

Gabriel Takaoka Nishimura  
Felippe Demarqui Ramos  
Vivian Kimie Isuyama

***Light Cyber: Um estudo sobre a  
implementação de Light Fidelity utilizando a  
norma IEEE 802.15.7***

São Paulo

2016

Gabriel Takaoka Nishimura

Felippe Demarqui Ramos

Vivian Kimie Isuyama

***Light Cyber: Um estudo sobre a implementação de Light Fidelity utilizando a norma IEEE 802.15.7***

Trabalho de Conclusão de Curso apresentado  
ao Departamento de Engenharia de Computa-  
ção e Sistemas Digitais da Escola Politécnica  
da Universidade de São Paulo para obtenção  
do título de Bacharel de Engenharia

Universidade de São Paulo - USP

Escola Politécnica

Engenharia Elétrica com ênfase em Computação

Orientador: Prof. Dr. Bruno Carvalho de Albertini

São Paulo

2016

# Agradecimentos

À instituição de ensino e todos os seus docentes por fornecerem uma aventura de graduação inesquecível aos integrantes do grupo.

Ao nosso orientador, Bruno de Carvalho Albertini, por todo o apoio e paciência com o grupo. Seu otimismo e bom humor sempre foram grande fonte de motivação.

Aos docentes e técnicos do Departamento de Sistemas Eletrônicos pelas aulas extracurriculares de eletrônica.

À Carolina Ribeiro Minchin e Caique Bernardes Queiroz Magalhães por suas exímias revisões. A gentileza foi imensurável.

Às nossas famílias pelo amor, incentivo e apoio incondicional.

Por fim aos nossos amigos, que acreditaram no nosso potencial.

*“In a dark place we find ourselves,  
and a little more knowledge lights our way.  
(Yoda, Episode 3: Revenge of the Sith)*

# Resumo

O presente trabalho analisa a comunicação por luz visível, dentro do contexto da norma IEEE 802.15.7. Verificaram-se, pois, alternativas para uma implantação da primeira camada física definida nesta norma, ressaltados os mecanismos de codificação, decodificação e correção de erros, transmissão e recepção pela luz visível. A partir desta análise foi definido um projeto, cuja execução final gerou um módulo receptor e um módulo transmissor, entre os quais acontece transferência de dados pela luz, sem presença de cabeamento entre os dois. Com efeito, os resultados finais indicam a viabilidade desse tipo de comunicação para projetos futuros, com destaque para aplicações no uso doméstico ou privado.

**Palavras-chave:** comunicação por luz visível. Li-Fi. IEEE 802.15.7. PHY I.

# Abstract

The current project analyses the concept of visible light communication within the context of the IEEE 802.15.7 standard. The main activities included the assessment of alternatives for the deployment of the first physical layer defined by this standard. Therefore, methods concerning encoding, decoding, error correction, transmission and reception through visible light were emphasized. Through this evaluation the project was defined. Its final execution has generated reception and transmission units, which are able to transfer data without any wiring system. Effectively, the final deliverables point out the viability of this type of communication for future projects, highlighting domestic and private applications.

**Keywords:** visual light communication. Li-Fi. IEEE 802.15.7. PHY I.

# Listas de ilustrações

Figura 1 – Caracterização dos espectros de frequências, de 0Hz a 1000THz - frequências reguladas estão em laranja. . . . .	17
Figura 2 – Arquitetura do sistema VLC do Instituto Fraunhofer . . . . .	20
Figura 3 – Configuração experimental da transmissão WDM. AWG: Gerador de Funções; APD: Fotodiodo de Avalanche . . . . .	21
Figura 4 – Arquitetura do sistema VLC desenvolvido pela Universidade de Fudan. Observa-se a modulação por comprimento de onda com as três cores do LED e o processamento adicional para converter os símbolos em CAP. . . . .	22
Figura 5 – Estrutura Analítica do Projeto Light Cyber. . . . .	23
Figura 6 – Diagrama de Gantt do projeto LiCy. . . . .	24
Figura 7 – Requisitos Funcionais de Hardware Digital. . . . .	25
Figura 8 – Requisitos Funcionais de Hardware Analógico. . . . .	26
Figura 9 – Requisitos Não Funcionais. . . . .	27
Figura 10 – Arquitetura de dispositivos VPAN. . . . .	28
Figura 11 – Diagrama de blocos da codificação da mensagem. . . . .	29
Figura 12 – Diagrama de blocos da codificação da mensagem. . . . .	32
Figura 13 – Circuito lógico de codificação Reed Solomon (15,7). . . . .	32
Figura 14 – Esquemático do Codificador Convolucional com taxa 1/3, os quadrados representam registradores e os retângulos os somadores. . . . .	34
Figura 15 – Padrão de <i>puncture</i> para obter código a taxa 1/2. . . . .	35
Figura 16 – Padrão de repetição utilizado para obter um código com taxa 1/4. . . .	35
Figura 17 – Diagrama de blocos da decodificação da mensagem. . . . .	36
Figura 18 – Visão de máquina de estados de um codificador convolucional. . . . .	37
Figura 19 – A tabela Trellis é útil para entender o processo de decodificação de uma máquina de estados. . . . .	38
Figura 20 – Módulo genérico de cálculo de uma das síndromes . . . . .	40
Figura 21 – Diagrama funcional de decodificação Reed Solomon. . . . .	42
Figura 22 – Estrutura da mensagem, composta pelo Cabeçalho de Sincronização (SHR), Cabeçalho da Camada Física (PHR) e Unidade de Dados de Serviço PHY (PSDU). . . . .	43
Figura 23 – Estrutura do SHR - Cabeçalho da Camada Física. . . . .	44
Figura 24 – Estrutura do PSDU - Mensagem e <i>tail bits</i> . . . . .	45
Figura 25 – Diagrama de blocos da conversão digital-analógica e analógica-digital da transmissão de dados. . . . .	46
Figura 26 – Equivalência de circuitos com MOSFET em região de corte. . . . .	48
Figura 27 – Equivalência de circuitos com MOSFET em região de saturação. . . . .	48

Figura 28 – FET-n em modo de fonte comum. . . . .	49
Figura 29 – Circuito polarizador de LED, com resistor limitador de corrente. . . . .	49
Figura 30 – Símbolo esquemático de um LED, similar ao do diodo. . . . .	50
Figura 31 – Circuito Amplificador de Transimpedância. . . . .	52
Figura 32 – Comparador e curva de resposta. . . . .	53
Figura 33 – Saída esperada após remoção do acoplamento DC, aplicando filtro passa-alta RC de primeira ordem (à direita), com defasagem de 45°. A área hachurada representa tensão inadequada para o comparador. . . . .	55
Figura 34 – Circuito amplificador de diferenças. . . . .	57
Figura 35 – Desenho esquemático da arquitetura do sistema. . . . .	61
Figura 36 – Simulação de codificação do codificador Reed Solomon. . . . .	62
Figura 37 – Diagrama esquemático do fluxo de dados do <i>Interleaver</i> . . . . .	63
Figura 38 – Diagrama esquemático do <i>Interleaver</i> , com unidade de controle e memória. Os sinais azuis representam entradas para melhor visualização. . . . .	64
Figura 39 – Ordem do algoritmo entrelaçador para gravar a memória. . . . .	65
Figura 40 – Simulação da etapa de gravação do módulo <i>Interleaver</i> . . . . .	66
Figura 41 – Simulação da etapa de saída do módulo <i>Interleaver</i> . . . . .	66
Figura 42 – Simulação do Codificador Convolucional. Nota-se a forma que ele serializa os dados do <i>Interleaver</i> em ccEncoderInput antes de utilizá-los, para gerar a saída em ccEncoderOutput. . . . .	67
Figura 43 – Simulação do Codificador Manchester. . . . .	67
Figura 44 – Relação entre vazão de <i>bits</i> entre cada módulo do Codificador. . . . .	68
Figura 45 – Diagrama esquemático do <i>Sync</i> . O <i>clock</i> do registrador de deslocamentos foi omitido para simplificar a representação. . . . .	68
Figura 46 – Simulação do comportamento do Sincronizador. Quando o sinal FLP é habilitado, o gerador de <i>clock</i> reinicia. . . . .	70
Figura 47 – Visão de máquina de estados do decodificador Manchester. . . . .	70
Figura 48 – Simulação do Decodificador Manchester. . . . .	70
Figura 49 – Simulação do funcionamento do Decodificador Viterbi <i>Fangled</i> . Observa-se que o componente apenas conclui qual o símbolo recebido após quatro ciclos. . . . .	72
Figura 50 – Simulação do cálculo das síndromes. . . . .	72
Figura 51 – Simulação do módulo de Berlekamp-Massey. . . . .	73
Figura 52 – Simulação do módulo de Forney. . . . .	74
Figura 53 – Quadro de mensagem da UART utilizada no projeto. . . . .	74
Figura 54 – Relação entre vazão de <i>bits</i> entre cada módulo do Decodificador. . . . .	75
Figura 55 – Circuito de transmissão com filtro de ponta de prova. . . . .	77
Figura 56 – Circuito de transmissão sem o filtro de ponta de prova. . . . .	79

Figura 57 – Circuito final de transmissão de dados LiCy, utilizando um filtro passa-altas. . . . .	80
Figura 58 – Circuito e saída de um amplificador de impedâncias. No circuito, o fotodiodo é reversamente polarizado por $V_{bias}$ . No osciloscópio, a saída está marcada em amarelo enquanto a saída do LED está em verde. . . . .	81
Figura 59 – Circuito esquemático final de recepção de dados LiCy. . . . .	82
Figura 60 – Arquitetura do módulo de cálculo das síndromes. . . . .	91
Figura 61 – Arquitetura do módulo de Berlekamp-Massey. . . . .	92
Figura 62 – Arquitetura do módulo de busca de Chien: localização de erros. . . . .	93
Figura 63 – Arquitetura do módulo de busca de Chien: valores de erros. . . . .	93
Figura 64 – Arquitetura do módulo de <i>Interleaver</i> . . . . .	94
Figura 65 – Arquitetura do módulo de Sincronização. . . . .	94
Figura 66 – Arquitetura do módulo de Decodificação Manchester. . . . .	94
Figura 67 – Arquitetura do módulo de Decodificação Viterbi. . . . .	95

# Lista de Gráficos

1	Crescimento do tráfego IP do ano 2014 ao 2019 . . . . .	16
2	Curva característica de um MOSFET, com indicações das regiões de saturação, corte e triodo. . . . .	47
3	Zona de operação do diodo. . . . .	50
4	Saída esperada do Amplificador de Transimpedância, com amplitude de $i_D \cdot R_F$ , com componente $V_{DC} = L$ variável. . . . .	53
5	Gráfico com adição de componente DC $V_{DC} = V_{ref}$ fixa e filtro passa-altas modificado. . . . .	56
6	Saída esperada após a etapa de comparação em verde. . . . .	56
7	<i>Oversampling</i> com <i>baud clock</i> . . . . .	75
8	Características de transferência do MOSFET de potência IRLZ14. . . . .	76
9	Comportamento do circuito de transmissão com filtro de ponta de prova em série com a entrada. A onda azul é saída do gerador de funções enquanto a onda amarela é a tensão submetida ao LED. Observa-se o comportamento de subamortecimento em ambas. . . . .	78
10	Operação de circuito transmissor sem ponta de prova. A onda amarela representa voltagem no LED sem filtro de ponta de prova em frequências mais altas. O gerador de funções é medido e gera a forma de onda verde. .	79
11	Forma de onda após adicionar um capacitor que age como filtro passa-altas. Está defasada em $180^\circ$ . . . . .	80
12	Saída do transmissor em verde e saída digital convertida do receptor em amarelo. É possível observar uma defasagem de $90^\circ$ em relação às ondas. .	82

# **Lista de tabelas**

Tabela 1 – Análise Comparativa entre os trabalhos sobre VLC encontrados.	22
Tabela 2 – Modos de operação da camada PHY I de Li-Fi.	28
Tabela 3 – Corpos de Galois (16)	30
Tabela 4 – Parâmetros do Código Convolucional da camada PHY I.	34
Tabela 5 – Codificação Manchester.	36
Tabela 6 – Definição de Topologias para o TDP.	44
Tabela 7 – Definição dos campos do cabeçalho PHY.	45
Tabela 8 – Comparaçao entre modos de operação do fotodiodo.	51
Tabela 9 – Comportamento real de um comparador com amplificador operacional.	54
Tabela 10 – Fonte: Autores.	54
Tabela 11 – Características Dinâmicas do MOSFET IRLZ14.	77

# **Lista de abreviaturas e siglas**

AC	Corrente Alternada (do inglês <i>Alternating Current</i> )
ACS	Adicionar, Comparar e Selecionar (do inglês <i>Add, Compare and Select</i> )
ADC	Conversor Digital Analógico (do inglês <i>Analog Digital Converter</i> )
BM	Métrica de Ramificação (do inglês <i>Branch Metrics</i> )
BER	Taxa de Erro de <i>bit</i> (do inglês <i>Bit Error Rate</i> )
CAP	<i>Carrier-less amplitude and phase</i>
DC	Corrente Contínua (do inglês <i>Direct Current</i> )
DMT	Multi-Tom Discreto (do inglês <i>Discrete Multi-Tone</i> )
EAP	Estrutura Analítica de Projeto
FET	Transistor de Efeito de Campo (do inglês <i>Field Effect Transistor</i> )
FLP	Padrão de Rápido Travamento (do inglês <i>Fast Locking Pattern</i> )
FPGA	Arranjo de Portas Programável (do inglês <i>Field Programmable Gate Array</i> )
GF	Campos de Galois (do inglês <i>Galois Field</i> )
GND	Terra (do inglês <i>Ground</i> )
GPIO	Pinos de Entrada e Saída de Propósito Geral (do inglês <i>General Purpose Input/Output</i> )
HDL	Linguagem de Descrição de Hardware (do inglês <i>Hardware Description Language</i> )
IEEE	Instituto dos Engenheiros Elétricos e Eletrônicos (do inglês <i>Institute of Electrical and Electronics Engineers</i> )
IP	Propriedade Intelectual (do inglês <i>Intellectual Property</i> )
ISI	Interferência Inter-Símbolo (do inglês <i>Inter-symbol Interference</i> )
LED	Diodo Emissor de Luz (do inglês <i>Light Emitting Diode</i> )
LUT	Tabela de Pesquisa (do inglês <i>Look Up Table</i> )
MAC	Controle de Acesso de Mídia (do inglês <i>Media Access Control</i> )

M-CMMA	<i>Modified Cascaded Multi-Modulus Algorithm</i>
MCS	Esquema de Modulação e Codificação (do inglês <i>Modulation and Coding Scheme</i> )
MOSFET	<i>Metal Oxide Semiconductor Field Effect Transistor</i>
OFDM	Multiplexação de Divisão de Frequência Ortogonal (do inglês <i>Orthogonal Frequency Divided Multiplexing</i> )
OOK	Chaveamento Alto-Baixo (do inglês <i>On-Off Keying</i> )
PHR	Cabeçalho do PHY (do inglês <i>PHY Header</i> )
PISO	Entrada Paralela para Saída Serial (do inglês <i>Parallel Input to Serial Output</i> )
PM	Métrica de Caminho (do inglês <i>Path Metrics</i> )
PPDU	Unidade de Dados da Camada Física (do inglês <i>Physical Layer Data Unit</i> )
PSDU	Unidade de Serviço de dados PHY I (do inglês <i>PHY Service Data Unit</i> )
QAM	Modulação de amplitude em quadratura (do inglês <i>Quadrature Amplitude Modulation</i> )
RLL	Comprimento de Execução Limitado (do inglês <i>Run Length Limited</i> )
RLS	<i>Recursive Least Square</i>
RGB	Vermelho Verde e Azul (do inglês <i>Red Green and Blue</i> )
RS	Reed Solomon
RTL	Nível de Transferência de Registradores (do inglês <i>Register Transfer Level</i> )
SHR	Cabeçalho de Sincronização (do inglês <i>Synchronization Header</i> )
SIPO	Entrada Serial para Saída Paralela (do inglês <i>Serial Input to Parallel Output</i> )
TDP	Padrão Dependente de Topologia (do inglês <i>Topology Dependant Pattern</i> )
TTL	Lógica de Transistor-Transistor (do inglês <i>Transistor-Transistor Logic</i> )

UART	Receptor/Transmissor Asíncrono Universal (do inglês <i>Universal Asynchronous Receiver/Transmitter</i> )
UC	Unidade de Controle
USB	Barramento Serial Universal (do inglês <i>Universal Serial Bus</i> )
VCC	Voltagem no Coletor Comum
VHDL	Linguagem Verilog de Descrição de Hardware (do inglês <i>Verilog Hardware Description Language</i> )
VPAN	<i>Visible-Light Communication Personal Area Network</i>
WDM	Multiplexação por Divisão de Frequência (do inglês <i>Wavelength Division Multiplexing</i> )

# Sumário

<b>Introdução</b>	16
<b>1 REVISÃO BIBLIOGRÁFICA</b>	19
<b>1.1 Lifi-X</b>	19
<b>1.2 Franhoufer-Gesellschaft</b>	19
<b>1.3 Scuola Superiore Sant'Anna</b>	20
<b>1.4 Fudan University</b>	21
<b>1.5 Análise Comparativa</b>	22
<b>2 METODOLOGIA</b>	23
<b>2.1 Planejamento</b>	23
2.1.1 Estrutura Analítica do Projeto	23
2.1.2 Cronograma	24
2.1.3 Requisitos	24
<b>2.2 Norma IEEE 802.15.7</b>	27
2.2.1 Transmissão	29
2.2.2 Recepção	36
2.2.3 Estrutura da mensagem	43
<b>2.3 Hardware</b>	45
2.3.1 Transmissão	45
2.3.2 Recepção	51
2.3.3 FPGA	58
<b>2.4 Suporte ao hardware digital</b>	59
2.4.1 VHDL	59
2.4.2 Quartus	59
<b>3 EXECUÇÃO</b>	61
<b>3.1 Arquitetura</b>	61
<b>3.2 Codificador Digital</b>	62
3.2.1 Codificador Reed Solomon	62
3.2.2 Interleaver	62
3.2.3 Codificador Convolucional	66
3.2.4 Codificador Manchester	67
3.2.5 Integração do Codificador	67
<b>3.3 Decodificador Digital</b>	68
3.3.1 Sync	68

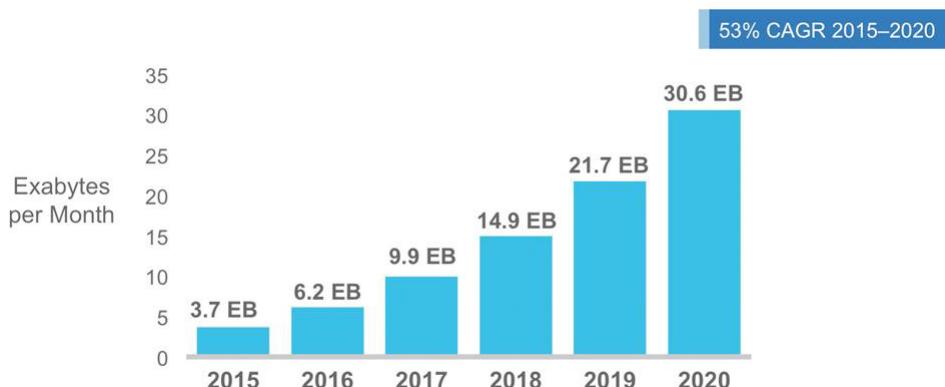
3.3.2	Decodificador Manchester . . . . .	70
3.3.3	Decodificador Viterbi Fangled . . . . .	71
3.3.4	Decodificador Reed Solomon . . . . .	72
3.3.5	UART . . . . .	74
3.3.6	Integração do Decodificador . . . . .	75
<b>3.4</b>	<b>Transmissor Analógico</b> . . . . .	<b>75</b>
3.4.1	Conversor Digital-Analógico . . . . .	76
3.4.2	Transmissão de Luz . . . . .	76
3.4.3	Versões Anteriores . . . . .	77
<b>3.5</b>	<b>Receptor Analógico</b> . . . . .	<b>81</b>
3.5.1	Recepção Luminosa . . . . .	81
3.5.2	Conversor Analógico-Digital . . . . .	82
<b>3.6</b>	<b>Integração Completa</b> . . . . .	<b>83</b>
3.6.1	Etapas da Integração . . . . .	83
<b>3.7</b>	<b>Testes</b> . . . . .	<b>83</b>
<b>4</b>	<b>CONCLUSÃO</b> . . . . .	<b>85</b>
<b>4.1</b>	<b>Trabalhos Futuros</b> . . . . .	<b>85</b>
	<b>REFERÊNCIAS</b> . . . . .	<b>87</b>
	<b>APÊNDICES</b> . . . . .	<b>89</b>
	<b>APÊNDICE A – DIAGRAMAS DA ARQUITETURA</b> . . . . .	<b>90</b>

# Introdução

## Objetivos

Há um aumento em escala global no uso de comunicação sem fio([WANG, 2015](#)). Apenas no ano de 2015, mais de meio bilhão de telefones celulares foram adicionados à rede de telefonia no mundo ([CISCO, 2016](#)). Em relação ao crescimento de aparelhos conectados como carro, geladeiras, TV's, DVD's, entre outros, existem previsões de que a sua quantidade chegará a 9 bilhões em 2020, com um crescimento de 900% em relação a 2015 ([ERRICSON, 2015](#)). O tráfego IP também fica cada vez mais denso, com previsão de que atinja 168 exabytes em 2019, como pode-se observar na [1](#).

