Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Eletrônica

Implementação de um Modulador para DVB-RCS2 em FPGA

Autor: Davi Antônio da Silva Santos
Orientador: Professor Doutor Daniel Mauricio Muñoz Arboleda

Brasília, DF
2019

Davi Antônio da Silva Santos

Implementação de um Modulador para DVB-RCS2 em FPGA

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Professor Doutor Daniel Mauricio Muñoz Arboleda

Brasília, DF

2019

Davi Antônio da Silva Santos
Implementação de um Modulador para DVB-RCS2 em FPGA/ Davi Antônio da Silva Santos. – Brasília, DF, 2019-
86 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Daniel Mauricio Muñoz Arboleda

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2019.

1. Modulação digital. 2. DVB-RCS2. I. Professor Doutor Daniel Mauricio Muñoz Arboleda. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementação de um Modulador para DVB-RCS2 em FPGA

CDU 02:141:005.6

Davi Antônio da Silva Santos

Implementação de um Modulador para DVB-RCS2 em FPGA

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 12 de dezembro de 2019 – Data da aprovação do trabalho:

Professor Doutor Daniel Mauricio Muñoz Arboleda
Orientador

Professor Doutor Wellington Avelino do Amaral
Convidado 1

Professor Doutor Gilmar Silva Beserra
Convidado 2

Brasília, DF
2019

Agradecimentos

A Deus, por ter me dado força e saúde para superar as dificuldades.

À minha família, pelo amor e apoio incondicionais.

Aos professores, pelo conhecimento transmitido e pelos desafios propostos.

Ao meu orientador, pelo apoio e confiança.

Aos amigos e colegas, pelo apoio e pelos conselhos.

A todos que contribuíram direta ou indiretamente para a minha formação.

"O temor do Senhor é o princípio sabedoria;
e o conhecimento do Santo é o entendimento.
(Bíblia Sagrada, Provérbios 9, 10)

Resumo

Este trabalho objetiva implementar em um FPGA as quatro modulações lineares do protocolo DVB-RCS2: $\pi/2$-BPSK, QPSK, 8PSK e 16QAM, em banda base e usando o filtro formatador de pulsos exigido pela norma, um cosseno levantado com raiz quadrada (SRRC) com fator de *roll-off* de 0,2. Foram construídos scripts interpretados pelo GNU/Octave para produção de valores de referência, geração de símbolos de teste e comparação dos dados experimentais com os modelos de referência. A quantização dos coeficientes do filtro e dos símbolos usados nas constelações foi feita em ponto fixo, sendo a quantidade de bits determinada através de uma análise de precisão numérica usando o GNU/Octave para comparar o erro quadrático médio entre os valores em ponto flutuante e os quantizados. Um script também foi desenvolvido para automatizar a descrição em VHDL do filtro SRRC a partir dos coeficientes do filtro, dos bits de quantização e do fator de *oversample*. O funcionamento do sistema descrito em VHDL foi verificado através de simulações comportamentais e da implementação com uma arquitetura com memórias que possuem vetores de teste em um kit de desenvolvimento. As simulações comportamentais do sistema com a arquitetura de testes a uma frequência de operação de 125 MHz informam que o sistema possui latência de 376 ns e taxa de transmissão de símbolos de 17,5 MHz. O modulador e uma arquitetura de testes descritos em VHDL e o hardware de instrumentação, usado para captura de dados, foram implementados em um FPGA XC7Z010-1CLG400C, presente no kit de desenvolvimento Zybo Revisão B a uma frequência de operação de 125 MHz. Verificou-se que o sistema sem o hardware de instrumentação inserido pelo *Integrated Logic Analyser* (ILA) pode ser implementado a uma frequência de 166,67 MHz, sendo que, na frequência padrão, são consumidos 3921 *Look-Up Tables* (LUTs), 4547 registradores, 2 memórias em bloco, 38 entradas e saídas, uma linha global de *clock* e 121 mW de potência. Os dados obtidos pelo ILA foram comparados com os modelos de referência e encontrou-se um erro quadrático médio da ordem de 10^{-6} para as partes real e imaginária das modulações implementadas.

Palavras-chaves: modulação digital. FPGA. VHDL. DVB-RCS2.

Abstract

This work intends to implement the four linear modulations employed in the DVB-RCS2 protocol: $\pi/2$-BPSK, QPSK, 8PSK e 16QAM. The modulations will be implemented in baseband using a Square Root Raised Cosine Filter for pulse formatting with a roll-off factor of 0,2. GNU/Octave scripts were developed for reference values and test vectors generation, and to compare experimentally obtained data with the reference models. The filter coefficients and the constellation symbols were quantised using fixed point. The bit depth used in the quantisation process was determined by a numerical precision analysis that used GNU/Octave to compare the mean square error between the floating-point values and the quantised ones. A script was developed in order to automatise the generation of the VHDL description of the SRRC filter based on its coefficients, the number of bits used for quantisation and the oversample factor. The system was validated using behavioural simulations and one hardware implementation in a development kit. This implementation contains one architecture with memories containing test vectors. The behavioural simulations were performed at a 125 MHz frequency and revealed the system's latency and symbol transmission rate: 376 ns and 17,5 MHz. The modulator and a test architecture, both described using VHDL, were implemented in a XC7Z010-1CLG400C FPGA contained in the Zybo Rev. B development kit at the operating frequency of 125 MHz. Verifications showed that the system without the instrumentation hardware inserted by the Integrated Logic Analyser (ILA) can be implemented at a 166,67 MHz operating frequency. Running at the standard 125 MHz frequency the system uses 3921 Look-Up Tables (LUTs), 4547 registers, 2 block memories, 38 input and output pins, one global clock buffer and 121 mW of power. The data obtained using the ILA was compared with the reference models and the average quadratic error found was close to 10^{-6} for the real and imaginary terms of the implemented modulations.

Key-words: digital modulation. FPGA. VHDL. DVB-RCS2.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ALPDU	*Addressed Link Protocol Data Unit*
ASIC	*Application Specific Integrated Circuits*
AXI	*Advanced Extensible Interface*
BPSK	*Binary Phase Shift Keying*
CCITT	*Comité Consultatif International Téléphonique et Télégraphique*
CPM	*Continuous Phase Modulation*
CRC	*Cyclic Redundancy Check*
CSV	*Comma Separated Values*
DAC	*Digital-to-Analog Converter*
DDS	*Direct Digital Synthesis*
DFS	*Digital Frequency Synthesizer*
DPSK	*Differential Phase-Shift Keying*
DSP	*Digital Signal Processor*
DVB-RCS2	*Digital Video Broadcasting - Interactive Satellite System, Return Channel by Satellite second generation*
DVB	*Digital Video Broadcast*
FEC	*Forward Error Correction*
FF	*Flip-Flop*
FIR	*Finite Impulse Response*
FPDU	*Frame PDU*
FPGA	*Field-Programmable Gate Array*
FSM	*Finite State Machine*
GPRS	*General Packet Radio Service*
GSE	*Generic Stream Encapsulation*

HDL	*Hardware Description Language*
ILA	*Integrated Logic Analyser*
IP	*Internet Protocol*
LE	*Logical Element*
LUT	*Look-Up Table*
MF-TDMA	*Multi-Frequency Time Division Multiple Access*
MIP	*Mobile Internet Protocol*
MUX	*Multiplexer*
NCO	*Numerically Controlled Oscillator*
PDU	*Protocol Data Unit*
PPDU	*Payload-adapted Protocol Data Unit*
PSK	*Phase-Shift Keying*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature Phase-Shift Keying*
RCS	*Return Channel by Satellite*
RLE	*Return Link Encapsulation*
ROM	*Read-Only Memory*
SDU	*Service Data Unit*
SMMH	Síntese de Modelagem Matemática em Hardware
SoC	*System on* Chip
SRRC	*Square Root Raised Cosine Filter*
TC	*Turbo Coding*
VHDL	*Very High Speed Integrated Circuits Hardware Description Language*
WHS	*Worst Hold Slack*
WNS	*Worst Negative Slack*
WPWS	*Worst Pulse Width Slack*

Lista de símbolos

f_N	Frequência/Taxa de Nyquist
r	Fator de *roll-off*
f_s	Frequência/Taxa de amostragem (*Sampling frequency*)
T_s	Tempo de símbolo
f_{symb}, R_s	Frequência/Taxa de símbolos
R_b	Frequência/Taxa de bits
M	Símbolos em uma modulação m-ária
$\mathbb{N}$	Conjunto dos números naturais
B_T	Largura de banda da transmissão
π	Número pi
j	Unidade imaginária

Sumário

1 Introdução

Para garantir maior eficiência nas operações e maior segurança de veículos é comum que empresas e indivíduos contratem serviços de rastreamento veicular e gestão de veículos os quais, devido à integração cada vez maior dos mercados. Esses serviços possuem perspectivas de crescimento, o que por sua vez, deve levar a uma competição cada vez maior entre as provedoras de tais serviços.

Uma das maneiras de uma provedora de serviços de rastreamento e gestão de frotas se diferenciar em um mercado cada vez mais dinâmico é o fornecimento de serviços que possam ser integrados às redes existentes de comunicação satelital, que garantem maior cobertura se comparadas às redes GPRS, que dependem da disponibilidade das operadoras. Um desses serviços extras que se integram facilmente às redes já usadas para rastreamento satelital é o serviço de *Mobile IP* (MIP), que permite o fornecimento de conexão à internet para os dispositivos de rastreamento e pode ser implementado usando diversos padrões já consolidados no mercado, como o IP sobre DVB.

É importante ressaltar que o fornecimento do serviço de MIP deve ser feito de maneira que não afete a confiabilidade do serviço. Deste modo, é interessante que a implementação de um serviço que forneça MIP integrado ao rastreamento e gestão garanta o desempenho do sistema com uma taxa de transmissão esperada; seja robusto, garantindo a integridade das informações enviadas e recebidas; e permita fácil atualização remota, sem necessidade de trocas de *hardware* constantes.

1.1 Justificativa

O número de dispositivos móveis habilitados para conexão à Internet cresce constantemente. É interessante para o usuário uma alta disponibilidade da conexão a rede mundial de computadores, e isso pode ser possível com a capacidade de roteamento transparente presente no *Mobile IP* (MIP) (PERKINS, 1997). Essa característica garante que havendo um meio de conexão disponível, os dispositivos móveis continuem online mesmo ao deslocarem, estabelecendo novos enlaces e se desligando dos antigos (PERKINS, 1997).

O protocolo DVB-RCS2 consegue fornecer conexão *Mobile IP* de forma transparente ao equipamento do usuário através do *Proxy Mobile IP* (ETSI; EBU, 2014b). Isso garante o acesso à rede mundial com os benefícios de uma conexão MIP na área de cobertura da constelação de satélites.

A comunicação satelital usa IP sobre DVB, um padrão já estabelecido pelo mercado, e, conforme regem as normas (ETSI; EBU, 2014a) (ETSI; EBU, 2014c), o enlace

direto será recebido codificado no protocolo DVB-S2 e o retorno para o satélite é realizado através do DVB-RCS2.

Além do que já foi apresentado, é possível agregar os benefícios de um *Software Defined Radio* (SDR) implementando o DVB-RCS2 usando SoC (*System on* Chip) contendo um FPGA. Essa abordagem permitirá que o sistema possa ser reutilizado e atualizado com facilidade (WYGLINSKI et al., 2016).

Diante do exposto, sugere-se o desenvolvimento de um produto que forneça o serviço de MIP em conjunto de um *hardware* já existente que faça o rastreamento e realize as medidas de telemetria exigidas pelo sistema de gestão de frotas. Tal produto será baseado em lógica programável, mais especificamente em FPGA, permitindo o cumprimento dos requisitos citados anteriormente com a adição de permitir uma futura implementação em circuito integrado (ASIC). Além disso, oferecerá a possibilidade de atualização do sistema por métodos remotos, permitindo a expansão das funcionalidades da solução através da implementação de novos protocolos, codificações e modulações.

1.2 Objetivos

1.2.1 Objetivo geral

Este trabalho objetiva demonstrar a implementação de um sistema embarcado baseado em FPGA que implemente as modulações lineares em banda base do protocolo DVB-RCS2 para atuar como link de retorno de um produto que forneça o serviço de MIP usando IP sobre DVB.

1.2.2 Objetivos específicos

- Detalhar a organização geral da solução proposta;
  - Uma arquitetura para os moduladores;
  - Uma arquitetura para o filtro cosseno levantado com raiz quadrada (SRRC);
- Implementar em um FPGA as seguintes modulações com pulsos cosseno levantado com raiz quadrada, de acordo com as normas do DVB-RCS2 (ETSI; EBU, 2014a) (ETSI; EBU, 2014c):
  - Modulação binária em fase ($\pi/2$-BPSK);
  - Modulação em quadratura em fase (QPSK);
  - Modulação 8-ária em fase (8-PSK);
  - Modulação 16-ária em fase e amplitude (16QAM).

1.3 Organização do documento

O documento é composto por seis capítulos, sendo o primeiro o que contém a introdução, a justificativa, o objeto e os objetivos deste trabalho. No segundo capítulo, há fundamentação teórica, onde são discutidos trabalhos anteriores sobre o assunto e os conceitos gerais abordados na pesquisa. O terceiro apresenta a metodologia proposta, onde são apresentados os passos utilizados na elaboração do objeto do trabalho, simulações, fluxo de projeto e detalhes da implementação. Já o quarto capítulo trata dos resultados obtidos. Ao final, a discussão dos resultados comparando os dados obtidos nas simulações de alto nível feitas no GNU/Octave e a implementação em FPGA do modulador descrito em VHDL, e a conclusão.

2 Fundamentação Teórica

2.1 Modulação digital

A modulação digital objetiva a transmissão, por meio de um canal analógico, de dados digitais. Os sistemas de comunicação digitais podem ser separados em um diagrama de blocos simples, presente na figura 1. Observa-se na mesma que uma sequência de dígitos binários é a entrada do sistema, que será modulada em banda base digital por meio da codificação de linha. Os pulsos formatados são então modulados digitalmente por uma portadora, multiplexados, transmitidos pelo canal e recuperados pelo demodulador.

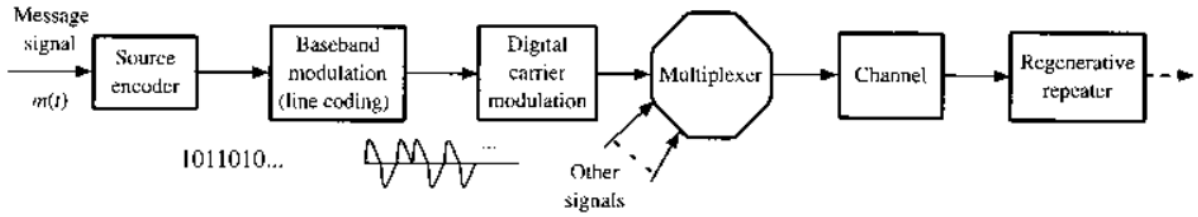

Figura 1 – Diagrama de blocos dos componentes de um sistema de modulação digital (LATHI; DING, 2010)

O processo de codificação de linha transforma a sequência de dados binária em formas de onda predeterminadas de modo a maximizar a facilidade em manter a informação íntegra e minimizar a banda de transmissão ocupada. Essa forma de onda então passa por um filtro formatador de pulsos que objetiva minimizar a interferência de um pulso com os seus vizinhos, chamada de Interferência entre Símbolos (IES).

$$\begin{cases} p(t) = 1 & \text{para} \quad t = 0 \\ p(t) = 0 & \text{para} \quad t = \pm nT_b \qquad n \in \mathbb{N}^* \\ R_b = \dfrac{1}{T_b} \end{cases} \tag{2.1}$$

A IES nula é garantida matematicamente por pulsos formatados que obedeçam ao Primeiro Critério de Nyquist para IES nula, disponível no conjunto de equações 2.1 (LATHI; DING, 2010), no qual T_b é o tempo de cada bit em segundos e R_b a taxa de transmissão por segundo. Esta equação permite deduzir que a mínima banda ocupada (B_T) para uma transmissão digital em banda base (baixa frequência) a uma taxa de R_b bits é fornecida por pulsos definidos no domínio do tempo pela função seno cardinal, e é

dada pela equação 2.2.

$$B_T = \frac{R_b}{2} \tag{2.2}$$

Entretanto, é difícil produzir filtros seno cardinal devido à grande janela necessária que decai com razão de $1/t$, o que o torna suscetível a problemas de sincronismo e gera maior consumo de recursos. Essa limitação é contornada por outros formatadores de pulso, que usam a simetria vestigial no domínio da frequência dos filtros propostos por 2.1. Estes outros filtros negociam maior consumo de banda de transmissão por maior facilidade de implementação.

A banda necessária para a transmissão da informação pode ser reduzida negociando-se um desempenho menor em ambientes com interferências usando-se sistemas m-ários, que transmitem M símbolos, que consistem em mais de um bit. A equação que determina a banda consumida para sinais modulados em banda passante é atualizada para levar em conta o excesso da banda ocupado $r \in [0, 1]$ e a taxa de símbolos R_s.

$$B_T = \frac{1}{2}R_s(1+r) = \frac{1}{2}\frac{R_b}{\log_2 M}(1+r) \tag{2.3}$$

O sinal modulado em banda base é então passado por moduladores digitais que operam em uma frequência normalmente acima da exigida pelo teorema de Nyquist-Shannon, que afirma que o sinal é reconstruído sem erros se for amostrada a uma taxa que seja pelo menos o dobro do seu maior componente de frequência (LATHI; DING, 2010). Esses são semelhantes aos moduladores analógicos, mas mapeiam cada símbolo ou bit transmitido em valores finitos de amplitudes, no caso das modulações lineares, e/ou deslocamento de fase ou frequências, para as modulações exponenciais (LATHI; DING, 2010).

2.2 Descrição do protocolo DVB-RCS2

O DVB-RCS2 foi criado para padronizar conexões interativas de banda larga como uma extensão de sistemas que usam o DVB. São definidos os detalhes das camadas de alto e baixo nível usadas na comunicação entre o *hub* do operador de satélite e os terminais móveis, bem como a camada de rede e as funções de controle básicas do sistema. De modo a manter a economia de escala, tornando a extensão do sistema mais vantajosa, o protocolo se aproveita dos padrões já implementados comercialmente DVB-S2 e GSE (ETSI; EBU, 2014c) e implementa técnicas que otimizam o desempenho em altas frequências como o uso da banda Ka e da codificação e modulação adaptativas (SKINNEMOEN et al., 2013).

2.2.1 Camadas inferiores do DVB-RCS2 - *Link* de retorno

As camadas inferiores do DVB-RCS2 podem ser separadas em um diagrama de blocos, o qual se apresenta na figura 2. O diagrama separa o sistema nos blocos responsáveis pelo encapsulamento das informações (*GEN.ALPDU*, *GEN.PPDU* e *GEN.FPDU*); por embaralhar a sequência transmitida (*E.DISPERSAL*); pelo cálculo do CRC (*GEN.CRC*); pela codificação para correção de erros (*FEC.CODING*); pela geração de símbolos pré-definidos (*GEN.SYMBOLS*) e pela modulação *MODULATOR*.

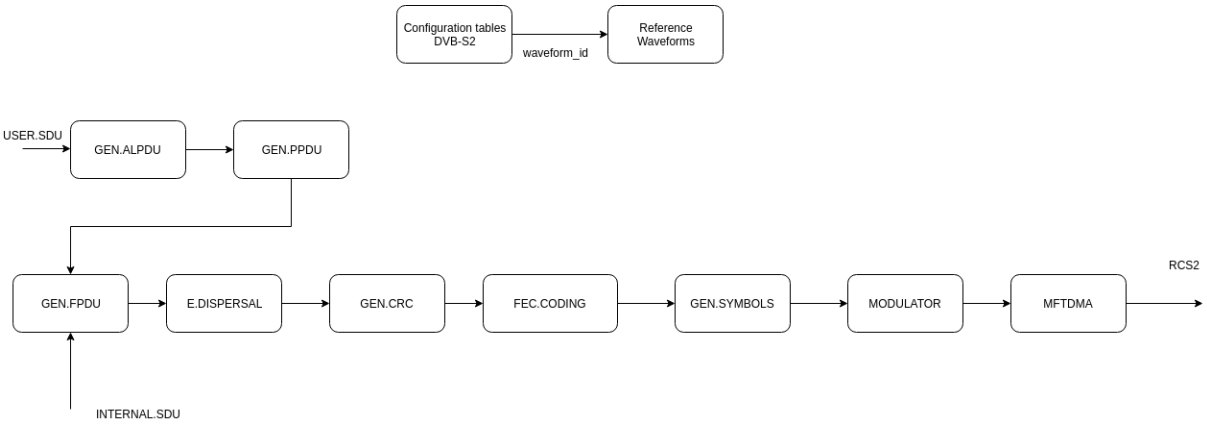

Figura 2 – Diagrama de blocos das camadas inferiores do DVB-RCS2

Os três primeiros blocos são responsáveis por encapsular os dados enviados no formato *Service Data Unit* (SDU) para o *Return Link Encapsulation* (RLE) de acordo com os dados enviados pelas tabelas de configuração recebidas a partir do enlace direto pelo DVB-S2. Então, os dados são enviados serialmente para serem embaralhados de maneira a reduzir a possibilidade de existirem sequências monotônicas (ETSI; EBU, 2014a) que possam afetar a qualidade da transmissão.

