain Amplifier: AD8366*. Analog Devices, http://www.analog.com/media/en/technical-documentation/evaluation-documentation/AD8366.pdf, 2010.

[Dev11] Analog Devices. *14-Bit, 170 MSPS/210 MSPS/250 MSPS, 1.8V Dual Analog-to-Digital Converter(ADC): AD9643*. Analog Devices, http://www.analog.com/media/en/technical-documentation/data-sheets/AD9643.pdf, 2011.

[Dev12] Analog Devices. *Wideband Synthesizer with Integrated VCO: ADF4351*. Analog Devices, http://www.analog.com/media/en/technical-documentation/data-sheets/ADF4351.pdf, 2012.

[Dev15a] Analog Devices. *AD-FMCOMMS1-EBZ Quick Start Guide on Xilinx FPGA Boards Without OS*. https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms1-ebz/quickstart/no_os_microblaze, 2015.

[Dev15b] Analog Devices. *ADI Reference Designs HDL User Guide*. https://wiki.analog.com/resources/fpga/docs/hdl#building_on_vivado, 2015.

[Ele] Digi-Key Electronics. Ad-fmcomms1-ebz-nd. http://www.digikey.com/product-detail/en/AD-FMCOMMS1-EBZ/AD-FMCOMMS1-EBZ-ND/3542764.

[Hay01] S. Haykin. *Communication Systems*. Wiley, 4th edition, 2001.

[MD13] U. Mengali and A. N. D'Andrea. *Synchronization Techniques for Digital Receivers*. Applications of Communications Theory. Springer US, 2013.

[NS09] H. H. Nguyen and E. Shwedyk. *A First Course In Digital Communications*. Cambridge University Press, 2009.

[OM88] M. Oerder and H. Meyr. Digital filter and square timing recovery. *Communications, IEEE Transactions on*, 36(5):605–612, May 1988.

[PS15] B. B. Purkayastha and K. K. Sarma. *A Digital Phase Locked Loop based Signal and Symbol Recovery System for Wireless Channel*. Signals and Communication Technology. Spinger India, 2015.

[Ric09] M. Rice. *Digital Communications: A Discrete-time Approach*. Pearson/Prentice Hall, 2009.

[Roh15] Rohde&Schwarz. *R&S®FSH Handheld Spectrum Analyzer: Specifications*. https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/FSH_dat-sw_en_5214-0482-22_v1900.pdf, February 2015.

[SDR] SDR forum. *SDRF Cognitive Radio Definitions*. http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf, November 2007.

[Skl01] B. Sklar. *Digital Communications: Fundamentals and Applications*. Prentice Hall Communications Engineering and Emerging Technologies Series. Prentice-Hall PTR, 2001.

[Sui15] Vivado Design Suite. *AXI Reference Guide*. XILINX, http://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf, 2015.

[Vit83] A. Viterbi. Nonlinear estimation of psk-modulated carrier phase with application to burst digital transmission. *Information Theory, IEEE Transactions on*, 29(4):543–551, Jul 1983.

[Wes09] K. Wesolowski. *Introduction to Digital Communication Systems*. WILEY, 2009.

[XIL] XILINX. *Defining Coefficient Values in a COE File*. XILINX, http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/cgn_c_define_values_coe_file.htm.

[XIL15a] XILINX. *CORDIC v6.0: LogiCORE IP Product Guide*. Vivado Design Suite, http://www.xilinx.com/support/documentation/ip_documentation/cordic/v6_0/pg105-cordic.pdf, November 2015.

[XIL15b] XILINX. *FIR Compiler v7.2: LogiCORE IP Product Guide*. Vivado Design Suite, http://www.xilinx.com/support/documentation/ip_documentation/cordic/v6_0/pg105-cordic.pdf, November 2015.

[XIL15c] XILINX. *Multiplier v12.0: LogiCORE IP Product Guide*. Vivado Design Suite, http://www.xilinx.com/support/documentation/ip_documentation/mult_gen/v12_0/pg108-mult-gen.pdf, November 2015.

[Zed] ZedBoard. ZedBoard. http://zedboard.org/product/zedboard.