Gráfico 1 – Crescimento do tráfego IP do ano 2014 ao 2019



Fonte: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, p. 5

Esse tipo de comunicação engloba diferentes tecnologias de transmissão e recepção de dados, incluindo algumas ainda pouco exploradas. Nesse sentido, surge um interesse relevante por uma tecnologia explorada há um período relativamente curto de tempo: a transferência de dados utilizando o espectro da luz visível ([Haas, 2011](#)). Essa comunicação é conhecida como VLC (Visible Light Communication) e posteriormente, com o uso de LED's, criou-se uma nova categoria, chamada de Li-Fi (Light Fidelity) ([HAAS, 2016](#)).

A partir do conhecimento dessas tecnologias, discutem-se alternativas para desenvolvê-las e utilizá-las de fato. Desta maneira, apresenta-se o cerne deste estudo, que é encontrar uma forma de realizar comunicação por luz visível e implementá-la. Dentre os estudos e referências que servem a esse propósito, a norma IEEE 802.15.7 ([IEEE, 2011](#)) (Li-Fi) se destaca como uma fonte confiável para o desenvolvimento de tecnologias nesta área.

Portanto, complementa-se o problema inicial, e o principal objetivo do projeto torna-se estabelecer comunicação por luz visível através da norma IEEE 802.15.7.

Em suma, o grupo tem o intuito de criar um transmissor em formato de luminária e um receptor anexado a um terminal, que pode ser um computador ou um dispositivo móvel. A comunicação deve ser simplex entre os dois módulos criados, para que o transmissor envie dados ao receptor, de modo a se trabalhar entre duas camadas físicas, a princípio. Posteriormente, podem-se integrar os módulos desenvolvidos a camadas mais altas, como Enlace, Rede, Transporte ou Aplicação.

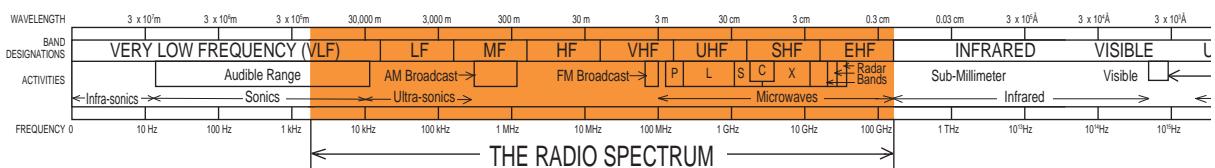
Finalmente, o escopo principal pode ser dividido em três partes:

- | Implementar em hardware uma das camadas definidas na norma IEEE 802.15.7;
- | Estabelecer transmissão e recepção de sinais entre um LED e um fotodiodo;
- | Integrar hardware digital e analógico.

## Motivação

No Brasil, a Anatel regulamenta uma banda de frequências que pertencem ao intervalo de 3KHz até 300GHz ([ANATEL, 2006](#)). Isso significa que existem frequências reservadas para tipos específicos de comunicação, com o intuito de diminuir possíveis interferências entre elas. No entanto, essa atribuição deixa aparelhos como telefones sem fio, tablets, roteadores e notebooks em frequências de livre uso, como 2.4GHz. Eventualmente, o número excessivo de dispositivos conectados em frequências livres pode causar interferência em áreas mais densas. Uma alternativa para reduzir essa possível saturação seria buscar comunicação pela luz visível, cuja frequência pertence ao intervalo de 430THz a 750THz ([BUSER, 1992](#)), e que não é regulamentada por nenhuma agência (vide [Figura 1](#)).

Figura 1 – Caracterização dos espectros de frequências, de 0Hz a 1000THz - frequências reguladas estão em laranja.



Fonte: U.S. Frequency Allocation Chart

Ademais, o uso da tecnologia Li-Fi oferece algumas vantagens ([GUPTA, 2015](#)). Não há, por exemplo, interferência com as bandas convencionais (como 2.4GHz e 5GHz), sendo portanto compatível com infraestruturas híbridas. Nota-se também uma eficiência energética do Li-Fi, pois o LED utilizado na transmissão pode aproveitar a infra-estrutura

de iluminação já existente em ambientes internos, o que justifica e reforça o uso de LED. Em sequência, pode-se citar o diferencial de que a luz não atravessa paredes, garantindo segurança e privacidade à rede.

É importante ainda ressaltar que este projeto tem como antecedentes experiências com comunicação via luz visível. Nascidas no contexto de uma disciplina de laboratório de processadores, as principais motivações foram a maior eficiência energética da VLC, bem como a diminuição de interferência com ondas de radiofrequência. No entanto, essas experiências esbarraram em desafios, principalmente relacionados à falta de um padrão de comunicação e de componentes adequados. Além disso, houve problemas relacionados à capacidade de processamento do microcontrolador utilizado, o que é consequência de um levantamento de requisitos incompleto.

Finalmente, este projeto surge como um amadurecimento dos objetivos anteriores. Neste segundo momento, foi escolhida uma norma que atendesse às expectativas de implementar comunicação por luz visível de forma eficiente. O padrão IEEE 802.15.7 é vantajoso por dois grandes motivos: (I) sua finalização em 2011, o que facilita a adequação do projeto e (II) este se encontra consolidado entre empresas e grupos industriais que formam o Consórcio Li-Fi.

# 1 Revisão Bibliográfica

Nesse capítulo será feita a verificação do estado atual de outros trabalhos sobre comunicação via luz. Serão analisados tanto o meio comercial quanto o acadêmico.

## 1.1 Lifi-X

A única solução de VLC anunciada disponível no mercado é o LiFi-X, de uma empresa chamada *pureLifi*. Que afirma fornecer:

- Comunicação *full duplex* com 40Mbps de *download* e *upload*;
- Módulo receptor portátil ligado via USB;
- Múltiplos usuários por ponto de acesso;
- *Handover* suportado entre múltiplos pontos de acesso;
- Comunicação sem fio segura contida por paredes;

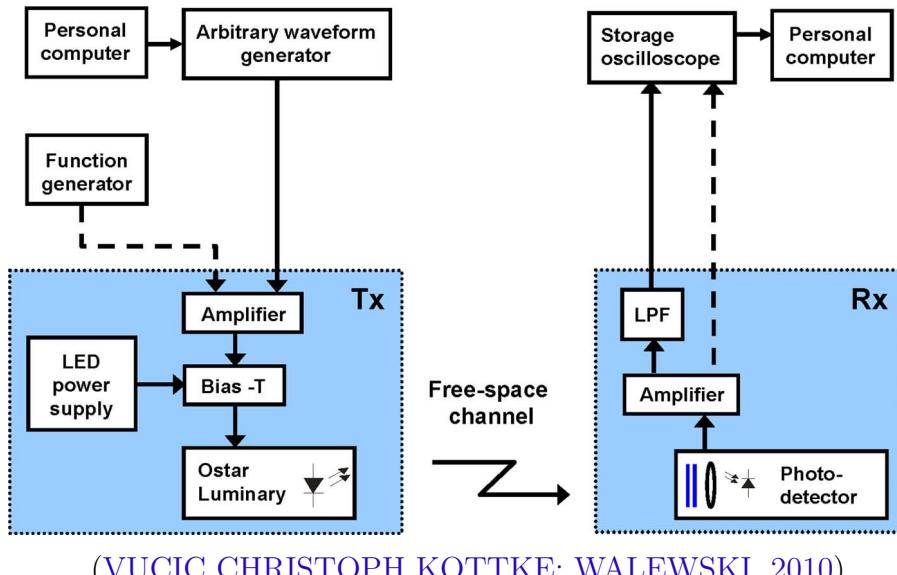
A descrição do produto não especifica a modulação utilizada, ou se existe algum código de correção de erro implementado. Sua velocidade de *download* é comparável ao 802.15.11g, que tem velocidades teóricas de 54Mbps. Entretanto, o produto está em fase de lançamento então não tem preço de venda ainda.

## 1.2 Franhoufer-Gesellschaft

Antes da publicação da norma IEEE 802.15.7 em Setembro de 2011, o Fraunhofer-Gesellschaft conseguiu criar um link de 513 Mbit/s utilizando VLC ([VUCIC CHRISTOPH KOTTKE; WALEWSKI, 2010](#)). A implementação utilizou modulação baseada em DMT (Discrete Multi-Tone) e QAM (Quadrature Amplitude Modulation), assim como técnicas de *bit* e *power loading* e *clipping* simétrico.

É utilizado um LED de alta potência para iluminar o receptor com 1000 lx e transmitir os dados modulados. Para possibilitar isso, o circuito analógico de transmissão utiliza um Bias-T, que adiciona uma componente de corrente contínua (DC) a uma onda com a mensagem codificada. A componente DC mantém a iluminação alta enquanto a mensagem é enviada ao receptor, que recebe os sinais luminosos com um fotodíodo e remove a componente de iluminação utilizando um filtro passa-baixas, facilitando o processo de recepção e decodificação. A [Figura 2](#) ilustra a arquitetura do sistema desenvolvido.

Figura 2 – Arquitetura do sistema VLC do Instituto Fraunhofer



O *overhead* de transmissão e recepção é removido com o pré e pós-processamento de dados: para transmitir é utilizado um gerador de funções programado com o *payload* e para receber é utilizado um osciloscópio com memória conectado a um computador, onde é realizada uma análise *offline* dos dados.

O artigo foca no estudo da eficácia das modulações citadas, de forma a obter uma relação distância de transmissão e vazão de dados boa. Ao fim do estudo, foram obtidas vazões de 513 Mbit/s a 30 centímetros de distância.

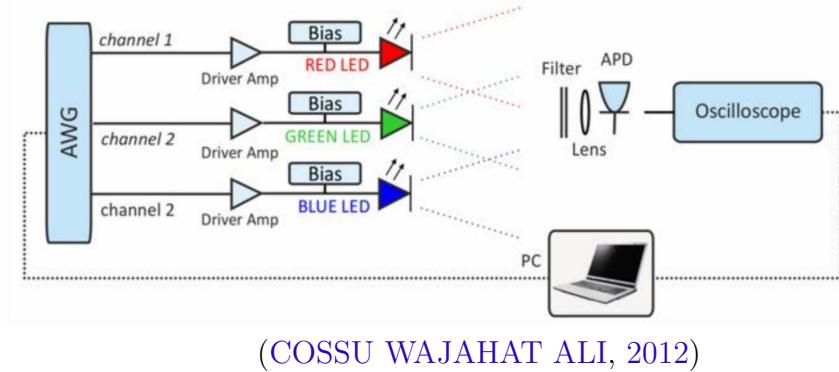
### 1.3 Scuola Superiore Sant'Anna

Em 2012, a Scuola Superiore Sant'Anna na Itália provou possível a transmissão de 3.4 Gbit/s com distâncias abaixo de 30cm ([COSSU WAJAHAT ALI, 2012](#)). O artigo se focou na transmissão de luz com modulação WDM (Wavelength Division Multiplexing) utilizando um LED Vermelho Verde e Azul (RGB). Cada comprimento de onda foi então modulado utilizando DMT, que divide o sinal em 512 portadoras, combinadas utilizando QAM.

Como no artigo anterior ([seção 1.2](#)), os autores da Scuola Superiore Sant'Anna apenas consideraram a camada física, previamente gravando a mensagem em um gerador de funções para diminuição do *overhead*.

As distâncias obtidas foram de 30cm, com níveis de iluminação de 410 lx no receptor. Como os níveis estavam abaixo do recomendado pela norma Européia para iluminação - que é 500 lx -, foi sugerido que em trabalhos futuros fossem adicionados mais componentes

Figura 3 – Configuração experimental da transmissão WDM. AWG: Gerador de Funções; APD: Fotodiodo de Avalanche



LED do lado do transmissor.

## 1.4 Fudan University

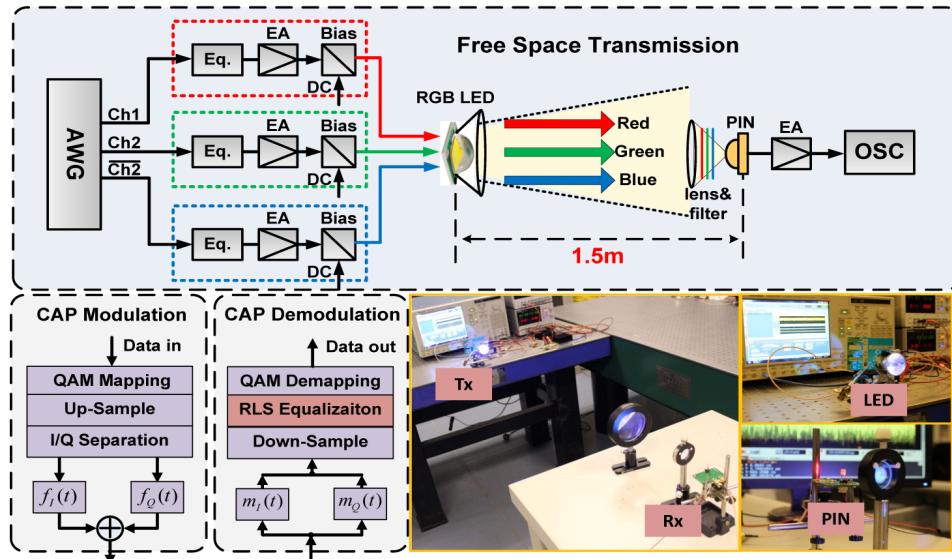
Três anos depois, na China, a Universidade de Fudan publicou um artigo sobre comunicação via luz a uma taxa de 4.5 Gbit/s (WANG XINGXING HUANG, 2015). O artigo se focou em resolver o problema da baixa distância de transmissão em sistemas VLC com velocidades na ordem de grandeza de *Gigabits*/s. Os LEDs utilizados estão disponíveis comercialmente.

Em sistemas de VLC de alta velocidade, a interferência inter-símbolo (ISI) é induzida pela dispersão multi-caminho ótica, degradando a performance do sistema e reduzindo a distância de transmissão. Para reduzir o ISI, os autores propõe a utilização de um equalizador adaptativo baseado no algoritmo *Recursive Least Square* (RLS), devido à suas vantagens sobre outro algoritmo equalizador chamado *Modified Cascaded Multi-Modulus Algorithm* (M-CMMA). Algumas das vantagens do RLS sobre o M-CMMA são a velocidade de convergência do algoritmo e a complexidade computacional total.

Além da etapa de equalização dos sistemas, ainda é descrita a modulação aplicada à mensagem. A sequência de bits da mensagem é mapeado para símbolos complexos em 64QAM, que são posteriormente enviados para modulação *Carrier-less amplitude and phase* (CAP). No artigo, são explicados ainda os motivos da escolha da modulação CAP sobre a *Orthogonal Frequency Divided Multiplexing* OFDM. Utilizando CAP, não é necessária a conversão elétrica ou ótica de números complexos para reais, além de dispensar a transformação de Fourier discreta, reduzindo a complexidade da computação e estrutura do sistema consideravelmente. A arquitetura do sistema e algumas fotos podem ser observadas na Figura 4.

Após as etapas de equalização e modulação, o sistema obteve até 1.5m de distância

Figura 4 – Arquitetura do sistema VLC desenvolvido pela Universidade de Fudan. Observa-se a modulação por comprimento de onda com as três cores do LED e o processamento adicional para converter os símbolos em CAP.



(WANG XINGXING HUANG, 2015)

entre o transmissor e receptor, com taxas de BER 7% abaixo de  $3.8 \times 10^{-3}$ .

## 1.5 Análise Comparativa

Na Tabela 1 se encontra uma análise comparativa das implementações pesquisadas. Foram considerados os parâmetros Velocidade de *download*, a distância de transmissão, a taxa de erros no canal, a modulação utilizada e se o trabalho utilizou algum código de correção de erro.

Tabela 1 – Análise Comparativa entre os trabalhos sobre VLC encontrados.

	LiFi-X	Franhoufer	Sant'Anna	Fudan University
Download	40Mbps	513Mbps	3.4Gbps	4.5Gbps
Distância	-	30cm	30cm	150cm
BER	-	$2 \times 10^{-3}$	$2 \times 10^{-3}$	$3.8 \times 10^{-3}$
Modulação	-	DMT + QAM	WDM + DMT + QAM	WDM + CAP + QAM
FEC	-	não	não	não

Fonte: Artigos citados nas seções 1.1, 1.2, 1.3 e 1.4.