Em seguida, as sequências recebem o valor do CRC16, caso sejam sequências de controle, ou CRC32, caso sejam de controle, (ETSI; EBU, 2014a) para posterior verificação de integridade das mesmas no receptor. Esse novo conjunto de informações passa, então, por codificadores definidos pelo tipo de modulação utilizada e por outros parâmetros também provenientes das tabelas de configuração fornecidas pelo DVB-S2. É utilizado um codificador do tipo *Turbo Code* (TC) para as modulações lineares $\pi/2$-BPSK, QPSK, 8PSK, 16QAM, e um Convolucional para a modulação CPM (ETSI; EBU, 2014a), que não será implementada neste trabalho.

As informações são convertidas nos símbolos usados pelas modulações selecionadas nas configurações por meio de permutações com regras pré-definidas. Os símbolos são passados aos moduladores selecionados nas tabelas de configuração e geram o sinal modulado, o qual será transmitido ao satélite de acordo com as divisões de tempo e frequência acordadas por meio das configurações do MF-TDMA.

O enlace de retorno é acessado através do MF-TDMA. Isso permite o acesso de múltiplos terminais através do compartilhamento de um número de frequências de portadoras em períodos de tempo determinados.

2.3 Hardware reconfigurável

A lógica digital reconfigurável consiste em circuitos com matrizes que podem ser configuradas para realizar funções lógicas específicas (HARRIS; HARRIS, 2013). Ela é implementada através de tecnologias de circuitos integrados programáveis, sendo o seu principal uso acelerar o desenvolvimento de um circuito integrado, permitindo a rápida prototipagem de circuitos customizados ou semicustomizados, ou até servindo como soluções finais para projetos em que o *time to market* é crítico (VAHID, 2008).

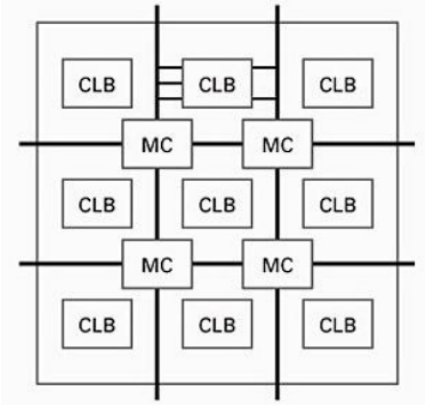

Figura 3 – Diagrama de blocos de um FPGA básico (VAHID, 2008)

Atualmente, a tecnologia de lógica programável mais utilizada é baseada em matrizes de portas programáveis em campo (FPGAs) (VAHID, 2008). Esses circuitos integrados possuem, em sua configuração atual, diversos elementos lógicos configuráveis organizados em blocos com tabelas de pesquisa (*lookup tables*) e registradores (*flip-flops*), bem como alguns blocos especializados, que vão desde blocos de processamento digital de sinal (DSPs) a microprocessadores (HARRIS; HARRIS, 2013). Isso pode ser visto na figura 4, um diagrama de blocos dos SoCs da série Zynq-7000 da Xilinx, que possuem um FPGA interconectado a processadores ARM.

Todos os elementos de um *chip* FPGA estão ligados por fios e multiplexadores que também podem ser programáveis. A figura 3 mostra o diagrama de blocos de um FPGA básico composto pelos blocos lógicos programáveis, identificados como CLB, e pelos fios e multiplexadores das matrizes de chaveamento, identificadas por MC.

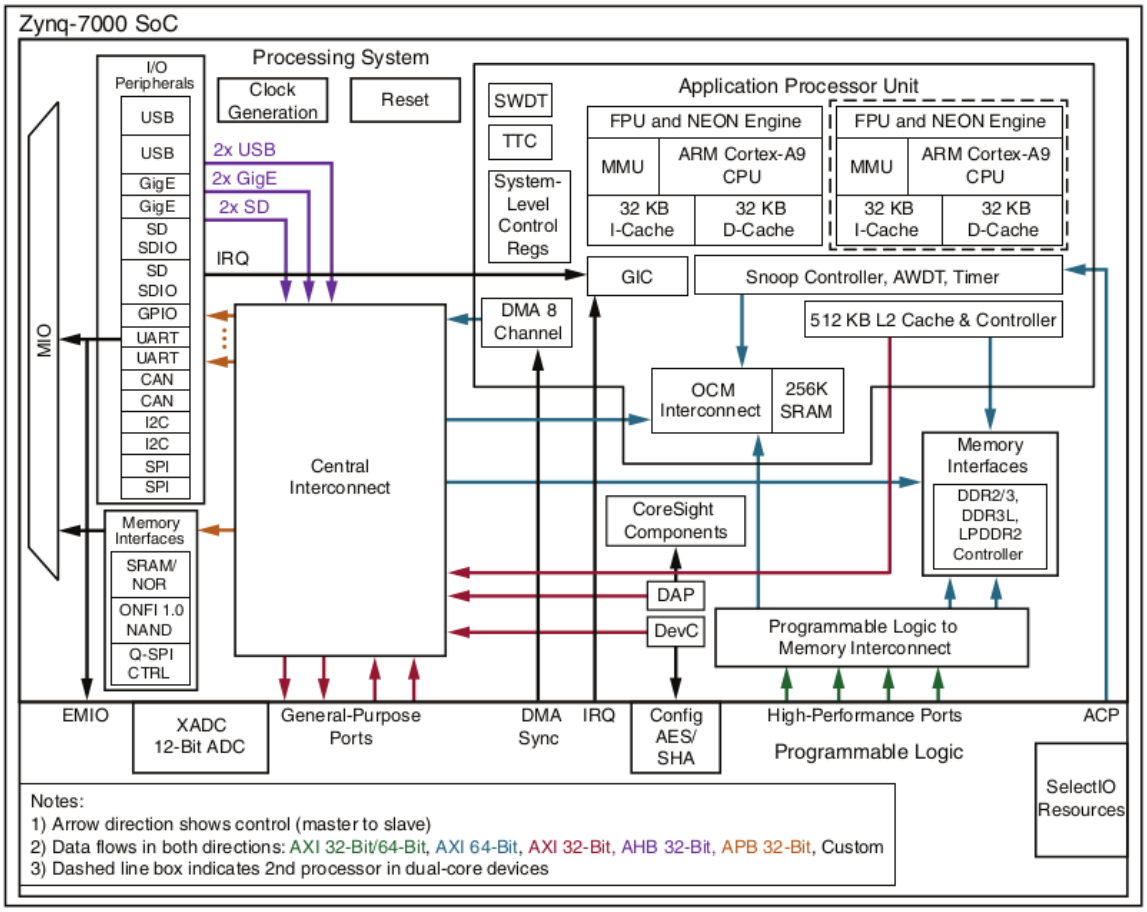

Figura 4 – Diagrama de blocos dos SoCs Zynq-7000 (XILINX, 2018)

A programação de um circuito integrado FPGA é feita por meio da descrição do sistema em uma linguagem de descrição de hardware. Essa especificação é convertida em um arquivo que especifica a configuração de todos os blocos do *chip* de maneira a implementar a descrição em HDL (HARRIS; HARRIS, 2013). Tal programação pode ser feita em campo, seja em casa ou no local de trabalho, sem necessidade de ser feito em uma planta de fabricação – por isso o nome deste tipo de tecnologia (VAHID, 2008).

2.4 Implementação de moduladores digitais e filtros cosseno levantado em FPGAs

A partir da observação dos trabalhos previamente realizados com o objetivo de implementar filtros cosseno levantado ou cosseno levantado com raiz quadrada (SRRC) e modulações digitais em tecnologia de lógica programável baseada em FPGAs, presentes na tabela 1, é possível depreender que tais implementações costumam consumir quantidade considerável de recursos do *chip*, principalmente em implementações que tentam alcançar maior desempenho explorando o paralelismo dos algoritmos ou usando grandes tabelas para acelerar os cálculos.

Tabela 1 – Estado da arte sobre implementação de geradores de símbolos, moduladores e filtros cosseno levantado em FPGAs

Autor	Ano	Modulações	Filtros	Técnica	Plataforma	Ferramentas de desenvolvimento	Consumo de recursos	Resultados
MUHAMMAD et al.	2016	BPSK QPSK 16-QAM	-	Portadoras com período de 10 *clocks*	Xilinx Spartan-6 xc6slx45-csg324c	Xilinx ISE 14.5 Xilinx Chipscope Altera Modelsim VHDL	398 LUTs 177 FFs 40 IOBs 4 BUFGs BUFGCTRLs BUFHCEs 5 DSP48A1s	**Taxa de dados** 99,85 Mbps (BPSK) 199,7 Mbps (QPSK) 399,4 Mbps (16QAM)
K.; PARIKH	2016	BPSK QPSK 8PSK 16PSK 32PSK	Cosseno levantado $r = 0,5$	Embaralhador CCITT V.35 Codificadores diferenciais e convolucionais ($k = 7$, $rate = 1/2$) Portadoras de 1 MHz sintetizadas por DDS	Xilinx Spartan-3AN	Xilinx DDS *Compiler* Xilinx FIR *Compiler* Xilinx *System Generator* Xilinx ISE Mathworks MATLAB Mathworks Simulink	1073 FFs 9100 LUTs 14 IOBs 2 BUFGMUXs 6 RAMB16BWEs	**Taxa de dados** 50 kHz (todas) 500 kHz (32PSK)
ANITHA et al.	2014	QPSK	-	Acumuladores e multiplicadores Booth Quatro ROMs e MUXs Um DDS e multiplicadores (convencional)	Xilinx Spartan-3E	Xilinx ISE 9 Verilog	**DDS** 46 LUTs 28 FFs **Booth** 90 LUTs 39 FFs **ROMs** 174 LUTs 55 FFs	**Frequência máxima de operação** 189 MHz (DDS) 104 MHz (Booth) 197 MHz (ROMs)
YASHODHA; HEMA-LATHA	2016	BPSK $\pi/2$-BPSK $\pi/2$-DPSK QPSK $\pi/4$-QPSK	-	Portadoras sintetizadas por DDS	-	Verilog	-	-
NINGDALLI; SUJATHA	2015	$\pi/2$-BPSK $\pi/4$-QPSK	-	Moduladores Demoduladores Portadoras sintetizadas por DDS MUXs *Phase shifter*	Xilinx Spartan-3E	Xilinx ISE 14.4 Mentor Graphics Modelsim 6.3 VHDL	-	-
TANAWADE; SUDHANSU	2017	BPSK QPSK	-	DDS ou Quatro ROMs	Xilinx xc3s50-5pq208	Xilinx ISE 13.2 Cadence NCSim 15.10-5011 Verilog	**ROMs** 80 FFs 112 LUTs 19 IOBs 1 BUFGMUXs 0 RAMB16s **DDS** 66 FFs 79 LUTs 19 IOBs 1 BUFGMUXs 1 RAMB16s	**Frequência máxima de operação** 228 MHz (ROMs) 189 MHz (DDS)
LUCENA et al.	2014	BPSK (portadora suprimida)	Filtro retangular	SMMH Simulação do canal	Xilinx Virtex 6 *devboard* AVNET ML605 DAC FMC150	Xilinx *System Generator* Mathworks Simulink	-	$f_{symb} = 1$ Mbps $f_{carrier} = 50$ MHz
COMBLOCK	2008	PSK QAM APSK BPSK	Cosseno levantado $r = 0,2$ $r = 0,25$ $r = 0,35$ $r = 0,4$ (escolhidos na síntese) com 30 *taps* quatro amostras por símbolo	NCO Entrada serial ou paralela 8 bits Saídas paralelas 10-14 bits (I-Q) Constelações em ponto fixo 18 bits	Xilinx Spartan-3 xc3s400-4ft256	Xilinx ISE 9.2 ou 18.2 VHDL	3807 FFs 4989 LUTs 157 IOBs 13 BRAMs 6 MULT18x18s 8 GCLK 2 DCMs	-
PILATO et al.	2017	-	SRRC com quatro fases (4x20 *taps*) quantizado em 11 bits (ponto fixo) $r = 0,35$	-	Xilinx Artix-7 xc7a15t	-	1867 LUTs 160 FFs	*clock* 50 MHz
WEILIANG et al.	2002	-	SRRC 13 *taps* $r = 0,6$ projetado pelo método Minimax	-	Altera EP20K60EFC144-1	-	1189 LEs 29 pinos	Erro de 6 % após aplicação do filtro complementar
KHAIRUDIN et al.	2011	-	SRRC com 65 *taps* $r = 0,22$ $f_{sample} = 15,36$ MHz $f_{cuttoff} = 1,92$ MHz	Estrutura MAC Multiplicadores Booth	Xilinx Virtex 3 Pro	Mathworks MATLAB 7 (FDA *tool*) Xilinx ISE 7.1 Mentor Graphics Modelsim SE 6.3f	2946 FFs 4930 LUTs 51 IOBs 20 MULT18x18SIO 1 GCLKs	-
PRIYANTO et al.	2003	OFDM 64QAM	SRRC 40 *taps* $r = 0,15$ Quatro amostras por símbolo $f_{symb} = 20$ MHz	NCO Modulador e demodulador	Xilinx xcv2-6000	Xilinx ISE Agilent ADS VHDL	-	**Frequência máxima de operação** 355 MHz (modulação) 453 MHz (demodulação)

A implementação de mais de uma modulação, bem como a presença de filtros formatadores de pulso tendem a aumentar o consumo de recursos. Isso ocorre devido à lógica extra necessária para multiplexar os diferentes sinais de saída presentes nos moduladores a serem implementados e aos blocos sobreamostradores necessários para a filtragem dos símbolos antes da geração de sinal em banda passante.

Tabela 2 – Características da solução comercial de modulador DVB-RCS2 da Creonic (CREONIC GMBH, 2018)

Característica	**Descrição**
Entrada	*Frames* PDU
Saída	Amostras em banda base ajustadas em ganho
Modulações	$\pi/2$-BPSK, QPSK, 8PSK, 16QAM
Interfaces	AXI-4 *Stream* para entrada e saída de dados AXI-4 *Lite* para ajuste de ganho e verificação do estado do sistema
Implementação	Código fonte VHDL ou sintetizado Descrição em alto nível em MATLAB, C ou C++
Componentes	Embaralhador Codificador CRC Codificador turbo Construtor de *frames* Mapeador Filtro SRRC com *roll-off* de 20 % Ajuste de ganho
Disponibilidade	ASICs FPGAs Xilinx e Altera

As soluções apresentadas até agora implementam parcialmente o modulador requerido pelo DVB-RCS2. Entretanto, há uma solução comercial desenvolvida pela Creonic que não só implementa todas as modulações lineares, mas também as etapas anteriores de recebimento do PDU, embaralhamento, geração de CRC, codificação e construção do *frame* a ser mapeado em símbolos e modulado segundo as exigências das normas. As características dessa solução estão condensadas na tabela 2.

3 Metodologia proposta

3.1 Simulação em alto nível

O filtro formatador de pulsos e o mapeador de símbolos para constelações foram simulados em alto nível através de scripts interpretados pelo programa GNU Octave versão 4.4.1 (EATON et al., 2018) com os pacotes de processamento de sinais, controle e comunicação digital. Tanto o programa principal quanto as extensões foram obtidas através do repositório do sistema operacional Debian GNU/Linux versão Buster e foram executados em um computador com arquitetura *amd64*.

Os scripts executados também geram os vetores de teste e de resposta esperada para serem usados na etapa de testes da implementação em linguagem de descrição de *hardware*, permitindo a comparação de resultados e a validação da implementação. Isso permite a comparação dos sinais intermediários com sinais de referência, evitando implementações errôneas que demandariam tempo para serem corrigidas uma vez na linguagem de descrição de *hardware*. Também é possível obter características teóricas do sistema como espectro utilizado e erro de quantização, que podem ser usados para validar o sistema implementado em linguagem de descrição de hardware.

3.2 Fluxo de projeto em *Hardware*

O sistema será implementado em um kit de desenvolvimento Zybo Revisão B, fabricado pela empresa Digilent. Ele contém, entre outros circuitos integrados, um *System-on-Chip* Xilinx Zynq-7000 XC7Z010-1CLG400C, composto por um FPGA e dois núcleos ARM. As ferramentas de desenvolvimento utilizadas para codificação, simulação, síntese e implementação foram as presentes no software Vivado IDE 2018.3.1, produzido pela empresa Xilinx.

Essas ferramentas permitem que se siga o fluxo usual de projeto de sistemas digitais baseados em FPGAs, disponível na figura 5. Ele consiste na elaboração de um projeto inicial, muitas vezes em diagrama de blocos ou em uma linguagem de alto nível. Esse projeto inicial é mapeado em uma linguagem de descrição de hardware, dividido em blocos lógicos, mapeado nos recursos presentes no chip e implementado fisicamente. Entre todas esses passos há etapas destinadas a verificar se o comportamento do projeto continua conforme o estabelecido nos modelos de alto nível.

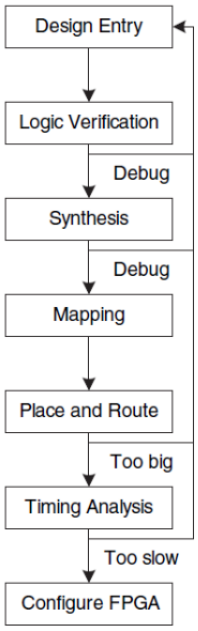

Figura 5 – Fluxo de projeto de um sistema digital implementado em FPGA (HARRIS; HARRIS, 2013)

3.2.1 Procedimentos gerais de projeto

Após a verificação a nível de diagrama de blocos gerada pelo ambiente de desenvolvimento integrado, o projeto descrito em VHDL passou pela simulação comportamental simples para verificação do funcionamento lógico. Observando-se que os resultados esperados foram produzidos, a etapa de síntese mapeou os recursos exigidos pelo sistema descrito em funções booleanas e elaborou um modelo intermediário do projeto no FPGA.

Ao fim desse processo foi possível verificar o funcionamento do circuito sintetizado através de uma simulação comportamental pós síntese e obteve-se uma estimativa dos recursos utilizados. Finalizado o processo de síntese, foram especificados os sinais do sistema a serem expostos para *debug* usando o analisador lógico integrado disponibilizado pelo Vivado, inserindo hardware de instrumentação no sistema.

Em seguida, executou-se no *software* as etapas de mapeamento, responsável por converter as equações geradas na síntese para recursos presentes no *chip*; posicionamento e roteamento, sendo essas últimas necessárias para a interconexão entre os blocos e outros elementos especializados disponíveis.

Terminado o mapeamento e roteamento, foi feita uma análise lógica do sistema já implementado com os blocos disponíveis e verificou-se a análise temporal detalhada do *netlist* implementado no chip. Em seguida foi gerado o arquivo de *bitstream*, que contém a programação das interconexões e os dados presentes entre todos os elementos lógicos, dados suficientes e necessários para implementar o sistema descrito. Finalmente, o sistema a ser analisado era programado no FPGA e os sinais marcados para *debug* na etapa de

síntese tinham os seus valores gravados e transmitidos para o computador, permitindo a comparação com os padrões gerados com a descrição em alto nível.