Observa-se que nenhum dos trabalhos utilizou códigos de correção de erro, principalmente pelo foco na velocidade de transmissão.

## 2 Metodologia

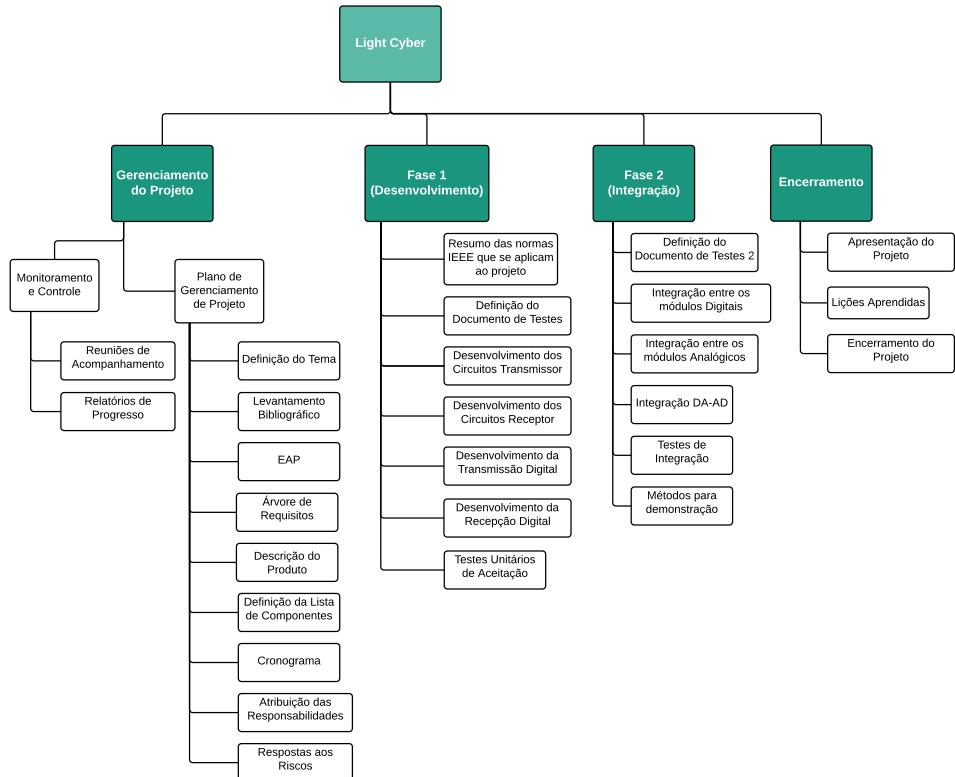
Neste capítulo serão analisadas a escolha da norma IEEE 802.15.7, bem como as especificações de *hardware* e *software*.

### 2.1 Planejamento

#### 2.1.1 Estrutura Analítica do Projeto

A Estrutura Analítica do Projeto (EAP) é uma ferramenta de gerenciamento que divide as principais entregas de um projeto. Cada entrega deve ser subdividida em tarefas até que sejam obtidos pacotes de trabalho, para que se estime recursos e tempo demandados. Neste projeto, as fases de trabalho foram definidas a partir da EAP. Observa-se que existe uma fase de gerenciamento, duas fases de desenvolvimento e uma fase de encerramento. Foi desenvolvida uma EAP para o projeto que se encontra na figura [Figura 5](#).

Figura 5 – Estrutura Analítica do Projeto Light Cyber.

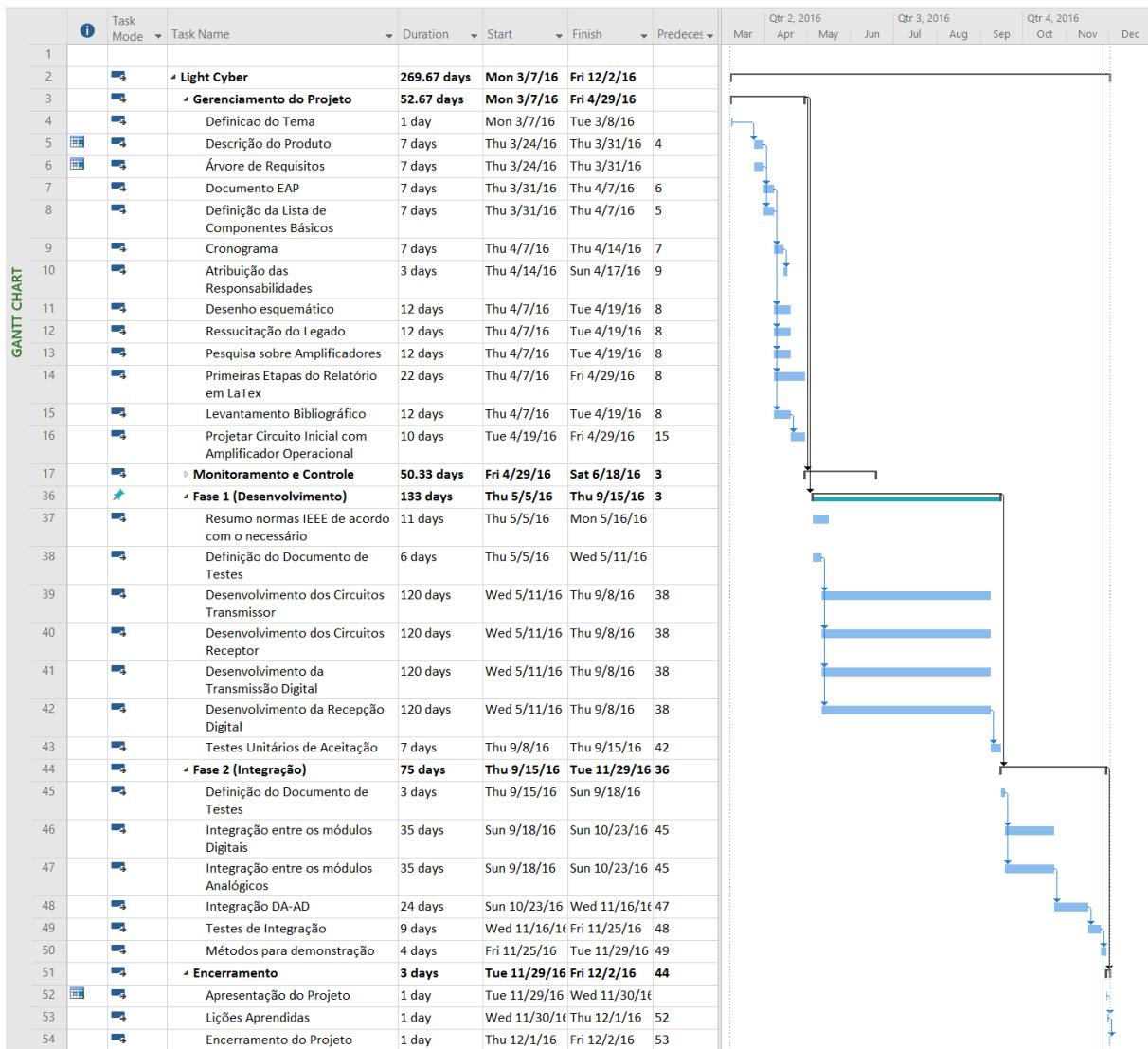


Fonte: Autores.

### 2.1.2 Cronograma

O cronograma do projeto foi planejado a partir dos pacotes de trabalho definidos na Estrutura Analítica de Projeto. Os participantes fizeram reuniões para estimar o tempo de cada um desses pacotes, com a aplicação das estimativas a um calendário, que também considerou as datas de entrega oficiais da disciplina. Com todas as estimativas feitas, a primeira e segunda fases do projeto durariam até o fim de Novembro de 2016, como pode-se observar no diagrama de Gantt na [Figura 6](#).

Figura 6 – Diagrama de Gantt do projeto LiCy.



Fonte: Autores.

### 2.1.3 Requisitos

Os requisitos foram divididos primeiramente em funcionais e não-funcionais. Além disso, existem outras divisões relacionadas a hardware, estética e módulos do sistema.

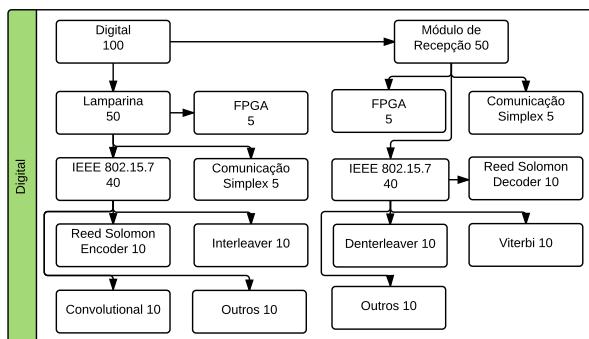
Para cada requisito foi dado um peso, conforme avaliação da relevância deste requisito para o projeto, de modo que no topo da árvore haja uma soma equivalente a 100.

## Requisitos Funcionais

### Requisitos de hardware digital

É importante observar que estes dividem-se entre a lamparina (luminária), que serve de ponto de acesso, e o módulo de recepção, com pesos iguais entre os dois, já que a comunicação é de um para um e ambos têm a mesma importância. Há requisitos em comum entre esses subsistemas, e alguns serão explicados mais à frente. A organização desses requisitos encontra-se abaixo, na [Figura 7](#).

Figura 7 – Requisitos Funcionais de Hardware Digital.



Fonte: Autores.

- Estar de acordo com a norma IEEE 802.15.7 - que tem o peso mais alto, já que o escopo principal deste trabalho comprehende implementá-la;
  - Codificador de Reed Solomon. Determinado pela norma, para garantir integridade através da correção de erros;
  - Entrelaçamento e desentrelaçamento. Métodos que evitam erros em sequências e garantem certo grau de confidencialidade;
  - Codificação convolucional e decodificador de Viterbi. Ajudam avaliar a chance da mensagem recebida conter erros, aproximando-a da enviada;
  - Outros requisitos. Dizem respeito a outra parte da codificação definida pela norma, além do pipeline do fluxo de dados e controle, bem como circuitos de sincronismo;
- Utilização da FPGA para implementar a camada física. As justificativas para o uso desse componente será apresentada nos próximos itens;

- Comunicação simplex um a um, que visa garantir a unilateralidade da comunicação entre dois nós, com recepção e transmissão exclusivas a cada uma das partes.

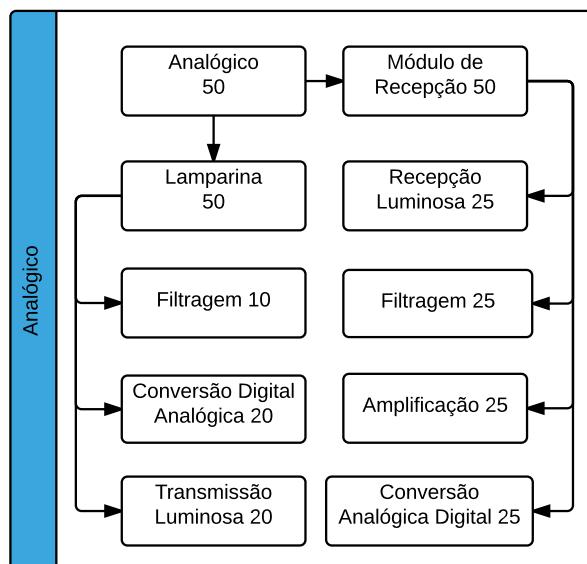
A lamparina possui requisitos de hardware próprios, incluindo a habilidade de funcionar como *Access Point*, ou seja, fornecer acesso a uma fonte de dados, que pode ser uma rede local, um computador, a internet, entre outros.

Já o módulo de recepção tem como requisito de hardware exclusivo a interface com um terminal para se observar os dados recebidos, que é essencialmente um computador ou dispositivo móvel.

### Requisitos de hardware analógico

Dividem-se entre a lamparina e o módulo receptor:

Figura 8 – Requisitos Funcionais de Hardware Analógico.



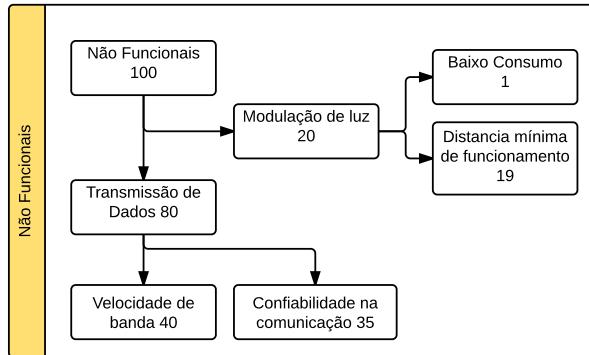
Fonte: Autores.

- Transmissão e recepção luminosa. Utilizar LED e fotodiodo, para transmitir e receber, respectivamente, os dados modulados através da luz. Adeuar a frequência de operação à norma;
- Filtragem de sinais. Filtrar a luz ambiente, que é primordial para que a recepção seja bem sucedida, do contrário pode haver muita interferência da luz que não tenha o LED como fonte. Deve fazer com que o sistema se aproxime ao máximo de um cenário ideal, onde há apenas um LED e um fotodiodo num ambiente escuro.
- Conversão Digital-Analógica / Analógica-Digital.

## Requisitos Não-Funcionais

Foram especificados através da seguinte estrutura:

Figura 9 – Requisitos Não Funcionais.



Fonte: Autores.

## Transmissão de dados

Uma das principais preocupações é manter a velocidade da banda conforme a especificação da norma IEEE 802.15.7 para cada camada física (PHY) correspondente. É também primordial garantir a confiabilidade na comunicação, de modo que se possa garantir que os *bits* recebidos estão corretos e quando possível e necessário, corrigí-los, observando as limitações que serão discutidas posteriormente no que diz respeito à norma (ver módulos de codificação e decodificação).

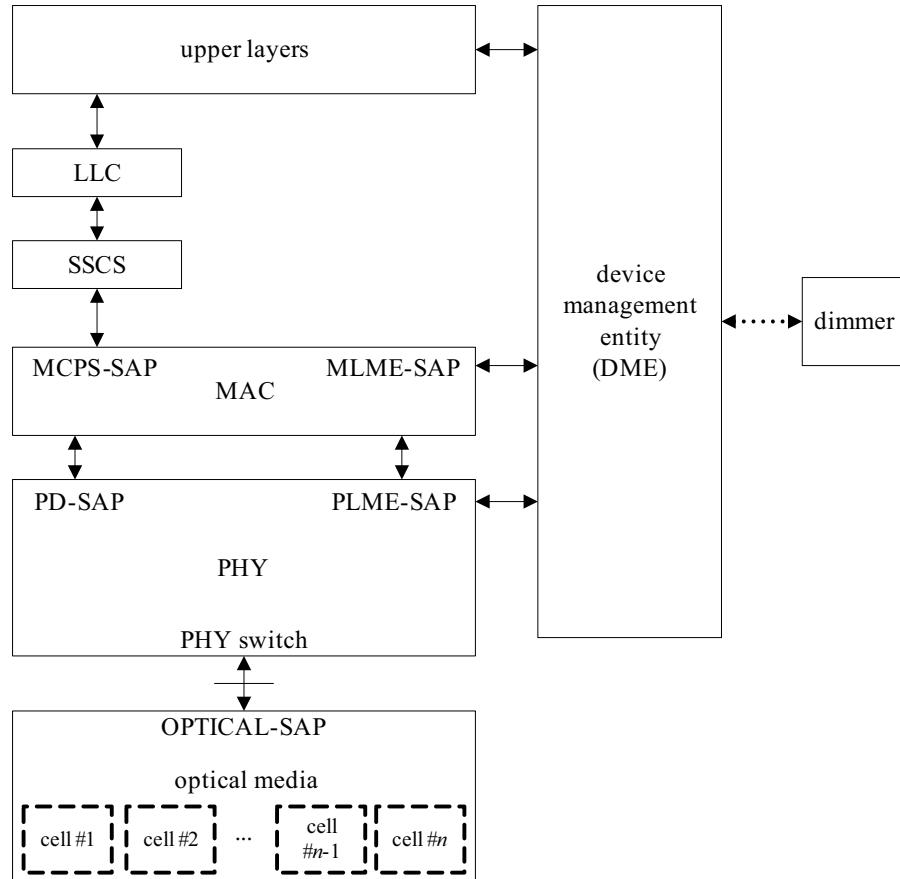
## Modulação de luz

O requisito mais importante é garantir o funcionamento do sistema a uma distância de até 1 metro, considerando uma aplicação residencial do produto. Secundariamente, há o requisito de manter um baixo consumo de energia, que apesar de fundamental em produtos de tecnologia da informação, foge do escopo deste estudo.

## 2.2 Norma IEEE 802.15.7

A norma 802.15.7 estabelece um padrão para comunicação via luz, abarcando definição de topologias de rede, codificação da informação, divisão das camadas da arquitetura do sistema de comunicação, ordem e conteúdo de cabeçalhos para cada camada e definições sobre a segurança do *link*. A arquitetura de um sistema LiFi completo está definida na Figura 10. O trabalho focará no estudo e implementação da camada PHY. Os modos de operação da subcamada PHY I estão listado na Tabela 2.

Figura 10 – Arquitetura de dispositivos VPAN.



Fonte: IEEE 802.15.7

Tabela 2 – Modos de operação da camada PHY I de Li-Fi.

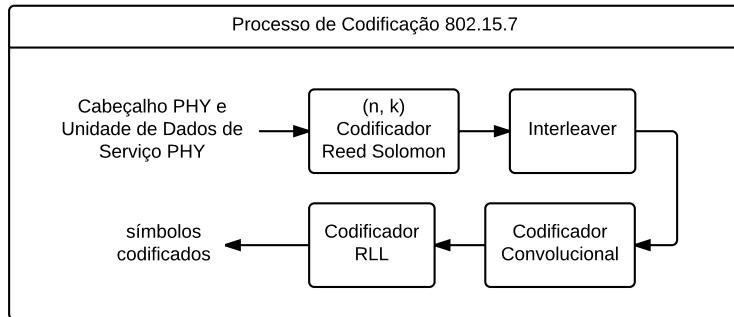
Modulation	RLL code	Optical clock rate	FEC		Data rate
			Outer code (RS)	Inner code (CC)	
OOK	Manchester	200 kHz	(15,7)	1/4	11.67 kb/s
			(15,11)	1/3	24.44 kb/s
			(15,11)	2/3	48.89 kb/s
			(15,11)	none	73.3 kb/s
			none	none	100 kb/s
VPPM	4B6B	400 kHz	(15,2)	none	35.56 kb/s
			(15,4)	none	71.11 kb/s
			(15,7)	none	124.4 kb/s
			none	none	266.6 kb/s

Fonte: IEEE 802.15.7

### 2.2.1 Transmissão

Nessa seção será discutido o processo de transmissão de dados via luz, levando em conta a norma.