3.3 Projeto do filtro SRRC

O filtro cosseno levantado com raiz quadrada exigido pela norma do DVB-RCS2 possui fator de *roll-off* $r = 0,2$ (ETSI; EBU, 2014a) e deve obedecer ao conjunto de equações 3.1. Para facilitar o projeto e a implementação em lógica programável de modo a obedecer aos requisitos exigidos pela norma, decidiu-se, assim como nos trabalhos expostos anteriormente, projetar um filtro de resposta finita ao impulso (FIR) que implementasse a resposta em frequência exigida por (ETSI; EBU, 2014a). No conjunto de equações 3.1, f_N representa a frequência de Nyquist, e r, o fator de *roll-off* utilizado no filtro.

$$\begin{cases} H(f) = 1 & \text{para} \quad |f| < f_N(1-r) \\ H(f) = \sqrt{\frac{1}{2} + \frac{1}{2}\sin\left(\frac{\pi(f_N - |f|)}{2rf_N}\right)} & \text{para} \quad f_N(1-r) \leqslant |f| \leqslant f_N(1+r) \\ H(f) = 0 & \text{para} \quad |f| > f_N(1+r) \end{cases} \tag{3.1}$$

Segundo JOOST (2010), as equações que definem a resposta em frequência do filtro podem ser implementadas em um filtro FIR com janela retangular usando-se a equação 3.2, que é o resultado da transformada de Fourier inversa da equação 3.1 descartando as constantes que multiplicam o resultado final. Tal equação possui duas singularidades removíveis em $t = 0$ e $t = \pm 1/4r$ cujos valores são constantes disponíveis nas equações 3.3 e 3.4, também obtidos em (JOOST, 2010).

$$h(tT_c) = \frac{\sin(\pi t(1-r)) + 4rt\cos(\pi t(1+r))}{\pi t(1-(4rt)^2)} \tag{3.2}$$

$$h(0) = (1-r) + \frac{4r}{\pi} \tag{3.3}$$

$$h\left(\pm\frac{1}{4r}T_c\right) = \frac{r}{\sqrt{2}}\left(\left(1+\frac{2}{\pi}\right)\sin\left(\frac{\pi}{4r}\right) + \left(1-\frac{2}{\pi}\right)\cos\left(\frac{\pi}{4r}\right)\right) \tag{3.4}$$

O script responsável por gerar os coeficientes do filtro tem como entradas a frequência de amostragem; o fator de *oversample*; um fator de ganho a ser aplicado nas equações 3.2, 3.3 e 3.4, que é mantido normalmente unitário; o número de elementos a ser usado na computação da transformada rápida de Fourier, usada para verificar a resposta no domínio da frequência, e a resolução dos gráficos produzidos em pontos por polegada.

Após a passagem dos parâmetros supracitados, o script gera os coeficientes em ponto flutuante e calcula a resposta no domínio da frequência do mesmo, fundamental

para se definir a atenuação na banda de rejeição. Em seguida, os coeficientes são normalizados no intervalo entre -1 e 1 e quantizados como inteiros com sinal e também é calculada a atenuação para o filtro quantizado, que ainda não garante a interferência entre símbolos nula. Para verificar a anulação da interferência intersimbólica, também é calculado o quantizado com o seu complementar através da convolução entre os coeficientes quantizados.

3.3.1 Conversão de ponto flutuante em ponto fixo

A implementação do filtro cosseno levantado com raiz quadrada descrito pela equação 3.2 através de um FIR exige a escolha somente do número de coeficientes, do fator de sobreamostragem e do número de bits usado para a quantização do mesmo, pois o parâmetro que define a banda de transição, o fator de *roll-off* já é fixado pela norma.

Para se evitar o grande consumo de recursos em chip causado pela implementação de operações em ponto flutuante, os coeficientes do filtro foram convertidos em ponto fixo. Essa conversão mapeou os números em ponto flutuante para ponto fixo sem parte inteira, permitindo o processamento dos mesmos como se fossem inteiros com sinal em complemento de dois.

O número de bits necessários para representação dos coeficientes foi definido experimentalmente através da execução de um script que calcula o erro quadrático médio entre o filtro original em ponto flutuante e o quantizado. Os parâmetros foram escolhidos após a realização de uma simulação por um outro script que testa a resposta em frequência e o erro médio quadrático para fatores de sobreamostragem com os valores de 4, 6, 8, 16, 32 e 64; ordens de filtro nos valores de 32, 64, 128, 256, 512, 1024 e 2048; bits para quantização nos no intervalo $[8, 12]$; frequência de amostragem com os valores de 500 kHz, 12,5 MHz, 25 MHz, 50 MHz, 100 MHz e 125 MHz.

Após a escolha dos valores dos parâmetros, a variação do erro quadrático médio foi plotada em um gráfico. A figura 6 demonstra que o valor de 8 bits é suficiente para representar o filtro com um erro pequeno e com relativa distância entre o erro quadrático máximo e o mínimo valor representável com essa quantização.

Os parâmetros e as características do filtro a ser implementado em linguagem de descrição de hardware estão dispostos na tabela 3. Os seus coeficientes foram calculados usando as equações 3.2, 3.3 e 3.4 (JOOST, 2010).

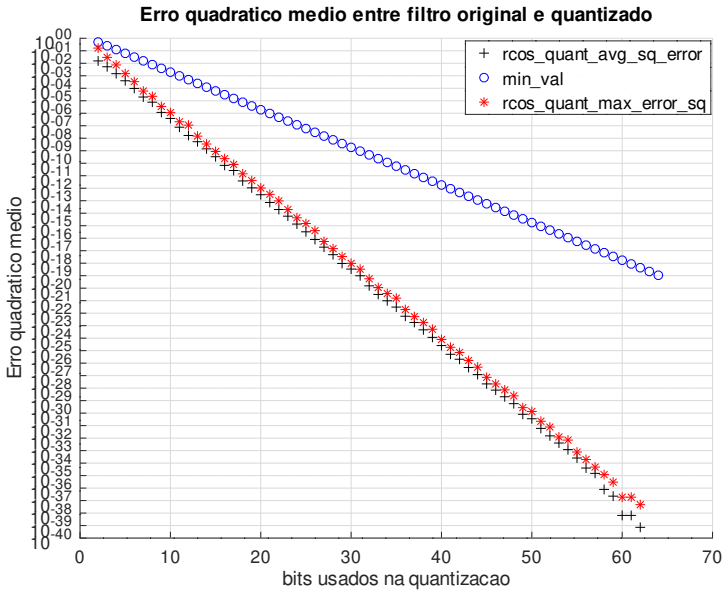

Figura 6 – Erro quadrático médio e máximo entre amostras do SRRC normalizado e o quantizado e valor mínimo representável

Tabela 3 – Parâmetros e resultados do filtro FIR a ser implementado em hardware

Parâmetro	**Valor**
Tipo de filtro	Cosseno levantado com raiz quadrada (SRRC)
Janelamento	Retangular (*sinc*)
Frequência de amostragem f_s	125 MHz
Ordem	64
Tamanho (*taps*)	65
Amostras por símbolo (*oversample*)	6
Frequência de símbolo f_{symb}	20,833 MHz
Tempo de símbolo T_s	48 ns
Fator de *roll-off* r	0,2
Atraso em amostras	32
Atraso em segundos	256 ns
Largura de banda	12,5 MHz
Resolução da quantização	8 bits
Erro quadrático médio da quantização	$7,68782 \times 10^{-6}$
Atenuação mínima na banda de rejeição	-32,570688 dB (normalizado) -31,279188 dB (quantizado) -62,55837 dB (quantizado após aplicar o complementar)

3.3.2 Resposta do filtro projetado

Os resultados da execução do script de geração dos coeficientes do filtro SRRC estão dispostos nas figuras: 7 e 8, que indicam respectivamente os coeficientes do filtro em ponto flutuante e os quantizados. Já as figuras 9, 10, 11 representam a resposta no domínio da frequência dos filtros em ponto flutuante, quantizado e quantizado com o complementar. A partir dos resultados do script de geração supracitado também foram produzidas as figuras 12, 13 e 14. Essas figuras demonstram a resposta em frequência normalizada para 0 dB e registram a atenuação na banda de rejeição.

Nota-se nas imagens 9, 10, 11, 12, 13 e 14, referentes à resposta em frequência do filtro em ponto flutuante, quantizado e quantizado com complementar, uma pequena oscilação no centro da banda passante do filtro. Ela é decorrente do fenômeno de Gibbs causado pelo janelamento retangular da resposta do filtro.

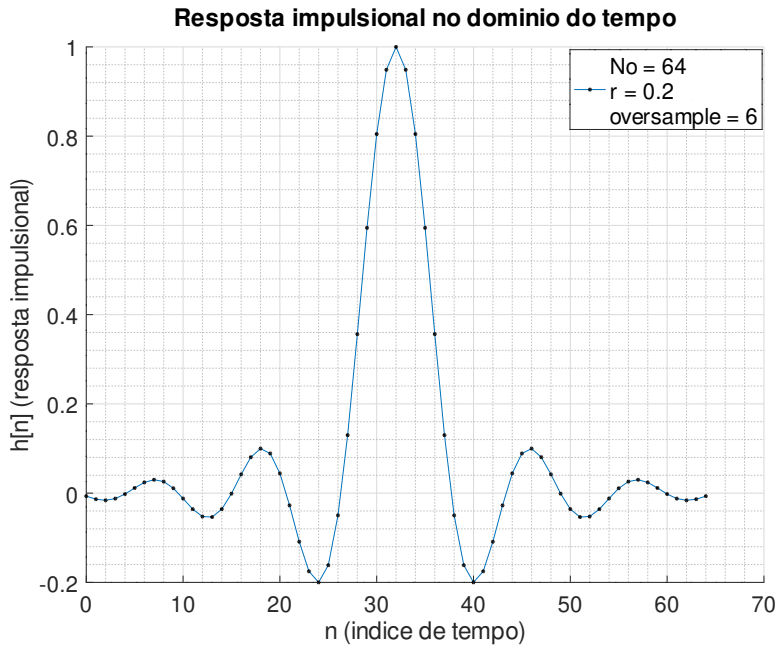

Figura 7 – Coeficientes do filtro FIR SRRC normalizado

O erro quadrático médio entre o filtro quantizado, que é novamente normalizado entre o intervalo entre -1 e 1, e a solução original em ponto flutuante normalizada para essa mesma faixa está presente na figura 15, que também detalha o erro para cada amostra do filtro.

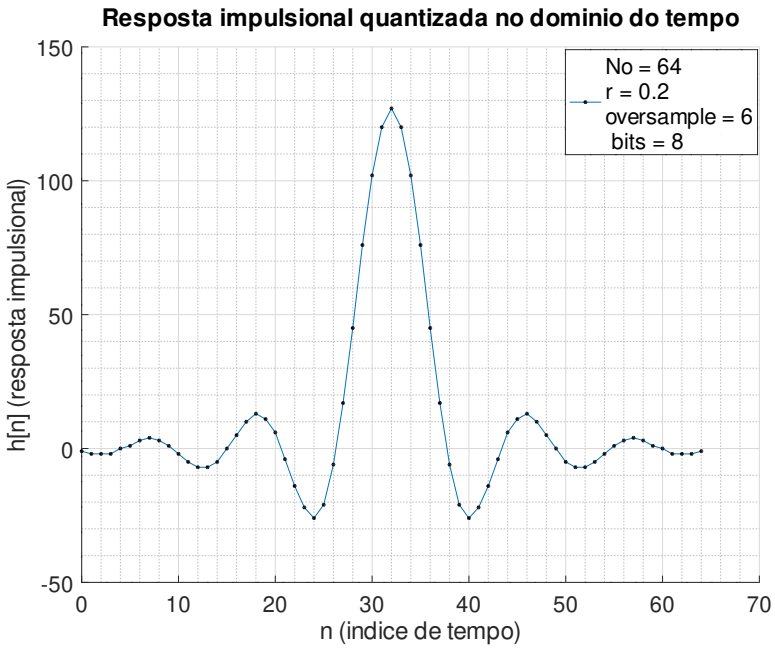

Figura 8 – Coeficientes do filtro FIR SRRC quantizado com 8 bits

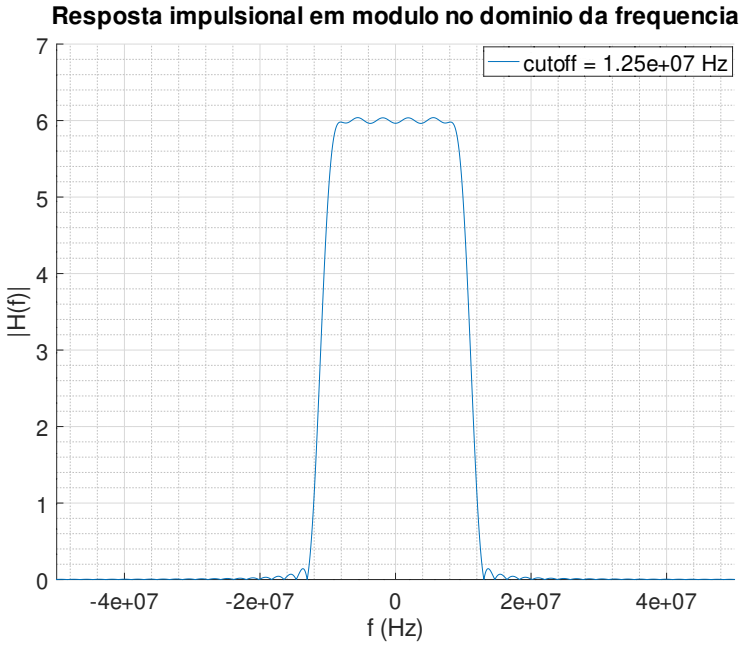

Figura 9 – Resposta em frequência em módulo do filtro FIR SRRC normalizado

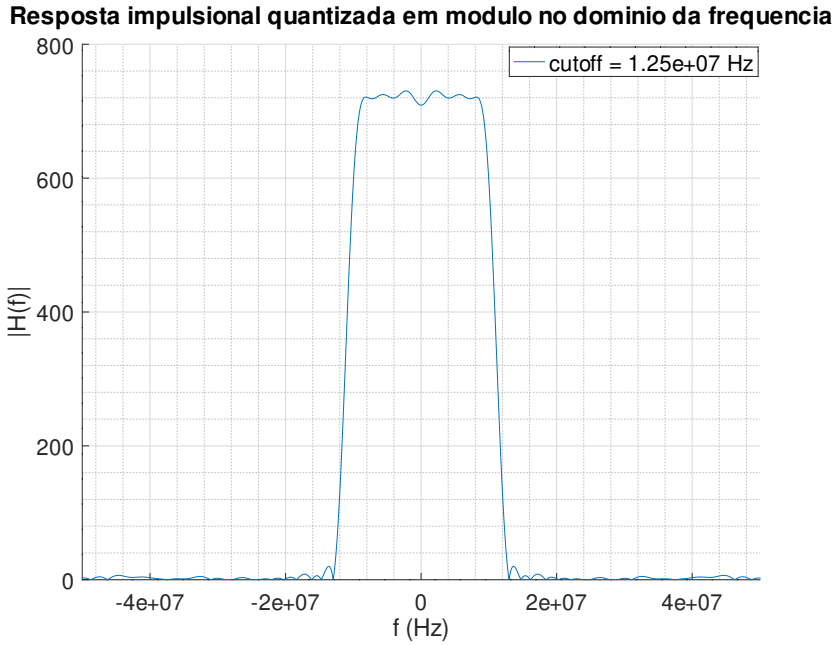

Figura 10 – Resposta em frequência em módulo do filtro FIR SRRC quantizado

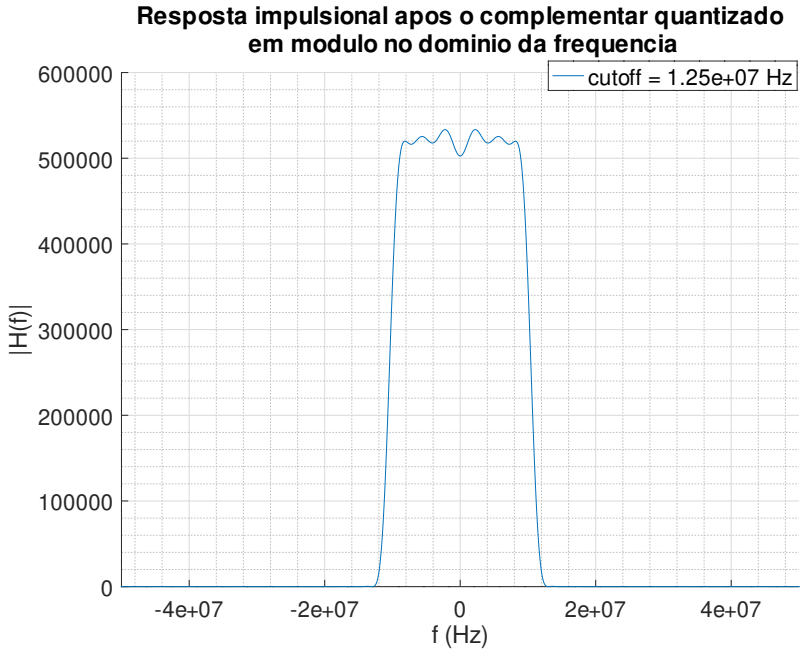

Figura 11 – Resposta em frequência em módulo do filtro FIR SRRC quantizado com o complementar

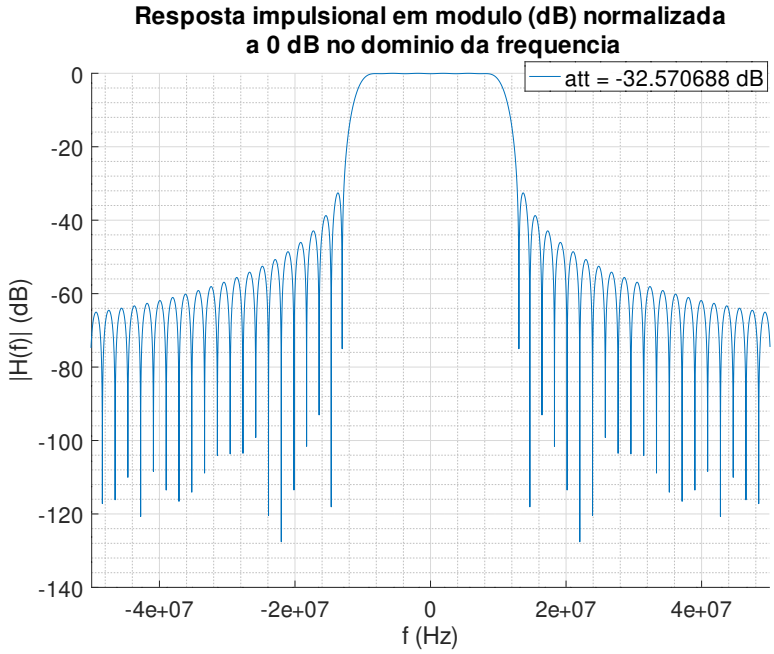

Figura 12 – Resposta em frequência em módulo do filtro FIR SRRC normalizado (dB)

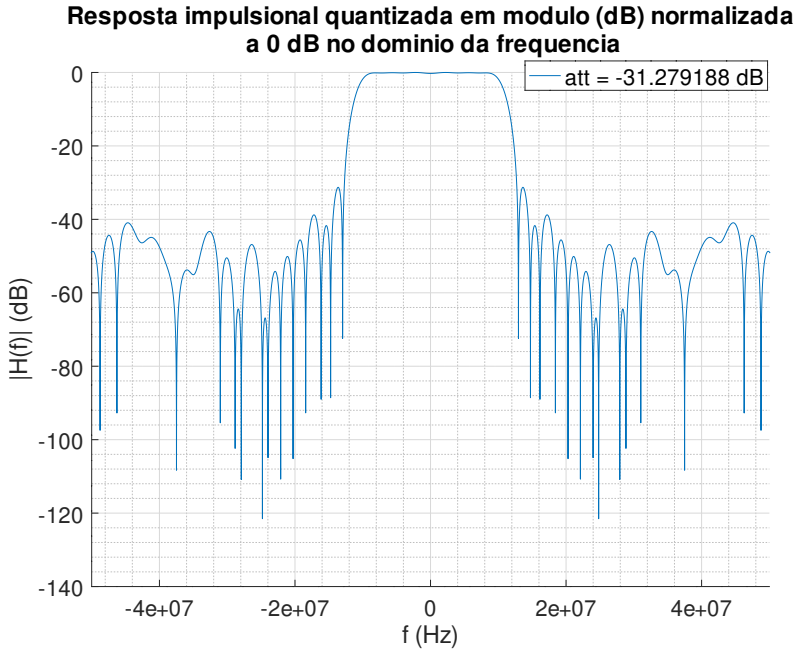

Figura 13 – Resposta em frequência em módulo do filtro FIR SRRC quantizado (dB)

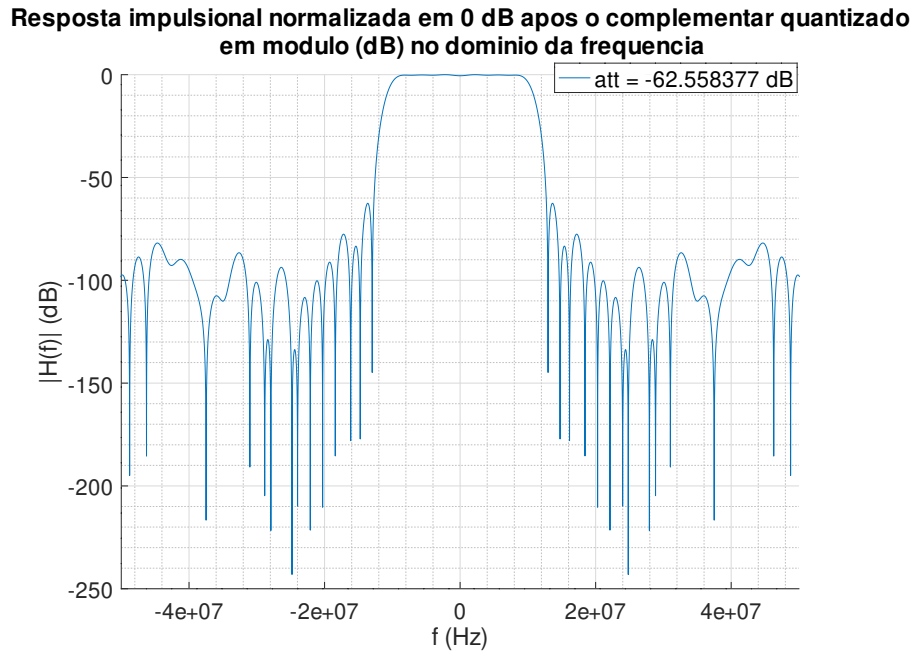

Figura 14 – Resposta em frequência em módulo do filtro FIR SRRC quantizado com o complementar (dB)

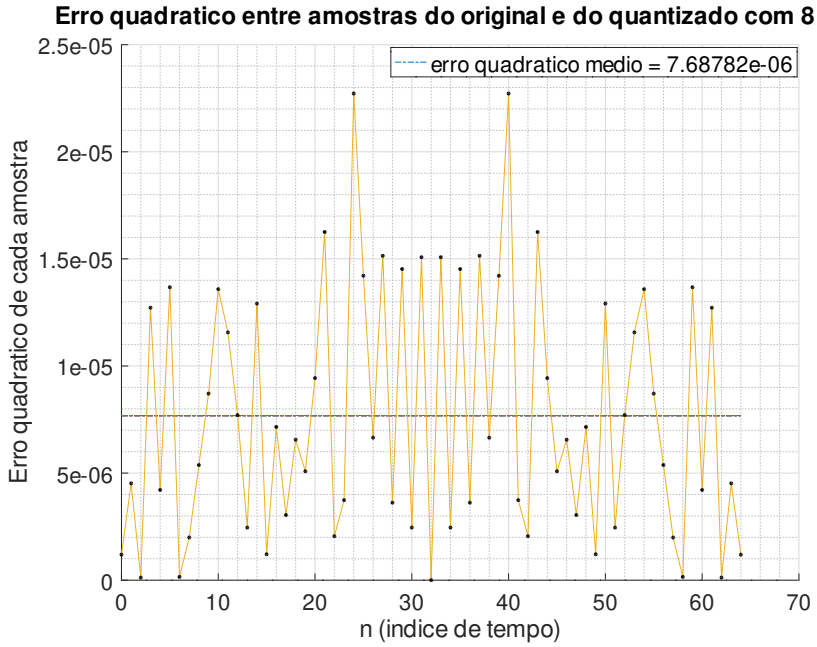

Figura 15 – Erro quadrático médio entre amostras do SRRC normalizado e o quantizado

3.3.3 Descrição em VHDL do filtro SRRC

O filtro FIR foi descrito com sua forma direta, disponível na figura 16. As amostras entram a cada ciclo de relógio do sistema e são deslocadas em uma fila implementada por um registrador de deslocamento. Os valores presentes nos registradores são multiplicados pelos 65 coeficientes do filtro e então somados através de uma árvore de somas de sete estágios, entregando o resultado da filtragem ao final. Todas as operações de soma e multiplicação são registradas formando uma *pipeline*, evitando assim problemas de temporização que possam vir a ser causados por somadores e multiplicadores implementados combinacionalmente.

A descrição do filtro em VHDL foi automatizada através de um script também interpretado pelo GNU/Octave. A partir dos coeficientes já quantizados, da largura em bits das amostras recebidas pelo filtro e do fator de sobreamostragem são produzidos um pacote de configurações e uma entidade com a arquitetura do filtro. O comprimento da palavra de saída é calculado automaticamente para se evitar *overflow* nas operações de soma e multiplicação realizadas.

Conforme supracitado, o filtro foi implementado com coeficientes de 8 bits e também recebe as amostras em 8 bits. Para que as multiplicações e somas em ponto fixo quantizado em complemento de 2 ocorressem sem risco de *overflow*, o armazenamento do resultado dessas operações com uma palavra de maior comprimento tornou-se necessária. Portanto, foi escolhido o comprimento de 23 bits para a saída do filtro: 16 bits são relativos às multiplicações entre as amostras da fila e os coeficientes, e os bits restantes são devido às 64 adições que acontecem no resto da *pipeline*. As somas são executadas em etapas dentro de uma árvore de somas de 7 estágios registrados.

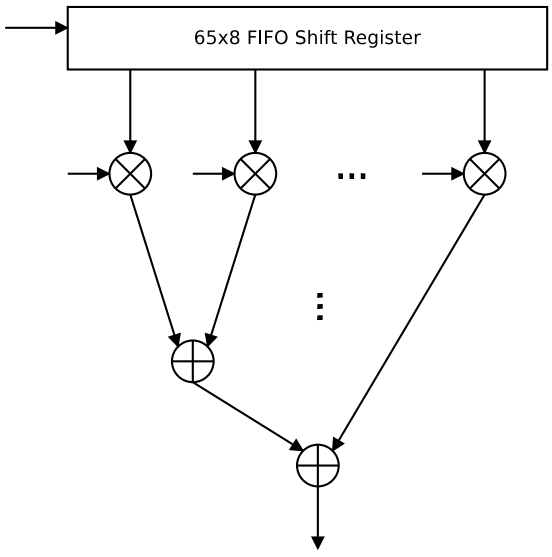

Figura 16 – Diagrama de blocos da arquitetura do filtro SRRC descrito em VHDL

Embora o bloco tenha sido descrito de modo que o mesmo receba as amostras

já sobreamostradas e com os zeros necessários para deslocar a última amostra relevante pela *pipeline* até o resultado final, ainda é necessário não passar para os próximos blocos as amostras inválidas, que saem do filtro até que ele tenha recebido uma quantidade de amostras igual ao seu atraso em amostras (verificar tabela 3).

Essa funcionalidade foi implementada através de um contador síncrono, que é incrementado sempre que o bloco precedente sinaliza que a amostra enviada ao filtro é válida. O contador está ligado a um comparador e quando o número de amostras lidas iguala-se ao atraso do filtro, no próximo ciclo de relógio a saída é sinalizada como válida sempre que o dado também o for. Em seguida, o contador é parado, sendo necessário que o sinal de *reset* de entidade seja levado a nível lógico alto para reiniciar a *pipeline* do filtro e o contador.

3.4 Projeto do sobreamostrador

O sobreamostrador não foi descrito em alto nível em um script do GNU/Octave pois já existe uma função no pacote *control* responsável pela sobreamostragem: *upsample*. Assim, este bloco do sistema foi descrito diretamente em VHDL, sendo o seu comportamento verificado através das simulações e da execução do mesmo no hardware com instrumentação para *debug* junto de outros módulos que compõem o sistema.

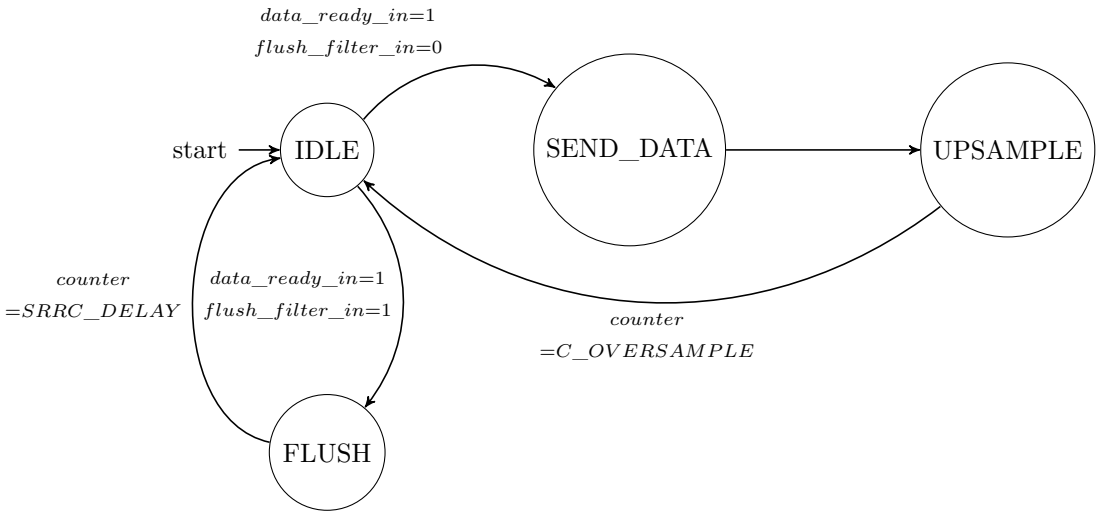

Figura 17 – Diagrama da FSM de controle do bloco sobreamostrador

Esse bloco foi descrito para ser o responsável por controlar a entrada de amostras no filtro FIR SRRC, passando o dado a partir da entrada, sobreamostrando-o e enviando um vetor de zeros quando solicitado o fim dos dados, retirando dos registradores de *pipeline* a última amostra válida. Esse controle é feito por meio de uma máquina de estados finitos, cujo diagrama encontra-se na figura 17.

3.5 Projeto dos mapeadores de símbolo para constelações

Os mapeadores de símbolo para constelação são responsáveis por mapear os símbolos entregues por blocos anteriores nos valores complexos exigidos pela norma (ETSI; EBU, 2014a) para cada uma das modulações: $\pi/2$-BPSK, QPSK, 8-PSK e 16-QAM.

3.5.1 Projeto em alto nível

Os mapeadores de símbolos para constelações foram descritos nos scripts Octave responsáveis por produzir os valores de referência para cada uma das modulações em banda base. O mapeamento das constelações $\pi/2$-BPSK, QPSK, 8-PSK e 16-QAM foi feito de acordo com a norma (ETSI; EBU, 2014a) e está descrito nas tabelas 4, 5, 6, sendo que a modulação $\pi/2$-BPSK utiliza a mesma constelação definida para a QPSK.

Tabela 4 – Constelação QPSK e $\pi/2$-BPSK segundo (ETSI; EBU, 2014a)

Índice	**Símbolo**	**Valor real**	**Valor imaginário**
0		0	0
1	S_0	$\cos(\frac{\pi}{4})$	$\text{sen}(\frac{\pi}{4})$
2	S_1	$\cos(\frac{7\pi}{4})$	$\text{sen}(\frac{7\pi}{4})$
3	S_2	$\cos(\frac{3\pi}{4})$	$\text{sen}(\frac{3\pi}{4})$
4	S_3	$\cos(\frac{5\pi}{4})$	$\text{sen}(\frac{5\pi}{4})$

Tabela 5 – Constelação 8-PSK segundo (ETSI; EBU, 2014a)

Índice	**Símbolo**	**Valor real**	**Valor imaginário**
0		0	0
1	S_0	$\cos(\frac{\pi}{8})$	$\text{sen}(\frac{\pi}{8})$
2	S_1	$\cos(\frac{3\pi}{8})$	$\text{sen}(\frac{3\pi}{8})$
3	S_2	$\cos(\frac{15\pi}{8})$	$\text{sen}(\frac{15\pi}{8})$
4	S_3	$\cos(\frac{13\pi}{8})$	$\text{sen}(\frac{13\pi}{8})$
5	S_4	$\cos(\frac{7\pi}{8})$	$\text{sen}(\frac{7\pi}{8})$
6	S_5	$\cos(\frac{5\pi}{8})$	$\text{sen}(\frac{5\pi}{8})$
7	S_6	$\cos(\frac{9\pi}{8})$	$\text{sen}(\frac{9\pi}{8})$
8	S_7	$\cos(\frac{11\pi}{8})$	$\text{sen}(\frac{11\pi}{8})$

Os bits recebidos pelo mapeador $\pi/2$-BPSK são mapeados na constelação QPSK de acordo com a equação 3.5 (ETSI; EBU, 2014a), na qual $u(n)$ é a polaridade, definida como positiva se o bit recebido é 0 e negativo caso seja 1, e n é o índice do bit recebido dentro da sequência. A equação foi implementada em alto nível através de um contador

Tabela 6 – Constelação 16-QAM segundo (ETSI; EBU, 2014a)

Índice	**Símbolo**	**Valor**
0		0
1	S_0	$-\frac{1}{\sqrt{10}} - \frac{1}{\sqrt{10}}j$
2	S_1	$-\frac{1}{\sqrt{10}} - \frac{3}{\sqrt{10}}j$
3	S_2	$-\frac{1}{\sqrt{10}} + \frac{1}{\sqrt{10}}j$
4	S_3	$-\frac{1}{\sqrt{10}} + \frac{3}{\sqrt{10}}j$
5	S_4	$-\frac{3}{\sqrt{10}} - \frac{1}{\sqrt{10}}j$
6	S_5	$-\frac{3}{\sqrt{10}} - \frac{3}{\sqrt{10}}j$
7	S_6	$-\frac{3}{\sqrt{10}} + \frac{1}{\sqrt{10}}j$
8	S_7	$-\frac{3}{\sqrt{10}} + \frac{3}{\sqrt{10}}j$
9	S_8	$+\frac{1}{\sqrt{10}} - \frac{1}{\sqrt{10}}j$
10	S_9	$+\frac{1}{\sqrt{10}} - \frac{3}{\sqrt{10}}j$
11	S_{10}	$+\frac{1}{\sqrt{10}} + \frac{1}{\sqrt{10}}j$
12	S_{11}	$+\frac{1}{\sqrt{10}} + \frac{3}{\sqrt{10}}j$
13	S_{12}	$+\frac{3}{\sqrt{10}} - \frac{1}{\sqrt{10}}j$
14	S_{13}	$+\frac{3}{\sqrt{10}} - \frac{3}{\sqrt{10}}j$
15	S_{14}	$+\frac{3}{\sqrt{10}} + \frac{1}{\sqrt{10}}j$
16	S_{15}	$+\frac{3}{\sqrt{10}} + \frac{3}{\sqrt{10}}j$

de módulo 4 e estruturas condicionais simples para seleção do símbolo QPSK equivalente conforme a entrada dos bits.

$$S(n) = u(n)\, e^{j\pi(\frac{2n+1}{4})} \tag{3.5}$$

Em todas as tabelas que descrevem a relação entre os símbolos e os números imaginários pertencentes às constelações implementadas foi adicionada uma linha para zeros, deslocando o índice de símbolos em um. Isso é necessário para que os zeros emitidos pelo sobreamostrador e necessários ao correto funcionamento do filtro SRRC não sejam confundidos com símbolos zero pelos mapeadores.

3.5.2 Conversão de ponto flutuante para ponto fixo

As constelações utilizadas pelo sistema: QPSK, 8PSK e 16QAM precisam ser quantizadas para que os símbolos sejam processados pelo filtro SRRC descrito em VHDL. Como a arquitetura do filtro já havia sido definida para usar coeficientes em ponto fixo com oito bits para a parte fracionária, utilizou-se também uma quantização semelhante como ponto

inicial, facilitando a codificação do filtro e reduzindo a complexidade combinacional para implementação das tabelas no hardware.

Tabela 7 – Erro quadrático médio dos valores reais e imaginários da constelação QPSK quantizada

	Parte Imaginária	**Parte Real**
Erro quadrático médio	31×10^{-6}	31×10^{-6}
Erro quadrático máximo	62×10^{-6}	62×10^{-6}

Tabela 8 – Erro quadrático médio dos valores reais e imaginários da constelação 8PSK quantizada

	Parte Imaginária	**Parte Real**
Erro quadrático médio	$20,334 \times 10^{-6}$	$20,334 \times 10^{-6}$
Erro quadrático máximo	62×10^{-6}	62×10^{-6}

Tabela 9 – Erro quadrático médio dos valores reais e imaginários da constelação 16QAM quantizada

	Parte Imaginária	**Parte Real**
Erro quadrático médio	$24,11 \times 10^{-6}$	$24,11 \times 10^{-6}$
Erro quadrático máximo	62×10^{-6}	62×10^{-6}

Para confirmar que o valor escolhido para quantização apresentava bom custo benefício para a implementação, executou-se um script para medir o erro quadrático médio e máximo entre o valor quantizado da parte real e da imaginária em relação aos originais em ponto flutuante. Os valores estão apresentados nas tabelas 7, 8 e 9 e demonstram que o valor de oito bits escolhido resulta em um erro em torno de 10^{-6} em relação ao original.

3.5.3 Projeto em VHDL

A implementação em linguagem de descrição de hardware também utilizou a abordagem em tabelas adotada na descrição em alto nível das modulações QPSK, 8PSK e 16QAM, definindo-as como vetores constantes de inteiros. Do mesmo modo, a implementação do mapeador da modulação $\pi/2$-BPSK também ficou muito parecida com a feita em scripts: um contador de módulo quatro que avança de maneira síncrona a cada símbolo recebido foi implementado, bem como a lógica síncrona de mapeamento do símbolo equivalente da constelação QPSK dependendo do valor do contador e do símbolo BPSK recebido.

3.6 Projeto do sistema completo

3.6.1 Simulação em alto nível

O sistema foi simulado em alto nível através de um script executado no GNU Octave que registra e plota os sinais de saída dos quatro moduladores: $\pi/2$-BPSK, QPSK, 8PSK e 16QAM. Estes valores, juntamente com as entradas, estão dispostos nas figuras 18, 19, 20 e 21. O script gera 256 símbolos pseudoaleatórios inteiros para cada uma das modulações: valores de 0 a 1 para a BPSK, valores de 0 a 2 para a QPSK, de 0 a 3 para a 8PSK e no intervalo de 0 a 4 para a 16QAM. Para garantir a repetibilidade dos testes, o gerador de sequências pseudoaleatórias do programa é inicializado com uma semente fixa em todas as execuções do script.

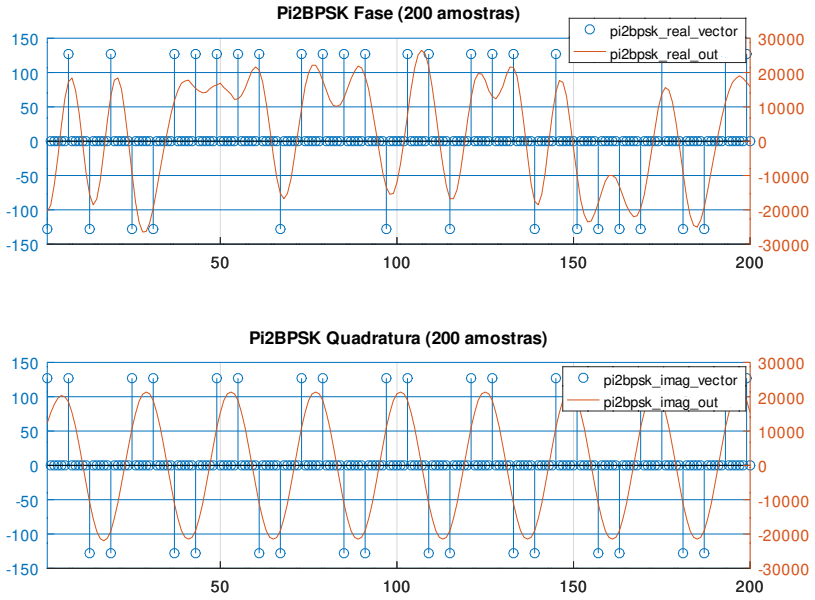

Figura 18 – Valores de referência e de entrada para o modulador $\pi/2$-BPSK

A próxima operação realizada é o mapeamento dos símbolos para os valores reais e imaginários já quantizados com oito bits de acordo com as regras mencionadas anteriormente de cada uma das constelações relativas às modulações utilizadas. As partes real e imaginária produzidas a partir do mapeamento são separadas em dois vetores e são sobreamostradas de acordo com o fator de 6, presente na tabela 3.

Os vetores relativos aos valores de fase e quadratura do sinal em banda base são filtrados pelo formatador de pulsos, o filtro SRRC com os coeficientes já quantizados. O atraso do filtro causado pela fase linear é compensado inserindo-se amostras zeradas ao final dos vetores e retirando-se as primeiras amostras relativas ao transiente após a filtragem. As amostras resultantes são então plotadas em conjunto com os sinais de entrada e são gravadas em arquivos para consultas posteriores.