Figura 11 – Diagrama de blocos da codificação da mensagem.



Fonte: Autores.

Para que os dados transmitidos sejam recebidos corretamente, a norma estabelece algumas regras para codificação da mensagem, a fim de garantir recuperação de erros, redução de erros em *burst*, facilidade na sincronia do *clock*, entre outros detalhes. A Figura 11 ilustra os passos necessários para compor a mensagem antes de sua transmissão. Esses passos estarão detalhados nas seções a seguir.

#### Codificação Reed Solomon

Com o intuito de adicionar redundância de informação na mensagem, a norma estabelece um mecanismo de correção de erros antecipada (FEC). Mais especificamente, o código de Reed Solomon, apresentado originalmente em (SOLOMON, 1960) e também discutido em (CLARKE, 2002).

Os códigos Reed Solomon são compostos por símbolos de mais de um *bit*. O número de erros é considerado como sendo o total de símbolos com pelo menos um *bit* com erro. Isso o torna efetivo para corrigir erros em rajadas, já que uma sequência de *bits* errados dentro de um mesmo símbolo é considerada como apenas um erro.

Os principais parâmetros de um código Reed Solomon são  $(n, k)$ , onde  $n$  é o número de símbolos de um bloco de código e  $k$  é o tamanho da mensagem em número de símbolos. Portanto, o bloco terá  $n - k$  símbolos de paridade, usualmente designados como  $2t$ , sendo que a capacidade de correção é de até  $t$  símbolos.

#### Corpos de Galois

A codificação e decodificação de Reed Solomon usam em seus algoritmos operações aritméticas de multiplicação e adição de corpos finitos. A norma IEEE 802.15.7 define

Tabela 3 – Corpos de Galois (16)

Elemento	Representação binária
$\alpha^0$	0001
$\alpha^1$	0010
$\alpha^2$	0100
$\alpha^3$	1000
$\alpha^4$	0011
$\alpha^5$	0110
$\alpha^6$	1100
$\alpha^7$	1011
$\alpha^8$	0101
$\alpha^9$	1010
$\alpha^{10}$	0111
$\alpha^{11}$	1110
$\alpha^{12}$	1111
$\alpha^{13}$	1101
$\alpha^{14}$	1001
$\alpha^{15}$	0000

que para a camada PHY 1, usa-se Galois Fields 16, ou na notação corrente na literatura: GF(16). Um corpo de Galois é um conjunto de  $2m - 1$  elementos, cada qual representado por um polinômio de grau  $m - 1$ , com coeficientes binários. Em suma, um elemento de GF(16) pode ser representado a partir de símbolos de 4 *bits*, totalizando 16 elementos de 0000 a 1111.

A adição e subtração de dois corpos de Galois é dada pela soma ou subtração dos coeficientes de dois polinômios. Além disso as operações são feitas *bit a bit*, sem que haja *bits* de “vai um” para coeficientes subsequentes. Caso os dois coeficientes tenham o mesmo valor, que pode ser 1 ou 0, o resultado da sua soma ou subtração deve ser 0. Caso contrário, a soma equivale a 1, de maneira que essa operação é perfeitamente representável pela função lógica de OU exclusivo, tanto para adição como subtração.

Os corpos de Galois são definidos com base no gerador polinomial de corpos, que é um polinômio irredutível, de grau  $m$ . Para GF(16) existe por exemplo, esta opção:

$$p(x) = x^4 + x + 1 \quad (2.1)$$

A norma IEEE 802.7.15 não especifica qual gerador polinomial deve ser usado, portanto neste estudo foi usada a opção do exemplo. O processo de geração neste caso resulta nos elementos finitos da Tabela 3.

A multiplicação entre dois corpos de Galois ocorre, em termos de aritmética comum, pela multiplicação de dois polinômios, seguida da divisão pelo gerador polinomial. Já a divisão de corpos finitos pode ser feita através da operação de inversão do divisor e multiplicação desse resultado pelo dividendo, seguindo os passos indicados anteriormente

para a multiplicação.

### Codificação

Para gerar um código de Reed Solomon a partir dos corpos finitos, é necessário um polinômio gerador de código. Este polinômio possui  $2t$  fatores, de maneira que o grau do polinômio representado por  $C(x)$  (bloco codificado, com grau  $n$ ) seja a soma do grau de  $M(x)$  (a própria mensagem dentro do bloco codificado, com grau  $k$ ) e  $G(x)$  (polinômio codificador, com grau  $2t = n - k$ ). O polinômio gerador de código deve ter a seguinte forma:

$$g(x) = (x + \alpha^i)(x + \alpha^{i+1})\dots(x + \alpha^{i+2t-1}) \quad (2.2)$$

As raízes são escolhidas como corpos finitos de Galois consecutivos. Para esse estudo, usaram-se como raízes os elementos consecutivos de  $\alpha^0$  a  $\alpha^7$ . Por exemplo, para um código (15, 7), usado posteriormente neste projeto:

$$g(x) = (x + \alpha^0)(x + \alpha^1)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)(x + \alpha^5)(x + \alpha^6)(x + \alpha^7) \quad (2.3)$$

$$g(x) = \alpha^{13}x^7 + \alpha^0x^6 + \alpha^1x^5 + \alpha^{13}x^4 + \alpha^8x^3 + \alpha^{14}x^2 + \alpha^4x + \alpha^{13} \quad (2.4)$$

Neste caso, a mensagem  $M(x)$  é codificada da seguinte maneira:

$$\frac{M(x)x^{n-k}}{G(x)} = q(x) + \frac{r(x)}{g(x)} \quad (2.5)$$

Desta forma, são produzidos um quociente  $q(x)$  e um resto  $r(x)$ . Por manipulação algébrica, temos que o bloco codificado pode ser representado da seguinte forma:

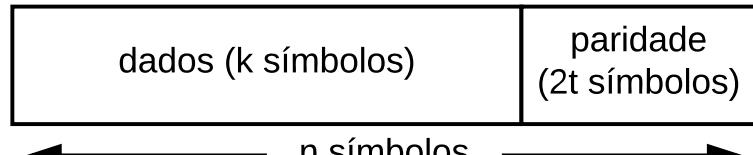
$$M(x)x^{n-k} + r(x) = g(x)q(x) \quad (2.6)$$

Portanto, o bloco codificado  $C(x)$  é composto por  $M(x)$  deslocada, somada a  $r(x)$ , na prática equivalendo à [Figura 12](#), com os símbolos de dados transmitidos em esquema de fila, seguidos dos símbolos de paridade.

Finalmente, a representação do codificador de Reed Solomon (15, 7) em hardware equivale a mostrada na [Figura 13](#).

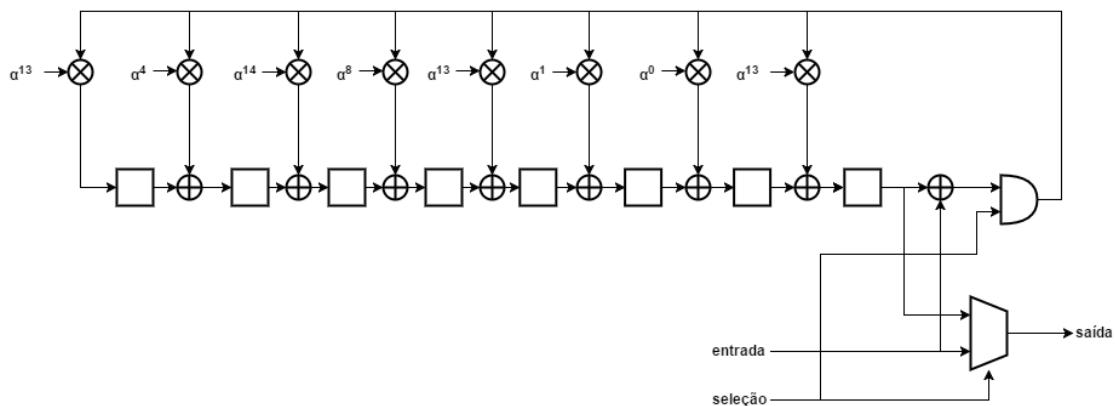
É possível observar nos registradores e multiplicadores de Galois a estrutura de um polinômio e seus índices. A mensagem a que se vai adicionar redundância deve passar

Figura 12 – Diagrama de blocos da codificação da mensagem.



Fonte: Autores.

Figura 13 – Circuito lógico de codificação Reed Solomon (15,7).



Fonte: Autores.

pela “entrada”, com um ciclo de *clock* para cada símbolo, levando portanto 7 ciclos para finalizar o cálculo das paridades para um código (15, 7). Os 8 símbolos de paridade devem então serem passados através de registradores de deslocamento, com a mudança do sinal de seleção, de maneira também a zerar a saída da porta lógica AND à direita. Sendo assim, os valores dos registradores não são afetados pelos somadores, já que as operações de multiplicação e soma de Galois estarão recebendo uma das entradas igual a 0000. Em suma, o processo de codificação leva tantos ciclos quantos forem os símbolos do bloco codificado, ou o valor denotado por *n*.

### Interleaver

O processo de entrelaçamento é utilizado para aprimorar a performance de mecanismos de correção de erros antecipados. Ele se baseia no fato de que canais de comunicação não são estocásticos, então erros ocorrem em sequência e não independentemente.

O entrelaçamento serve para distribuir os dados de forma que erros em rajadas serão distribuídos em múltiplas palavras de código, facilitando o processo de recuperação de erros.

Para melhor demonstrar a aplicação do entrelaçador, serão exemplificadas várias situações utilizando um sistema de recuperação de erros em um canal com ruídos (que

costumam ser *bit-flip*). Na primeira, não é aplicado o entrelaçamento e a mensagem é recuperável com redundância.

Burst = 2, Redundância = 3, Sem entrelaçamento	
Mensagem inicial:	abcdefg
Mensagem com redundância:	aaabbcccddeefffggg
Transmissão com erros em burst:	aaabbbc_dddeefffggg
Mensagem recuperada:	abcdefg

Mesmo após dois caracteres perdidos em sequência, foi possível recuperar as perdas do canal. Isso ocorreu porque o *burst* do erro foi menor que sua capacidade de recuperação. Na situação a seguir, novamente sem entrelaçamento, a mensagem não é recuperável com redundância.

Burst = 4, Redundância = 3, Sem entrelaçamento	
Mensagem inicial:	abcdefg
Mensagem com redundância:	aaabbcccddeefffggg
Transmissão com erros em burst:	aaabbbcc___eeefffggg
Mensagem recuperada:	abc_efg

Mesmo com a redundância aplicada, a informação *d* foi perdida devido a erros em *burst*. Aplicando entrelaçamento, a mesma mensagem pode ser recuperada facilmente, a seguir:

Burst = 4, Redundância = 3, Com entrelaçamento	
Mensagem inicial:	abcdefg
Mensagem com redundância:	aaabbcccddeefffggg
Interleaving:	abcdefgabcdefgabcdefg
Transmissão com erros em sequência:	abcdefgab___gabcdefg
Deinterleaving:	aaabbbc_cd_de_ef_fggg
Mensagem recuperada:	abcdefg

A mensagem recebida foi recuperada porque os erros foram distribuídos em partes diferentes da mensagem, que foi entrelaçada. Os dados resultantes puderam ser inferidos pela redundância aplicada.

### Puncture

O módulo de punção é utilizado para reduzir o *overhead* do *padding*, e é aplicado depois do *interleaver*. A função da punção é remover alguns bits de paridade da sequência,

e equivale a utilizar um algoritmo de correção de erros com menos redundância. Neste caso, visa os bits de *padding*, que não representam informação e são colocados apenas para completar o pacote até que o tamanho seja uma potência de dois. O efeito é uma redução na taxa do código, que será explicada nas seções subsequentes.

### Código Convolutional

O módulo de código convolucional é um código de correção de erros que converte a mensagem em símbolos de paridade. Ele se utiliza de uma janela deslizante para receber e armazenar uma parte dos dados de entrada. Alguns subgrupos de *bits* dentro da janela, definidos pelos geradores polinomiais, são utilizados para gerar os *bits* de paridade. A operação de *exclusive or* entre os *bits* de um subgrupo gera um *bit* de paridade. A norma define os parâmetros do código na [Tabela 4](#) de acordo com a camada PHY I.

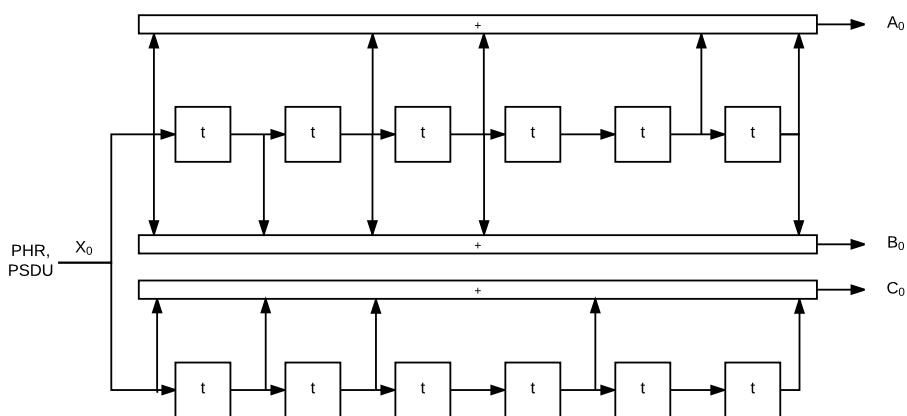
Tabela 4 – Parâmetros do Código Convolucional da camada PHY I.

Parâmetro	Valor	Descrição
K	7	comprimento da janela deslizante
R	1/4	taxa de <i>bits</i> / paridade
G	$g_0 = 133_8; g_1 = 171_8; g_2 = 165_8$	polinômio gerador

Fonte: Autores.

No documento da norma há uma figura que especifica a implementação do codificador. Estudando-a em detalhes, foram encontrados erros na figura, pois os *bits* utilizados para gerar as paridades eram extraídos de geradores polinomiais errados. A [Figura 14](#) esquematiza um codificador com os polinômios corrigidos. A cada *bit*  $X_n$  de entrada, o código convolucional gera três paridades:  $A_0$ ,  $B_0$  e  $C_0$ , por isso 1/3.

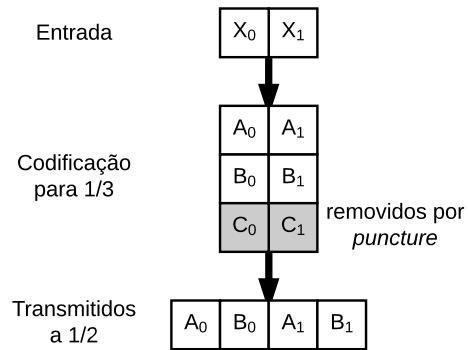
Figura 14 – Esquemático do Codificador Convolucional com taxa 1/3, os quadrados representam registradores e os retângulos os somadores.



Fonte: Autores

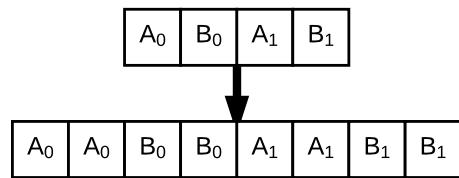
Nota-se que o codificador especificado pela norma tem taxa  $1/3$ , e não de  $1/4$ , como especificado na norma. Para obter uma taxa de  $1/4$  deve-se realizar operação de *puncture* no código de taxa  $1/3$  para um de taxa  $1/2$ , como mostrado na [Figura 15](#), e depois usando um código de repetição como na [Figura 16](#).

Figura 15 – Padrão de *puncture* para obter código a taxa  $1/2$ .



Fonte: Autores.

Figura 16 – Padrão de repetição utilizado para obter um código com taxa  $1/4$ .



Fonte: Autores.

Como a saída do codificador é serial de um *bit*, observa-se que esse código cria um *overhead* na velocidade da transmissão da mensagem. Os *bits* entram no codificador convolucional a uma taxa de  $t$  *bits/tempo* e saem a uma taxa de  $t/4$  *bits/tempo*.

### Codificação RLL

Para a primeira camada de implementação PHY I, será escolhida a codificação RLL Manchester. A codificação Manchester divide o *bit* do dado ao meio, de maneira que a primeira metade deve obrigatoriamente assumir um valor diferente da segunda metade. Em outras palavras, caso a entrada a ser codificada seja 0 por um tempo  $2t$ , a saída deve ser 01, com cada *bit* com tempo  $t$  de transmissão como se verifica na [Tabela 5](#). Analogamente, caso o *bit* de entrada seja 1, a saída deve ser 10. Uma maneira prática de se obter um codificador Manchester é através da aplicação da operação de OU exclusivo, ou XOR em dois sinais de entrada. Um destes sinais é a própria mensagem a ser codificada, o outro sinal deve ser um *clock* com o dobro da frequência da mensagem. É importante observar que para essa codificação ser eficiente, os dois sinais devem estar sincronizados, de maneira

Tabela 5 – Codificação Manchester.

<i>bit</i>	<i>manchester symbol</i>
0	01
1	10

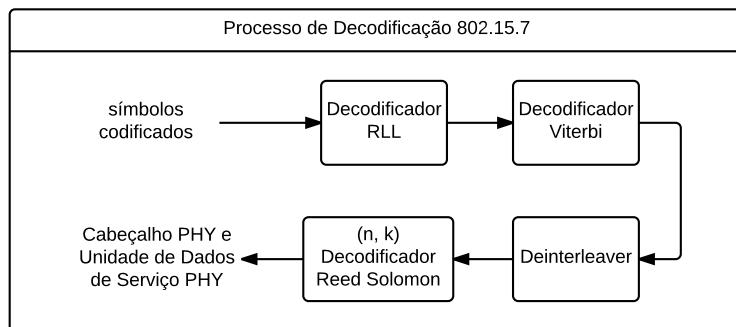
Fonte: Tabela 103 da norma 802.15.7

que um período de *clock* seja exatamente equivalente à duração do bit da mensagem de entrada.

## 2.2.2 Recepção

Após a codificação dos dados, é necessário realizar a sua decodificação. Esse processo inclui a reversão dos processos de codificação e correção de erros, caso existam e seja possível corrigir. No entanto isso não é trivial, pois a norma não especificou nenhum algoritmo para decodificar nem corrigir erros. Foi necessário realizar uma pesquisa para descobrir os métodos mais adequados para decodificação de cada etapa, que estão esquematizadas na Figura 17. Esses algoritmos estão descritos em detalhes abaixo.

Figura 17 – Diagrama de blocos da decodificação da mensagem.



Fonte: Autores.