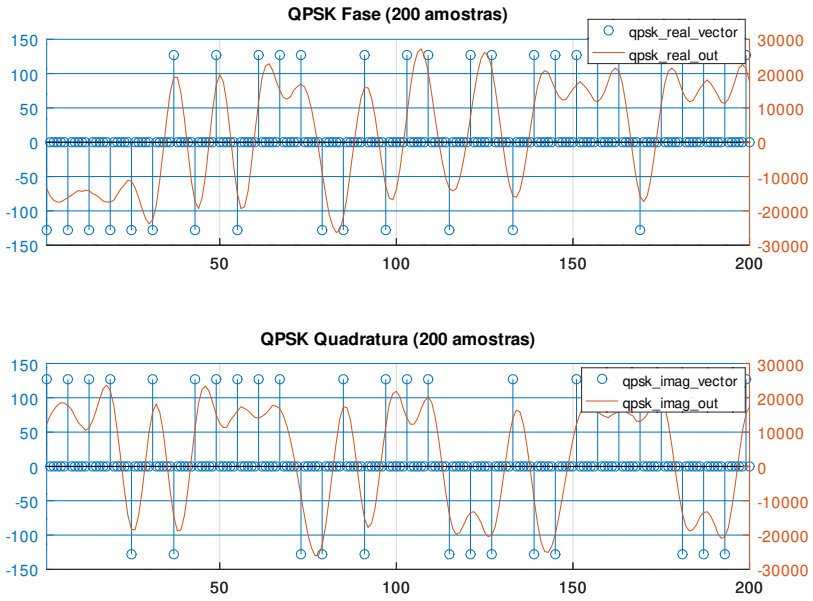

Figura 19 – Valores de referência e de entrada para o modulador QPSK

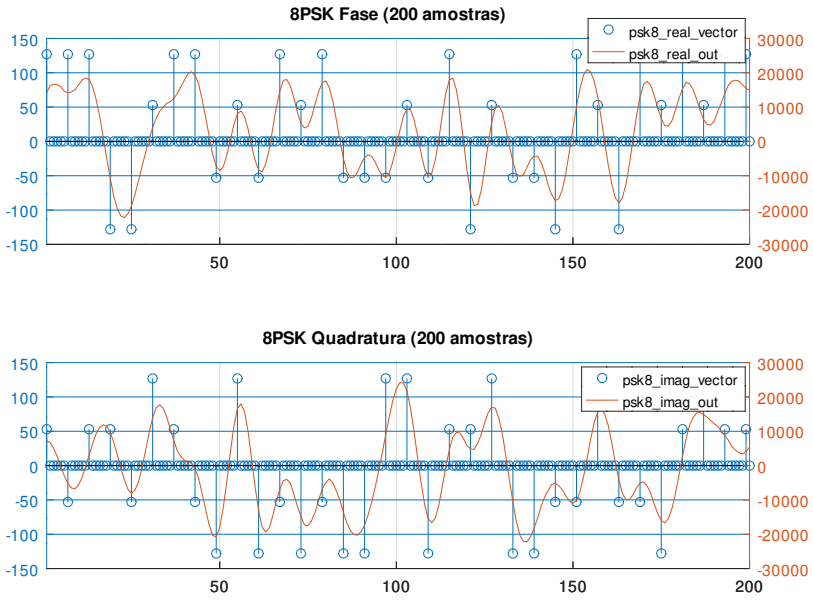

Figura 20 – Valores de referência e de entrada para o modulador 8PSK

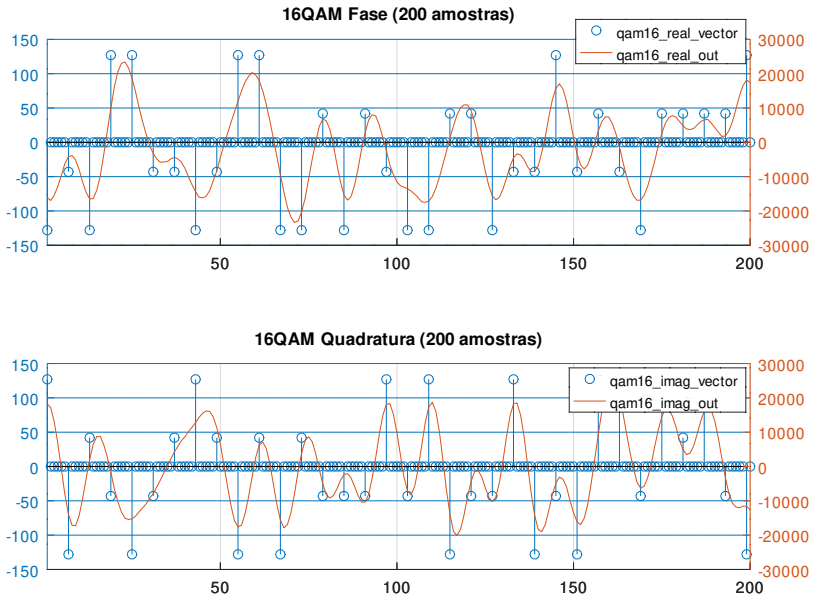

Figura 21 – Valores de referência e de entrada para o modulador 16QAM

3.6.1.1 Simulação do sistema em conjunto com um receptor

Com o objetivo de verificar o desempenho do filtro SRRC quantizado, também foi realizada uma simulação com a descrição em alto nível do modulador em banda base, um canal com ruído branco gaussiano aditivo e um receptor simples. A simulação produz as constelações dos símbolos recuperados na entrada do receptor para cada uma das modulações: $\pi/2$-BPSK, QPSK, 8PSK e 16QAM. Esses resultados estão nas figuras 22, 23, 24 e 25 respectivamente. Também foram produzidas as figuras 26, 27, 28 e 29, que indicam os símbolos recuperados no receptor para as modulações $\pi/2$-BPSK, QPSK, 8PSK e 16QAM.

Figura 22 – $\pi/2$-BPSK recebido na entrada do receptor

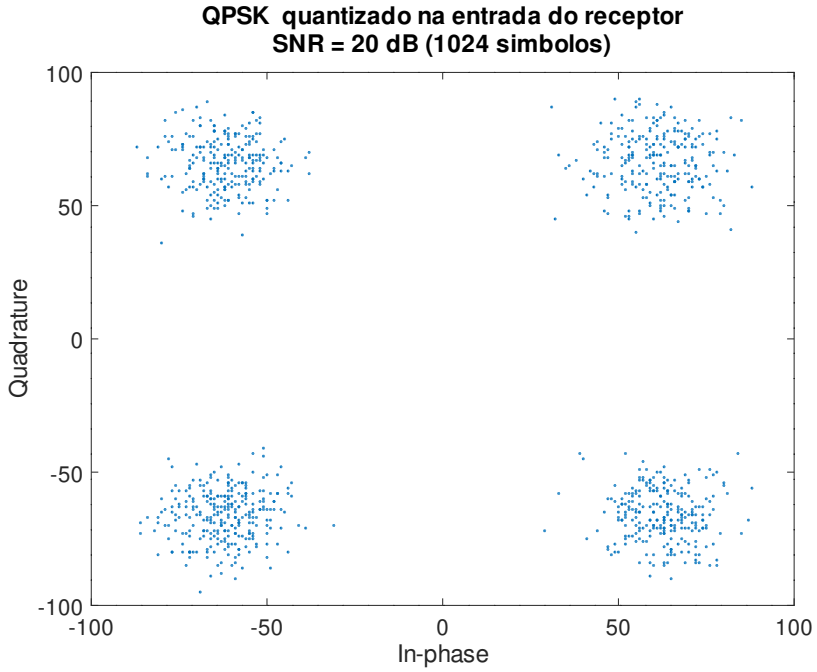

Figura 23 – QPSK recebido na entrada do receptor

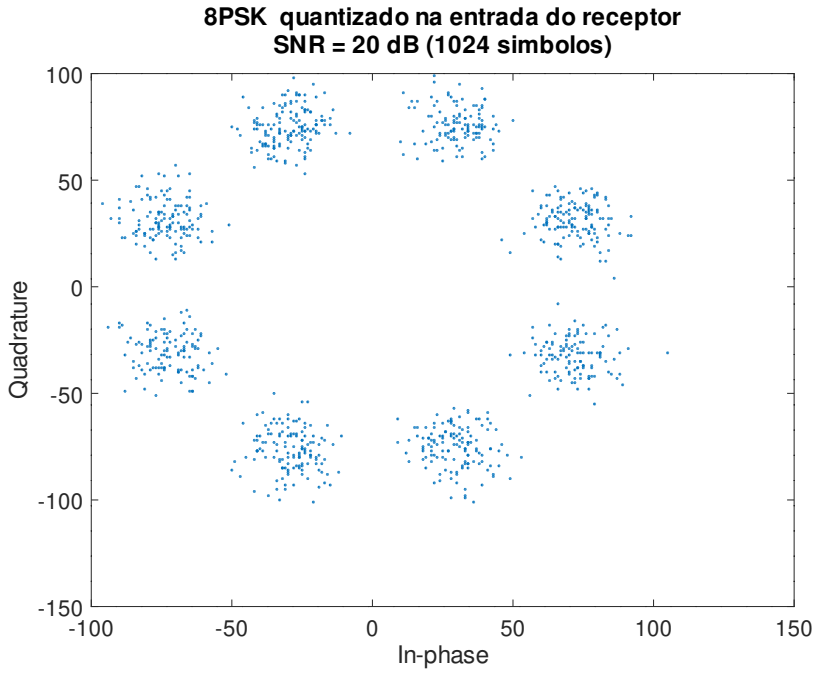

Figura 24 – 8PSK recebido na entrada do receptor

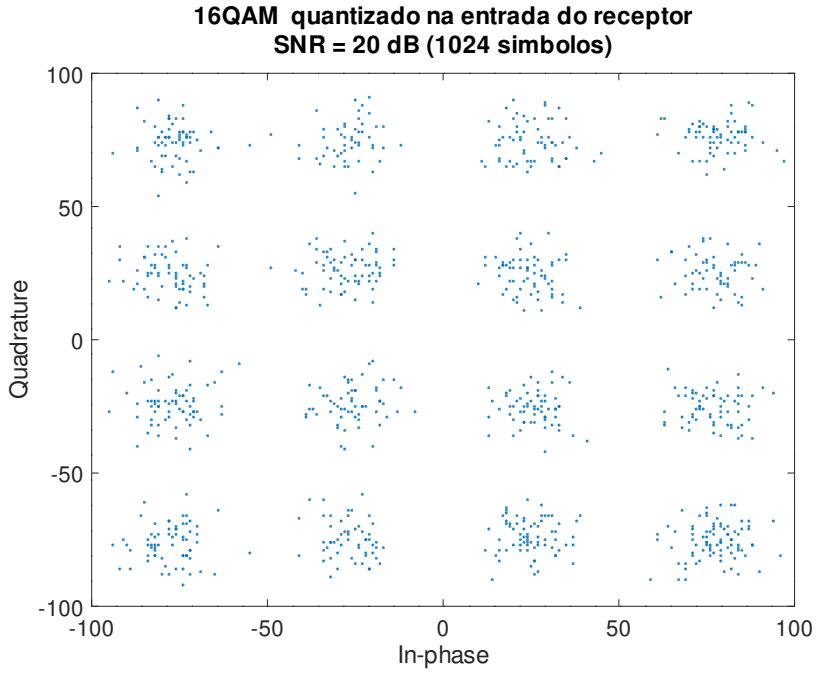

Figura 25 – 16QAM recebido na entrada do receptor

A simulação produz 1024 símbolos pseudoaleatórios para cada uma das modulações. Esses símbolos são sobreamostrados pelo fator requerido pelo filtro, mapeados nas constelações e filtrados pelos dois filtros SRRC quantizados, sendo um utilizado para os dados em fase e outro para os que estão em quadratura. Em seguida, o sinal produzido é normalizado para o intervalo $[-1, 1]$ e é degradado pelo ruído branco gaussiano aditivo, resultando em uma relação sinal ruído de 20 dB.

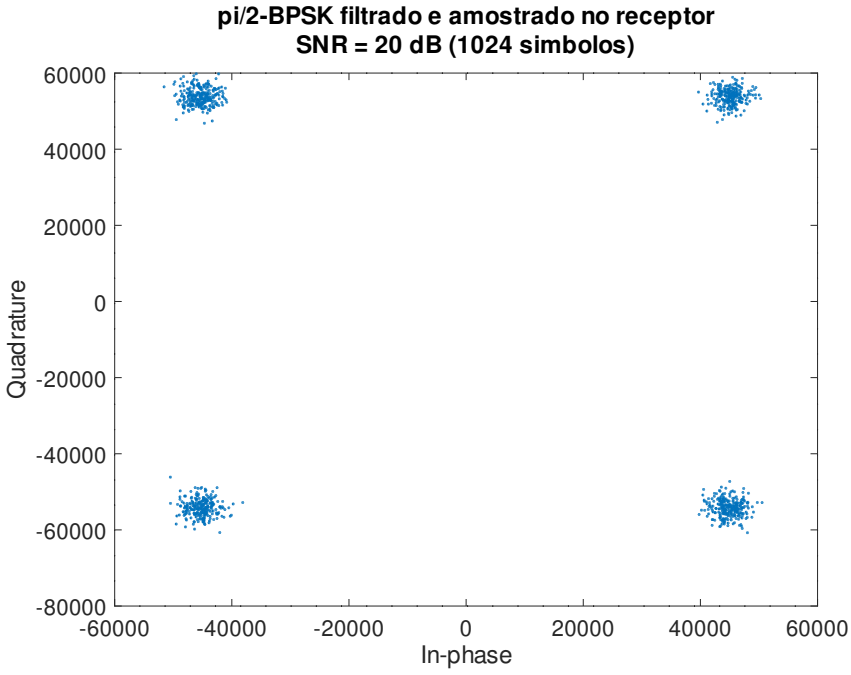

Figura 26 – $\pi/2$-BPSK recuperado no receptor

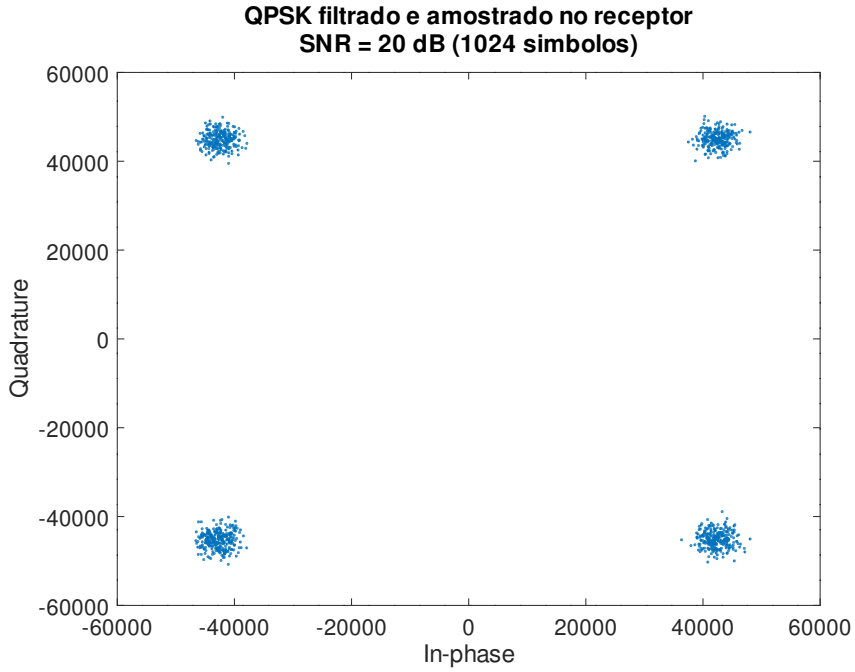

Figura 27 – QPSK recuperado no receptor

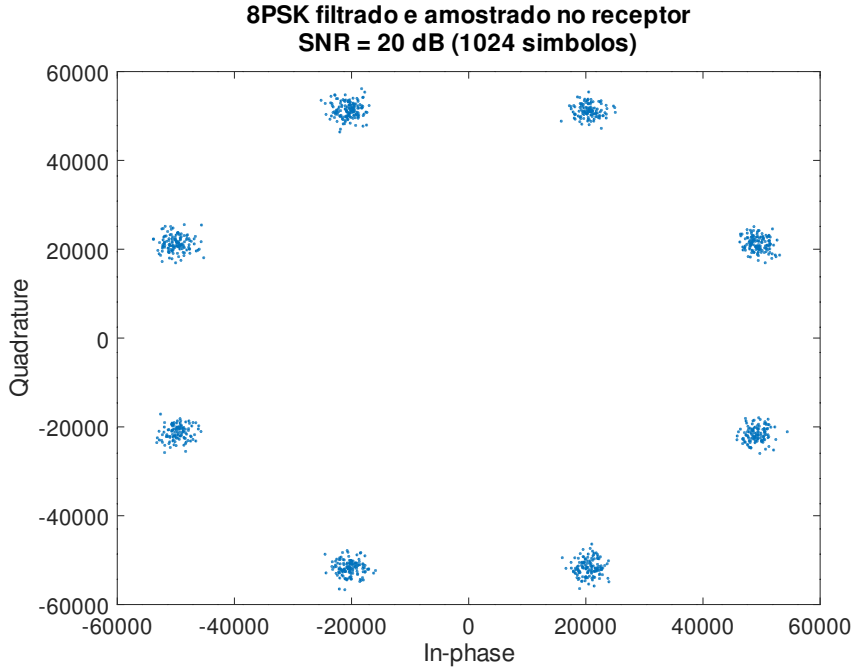

Figura 28 – 8PSK recuperado no receptor

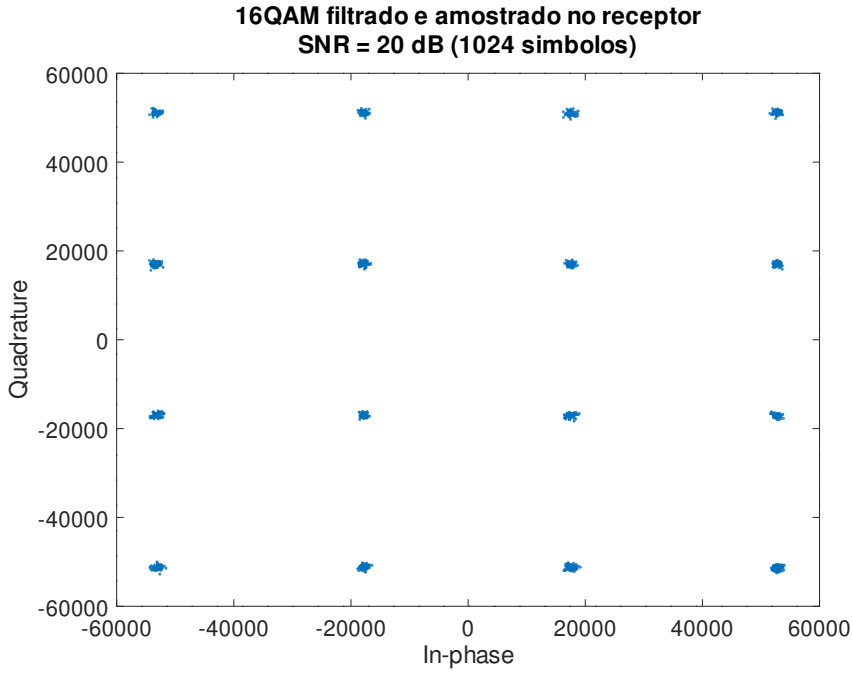

Figura 29 – 16QAM recuperado no receptor

O sinal complexo em ponto flutuante é então dividido em suas partes real e imaginária e, em seguida, quantizado para ponto fixo com 8 bits de parte fracionária. Os sinais resultantes são então filtrados novamente por dois filtros SRRC com coeficientes quantizados e são subamostrados para recuperação dos símbolos. O script supõe que o conversor analógico digital é ideal e que o tempo dos símbolos é conhecido pelo receptor, eliminando, assim, a necessidade de aplicação de filtros de sincronização e reamostragem.

3.6.2 Descrição em VHDL

O conjunto com as quatro modulações lineares do protocolo DVB-RCS2 foi subdividido em blocos, implementados como diferentes entidades integradas a uma entidade principal como componentes. A subdivisão do sistema em uma entidade principal, na qual residem as interfaces para outros blocos e uma máquina de estados finitos para controle do sistema; um sobreamostrador, responsável por entregar as amostras no formato esperado pelo filtro; um demultiplexador e um multiplexador para seleção da constelação e modulação desejada; e dois filtros SRRC, um para a parte real em fase e outro para a fase final em quadratura, está disponível na figura 30.

A máquina de estados finitos que controla o comportamento do sistema está disponível na figura 31. Após um sinal de reconfiguração assíncrono o sistema aguarda no estado *INITIAL* até que seja solicitada uma configuração do sistema. O pedido de configuração muda o estado para *GETCFG*, em que a máquina permanece até que seja enviado um sinal de configuração pronta, o que manda a máquina para o estado de aplicação das configurações recebidas *SETCFG* e em seguida, para um estado de espera *IDLE*. Essa

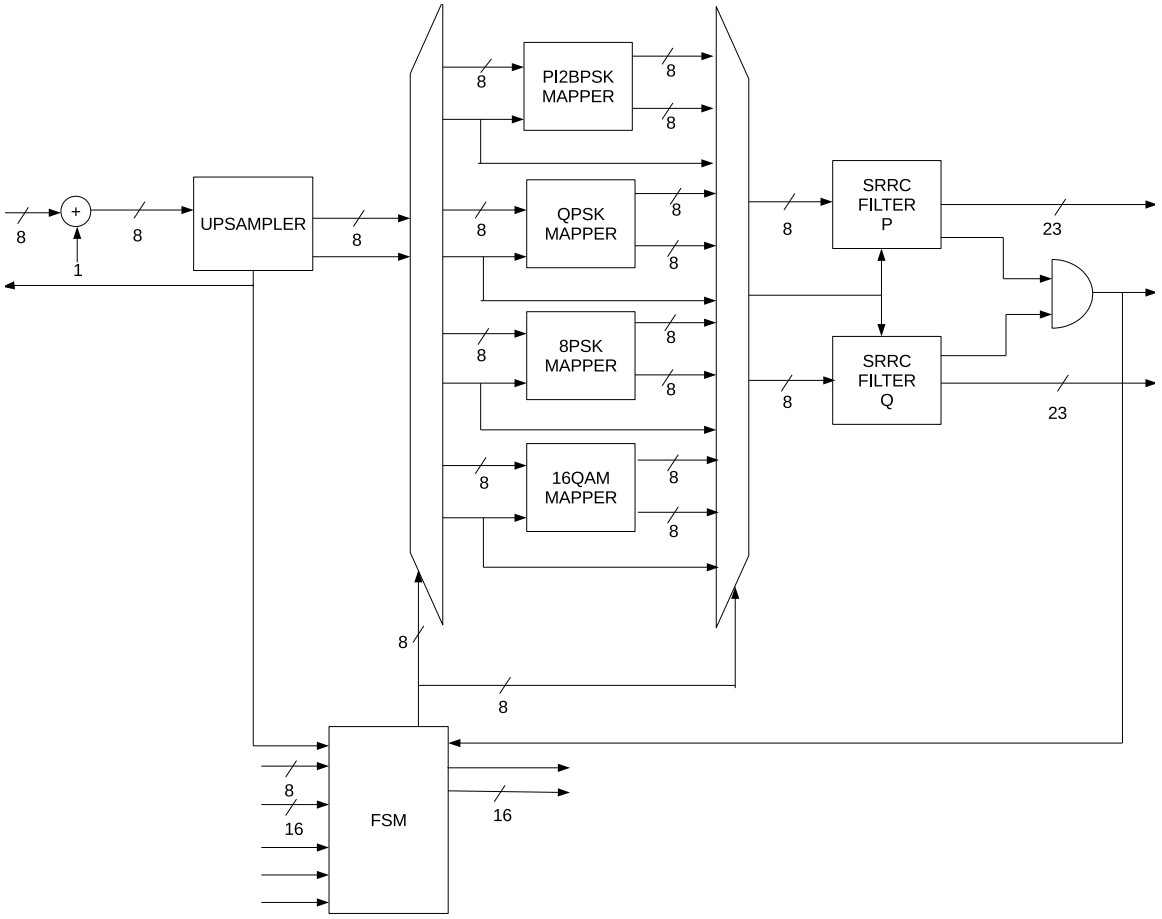

Figura 30 – Diagrama de blocos da arquitetura do modulador em banda base descrito em VHDL

abordagem garante que a máquina precisa obrigatoriamente de uma configuração inicial antes de transmitir a modulação escolhida e também permite que o componente que controlará o modulador em banda base tenha tempo de processar quais configurações devem ser aplicadas e possa defini-las somente quando estiverem prontas.

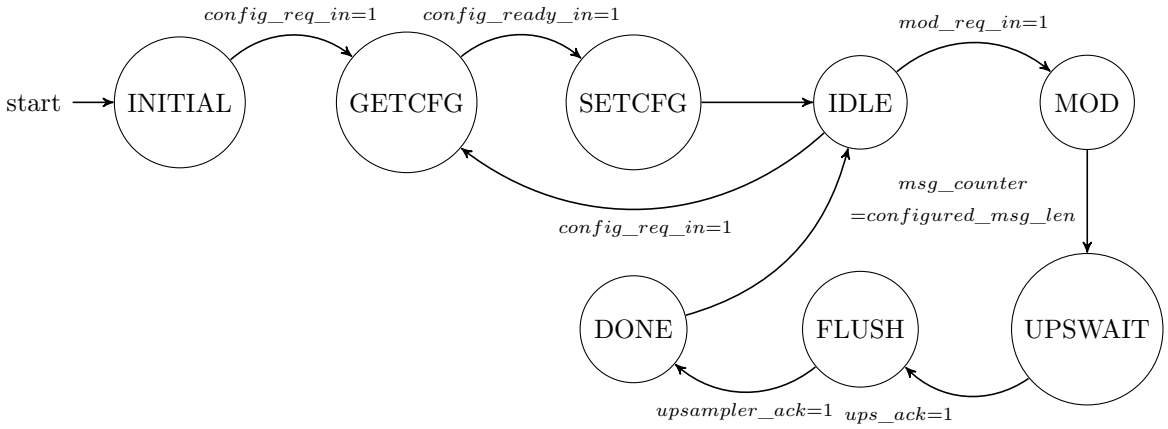

Figura 31 – Diagrama da FSM de controle do sistema

Quando no estado ocioso *IDLE*, a máquina pode mudar para o estado de confi-

guração *GETCFG*, caso seja sinalizado um pedido de configuração ou para o estado de modulação *MOD* se receber um pedido para modulação. No estado *MOD*, a operação do modulador é controlada diretamente pela máquina de controle do sobreamostrador e pelo contador interno dos dois filtros FIR. Esse estado sinalizará ao sistema responsável por enviar os símbolos ao modulador quando o filtro estiver pronto para receber um novo símbolo, e ao sistema responsável por receber os dados modulados em banda base quando o sinal de saída estiver válido.

Também no estado de modulação, são liberados para contagem dois contadores responsáveis pela contagem das amostras válidas enviadas para o bloco posterior e o número de amostras lidas do bloco anterior. Quando o número de amostras aceitas é igual ao número de amostras configuradas, o controlador entra no estado *UPSWAIT*, no qual o contador de amostras recebidas para e aguarda o estado ocioso do sobreamostrador para que possa solicitar a operação de esvaziamento dos registradores de *pipeline* do sistema no estado seguinte, *FLUSH*.

Ao término da operação de propagação da última amostra útil até a saída registrada do sistema, a máquina de controle do modulador entra no estado *DONE*. Nesse estado o contador de amostras produzidas é parado e o seu valor é registrado por um ciclo de relógio na saída do sistema. Em seguida o controlador retorna para o estado ocioso e espera que seja solicitada uma reconfiguração ou um comando de modulação.

3.6.2.1 Arquitetura para testes

Visto que o sistema completo apresenta grande número de entradas e saídas e de modo a possibilitar o teste do sistema descrito já implementado em hardware, foi criada uma arquitetura de testes. Essa arquitetura, disponível na figura 32, instancia o sistema, blocos de controle, memórias com os vetores de entrada do sistema e um somador para os sinais modulados em fase e quadratura. O somador foi instanciado para facilitar a obtenção dos dados de saída através do analisador lógico integrado que pode ser usado nos FPGAs da empresa Xilinx.

Para facilitar a codificação, a arquitetura de testes usa memórias em bloco instanciadas a partir do *IP Core Block Memory Generator* 8.4 disponível no Vivado. As quatro memórias, uma para cada modulação, foram carregadas com os 256 símbolos de teste de cada modulação e instanciadas em blocos de controle responsáveis por atualizarem os endereços e as saídas enviadas para o bloco principal.

As máquinas de estados finitos que controlam o avanço do endereço das memórias e os sinais de saída válida funcionam de modo semelhante à máquina que controla o sobreamostrador, pois ambas estão diretamente relacionadas (ver figura 17). O sistema sempre retorna para o estado inicial *IDLE* quando um *reset* assíncrono é recebido. Nesse estado são zerados os sinais de endereço da memória e de dado de saída válido.

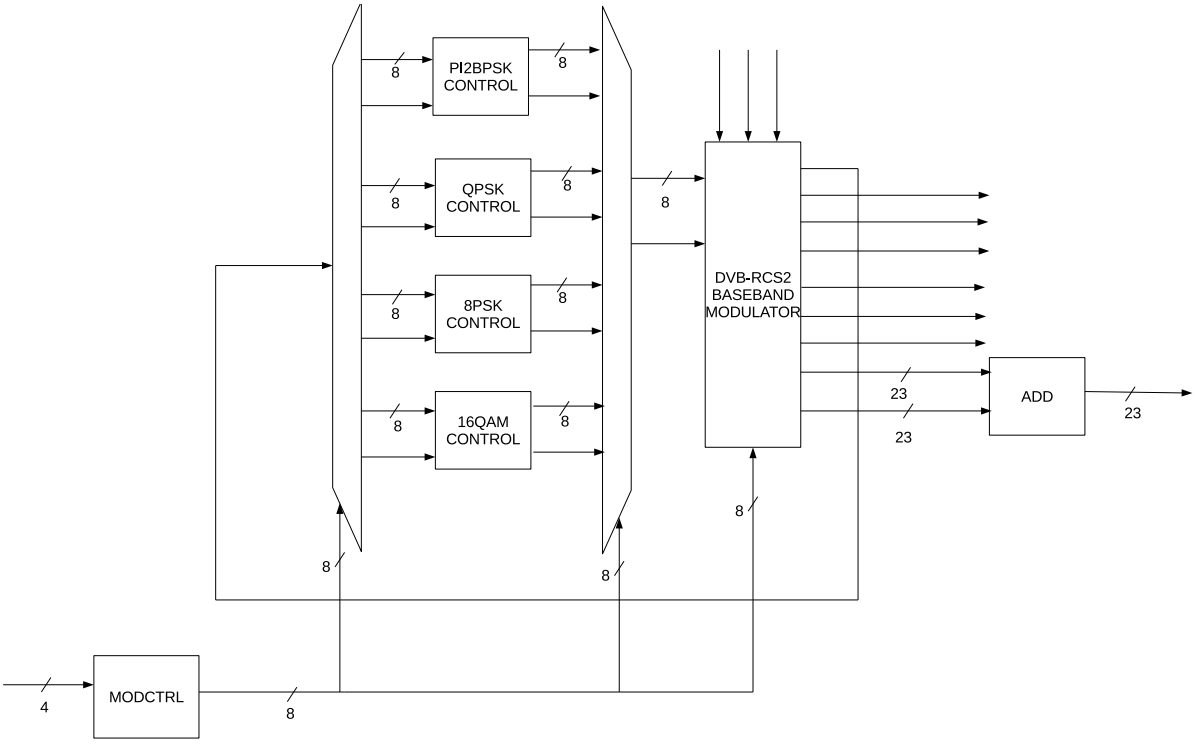

Figura 32 – Diagrama de blocos da arquitetura de testes do modulador em banda base descrito em VHDL

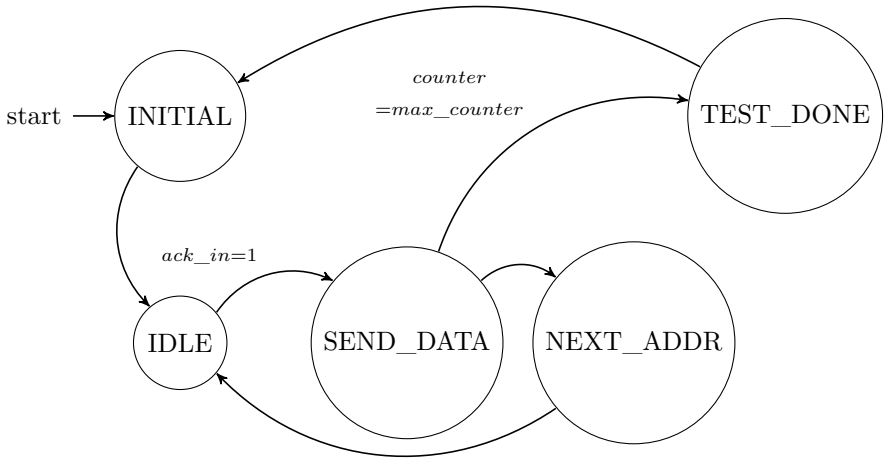

Figura 33 – Diagrama da FSM de controle das memórias de teste

Caso o controlador receba do sobreamostrador que este está pronto para receber uma amostra, o estado muda para *SEND_DATA*. O dado disponibilizado na saída da memória é então entregue ao sobreamostrador com a linha de dado válido em nível lógico alto. O próximo estado passa a ser *NEXT_ADDR*, no qual o endereço atual é incrementado e a máquina volta para *IDLE*.

Se o endereço enviado em *SEND_DATA* for o último disponível na memória, o controlador entra no estado de *TEST_DONE*. Nesse estado a saída é declarada inválida e em seguida o controlador zera o endereço da memória pois chega ao estado *INITIAL*. Não

há reinício do ciclo de leitura da memória já que o número de posições lidas é configurado anteriormente ao início da requisição de dados pelo sobreamostrador (ver diagrama 31 referente à máquina de estados principal).

O bloco de testes também mapeia não só as entradas e saídas do módulo principal para pinos de entrada e saída do chip, mas também adiciona entradas de controle manual para configuração da modulação desejada e configuração da máquina de estados finitos do bloco modulador em banda base.

Os valores obtidos após a configuração da máquina para as quatro modulações são então registrados no computador através de um ILA com 4096 amostras para cada sinal de interesse: sinais modulados em banda base e sinais de controle do modulador. O esquemático gerado após a síntese desse projeto está disponível no Apêndice A. Após a captura, os dados obtidos são exportados em CSV para que possam ser comparados com os valores de referência usando um script interpretado pelo GNU/Octave.

4 Resultados

O modulador em banda base das modulações lineares do DVB-RCS2 instanciado com a arquitetura de testes com e sem o hardware de instrumentação foram implementados no kit de desenvolvimento Zybo Revisão B da empresa Digilent usando as estratégias padrões de síntese e implementação do Vivado 2018.3.1. Todas as implementações usam a frequência de relógio padrão de 125 MHz, a não ser quando especificado o contrário.

Além disso, os dados de consumo de recursos do FPGA e da temporização do circuito implementado foram retirados dos relatórios pós implementação do Vivado e os gráficos de sinais modulados e erros quadráticos médios foram produzidos através de um script interpretado pelo GNU/Octave. Esse script compara os dados de referência produzidos anteriormente com os obtidos pelo ILA.

4.1 Modulador em banda base e arquitetura de testes

Os resultados de consumo de recursos da implementação no FPGA nas condições descritas anteriormente estão disponíveis na tabela 10 e os parâmetros de temporização do circuito, *Worst Negative Slack* (WNS), *Worst Hold Slack* (WHS) e *Worst Pulse Width Slack* (WPWS), na tabela 11. Já a figura 34 apresenta o leiaute do circuito implementado, sendo que as informações de roteamento foram ocultadas para melhor visualização dos componentes.

Tabela 10 – Consumo de recursos pós-implementação do modulador e da arquitetura de testes

Recurso	**Uso**	**Uso (%)**
LUT	3921	22,28
FF	4547	12,92
BRAM	2	3,33
IO	38	38,00
BUFG	1	3,13

As simulações comportamentais feitas no Vivado 2018.3.1 estão presentes na figura 36 e caracterizaram o sistema na sua frequência de operação padrão de 125 MHz, cujo período é de 8 ns. Após a síntese e a implementação, também foi obtido o consumo estimado de potência do circuito, presente na figura 35.

A latência foi medida tomando-se o intervalo entre a primeira borda de subida do relógio com o sinal de requisição de modulação *mod_req_in* em nível alto. A taxa de

Tabela 11 – Temporização pós-implementação do modulador e da arquitetura de testes a 125 MHz

Parâmetro	**Valor**
WNS	1,533 ns
WHS	0,038 ns
WPWS	3,5 ns

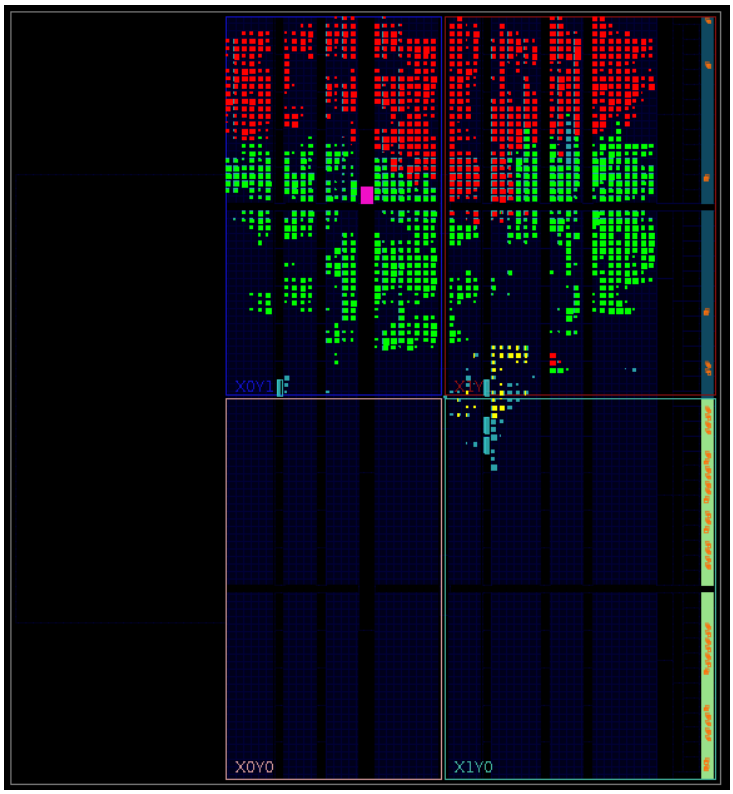

Figura 34 – Leiaute do circuito com o modulador e a arquitetura de testes. O filtro SRRC em fase está destacado em vermelho, o filtro em quadratura em verde, e o sobreamostrador em amarelo

transmissão foi calculada de modo semelhante: mediu-se o intervalo entre a requisição de modulação e a sinalização de modulação *modulation_done_out*, sendo que os 256 símbolos foram transmitidos em um intervalo de 14624 ns. Esses dados são iguais para as quatro modulações e estão disponíveis na tabela 12. Eles consideram o atraso de um ciclo de relógio para atualização de cada uma das 256 amostras que entram no modulador.

A figura 34 também destaca os blocos do sistema que mais consumiram os recursos: o filtro SRRC para os dados reais (em fase) está destacado na cor vermelha, o filtro para os dados em quadratura está na cor verde, e o sobreamostrador na cor amarela. Os blocos na cor azul claro representam outros componentes do sistema que não ocupam muitos recursos do sistema ou foram otimizados na implementação, como as máquinas de estados, as memórias de teste e os mapeadores.