### Decodificação RLL

A decodificação do código Manchester ocorre da mesma forma que a codificação, porém com um sinal de *clock* cuja frequência é idêntica à do *clock* de codificação. Portanto, o período do *clock* deve ser o dobro do tempo de duração do *bit* da mensagem. Uma consequência dessa definição é que os dados transmitidos entre um codificador e um decodificador Manchester têm o dobro da frequência da mensagem original, embora a frequência com que os dados efetivos são transmitidos seja a mesma.

A identificação de erros pode acontecer de diversas formas. É possível, por exemplo, criar uma máquina de estados que identifique a entrada de três *bits* idênticos consecutivos, o que sinaliza erro no caso do código de Manchester. Um dos motivos para esse erro pode

ser um código deslocado em até metade de um ciclo do *clock* da entrada do decodificador, e neste caso pode-se corrigir o erro através de uma correção simples, como um *shift register*. Caso o número de *bits* idênticos consecutivos seja superior a dois, a correção do erro se torna muito mais trabalhosa, motivo pelo qual esse tipo de correção não foi abordado neste estudo.

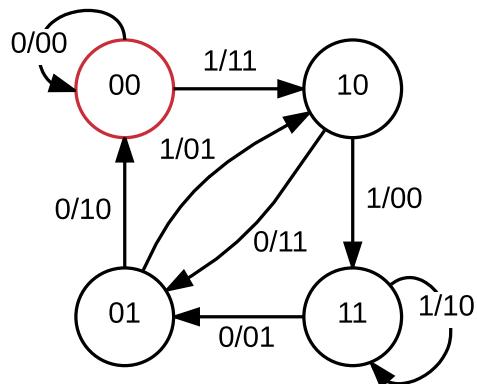
### Decodificação Viterbi

A seção a seguir explica os conceitos fundamentais para o entendimento do decodificador Viterbi: máquinas de estado, tabela Trellis, o algoritmo de decodificação *Viterbi* e sua simplificação *Fangled Viterbi*.

### Máquina de Estados

A visão de máquinas de estado é uma maneira elegante de explicar o funcionamento do transmissor (e do receptor) e é explicada a seguir. O transmissor começa no estado inicial (marcado como em vermelho na Figura 18) e processa um *bit* de cada vez. Para cada *bit* de mensagem, há a transição do estado atual para um novo dependendo do valor do *bit* de entrada, e o envio da paridade no arco correspondente. A nomenclatura tomada na figura é a idêntica à utilizada em máquinas de Mealy: X/Y, onde X representa a entrada e Y representa a saída. O receptor, é claro, não tem conhecimento direto das transições do

Figura 18 – Visão de máquina de estados de um codificador convolucional.



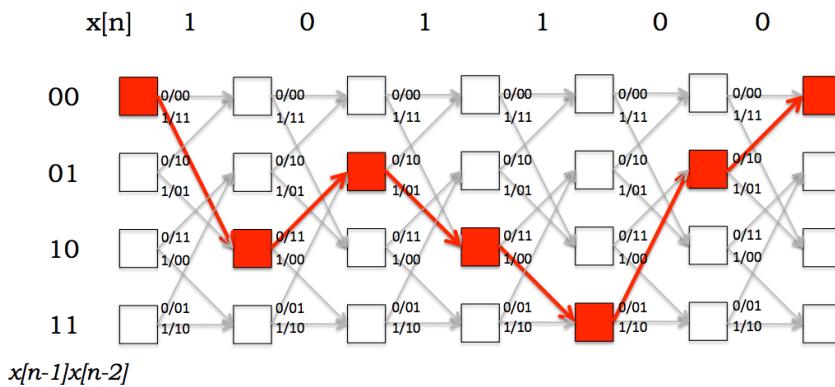
Fonte: Autores.

transmissor. Ele só pode ver a sequência de paridades recebidas, possivelmente corrompidas. Sua tarefa é determinar a **melhor** sequência possível de estados transmitidos que podem ter gerado a sequência de paridades. Essa tarefa é chamada de decodificação, que será detalhada a seguir.

### Tabela Trellis

Como mencionado acima, o receptor deve determinar a **melhor** sequência de estados que o transmissor passou. Essa sequência forma um caminho dentro da máquina de estados, que pode ser deduzido pelo decodificador. Trellis é a estrutura derivada da máquina de estados, ela ajuda o decodificador a escolher os melhores caminhos para recuperar a mensagem. Um exemplo de tabela Trellis é dado na [Figura 19](#). Nessa figura,

Figura 19 – A tabela Trellis é útil para entender o processo de decodificação de uma máquina de estados.



Fonte: ([MIT, 2010](#))

cada coluna tem um grupo de estados; cada estado é conectado a dois estados na próxima coluna; cada seta indica qual saída e próximo estado para cada entrada. A figura mostra o caminho percorrido pelo transmissor com a mensagem 101100.

O decodificador deve identificar o melhor caminho utilizando a tabela Trellis, explicando por meio dela a sequência de paridades observadas e possivelmente corrompidas. Esse decodificador é chamado de Viterbi.

### Algoritmo

O algoritmo de decodificação Viterbi é composto por três componentes principais: a métrica de ramificação (BM), a métrica de caminho (PM) e o *Traceback Unit*.

A métrica de ramificação calcula as distâncias entre a entrada e todas as possíveis entradas. Em alguns casos ela também é conhecida como distância de Hamming.

A métrica de caminho calcula todos caminhos possíveis na tabela Trellis. Esse módulo é composto por múltiplos submódulos chamados ACS (Add, Compare and Select), que representam um estado e realizam a decisão de qual será o próximo estado. Após realizar a decisão, o ACS grava o caminho até o próximo estado e a distância acumulada na memória do *Traceback Unit*. Esse processo se repete à medida que cada *bit* da mensagem entra no Viterbi e apenas acaba quando a mensagem termina.

Ao fim da mensagem, o *Traceback Unit* escolhe o caminho com a menor distância acumulada e o percorre de forma invertida.

### *Fangled Viterbi*

Pode-se observar que esse método necessita de muitos recursos e computação, pois grava e compara todos os caminhos possíveis durante sua execução. O algoritmo pode ser simplificado removendo o módulo de *Traceback Unit* e utilizando apenas uma unidade modificada de ACS. Essa unidade escolhe a menor distância de acordo com o estado atual e percorre o caminho, removendo a necessidade de gravar as possibilidades em memória. Esse método é chamado de *Fangled Viterbi*.

### *Deinterleaver*

O processo de *deinterleaving* é análogo ao processo de *interleaving* e é o único que não necessita de atenção especial para um estudo de um novo algoritmo.

### Decodificação *Reed Solomon*

Esta seção discorre sobre as quatro etapas principais do processo de decodificação de Reed Solomon abordados na execução do projeto. Alguns dos algoritmos são discutidos com mais detalhes em ([SHIH, 1992](#)).

### Cálculo das síndromes

As síndromes são calculadas através da mensagem recebida e dos símbolos de paridade. O objetivo desse cálculo é identificar, num primeiro momento, a presença de erros no bloco que foi recebido em relação ao que foi transmitido. Utilizaremos aqui a notação de  $R(x)$  para o bloco recebido e  $E(x)$  para os possíveis erros. Portanto:

$$R(x) = C(x) + E(x) \quad (2.7)$$

Caso a mensagem recebida não tenha erros, teremos  $E(x)$  formado por 15 símbolos de 0000 no caso de Reed Solomon (15, 7). Se pelo menos um símbolo de  $E(x)$  possuir pelo menos um bit diferente de 0, configura-se um erro, e o cálculo das síndromes deverá apontá-lo. Caso o número de símbolos com erro exceda  $t$ , ou 4 símbolos para um código (15, 7), o erro será identificado nas síndromes, porém não será possível corrigi-lo posteriormente.

No caso de Reed Solomon, as síndromes podem se definir da seguinte maneira:

$$S_i = Q_i(x)(x + \alpha^i) + R(x) \quad (2.8)$$

No caso específico em que  $x$  é  $\alpha^i$ , temos que o primeiro termo da soma é nulo, já que  $x + \alpha^i = 0000$ . Portanto:

$$S_i = R(\alpha^i) \quad (2.9)$$

E haverá um total de  $2t$  síndromes, uma para cada raiz do polinômio codificador  $G(x)$ . Considerando que existem  $v$  símbolos com erros, com  $v \leq t$ , teremos:

$$E(x) = V_1 x^{L_1} + \dots + V_v x^{L_v} \quad (2.10)$$

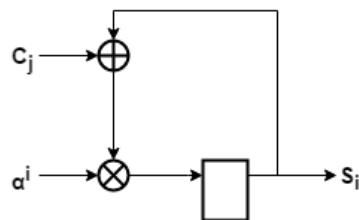
Com  $V_i$  representando os valores do erro, e  $L_i$  indicando a localização desses erros. Portanto, para cada síndrome  $S_i$ :

$$S_i(\alpha^i) = V_1 X_1^i + V_2 X_2^i + \dots + V_v X_v^i \quad (2.11)$$

Os valores de  $X_i$  são usualmente designados para representar os localizadores de erro. Observe que os índices numéricos não necessariamente indicam o coeficiente do polinômio.

Uma possível representação para o cálculo da síndrome, em hardware, seria a Figura 20. Observe que cada símbolo  $C_j$  do código é uma das entradas para cada ciclo do cálculo. O parâmetro  $\alpha^i$  é fixo e seu índice é o mesmo da síndrome de saída. Portanto, para um código Reed Solomon (15, 7), existem 15 símbolos de entrada e 8 síndromes - uma para cada raiz do polinômio codificador.

Figura 20 – Módulo genérico de cálculo de uma das síndromes



Fonte: Autores.

### Localização dos erros

O polinômio localizador de erros é construído com auxílio dos localizadores de erro, denotados por  $X_1$  a  $X_v$ .

$$\Lambda(x) = (1 + xX_1)(1 + xX_2)\dots(1 + xX_v) \quad (2.12)$$

$$\Lambda(x) = 1 + \Lambda_1 x^1 + \dots + \Lambda_{v-1} x^{v-1} + \Lambda_v x^v \quad (2.13)$$

Dessa forma teremos os inversos de  $X_1$  a  $X_v$  como raízes deste polinômio, sendo que cada uma dessas raízes torna  $\Lambda(x)$  igual a zero.

$$\Lambda(X_j^{-1}) = \Lambda_v X_j^{-v} + \dots + \Lambda_1 X_j^{-1} + 1 = 0 \quad (2.14)$$

Multiplicando essa igualdade por  $V_j * X_j^{i+v}$ , temos:

$$Y_j X_j^{i+v} + \Lambda_1 Y_j X_j^{i+v-1} + \dots + \Lambda_v Y_j X_j^i \quad (2.15)$$

Para diferentes valores de  $j$ , temos:

$$S_{i+v} + \Lambda_1 S_{i+v-1} + \dots + \Lambda_v S_i = 0 \quad (2.16)$$

Dessa forma, pode-se construir a seguinte matriz com um sistema de equações:

$$\begin{pmatrix} S_{v-1} & S_{v-2} & \dots & S_0 \\ S_{v-1} & S_{v-2} & \dots & S_0 \\ \dots & \dots & \dots & \dots \\ S_{2v-2} & S_{2v-3} & \dots & S_{v-1} \end{pmatrix} \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \\ \dots \\ \Lambda_v \end{pmatrix} = \begin{pmatrix} S_v \\ S_{v+1} \\ \dots \\ S_{2v-1} \end{pmatrix} \quad (2.17)$$

### Algoritmo de Berlekamp-Massey

Resolvendo o sistema anterior, encontram-se os coeficientes do polinômio localizador de erros. Essa resolução pode se feita de várias formas, porém neste estudo foi selecionado o método de Berlekamp-Massey. Este algoritmo resolve o sistema por meio de aproximações sucessivas do polinômio localizador, iniciando com um  $\Lambda(x)$  capaz de produzir  $S_v$ , ou a primeira síndrome. Conforme a seção anterior, calcula-se o próximo termo,  $S_{v+1}$ , considerando:

$$S_{v+1} = S_v \Lambda_1 + S_{v-1} \Lambda_2 + \dots + S_1 \Lambda_v \quad (2.18)$$

Os valores de  $\Lambda$  fazem parte da primeira aproximação. Caso esse valor da síndrome calculada pelo polinômio seja o mesmo valor da síndrome real, ou a diferença  $d = 0000$ , calcula-se  $S_{v+2}$ . Caso contrário, o polinômio deve ser refinado:

$$\Lambda'(x) = \Lambda(x) + Kx^z \Lambda''(x) \quad (2.19)$$

$\Lambda''(x)$  é o último polinômio localizador estimado antes de haver uma diferença  $d \neq 0000$ .  $K$  é definida como  $d''^{-1}d$ , com  $d''^{-1}$  sendo o inverso da diferença de quando o polinômio  $\Lambda(x)$  foi modificado da última vez. Já  $z$  é o número de estágios entre  $\Lambda''+1(x)$  e  $\Lambda(x)$ .

Uma vez calculados os coeficientes do polinômio localizador, deve-se descobrir o valor de suas raízes. Como apontado anteriormente, o inverso destas aponta a localização dos erros. Esses valores são encontrados através do método de busca de Chien, que utiliza tentativa e erro, com todos os possíveis valores das raízes.

### Valores dos erros

Existem diversas maneiras de encontrar os valores dos erros. O método abordado aqui - o algoritmo de Forney - utiliza o polinômio avaliador de erros, definido como:

$$\Omega(x) = \Omega_{v-1}x^{v-1} + \dots + \Omega_1x + \Omega_0 \quad (2.20)$$

Os valores dos erros são dados por sua vez por:

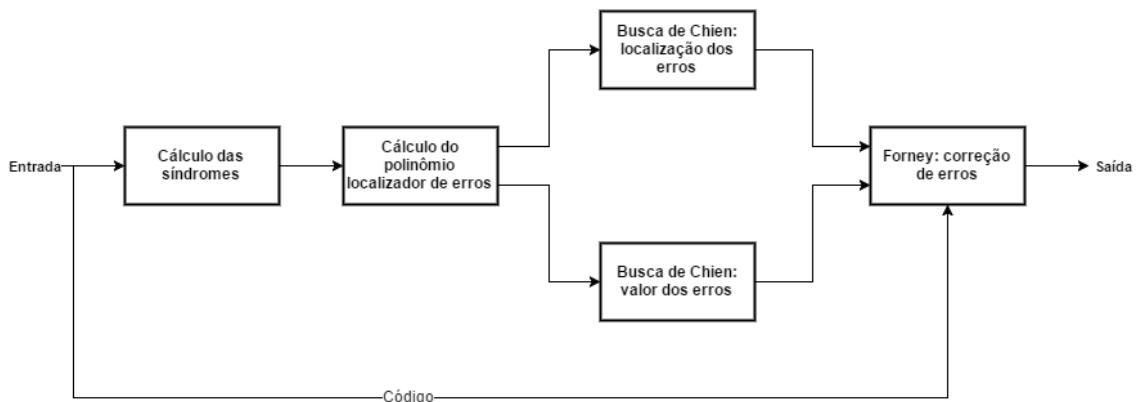
$$V_j = X_j^{1-b} \frac{\Omega(X_j^{-1})}{\Lambda'(X_j^{-1})} \quad (2.21)$$

$\Lambda'(x)$  é a derivada de  $\Lambda(x)$ , e  $b$  é o índice do primeiro  $\alpha$  usado no polinômio codificador, que neste caso será 0.

### Síntese

Um decodificador de Reed Solomon pode ser dividido funcionalmente nos seguintes módulos:

Figura 21 – Diagrama funcional de decodificação Reed Solomon.



Fonte: Autores.

Em suma, a entrada compreende os símbolos de mensagem e de paridade, que são enviados sequencialmente para o cálculo das síndromes. Uma vez calculadas as síndromes, seus valores são usados para se obter os polinômios localizador e avaliador de erros, cujos coeficientes determinam quando se realiza uma correção e qual o valor da correção que, somado à mensagem recebida, corrige-a.

### Estratégia

O primeiro passo para aproximar os algoritmos de uma solução em hardware foi procurar níveis de abstração intermediários. O nível encontrado foi a programação em alto nível, reduzindo módulos como multiplicadores e somadores a funções. Uma das vantagens deste método foi a geração de saídas esperadas para cada módulo funcional, incluindo o bloco de saída final, corrigido ou a ser descartado. Esses dados foram fundamentais para a depuração e correções em hardware. Entretanto, uma das limitações deste método era a de que a linguagem de programação pertence a um paradigma diferente do da descrição de hardware, o que envolveu o uso de máquinas de estado no lugar de loops, por exemplo.

#### 2.2.3 Estrutura da mensagem

Como neste trabalho apenas será implementado a camada física PHY I, a estrutura da mensagem não conterá elementos da camada MAC. Define-se portanto o elemento básico de transmissão, a Unidade de dados da camada física, ou PSDU, de acordo com a norma. Seu formato pode ser visto na [Figura 22](#).

Figura 22 – Estrutura da mensagem, composta pelo Cabeçalho de Sincronização (SHR), Cabeçalho da Camada Física (PHR) e Unidade de Dados de Serviço PHY (PSDU).

Preâmbulo	Cabeçalho PHY	HCS	Campos Opcionais	Payload PHY
SHR	PHR			PSDU

Fonte: Autores.

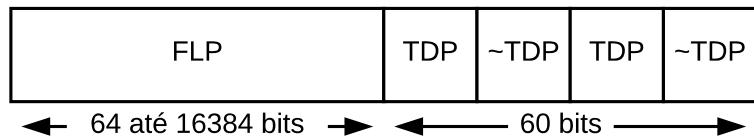
#### Preâmbulo - SHR

O campo de preâmbulo é usado pelo transmissor de data para obter sincronização ótica de *clock* com uma mensagem recebida. A norma define um padrão de travamento rápido (FLP), seguido de quatro padrões dependentes da topologia (TDP) para fins de distinção de topologias de PHY diferentes. O preâmbulo deve ser enviado em uma taxa de *clock* escolhida pelo transmissor e suportada pelo receptor. Ele é uma sequência de domínio de tempo e não possui nenhuma codificação de canal ou linha.

O preâmbulo começa com um FLP com pelo menos 64 zeros e uns alternados. O FLP é fixado para iniciar com um padrão "**1010...**" i.e., termina com um '0'. A sequência de transição máxima é usada para travar o circuito de recuperação de *clock* e dados (CDR).

O comprimento do padrão de travamento rápido não pode exceder o máximo de 16384 bits. Após o FLP, quatro repetições de TDP devem ser enviados. O TDP deve conter 15 bits de comprimento e deve ser invertido a cada repetição para providenciar *DC balance*. O diagrama da estrutura do SHR está na [Figura 23](#).

Figura 23 – Estrutura do SHR - Cabeçalho da Camada Física.



Fonte: Autores.

O preâmbulo deve ser enviado utilizando modulação OOK (*on-off keying*). O campo deve ser formatado com um TDP fixo e a lista de topologias está definida na [Tabela 6](#).