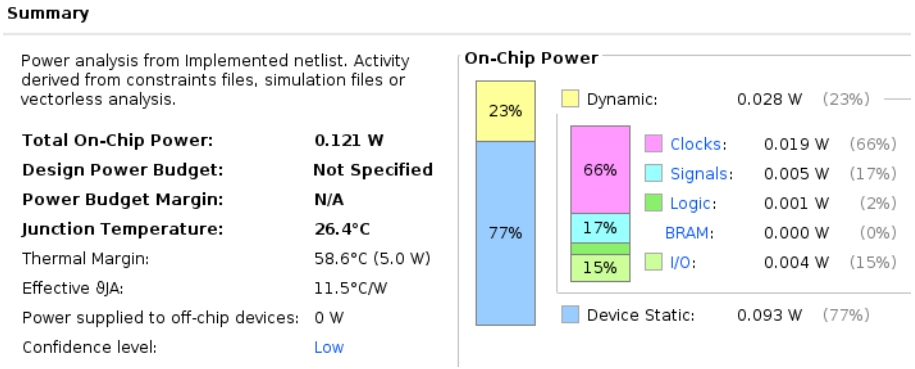

Figura 35 – Consumo de potência do modulador e da arquitetura de testes a 125 MHz

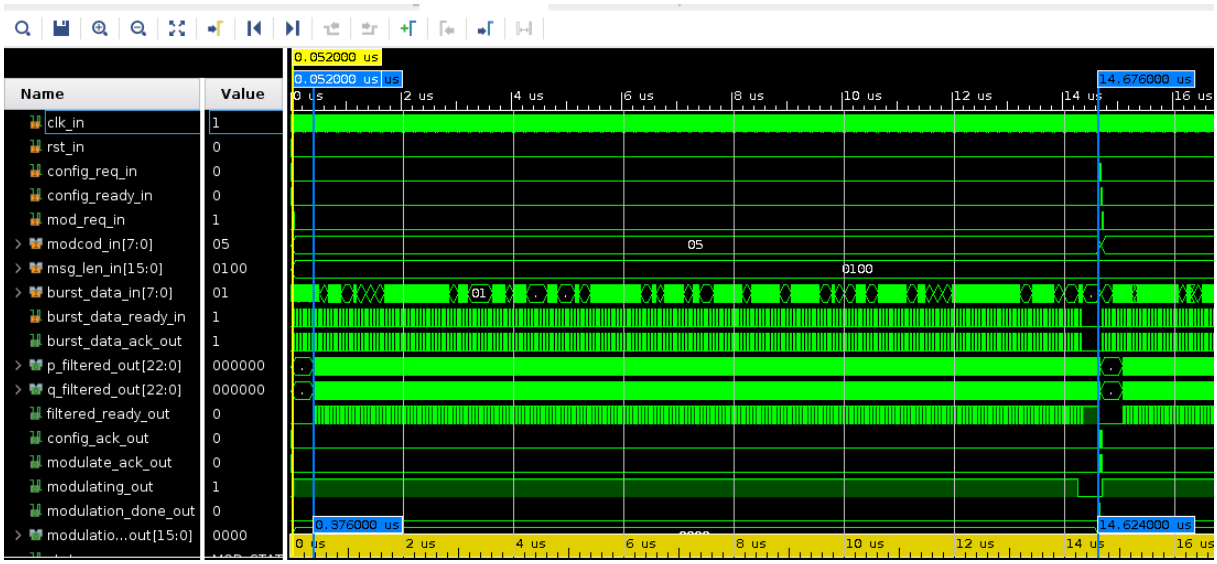

Figura 36 – Simulação comportamental do modulador e da arquitetura de testes

Tabela 12 – Latência e taxa de transmissão do modulador e da arquitetura de testes a 125 MHz

Parâmetro	Valor
Latência	376 ns
Taxa de transmissão	17,50547 MHz

Através de modificações no arquivo de restrições foi possível implementar o conjunto modulador e arquitetura de testes com uma frequência de operação 33% maior que a utilizada por padrão no kit de desenvolvimento. Os dados de temporização do circuito implementado com uma frequência de relógio de 166,67 MHz, ou seja, com um período de relógio de 6 ns, estão disponíveis na tabela 13.

Tabela 13 – Temporização pós-implementação do modulador e da arquitetura de testes a 166,67 MHz

Parâmetro	**Valor**
WNS	0,088 ns
WHS	0,031 ns
WPWS	2,5 ns

4.2 Modulador em banda base, arquitetura de testes e hardware de instrumentação

O conjunto modulador, arquitetura de testes e hardware de instrumentação foi implementado somente na frequência padrão de 125 MHz devido ao alto consumo de recursos do chip gerado pela instanciação do ILA. Os dados de consumo de recursos e de temporização pós implementação estão disponíveis, respectivamente, nas tabelas 14 e 15.

Tabela 14 – Consumo de recursos pós-implementação do modulador, da arquitetura de testes e do hardware de instrumentação

Recurso	**Uso**	**Uso (%)**
LUT	10403	59,11
LUTRAM	1867	31,12
FF	11671	33,16
BRAM	10	16,67
IO	38	38,00
BUFG	2	6,25

Tabela 15 – Temporização pós-implementação do modulador, da arquitetura de testes e do hardware de instrumentação a 125 MHz

Parâmetro	**Valor**
WNS	0,285 ns
WHS	0,029 ns
WPWS	2,75 ns

Na figura 37 estão destacados os componentes que mais consomem recursos do chip. Em marrom estão os componentes do hardware de instrumentação inseridos pelo ILA, em amarelo, o bloco sobreamostrador, em vermelho o filtro SRRC dos dados reais e em verde o filtro dedicado aos dados imaginários. Os demais componentes em azul claro não ocupam muitos recursos individualmente.

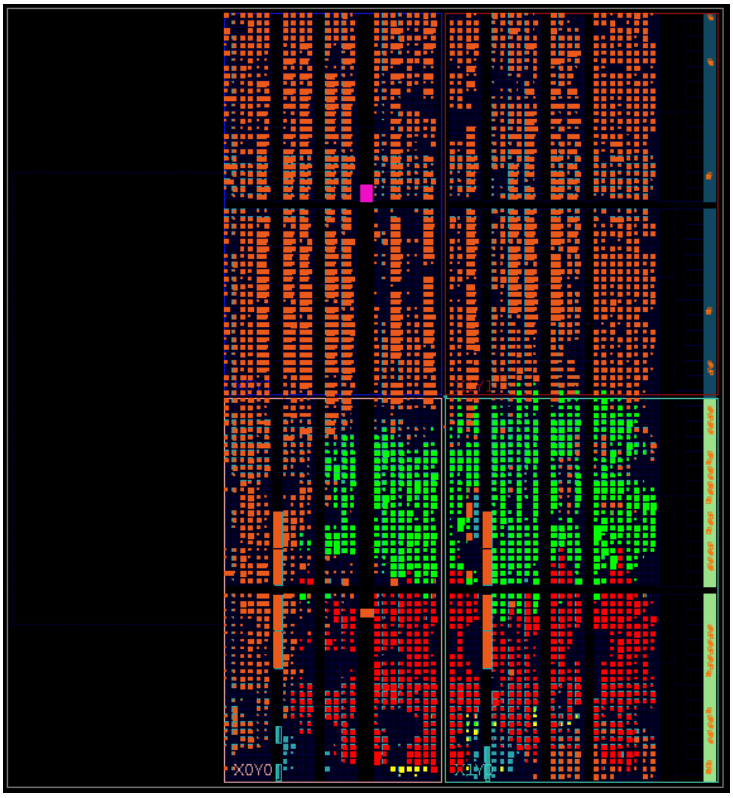

Figura 37 – Leiaute do circuito com o modulador, a arquitetura de testes e o hardware de instrumentação. O filtro SRRC em fase está destacado em vermelho, o filtro em quadratura em verde, o ILA em marrom, e o sobreamostrador em amarelo

4.2.1 Comparação com os modelos de referência

Com os dados obtidos e processados por um script de comparação, foram gerados gráficos que comparam as saídas obtidas através dos CSVs exportados do ILA e os valores de referência gerados pelos outros scripts para cada uma das modulações. As figuras 38, 39, 40 e 41 são gráficos nos quais foram plotadas as respostas experimentais e teóricas das modulações $\pi/2$-BPSK, QPSK, 8PSK e 16QAM respectivamente.

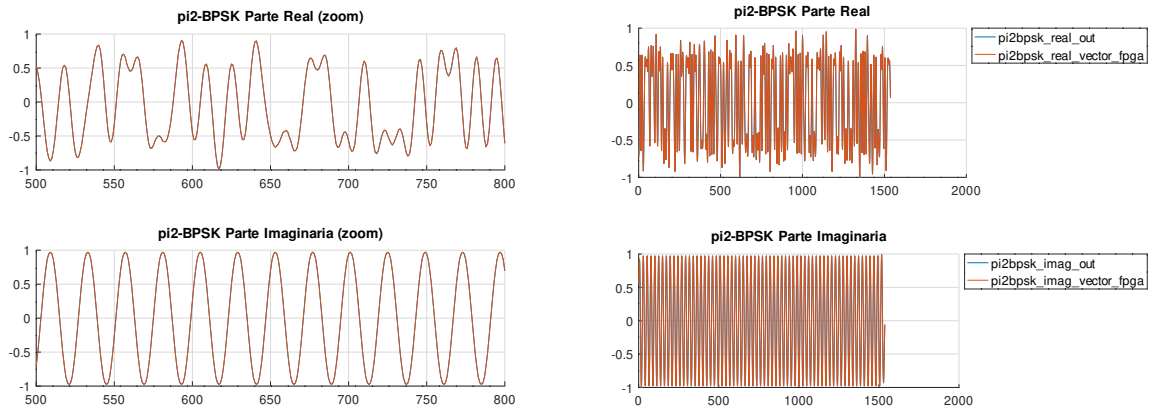

Figura 38 – Comparação entre os sinais de referência e experimental para a modulação $\pi/2$-BPSK

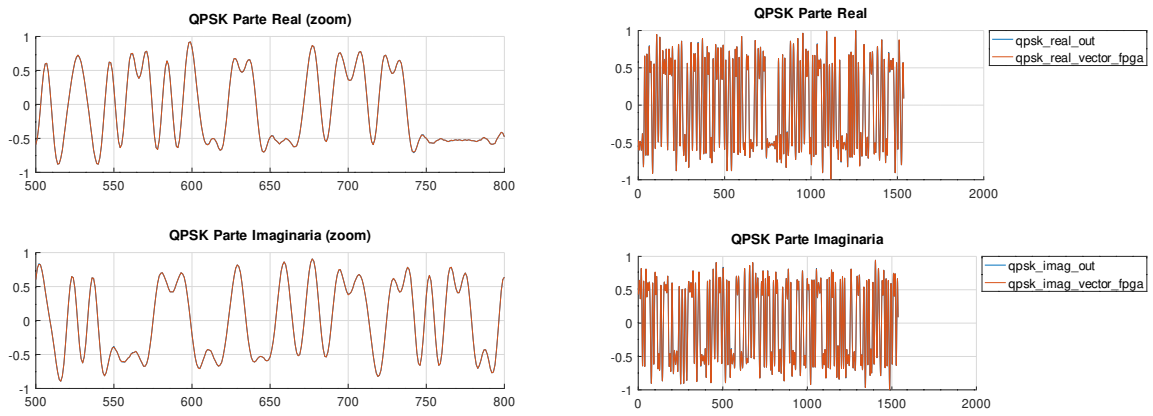

Figura 39 – Comparação entre os sinais de referência e experimental para a modulação QPSK

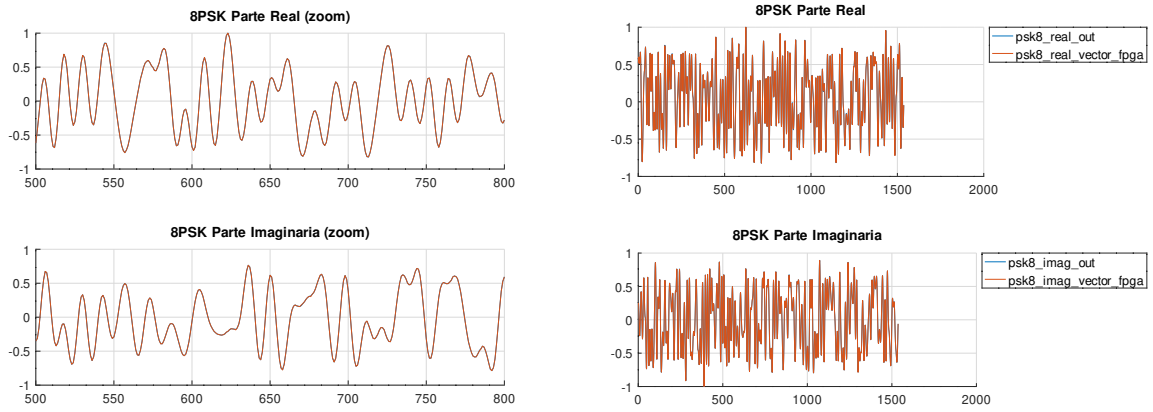

Figura 40 – Comparação entre os sinais de referência e experimental para a modulação 8PSK

Calculou-se o erro quadrático de cada uma das amostras e o erro quadrático médio para as partes real e imaginária dos sinais em banda base. Os gráficos com os erros amostral e médio estão disponíveis nas figuras 42, 43, 44 e 45 para as modulações $\pi/2$-BPSK, QPSK, 8PSK e 16QAM respectivamente. A tabela 16 contém os erros quadráticos médios entre o que foi aferido experimentalmente e o que foi simulado para todas as modulações testadas.

Tabela 16 – Erro quadrático médio e amostral entre os sinais de referência e experimentais

Modulação	**Parte real**	**Parte imaginária**
$\pi/2$-BPSK	$6,91 \times 10^{-6}$	$1,1 \times 10^{-5}$
QPSK	$6,31 \times 10^{-6}$	$6,1 \times 10^{-6}$
8PSK	$4,06 \times 10^{-6}$	$3,78 \times 10^{-6}$
16QAM	$3,66 \times 10^{-6}$	$4,83 \times 10^{-6}$

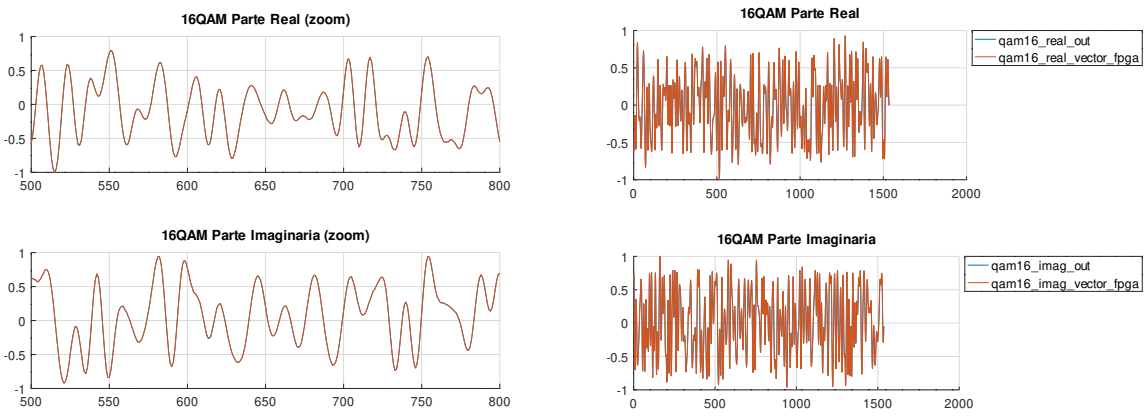

Figura 41 – Comparação entre os sinais de referência e experimental para a modulação 16QAM

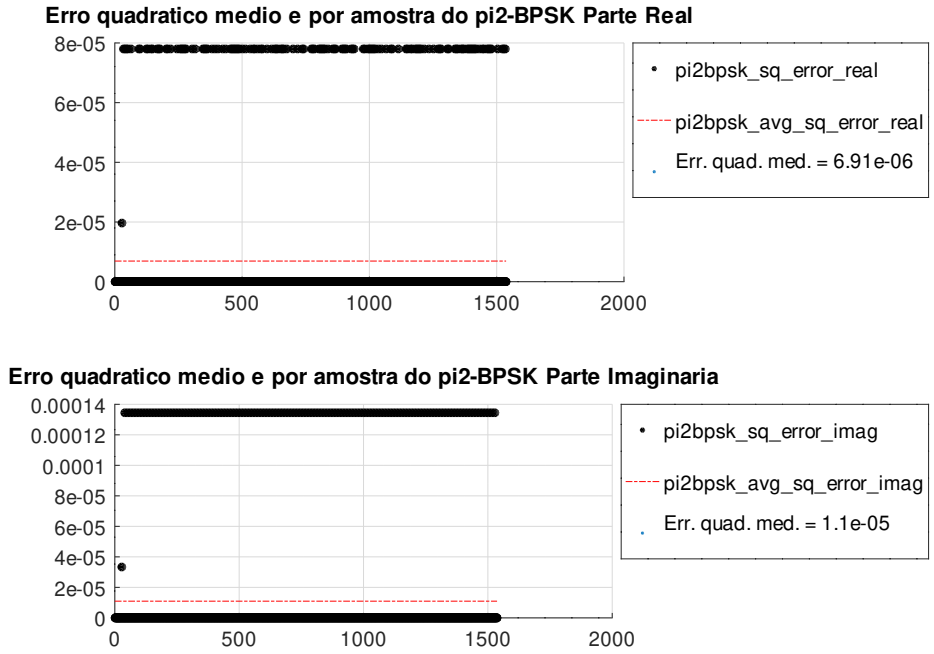

Figura 42 – Erro quadrático médio e amostral entre os sinais de referência e experimental para a modulação $\pi/2$-BPSK

Além disso, foram produzidos gráficos para verificação visual do funcionamento dos dois filtros SRRC. As figuras 46, 47, 48, e 49 representam os símbolos recuperados das quatro modulações usando os dados obtidos pelo ILA. Os dados obtidos foram apenas subamostrados para a obtenção dos 256 símbolos, supondo que o ILA sempre produz o mesmo número de amostras por símbolo e que a temporização das amostras permanece inalterada.

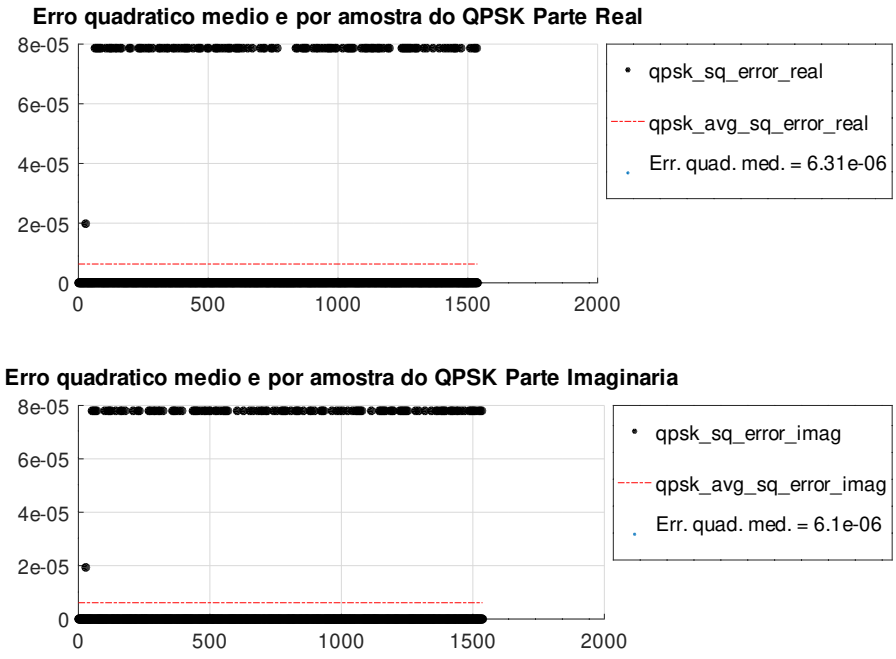

Figura 43 – Erro quadrático médio e amostral entre os sinais de referência e experimental para a modulação QPSK

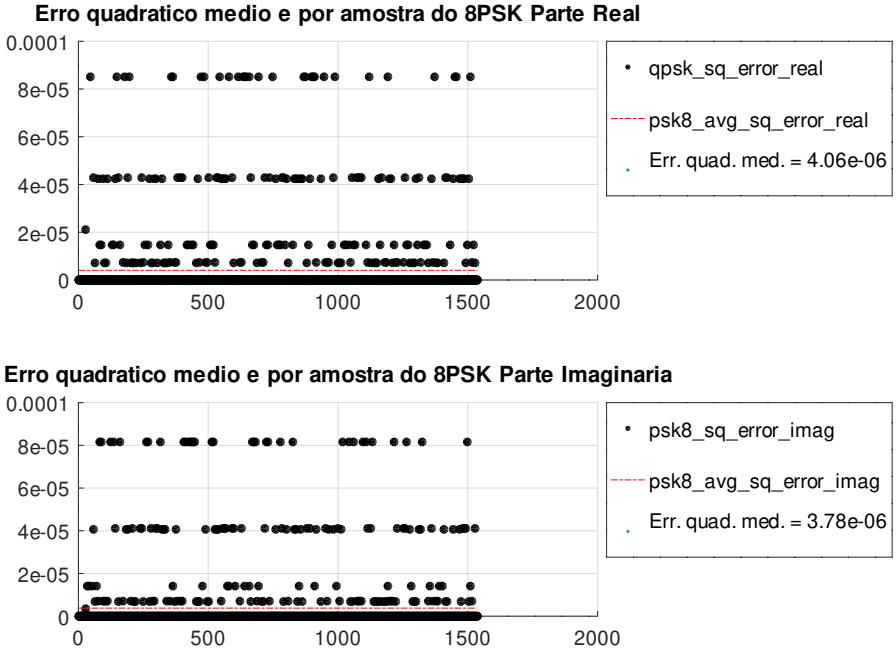

Figura 44 – Erro quadrático médio e amostral entre os sinais de referência e experimental para a modulação 8PSK