Tabela 6 – Definição de Topologias para o TDP.

TDP	Topologia	Valor
P1	Independente	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
P2	Peer-to-peer	0 0 1 0 1 1 1 0 1 1 1 1 1 1 0
P3	Estrela	1 0 0 1 1 0 0 0 0 0 1 0 0 1 1
P4	Broadcast	0 1 0 0 0 0 1 1 0 1 0 0 1 0 1

Fonte: Figura 120 da norma 802.15.7

### Sincronização de *Clocks*

Se refere ao ato de sincronizar os *clocks* de sistemas independentes. Para a transmissão de dados via luz, como ambos os módulos (receptor e transmissor) não estão conectados via cabo, o sinal de *clock* não é compartilhado. Dessa forma, é certo dizer que eles estão fora de fase. Para isso a norma especifica o FLP, um sinal com período fixo utilizado pelo receptor para sincronizar seu *clock* com o do transmissor.

### Cabeçalho PHY - PHR

O cabeçalho PHY, como mostrado na [Tabela 7](#), deve ser transmitido utilizando modulação OOK. Ele deve ser enviado na menor velocidade listada da camada, que no caso de PHY I é 11.67kb/s e fica no cabeçalho como 000000.

A maioria dos campos não será usada por motivos de redução de escopo, portanto o receptor e transmissor não terão habilitado o modo *Burst*, canais e dimerização OOK. O

Tabela 7 – Definição dos campos do cabeçalho PHY.

Campo do cabeçalho	Comprimento em Bits
Modo de Burst	1
Número do Canal	3
MCS ID	6
Comprimento PSDU	16
Extensão de dimerização OOK	1
Campos Reservados	5

Fonte: Tabela 81 da norma 802.15.7

único campo variável do cabeçalho será o Comprimento PSDU, que dependerá do conteúdo da mensagem enviada.

#### Unidade de Dados de Serviço PHY - PSDU

Como o escopo do trabalho não engloba a camada MAC, o campo de PSDU conterá apenas a mensagem. A estrutura adaptada sem os cabeçalhos MAC está na [Figura 24](#). A norma ainda especifica *tail bits* para reiniciar o processo do código convolucional.

Figura 24 – Estrutura do PSDU - Mensagem e *tail bits*.

Fonte: Autores.

## 2.3 Hardware

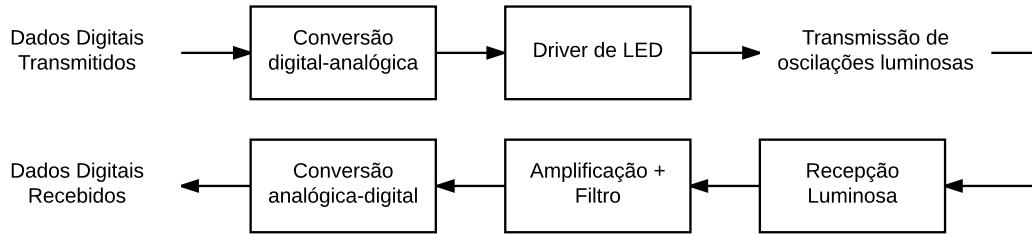
A seguir serão detalhadas as implicações da implementação da norma IEEE 802.15.7 em um escopo de hardware analógico, a partir da necessidade de transmitir, receber e converter sinais digitais em analógicos, que então serão convertidos de e para oscilações luminosas. A [Figura 25](#) representa o fluxo dos dados:

Os dois primeiros módulos são relativos ao circuito de recepção, enquanto os três últimos são relativos ao transmissor.

### 2.3.1 Transmissão

O transmissor do sistema de comunicação LiCy será composto por um conversor digital-analógico e um LED de potência. Os dados serão enviados a partir da FPGA e

Figura 25 – Diagrama de blocos da conversão digital-analógica e analógica-digital da transmissão de dados.



Fonte: Autores.

codificados em 0 e 3.3V, representando os sinais 0 e 1 digitais.

Para cada um desses módulos deve haver um minucioso estudo sobre suas características eletro-eletrônicas, pois existem muitas restrições que devem ser levadas em conta com a frequência de operação  $F_{op} = 200\text{kHz}$ , como:

- Amplificação de ruídos;
- Capacitâncias e Indutâncias parasitas em todos componentes, com ênfase no LED e no transistor;
- Prototipação dificultada, pois a *breadboard* possui muita capacidade em seus contatos;
- Alta corrente para ligar um LED de potência.
- Componentes limitados pela frequência de operação.

Uma vez contornadas e solucionadas todas as restrições listadas acima, o circuito resultante deverá enviar sinais luminosos a uma frequência de 200kHz, que é a velocidade definida pela norma da camada PHY I.

### Conversor Digital-Analógico

Como o LED será de alta potência, a corrente necessária para ligá-lo corretamente é na ordem de grandeza de amperes. Essa corrente de polarização é incompatível com a fornecida pela porta de saída da FPGA (aproximadamente mil vezes menor). De forma a proteger o circuito da FPGA e também a chavear o LED, foi utilizado um transistor de potência.

Além disso, deve-se notar que o método de modulação desejado é OOK (*on-off keying*). Isso significa que, na presença de portadora (sinal 1), a luz deve estar ligada, enquanto em sua ausência (sinal 0), a luz deve estar desligada.

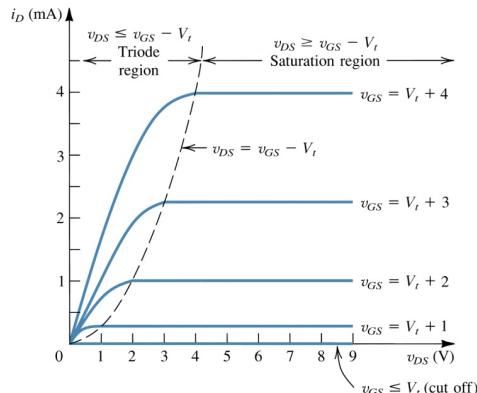
## Curva característica de FET

Para projetar um circuito chaveador com um transistor de potência, é importante entender seu funcionamento. Um FET (*Field Effect Transistor*) é um transistor que utiliza campo elétrico para controlar a condutividade de um tipo de carga de um canal em um semicondutor. Existem dois tipos de canais para transistores: n e p.

Em um *n-channel* FET, a aplicação de voltagem positiva no *gate* em relação à *source* permite a livre passagem de elétrons pelo caminho *drain-source*. Analogamente, em um *p-channel*, a aplicação de voltagem positiva no *gate* bloqueia a passagem de elétrons. Nesse caso, a voltagem aplicada deve ser zero ou negativa para polarizá-lo. Esse comportamento deve ser melhor detalhado utilizando uma curva característica de transistores tipo FET.

O gráfico 2 caracteriza a resposta  $i_D$  do transistor tipo FET de acordo com a tensão  $V_{GS}$  fornecida, indicando as regiões operação do componente. As regiões de interesse para o chaveamento do LED são as de saturação e corte, e serão detalhadas abaixo.

Gráfico 2 – Curva característica de um MOSFET, com indicações das regiões de saturação, corte e triodo.



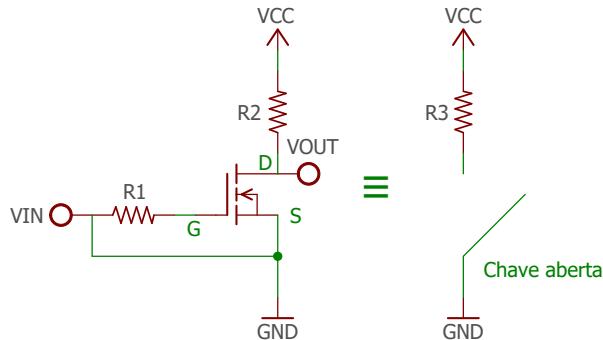
Fonte: ([SEDRA/SMITH, 2014](#))

**Região de Corte** Na região de corte, o transistor se comporta como uma chave aberta. Esse comportamento é visto em todos transistores FET em que se aplica uma tensão  $V_{GS} \leq V_t$ , onde  $V_t$  indica a tensão de limiar característica de cada componente. A Figura 26 mostra um circuito com comportamento equivalente.

Nesse esquemático, R1 representa o resistor que controla a tensão no *gate*, enquanto R2 e R3 representam a carga.

Para a aplicação OOK, é interessante que a chave do LED esteja aberta quando a FPGA gerar um 0 digital. A restrição mínima da tensão de limiar é  $V_{th} > 0V$ , pois a FPGA não consegue gerar sinais negativos sem circuito auxiliar.

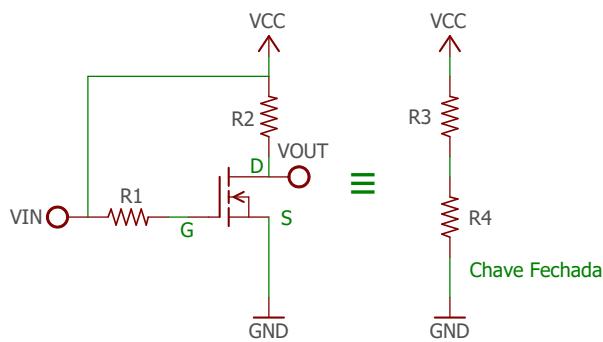
Figura 26 – Equivalência de circuitos com MOSFET em região de corte.



Fonte: Autores.

**Região de Saturação** Na região de saturação, o transistor se comporta como uma chave fechada. Em uma situação real, dependendo de qual  $V_{GS} \geq V_t$  aplicado, é permitida maior ou menor passagem de corrente  $i_D$ , como pode-se observar na 2. Os valores  $i_D$  não são fiéis à realidade, uma vez que um MOSFET de potência permite passagem de corrente muito maior. O esquemático na Figura 27 mostra um circuito com FET na região de saturação com um circuito com comportamento equivalente, utilizando apenas componentes passivos.

Figura 27 – Equivalência de circuitos com MOSFET em região de saturação.



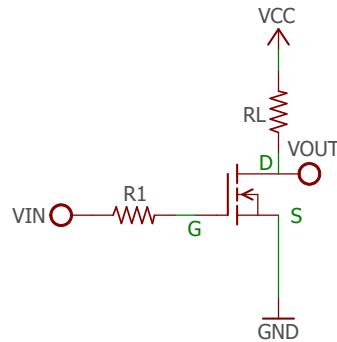
Fonte: Autores.

Nesse esquemático, R1 representa o resistor que controla a tensão no *gate*, enquanto R2 e R3 representam a carga e R4 a resistência interna  $R_{DS}$  do MOSFET.

## Conclusão

O transistor deve ser deixado na configuração de fonte comum da Figura 28, agindo como chave entre o circuito do LED, VCC e GND. Quando a FPGA gerar o sinal digital 1, o circuito deve atender  $V_{in} \geq V_{(i_D > CorrentedeLEDaceso)}$ . Da mesma forma, quando o sinal gerado for 0, é necessário projetar  $V_{in} < V_{i_D \approx 0}$ .

Figura 28 – FET-n em modo de fonte comum.



Fonte: Autores.

## LED

De acordo com a [Tabela 2](#) é necessário suportar uma frequência de oscilação luminosa mínima de 200kHz. Para não tornar o LED limitante, seria interessante se ele também suportasse a frequência de transmissão de 120MHz, que é a frequência para PHY II. Além disso, ele deve realizar a transmissão de luz a distâncias de pelo menos um metro, conforme requisito funcional do projeto.

Será então necessária a utilização de um LED de potência. É muito importante que o LED escolhido tenha resposta luminosa de acordo com o chaveamento, de forma que a tensão fornecida seja proporcional à luz emitida. Como o grupo não tem nenhum medidor de intensidade luminosa, será muito difícil de caracterizar um problema dessa natureza como, por exemplo, a possibilidade de o LED atrasar em relação a excitações em seus terminais. O circuito polarizador de um LED de alta potência pode ser simples, apenas ligando-o a um resistor de alta potência chamado de resistor limitador de corrente, como na [Figura 29](#).

Figura 29 – Circuito polarizador de LED, com resistor limitador de corrente.



Fonte: Autores

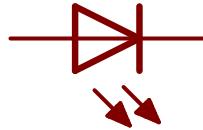
A corrente que passa pelo LED, que não pode exceder valores definidos de fábrica, pode ser calculada pela fórmula:

$$V_{LED} = \frac{V_{CC}}{R_{limit}} \quad (2.22)$$

## Resposta Luminosa

Como definido no próprio nome, o LED é um Díodo Emissor de Luz. Então, para efeitos de simulação, pode-se considerar que ele é um diodo ([Figura 30](#)).

Figura 30 – Símbolo esquemático de um LED, similar ao do diodo.

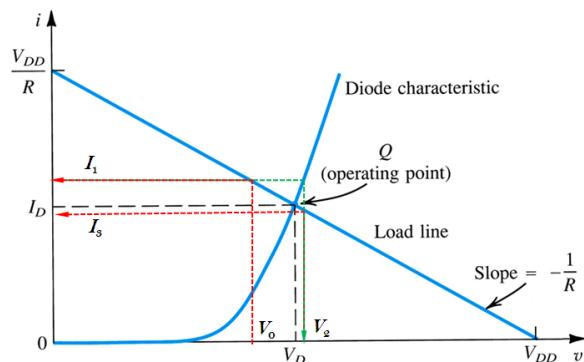


Fonte: Autores.

O circuito proposto visa oscilar todos os componentes a 200kHz. Isso significa oscilar entre a região de saturação e corte. Ao realizar o fechamento ou abertura de seus terminais, é possível que o componente responda com efeitos de um capacitor. As componentes capacitivas passivas e ativas dos componentes criam respostas indesejadas no circuito. Para atenuá-las, é possível utilizar um filtro, que remove esse comportamento indesejado, aplicando uma onda de acordo com a entrada (da FPGA) no LED.

Ainda existe outra opção para não gerar a componente capacitiva ativa no circuito: oscilar a voltagem de entrada apenas dentro da região de saturação do LED. Essa modificação consiste em aplicar uma componente contínua  $V_{DC}$  no LED e uma alternada  $V_{AC}$  pequena com a portadora do sinal gerado pela FPGA. Isso é muito utilizado em circuitos de fibra ótica, utilizando um componente chamado *Bias-Tee*.

Gráfico 3 – Zona de operação do diodo.



Fonte: ([SEDRA/SMITH, 2014](#))

No gráfico 3 é possível observar a curva de operação do diodo crescente. Ao submetê-lo a uma tensão, há resposta em  $I_D$ , que no caso do LED corresponde a intensidade luminosa. Cada diodo (ou LED) tem sua curva, e será necessário projetar o circuito levando esses parâmetros em consideração.

No entanto, é possível que essa estratégia cause modificações também no circuito do receptor, pois o LED deixará de apagar e acender, apenas oscilando sua intensidade.

### 2.3.2 Recepção

É de responsabilidade do receptor realizar três processos: (i) a recepção dos sinais luminosos, (ii) sua amplificação com filtragem e (iii) a conversão das ondas em sinais digitais. Com o funcionamento pleno deles, será possível receber um sinal equivalente ao transmitido pela FPGA. Os processos e todos os métodos envolvidos serão explicados em detalhes a seguir nas seções 2.3.2, 2.3.2, 2.3.2, 2.3.2 e 2.3.2.

#### Recepção Luminosa

O módulo de recepção luminosa é o mais crítico do receptor, pois recebe oscilações luminosas e as converte em sinais analógicos. É muito empregado em circuitos receptores de fibra óptica, CD players, televisões, câmeras e até em medidores de batimento cardíaco. Os fotorreceptores mais comuns são fototransistores ou fotodiodes, que são classificados principalmente de acordo com sua velocidade de chaveamento.

Um fototransistor é composto por um transistor de junção que responde à incidência de luz gerando e amplificando corrente elétrica. Seu tempo de resposta é da ordem de  $\mu\text{s}$ . Já o fotodiode é um diodo semicondutor que gera diferença de potencial ou mudança em sua resistência quando iluminado, com o tempo de resposta usualmente em ns, devido a sua natureza semicondutora e arquitetura de silício.

Para o projeto, é mais interessante a escolha do fotodiode, pois possui alta velocidade de resposta, tornando a onda recebida o mais fiel possível à realidade.

O fotodiode tem dois principais modos de operação, com as seguintes características na Tabela 8.

Tabela 8 – Comparação entre modos de operação do fotodiodo.

Modo de Operação	Polarização	Características
Fotovoltaico	não	$I_{dark}$ menor, resposta lenta
Fotocondutivo	negativa	$I_{dark}$ maior, resposta rápida

Fonte: Autores.

Como velocidade de resposta é prioritária na aplicação de comunicação, será utilizado o modo fotocondutivo. A polarização reversa aumenta o comprimento da região de depleção e diminui a capacidade de junção, aumentando a performance em altas frequências.

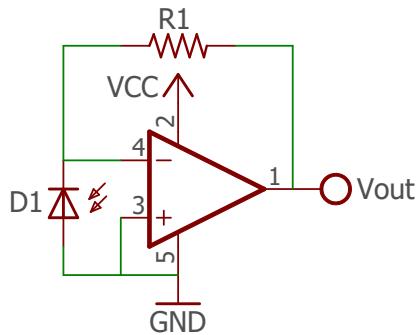
Mas como o fotodiodo apenas altera a corrente de polarização proporcionalmente à luz que recebe, portanto é necessário convertê-la em tensão para amplificar o sinal e filtrar os eventuais ruídos. A conversão será feita por um amplificador de transimpedância.

### Amplificador de Transimpedância

O comportamento de um amplificador de transimpedância é simples: converter corrente em tensão. Ele normalmente é utilizado com componentes em que a resposta em corrente é mais linear que a resposta em tensão, como no caso do fotodiodo.

O amplificador apresenta baixa impedância de entrada para isolar o fotodiodo da voltagem de saída. Em relação à saída, ao contrário do fototransistor, que tem corrente de saída a  $\mu\text{A}$ , o fotodiodo gera uma corrente na ordem de grandeza de pA. Para converter essa corrente baixa em voltagem útil para o circuito, é necessário realizar a amplificação desse sinal utilizando um resistor de feedback  $R_F$ , rotulado como R1 na Figura 31.

Figura 31 – Circuito Amplificador de Transimpedância.



Fonte: Autores.