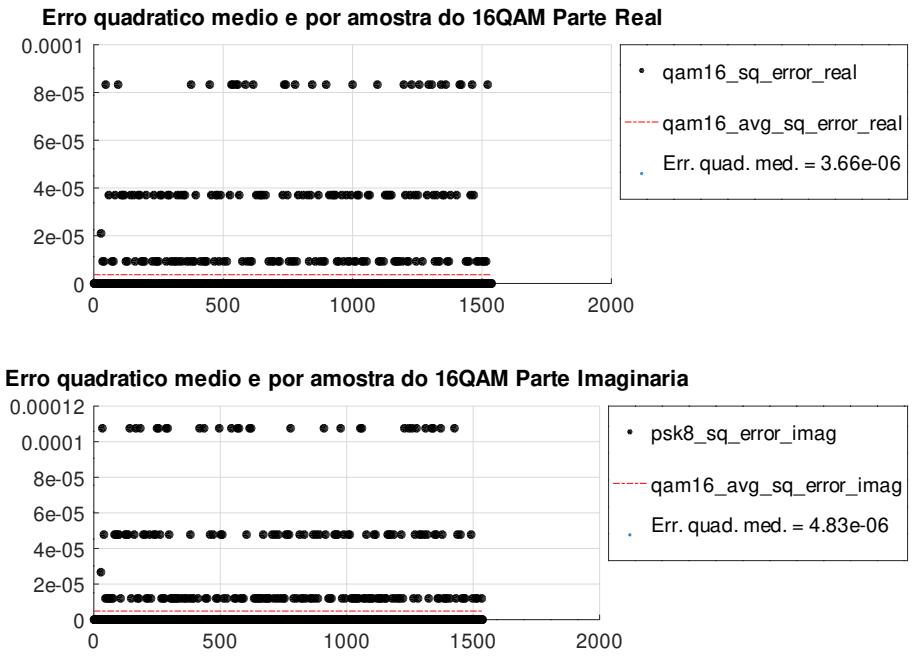

Figura 45 – Erro quadrático médio e amostral entre os sinais de referência e experimental para a modulação 16QAM

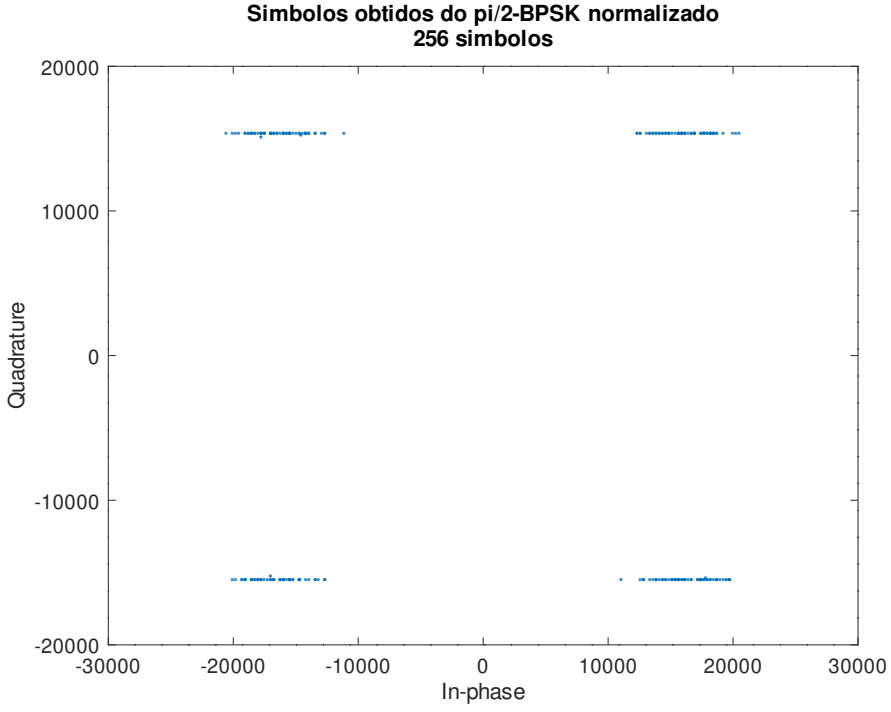

Figura 46 – Símbolos $\pi/2$-BPSK obtidos dos sinais experimentais

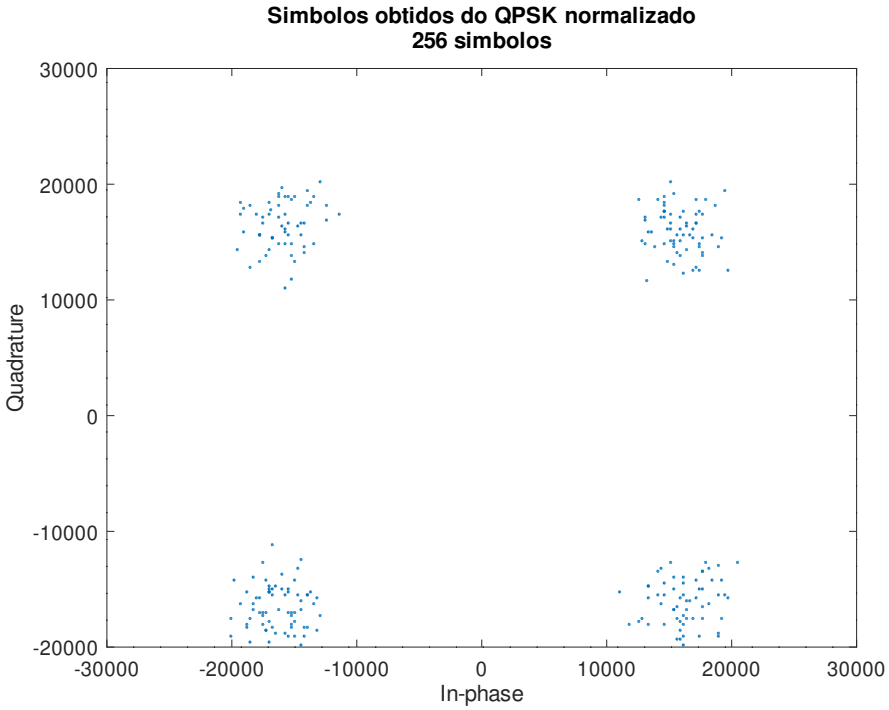

Figura 47 – Símbolos QPSK obtidos dos sinais experimentais

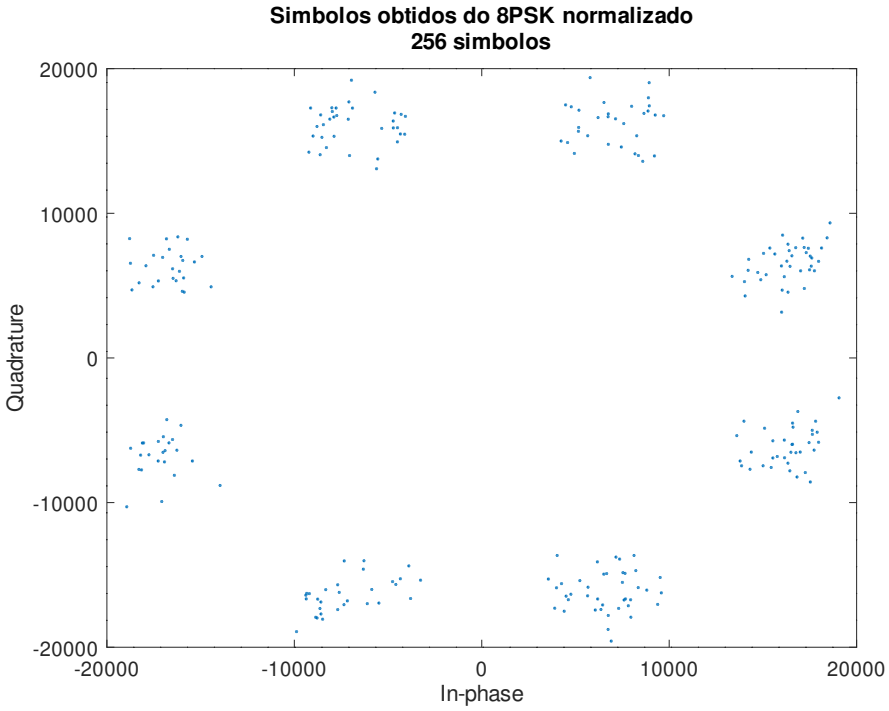

Figura 48 – Símbolos 8PSK obtidos dos sinais experimentais

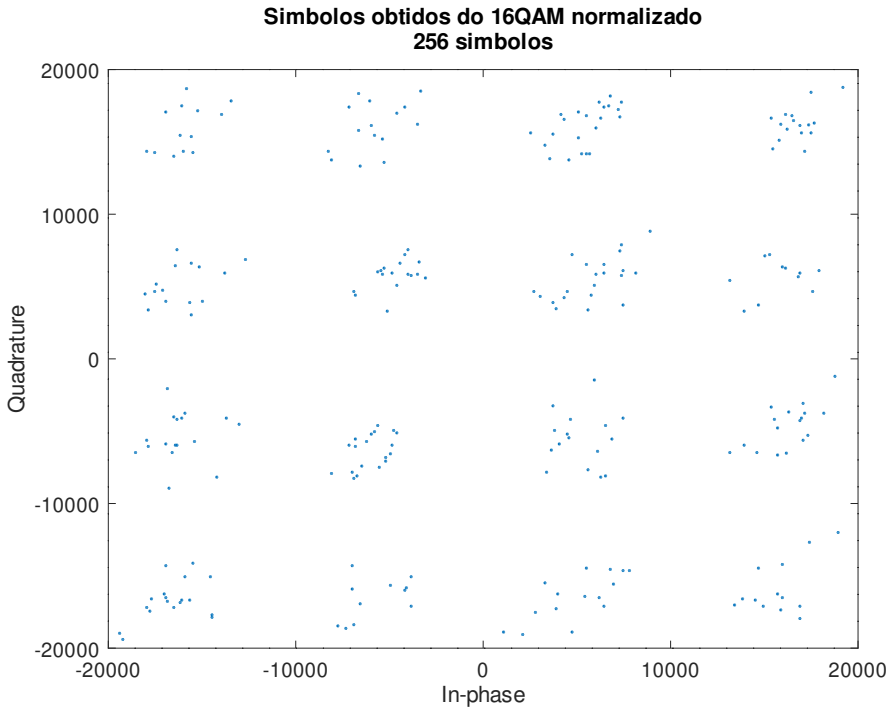

Figura 49 – Símbolos 16QAM obtidos dos sinais experimentais

5 Discussão

É possível depreender dos resultados de implementação do conjunto modulador em banda base com a arquitetura de testes (figura 34, tabelas 10 e 11) que o maior consumo de recursos no chip é devido a instanciação paralela dos dois filtros SRRC. A arquitetura destes filtros possui uma grande *pipeline* formada pela fila de amostras, pelos multiplicadores e pela árvore de somas. Como todas as operações estão registradas há um alto consumo de registradores com o benefício da diminuição de problemas de temporização, e isso é preferível pois o bloco implementado será integrado nos estágios finais de um sistema maior.

O possível aumento em 33% da frequência de operação do bloco sem hardware de instrumentação, de 125 MHz para 166,67 MHz, pode ser atribuído às causas citadas no parágrafo anterior. Apesar das folgas apertadas, disponíveis na tabela 13, ao ser implementado para operar em uma frequência maior, o desempenho do circuito nessa frequência ainda pode ser satisfatório devido à arquitetura possuir registradores nas saídas dos blocos mais complexos.

Comparando os relatórios de temporização do circuito antes e após a adição do hardware de instrumentação adicionado pelo ILA (tabelas 13 e 15), é possível observar uma piora na temporização do sistema, com folgas muito mais apertadas. Isso se deve ao grande aumento nos recursos do FPGA usados, o que dificultou um roteamento otimizado entre os blocos espalhados por todo o sistema (figura 37).

Os resultados das quatro modulações em banda base coletados pelo ILA estão razoavelmente próximos dos modelos de referência quantizados conforme pode ser visto nas figuras 42, 43, 44 e 45. Nas figuras 46, 47, 48 e 49 é possível verificar que, conforme esperado, o filtro SRRC não garante interferência entre símbolos nula.

O pequeno erro quadrático médio das partes real e imaginária dos sinais modulados pode ser atribuído a erros de quantização, pois pioram nas modulações com menor distância intersimbólica, como a 16QAM. Esses erros podem ser reduzidos com um aumento no número de bits usados para quantizar os coeficientes do filtro e os símbolos das constelações.

6 Conclusão

As quatro modulações lineares usadas pelo protocolo DVB-RCS2 foram implementadas em um kit de desenvolvimento Zybo Rev. B, que contém o SoC XC7Z010-1CLG400C. A implementação foi descrita em VHDL e divide o sistema em blocos: sobreamostrador, mapeadores de símbolos para constelações, filtros formatadores de pulso (SRRC) para as partes real e imaginária, uma máquina de estados finitos para gerenciamento das configurações e uma arquitetura de testes para facilitar a obtenção dos sinais de saída. A síntese, implementação e gravação do *netlist* gerado foram feitas com o Vivado 2018.3.1.

Além da FSM de controle do sistema principal, responsável pelo controle da configuração e da modulação, há uma máquina de controle do sobreamostrador. Ela é responsável por controlar a entrada de dados no sistema e inserir as amostras zeradas necessárias para a correta operação do filtro, que opera com sobreamostragem, e para limpar os registradores presentes na *pipeline* do sistema.

Os modelos descritos em alto nível em scripts interpretados pelo GNU/Octave foram usados para gerar a os coeficientes do filtro formatador de pulso, arquitetura em VHDL do filtro SRRC, as entradas, os padrões de referência para o sistema e para comparar os dados obtidos com os modelos teóricos. As amostras experimentais foram retiradas do sistema com o uso do ILA, um hardware de instrumentação disponibilizado pelo Vivado.

Foram testadas duas versões do modulador instanciado com a máquina de testes: a primeira não possui o hardware de instrumentação, e a segunda o possui. Foi possível implementar a primeira solução com uma frequência de operação de até 166,67 MHz. Já a segunda usa a frequência de operação padrão da placa de desenvolvimento, 125 MHz, e foi usada para obtenção das amostras experimentais das quatro modulações.

A análise do erro quadrático médio entre as amostras geradas pelos modelos GNU/Octave e as obtidas experimentalmente revelou que os sistemas estão razoavelmente próximos, e a análise do relatório de temporização do circuito sem o ILA mostrou razoáveis folgas. É possível, portanto, atestar o funcionamento correto do sistema implementado.

Referências

ANITHA, K.; UMESHARADDY; B.K.SUJATHA. FPGA Implementation of High Throughput Digital QPSK Modulator using Verilog HDL. *International Journal of Advanced Computer Research*, v. 4, n. 1, mar. 2014. ISSN 2277-7970. Citado na página 32.

COMBLOCK. *COM-1402 PSK / QAM / APSK modulator VHDL SOURCE CODE OVERVIEW*. [S.l.], 2008. Disponível em: <https://comblock.com/download/com1402soft.pdf>. Citado na página 32.

CREONIC GMBH. *DVB-RCS2 Modulator Product Brief*. [S.l.], 2018. Citado 2 vezes nas páginas 15 e 33.

EATON, J. W. et al. *GNU Octave version 4.4.0 manual: a high-level interactive language for numerical computations*. [S.l.], 2018. Disponível em: <https://www.gnu.org/software/octave/doc/v4.4.1/>. Citado na página 35.

ETSI; EBU. *EN 301 545-2 - V1.2.1 - Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 2: Lower Layers for Satellite standard*. [S.l.], 2014. Disponível em: <http://www.etsi.org/deliver/etsi_en/301500_301599/30154502/01.02.01_60/en_30154502v010201p.pdf>. Citado 7 vezes nas páginas 15, 23, 24, 29, 37, 47 e 48.

ETSI; EBU. *ETSI TR 101 545-5 - V1.1.1 - Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 5: Guidelines for the Implementation and Use of TS 101 545-3*. [S.l.], 2014. Citado na página 23.

ETSI; EBU. *TS 101 545-1 - V1.2.1 - Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 1: Overview and System Level specification*. [S.l.], 2014. Disponível em: <http://www.etsi.org/deliver/etsi_ts/101500_101599/10154501/01.02.01_60/ts_10154501v010201p.pdf>. Citado 3 vezes nas páginas 23, 24 e 28.

HARRIS, D. M.; HARRIS, S. L. *Projeto Digital e Arquitetura de Computadores*. 2. ed. [S.l.]: Imagination Technologies, 2013. ISBN 978-0-9954839-0-3. Citado 4 vezes nas páginas 13, 30, 31 e 36.

JOOST Michael. *Theory of Root-Raised Cosine Filter*. 2010. Disponível em: <https://michael-joost.de/rrcfilter.pdf>. Citado 2 vezes nas páginas 37 e 38.

K., S.; PARIKH, K. S. FPGA Design and Implementation of Selectable M-PSK Modulators. *International Journal of Engineering Research and Reviews*, v. 4, n. 4, p. 74–87, out. 2016. ISSN 2348-697X. Citado na página 32.

KHAIRUDIN, N. et al. Implementing Root Raised Cosine (RRC) filter for WCDMA using Xilinx. In: *2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA)*. [S.l.: s.n.], 2011. p. 203–207. ISSN 2159-2055. Citado na página 32.

LATHI, B. P.; DING, Z. *Modern Digital and Analog Communication Systems*. 4. ed. New York, NY, USA: Oxford University Press, Inc., 2010. ISBN 978-0-19-538493-2. Citado 3 vezes nas páginas 13, 27 e 28.

LUCENA, A. M. P. de et al. *Gerador de Sinais BPSK em FPGA para Aplicações Espaciais*. [S.l.], 2014. Disponível em: <http://urlib.net/8JMKD3MGP3W34P/3HJ29LP>. Citado na página 32.

MUHAMMAD, A. B.; DARLIS, D.; FAHMI, A. Design and Realization of Digital Modulator BPSK, QPSK and 16-QAM on FPGA. *Journal of Measurement, Electronic, Communication and Systems*, 2016. ISSN 9772477798001. Citado na página 32.

NINGDALLI, A.; SUJATHA, B. K. FPGA Implementation of PI/2 BPSK and PI/4 QPSK for IEEE 802.15.6 WBAN Standard. *International Research Journal of Engineering and Technology*, v. 2, n. 3, jun. 2015. ISSN 2395-0056. Disponível em: <https://www.irjet.net/archives/V2/i3/Irjet-v2i3109.pdf>. Citado na página 32.

PERKINS, C. E. Mobile ip. *IEEE Communications Magazine*, p. 84–99, May 1997. Citado na página 23.

PILATO, L.; MEONI, G.; FANUCCI, L. Design and quantization limits of root raised cosine digital filter. In: *2017 3rd International Conference on Frontiers of Signal Processing (ICFSP)*. [S.l.: s.n.], 2017. p. 59–62. Citado na página 32.

PRIYANTO, B. E.; LAW, C. L.; GUAN, Y. L. Design & implementation of all digital I-Q modulator and demodulator for high speed WLAN in FPGA. In: *2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing (PACRIM 2003) (Cat. No.03CH37490)*. [S.l.: s.n.], 2003. v. 2, p. 659–662 vol.2. Citado na página 32.

SKINNEMOEN, H. et al. DVB-RCS2 overview. *International Journal of Satellite Communications and Networking*, v. 31, n. 5, p. 201–217, 2013. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/sat.1025>. Citado na página 28.

TANAWADE, N.; SUDHANSU, S. FPGA Implementation of Digital Modulation Techniques BPSK and QPSK using HDL Verilog. *International Journal of Computer Applications*, v. 165, n. 12, maio 2017. ISSN 0975–8887. Disponível em: <https://pdfs.semanticscholar.org/b337/06bea637bf9905f166f540b8a24b0278e2dc.pdf>. Citado na página 32.

VAHID, F. *Sistemas Digitais - Projeto, Otimização e HDLs*. [S.l.]: Bookman, 2008. ISBN 978-85-7780-190-9. Citado 3 vezes nas páginas 13, 30 e 31.

WEILIANG, Z. et al. Design and FPGA implementation of high-speed square-root-raised-cosine FIR filters. In: *Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop.* [S.l.: s.n.], 2002. p. 232–235. Citado na página 32.

WYGLINSKI, A. M. et al. Revolutionizing software defined radio: case studies in hardware, software, and education. *IEEE Communications Magazine*, v. 54, n. 1, p. 68–75, January 2016. ISSN 1558-1896. Citado na página 24.

XILINX. *Zynq-7000 SoC Data Sheet: Overview (DS190)*. [S.l.], 2018. Disponível em: <https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf>. Citado 2 vezes nas páginas 13 e 31.

YASHODHA, H.; HEMALATHA, N. Design and analysis of different types of PSK modems on FPGA for SDR. *International Journal of Recent Advances in Engineering & Technology*, v. 4, n. 9, 2016. ISSN 2347-2812. Disponível em: <http://www.irdindia.in/journal_ijraet/pdf/vol4_iss9/20.pdf>. Acesso em: 20 jun. 2019. Citado na página 32.

Apêndices

APÊNDICE A – Diagrama de blocos pós-síntese da arquitetura de testes com hardware de instrumentação