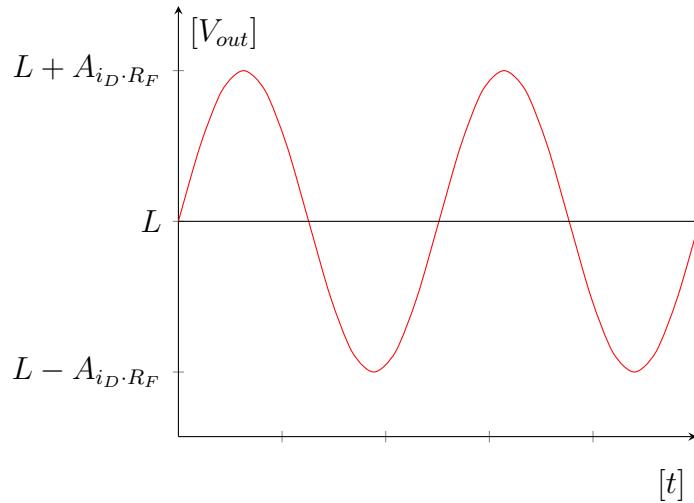
O ganho de amplificação é dado pela fórmula:

$$V_{out} = -I_{fotodiodo} \cdot R_{feedback} \quad (2.23)$$

Como  $I_{fotodiodo}$  é da ordem de pA,  $R_F$  deve ser grande ( $M\Omega$ ) para tornar a voltagem compatível com níveis TTL. Nota-se que  $V_{out}$  é negativo pois o amplificador se encontra na configuração inversora. O gráfico 4 ilustra o comportamento da saída do Amplificador de Transimpedância.

Após esse estágio de recepção, obtém-se uma onda com componente  $V_{DC} = L$ . Esse componente é variável e depende de alterações na intensidade de luz, variações na distância do transmissor com receptor, entre outros fatores. A amplitude da onda depende da corrente  $i_D$  do fotodiodo e da resistência de feedback  $R_F$  do circuito amplificador.

Gráfico 4 – Saída esperada do Amplificador de Transimpedância, com amplitude de  $i_D \cdot R_F$ , com componente  $V_{DC} = L$  variável.



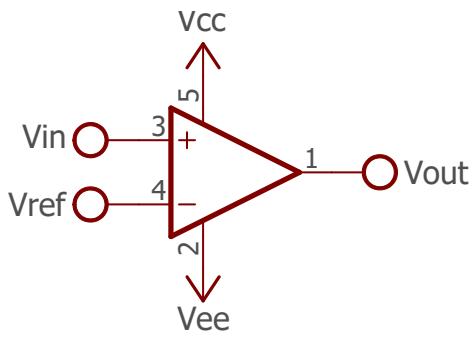
Fonte: Autores.

### Conversor Analógico-Digital

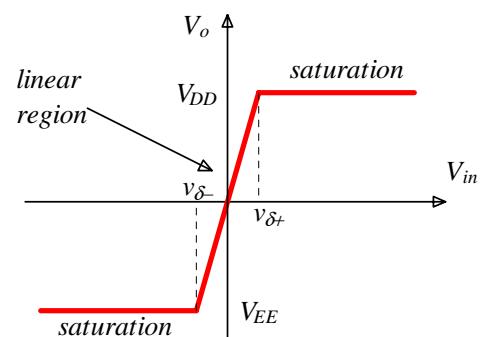
Para poder realizar estágios adicionais de amplificação e comparação a fim de obter uma saída compatível com a entrada da FPGA, deve-se entender o funcionamento de um comparador. O circuito comparador utiliza um amplificador operacional com alto ganho sem realimentação e recebe dois sinais: uma voltagem de entrada  $V_{in}$  e uma voltagem de referência  $V_{ref}$ . Ele realiza a comparação do nível de tensão analógico  $V_{in}$  com  $V_{ref}$  e produz saída de acordo com essa comparação. Seu desenho esquemático pode ser visto na Figura 32a e o comportamento de sua saída na Tabela 9.

Figura 32 – Comparador e curva de resposta.

(a) Circuito comparador simples.



(b) Curva de resposta de comparador real.



Fonte: Autores.

Fonte: (CHANIOTAKIS; CORY, 2006)

Na primeira e terceira cláusulas da Tabela 9, o amplificador operacional opera na região de saturação. No entanto, seu funcionamento para entrada  $V_{in} \approx V_{ref}$  polariza o

Tabela 9 – Comportamento real de um comparador com amplificador operacional.

$V_{in}$	$V_{out}$
$V_{in} > V_{ref}$	$V_{CC}$
$V_{in} \approx V_{ref}$	depende de $V_{\delta-}$ e $V_{\delta+}$
$V_{in} < V_{ref}$	$V_{EE}$

Tabela 10 – Fonte: Autores.

semicondutor em uma região linear que depende de dois parâmetros adicionais:

$$V_{\delta+} = \frac{V_{DD}}{A} \quad (2.24)$$

$$V_{\delta-} = \frac{V_{EE}}{A} \quad (2.25)$$

Nesse intervalo  $V_{\delta-} < V_{in} < V_{\delta+}$  (ilustrado em [Figura 32b](#)), o comportamento da saída é diferenciado e depende do ganho de *loop* aberto A, que para circuitos reais é muito grande  $A = 20000$ . Os valores  $V_{\delta-}$  e  $V_{\delta+}$  então são da ordem de  $\mu\text{V}$ , definindo a tensão mínima (ou máxima) da entrada para que o comparador realize uma comparação adequada. Com uma amplitude  $V_{in_{AC}}$  alternada suficientemente alta, é possível tornar a curva de resposta mais íngreme melhorando o tempo de resposta do componente. A amplitude pode ser ajustada com o resistor de feedback  $R_f$  do estágio anterior. Mesmo com essas correções, a onda comparada pode defasar um pouco na saída.

### Filtro Passa-Altas

Como foi observado na [4](#), a onda recebida possui um componente  $V_{DC} = L$  variável. Essa ela não tem valor fixo, então não é possível definir a tensão  $V_{ref}$  para comparar. Nesse caso existe a necessidade da eliminação da componente  $V_{DC} = L$ .

Para isso, pode-se utilizar um simples circuito RC de primeira ordem, que age como filtro passa-altas, esquematizado na [Figura 33](#). Seu funcionamento se baseia no capacitor C, que em baixas frequências apresenta alta reatância, atuando como um aberto e bloqueando qualquer sinal abaixo da frequência de corte  $f_c$ . Acima dessa frequência, a reatância do capacitor é reduzida suficientemente para agir como um curto-círcuito, permitindo qualquer entrada  $V_{in}$  passar diretamente para saída  $V_{out}$ . A frequência de corte  $f_c$  só pode ser obtida com a presença do resistor R e do capacitor C juntos, que geram a constante de tempo  $\tau$ . O cálculo desses valores se encontra abaixo:

$$\tau = R \cdot C \quad (2.26)$$

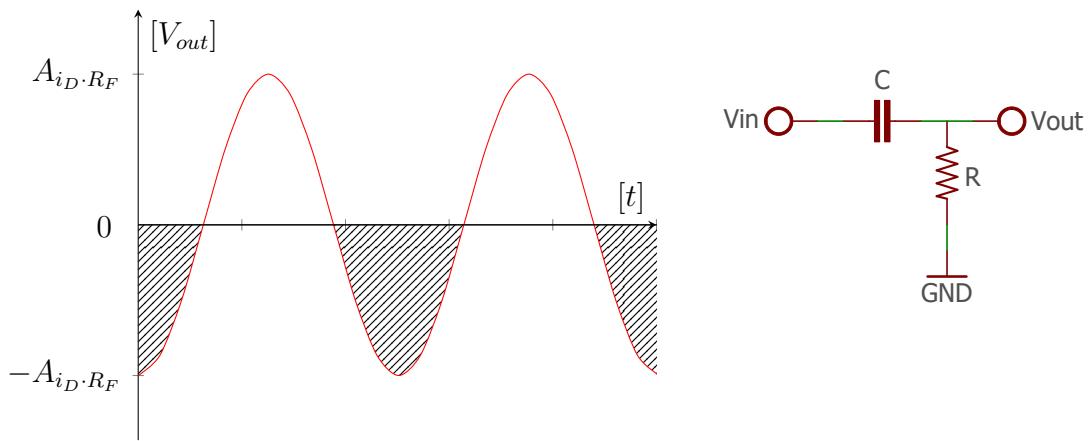
$$f_c = \frac{1}{2 \cdot \pi \cdot \tau} \quad (2.27)$$

Ainda existe a possibilidade da fase de resposta do circuito se deslocar devido à presença do capacitor. A fórmula que calcula sua defasagem/adiantamento  $\phi$  é:

$$\phi = \arctan\left(\frac{1}{2 \cdot \pi \cdot f_{op} \cdot R \cdot C}\right) \quad (2.28)$$

A [Figura 33](#) ilustra o comportamento da saída do filtro passa-altas juntamente com o esquemático do filtro passa-altas.

Figura 33 – Saída esperada após remoção do acoplamento DC, aplicando filtro passa-alta RC de primeira ordem (à direita), com defasagem de  $45^\circ$ . A área hachurada representa tensão inadequada para o comparador.



Fonte: Autores.

### Acoplamento DC

Com a aplicação do filtro passa-altas agora a onda gerada tem componente  $V_{DC} = 0$  fixa - observada na parte hachurada da [Figura 33](#), que ainda necessita correção, pois é uma tensão incompatível com o circuito.

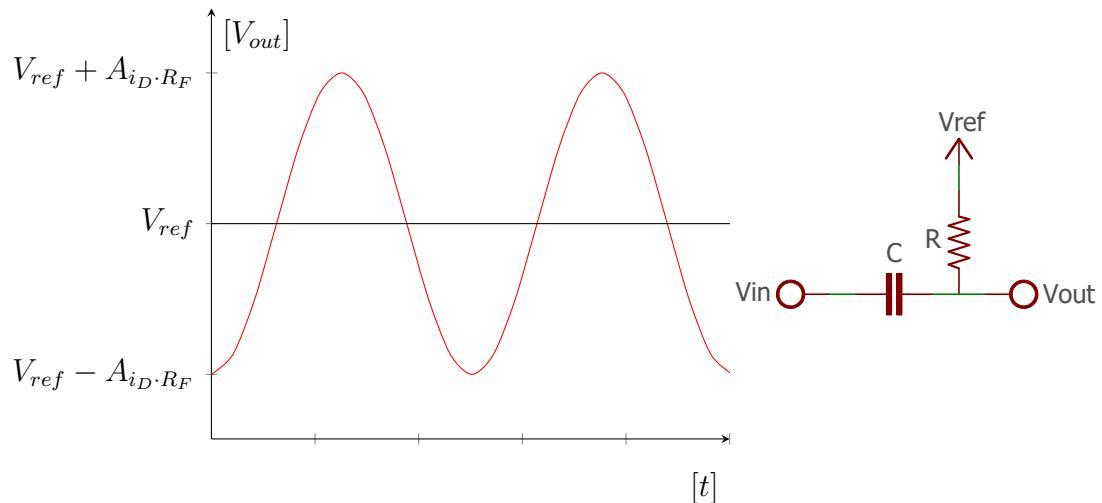
O funcionamento do comparador (e de qualquer amplificador operacional) depende altamente do fato de que as voltagens de entrada  $V_{in}$  e  $V_{ref}$  estejam dentro do intervalo de alimentação. Caso as tensões aplicadas nas entradas sejam muito altas (ou baixas), o amplificador pode operar incorretamente, ser reversamente polarizado e até ser danificado. Esses valores costumam estar presentes em *datasheets* de componentes em uma tabela chamada *Absolute Maximum Ratings*. Como foi decisão de projeto o fato de que o sistema de comunicação utilizaria apenas uma fonte de alimentação, essa restrição será aplicada para todos os componentes do circuito.

Apenas uma fonte significa que o circuito terá apenas alimentação positiva disponível em seus terminais:

$$V_{in}, V_{ref} \in [V_{EE} \geq 0, V_{CC} > V_{EE}] \quad (2.29)$$

Portanto, o componente não conseguirá realizar a comparação corretamente. Para resolver esse problema pode-se adicionar um acoplamento  $V_{DC} = V_{ref}$  fixo, que também será utilizado na entrada de referência do comparador. E a etapa de filtro pode ser realizada juntamente com a adição de  $V_{ref}$ , utilizando um circuito levemente modificado.

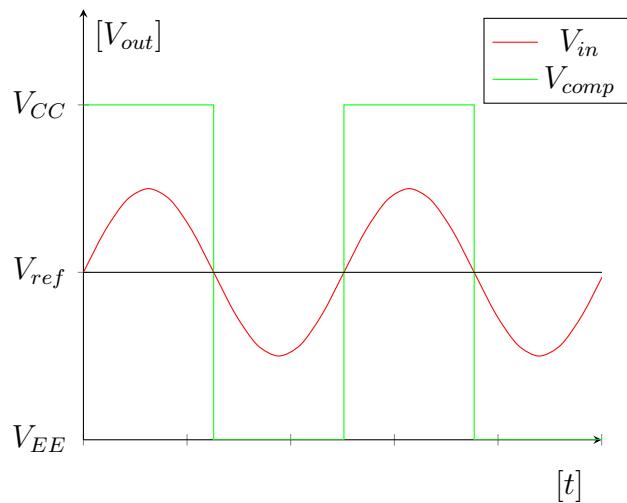
Gráfico 5 – Gráfico com adição de componente DC  $V_{DC} = V_{ref}$  fixa e filtro passa-altas modificado.



Fonte: Autores.

O gráfico 5 ilustra a onda modificada, agora com voltagem de referência fixa. Após todas essas etapas, é possível realizar de fato a etapa de comparação. Idealmente a onda ficaria similar ao gráfico 6.

Gráfico 6 – Saída esperada após a etapa de comparação em verde.



Fonte: Autores.

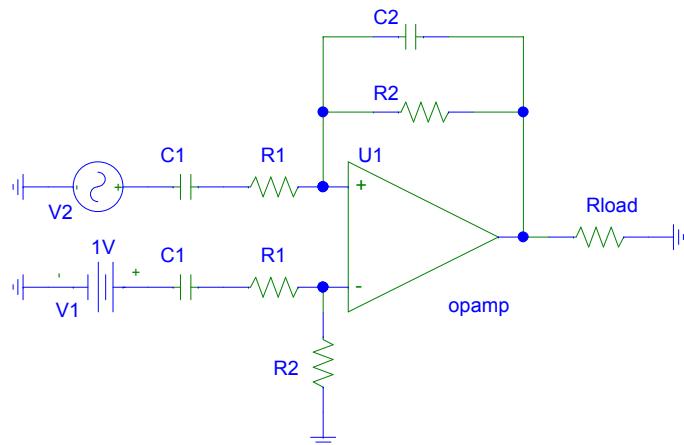
Com a sensibilidade ajustada de acordo, é possível que o sinal recebido seja muito semelhante (se não exatamente igual) ao enviado. Caso a saída não esteja adequada, é

possível que seja necessário realizar mais um estágio de amplificação e filtração após o filtro passa-altas.

### Filtro Passa-Faixas

Com intuito de reduzir o ruído em circuitos de amplificação com amplificadores operacionais, especifica-se um modelo de circuito passa-faixas na Figura 34.

Figura 34 – Circuito amplificador de diferenças.



Fonte: Autores

De acordo com a tabela 2 e a norma PHY II, o intervalo da frequência do *clock* luminoso é de 200kHz até 120MHz. Deseja-se então permitir a passagem dessas, limitando as frequências fora desse intervalo. Com o uso do passa-baixas e passa-altas da figura acima, pode-se realizar essa limitação adequando os resistores e capacitores do circuito.

Deve-se levar em conta também o ganho do circuito, que é dado pela fórmula:

$$V_{out} = V_{in} \cdot \frac{R2}{R1} \quad (2.30)$$

O cálculo da frequência de corte inferior do circuito é dado por:

$$f_{ci} = \frac{1}{2 \cdot \pi \cdot R1 \cdot C1} \quad (2.31)$$

No caso da frequência de corte superior, pode-se realizar o cálculo utilizando a fórmula abaixo:

$$f_{cs} = \frac{1}{2 \cdot \pi \cdot R2 \cdot C2} \quad (2.32)$$

O cálculo da frequência central é feita pela média geométrica das frequências calculadas em 2.31 e 2.32 acima:

$$f_o = \sqrt{f_{ci} \cdot f_{cs}} \quad (2.33)$$

A banda de passagem do circuito é calculada pela subtração da frequência de corte superior pela inferior:

$$BW = f_{cs} - f_{ci} \quad (2.34)$$

### 2.3.3 FPGA

FPGA (*Field-Programmable Gate Array*) é uma tecnologia que utiliza *chips* de silício reprogramáveis para obter circuitos customizáveis, latência a nível de hardware e vazão muito alta. No projeto LiCy, como o fluxo de informações é muito alto (requisito de projeto de 1Gbps), é necessário realizar múltiplos processos ao mesmo tempo para lidar com os dados:

- Recebimento/Transmissão de dados
- Cálculo de paridade
- Correção de erros quando do recebimento
- Codificação/Decodificação de dados

Essa tecnologia é ideal para criar protótipos de circuitos digitais. Existe neste caso a facilidade de mudar o arranjo de componentes e vias de dados, bem como a possibilidade de simulá-los e depurá-los antes mesmo de gravar as configurações no hardware. Estes são elementos chave para a criação de um ambiente controlado e que possibilita testar as funcionalidades do produto desenvolvido, sendo um dos fatores que mais motivou a escolha dessa tecnologia neste projeto.

O projeto de uma FPGA possui características que anteriormente estavam associadas a sistemas baseados em processadores. Uma delas é o uso de ferramentas e interfaces de alto nível, usando diagramas de bloco representando comportamentos e linguagens de descrição de hardware (na sigla em inglês, HDL), que no entanto diferem em muito de linguagens de programação como C.

A principal vantagem do uso de uma FPGA em relação a microcontroladores está na independência das operações de processamento que, por estarem em circuitos paralelos, não precisam dividir recursos como um mesmo núcleo, por exemplo. Entretanto, o alto

grau de paralelismo pode resultar em maiores desafios, como garantir o sincronismo e lidar com inconsistência de dados no nível de circuito. Ainda assim, é preferível neste projeto lidar com esses desafios para se beneficiar do ganho em latência e vazão de dados.

## 2.4 Suporte ao hardware digital

Esta seção explica outras escolhas feitas no aspecto de desenvolvimento de circuitos digitais do projeto.

### 2.4.1 VHDL

VHDL, sigla do inglês para *Very High Speed Integrated Circuit Hardware Description Language*, é uma linguagem de descrição de hardware. Entre suas características incluem-se uma tipagem forte, comportamento determinístico em comparação com outras linguagens de descrição de hardware como Verilog, e aplicação maior em FPGAs na área acadêmica. Por esses motivos, a curva de aprendizado pode ser mais inclinada no início, mas uma vez ultrapassada a barreira da maior rigidez no processo de compilação, a detecção de erros em código é facilitada.

Uma linguagem de descrição de hardware tem por objetivo modelar sistemas digitais através de código. Alguns parâmetros importantes dessa modelagem são o comportamento, a estrutura e o tempo - todos cobertos por VHDL. Analogamente às linguagens de programação, VHDL possui categorias de dados, como sinais, variáveis e constantes, que servem para representar fios, registradores e outros elementos de circuitos relacionados a dados.

Além disso, a linguagem VHDL é uma ferramenta interessante para a construção de máquinas de estado finitos. A síntese desse tipo de circuito é essencial para a implantação em hardware de algoritmos que envolvem iteração, o que é mais uma forte justificativa para a escolha desta ferramenta neste estudo. Outro fator importante levado em consideração para essa escolha foi a familiarização prévia com VHDL durante o curso, nas matérias de Laboratório, Organização e Arquitetura de Sistemas Digitais, o que auxiliou na escalada da curva de aprendizado.

### 2.4.2 Quartus

Quartus é um software de programação de dispositivos lógicos, desenvolvido pela Altera, subsidiária da Intel, que fabrica dispositivos lógicos programáveis, como as FPGAs citadas anteriormente. Algumas das vantagens dessa ferramenta são a possibilidade de sintetizar circuitos a partir de diagramas lógicos, simular o comportamento de circuitos no

tempo, controlando entradas através de *testbenches*, e uma interface relativamente simples para se programar uma FPGA.

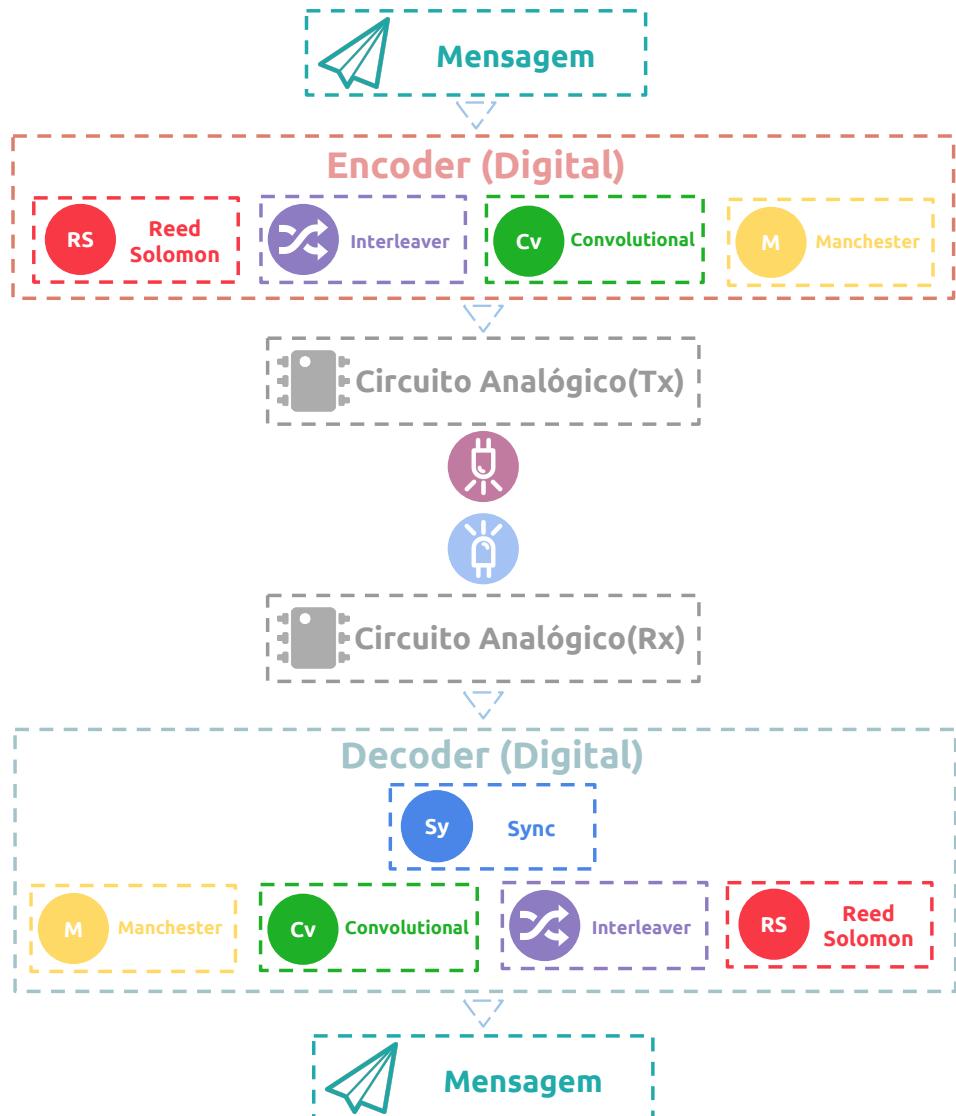
Dentro do contexto do estudo de sistemas digitais, os integrantes do grupo puderam se familiarizar com o software Quartus. Ademais, o fato desta ferramenta se integrar com famílias de FPGAs como Cyclone ou MAX (PLD) foi interessante, já que isto tornou possível o uso da infra-estrutura presente nos laboratórios da Universidade.

# 3 Execução

## 3.1 Arquitetura

Esta sessão visa apresentar a arquitetura dos sistemas digitais desenvolvidos durante o projeto. Conforme o embasamento teórico apresentado no capítulo de Metodologia, todos os circuitos apresentados a seguir servem a um mesmo objetivo: enviar uma mensagem pela camada física por um dos nós da comunicação e recebê-la integralmente através do outro nó. Separamos, portanto, na [Figura 35](#) a Transmissão na metade superior, e Recepção na inferior.

Figura 35 – Desenho esquemático da arquitetura do sistema.



Fonte: Autores.

A seguir, explicam-se os componentes indicados dentro de Encoder e Decoder, que em conjunto sintetizam todos os circuitos digitais do estudo. Explicitar-se-ão as decisões mais importantes de projeto e a forma pela qual se integraram e se testaram os dois módulos.

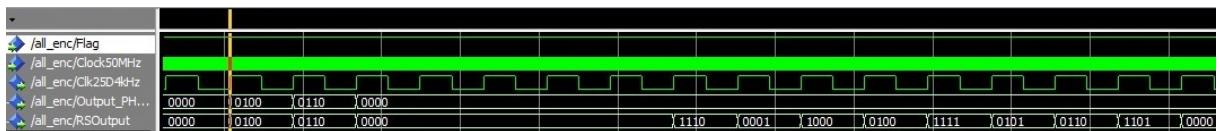
## 3.2 Codificador Digital

### 3.2.1 Codificador Reed Solomon

Conforme apontado no capítulo de Metodologia, a arquitetura do codificador Reed Solomon foi construída para um código (15, 7), portanto contando com 8 registradores em série. Os multiplicadores foram feitos com uma tabela, que recebe duas entradas e retorna uma saída, de maneira a simplificar o trabalho de fazer uma função utilizando lógica combinatória. Já os somadores são equivalentes à função de OU exclusivo, ou XOR, com duas entradas e uma saída de 4 *bits* cada - ou seja, um símbolo. Além destes, os módulos básicos ainda compreendem um multiplexador, que seleciona um entre dois símbolos. A arquitetura desenvolvida pode ser observada na [Figura 13](#).

O comportamento de codificação de uma mensagem pode ser exemplificado pela [Figura 36](#), onde se observa a entrada de uma mensagem de 7 símbolos e a saída de um bloco de 15 símbolos, sendo 8 de paridade.

Figura 36 – Simulação de codificação do codificador Reed Solomon.



Fonte: Autores.

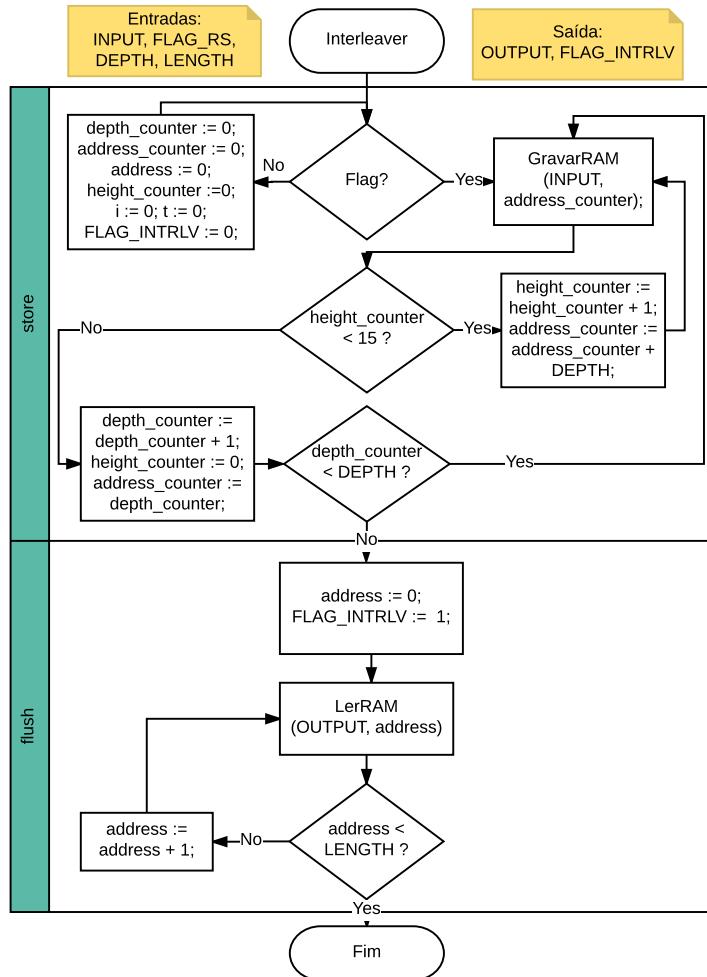
### 3.2.2 Interleaver

O módulo do Interleaver é composto por três componentes principais: (i) a memória, (ii) a unidade de controle do cabeçalho e (iii) a unidade de controle do payload.

#### Memória

Para realizar o processo de entrelaçamento, é necessário que o *Interleaver* guarde a mensagem inteira dentro de uma memória, para manipular sua ordem de disposição.

Deve-se observar, no entanto, que o tamanho da mensagem a ser guardado é maior que apenas o tamanho máximo do *payload* da PHY I (1023 *bytes*). Isso acontece porque o módulo anterior ao *Interleaver* - o Codificador Reed Solomon - transforma cada 28 *bits*

Figura 37 – Diagrama esquemático do fluxo de dados do *Interleaver*.

Fonte: Autores.

de mensagem em 60 bits (Reed Solomon (15,7)). A mensagem com paridade adicionada ao fim tem portanto 2191 bytes. Além disso, ainda deve ser considerado o cabeçalho, sua paridade e seu padding (4 + 3 + 8 bytes). Portanto o Interleaver deve ser, então, capaz de guardar 2206 bytes. Caso a mensagem seja maior que o tamanho máximo do payload, ela deve ser dividida em outro payload, para ser enviada posteriormente.

A princípio, é possível realizar esse armazenamento de três formas na FPGA: (i) criando vetores dentro de um arquivo .vhd, (ii) utilizando uma memória externa a FPGA e (iii) utilizando uma memória interna da FPGA. A primeira solução ocupa muito espaço disponível nas unidades lógicas do chip. A segunda proposta requer uma memória acoplada à placa de desenvolvimento utilizada, restringindo a gama de placas possíveis. A terceira e última opção é a mais adequada, pois a arquitetura da FPGA utilizada (EP4) possui memória auxiliar incorporada na FPGA.

Para implementar a memória, foi utilizado o componente RAM de duas portas do Catálogo IP (Intellectual Property) do programa Altera Quartus. De forma que a

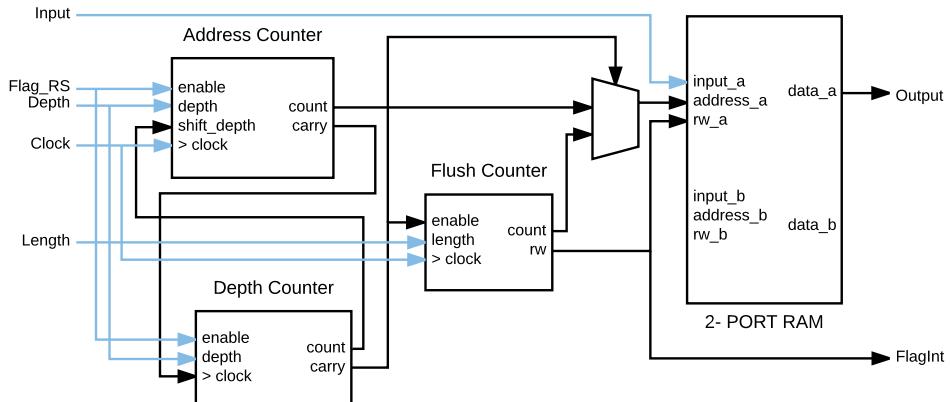
capacidade de memória seja no mínimo 2206 *bytes*, foi instanciada uma de 4096 *bytes*, pois devem ser potências de 2.

### Unidade de controle do cabeçalho

O propósito deste módulo é receber os dados do Codificador Reed Solomon, gravá-los na memória interna e depois enviá-los de forma entrelaçada. A abordagem tomada pelo projeto foi gravar esses dados em ordem entrelaçada, para que a leitura seja feita apenas varrendo os endereços da memória em ordem crescente. A [Figura 37](#) ilustra o algoritmo desenvolvido para escolher os endereços entrelaçados e gerar a saída. Ele foi implementado em Javascript para melhor visualização da ordem certa de gravação.

Em VHDL, a solução é composta por três componentes: (i) contador de endereços, (ii) contador de profundidade e (iii) contador de saída. Eles estão integrados com à memória na [Figura 38](#).

Figura 38 – Diagrama esquemático do *Interleaver*, com unidade de controle e memória. Os sinais azuis representam entradas para melhor visualização.



Fonte: Autores.

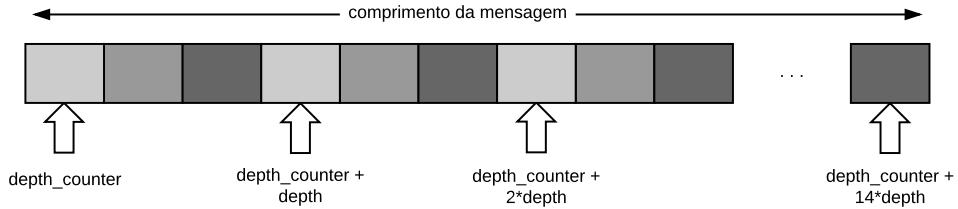
### Contador de endereços

Responsável por selecionar endereços para a etapa de gravação da memória. Recebe a profundidade atual e o valor da profundidade da mensagem.

Inicialmente, a profundidade atual é utilizada como endereço na etapa de gravação da mensagem na memória. As iterações posteriores utilizam também o valor de profundidade da mensagem para deslocar os dados na memória, realizando esse processo quatorze vezes, uma vez para cada símbolo do codificador Reed Solomon, como ilustrado na [Figura 39](#). A equação abaixo também auxilia no entendimento do processo:

$$Address = depth_{counter} + iteration \cdot depth \quad (3.1)$$

Figura 39 – Ordem do algoritmo entrelaçador para gravar a memória.



Fonte: Autores.

Na última iteração desse módulo, há o sinal de *carry*. Esse sinaliza ao contador de profundidade que deve-se deslocar uma profundidade na gravação.

#### Contador de profundidade

Contador que desloca o endereço base selecionado na [Figura 39](#), preenchendo toda a memória com dados entrelaçados. O processo de gravação termina quando a profundidade atual atinge o valor da profundidade da mensagem. Nesse momento, é sinalizado ao contador de saída que é possível gerar a saída.

#### Contador de saída

O contador de saída, quando habilitado pelo contador de profundidade, troca o modo de operação da memória de gravação para leitura. Esse módulo também seleciona endereços para ler de forma crescente, gerando uma saída entrelaçada, juntamente com sinal de FlagInt mostrando que os dados atuais são válidos.

#### Unidade de controle do *payload*

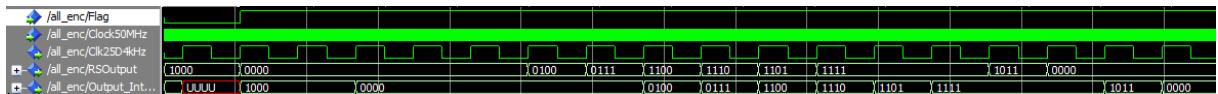
O comportamento da UC (unidade de controle) do *payload* é análoga à explicada anteriormente ([seção 3.2.2](#)). Entretanto, é necessário que eles funcionem de forma coordenada, de acordo com o recebimento do cabeçalho e em seguida dos dados. Algumas modificações foram feitas no esquema apresentado ([Figura 38](#)) para o funcionamento pleno do *Interleaver*:

- UC duplicada para o *payload*;
- Saída da segunda UC conectada na segunda porta da RAM;
- Segunda UC habilitada quando a primeira UC acaba o processo de gravação do cabeçalho;
- Segunda UC modificada para gravar depois do cabeçalho;

- Contador de saída habilitado quando a segunda UC acaba o processo de gravação;
- Critério de parada do contador de saída modificado para *comprimento*(*cabeçalho + mensagem*).

A simulação para a gravação dos dados está disponível na Figura 40. A entrada é uma mensagem Reed Solomon válida e vários zeros em seguida.

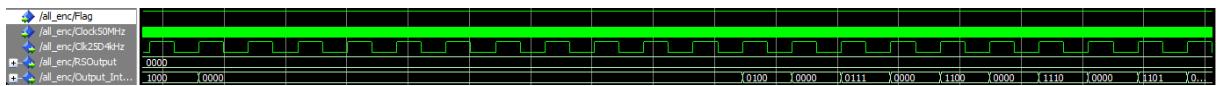
Figura 40 – Simulação da etapa de gravação do módulo *Interleaver*.



Fonte: Autores.

A etapa de saída dos dados está disponível na Figura 41. Como a profundidade da mensagem é 2, então o *Interleaver* irá alternar entre os primeiros 15 dados e os 15 seguintes. Esse comportamento é facilmente observado quando a saída alterna paridade do Reed Solomon com zeros, presentes na segunda mensagem.

Figura 41 – Simulação da etapa de saída do módulo *Interleaver*.



Fonte: Autores.

Como o componente é complexo, seu RTL encontra-se separado no Apêndice na Figura 64.

### 3.2.3 Codificador Convolucional

O codificador convolucional foi baseado no esquema corrigido da norma em Figura 14, com seis registradores representando a janela deslizante e o *exclusive OR* dos geradores polinomiais.

Nesse módulo, existe uma grande diferença da vazão da mensagem do *Interleaver* com a vazão suportada pelo Codificador Convolucional. A figura Figura 44 mostra que o entrelaçador fornece 4 *bits/clock* enquanto o convolucional apenas consome 1 *bit/clock*. Essa diferença foi tratada utilizando um módulo chamado PISO (Parallel Input to Serial Output), que transforma os dados paralelos em seriais, desde que o *clock* do convolucional seja mais rápido. Mais detalhes com relação a integração entre componentes do codificador na subseção 3.2.5.

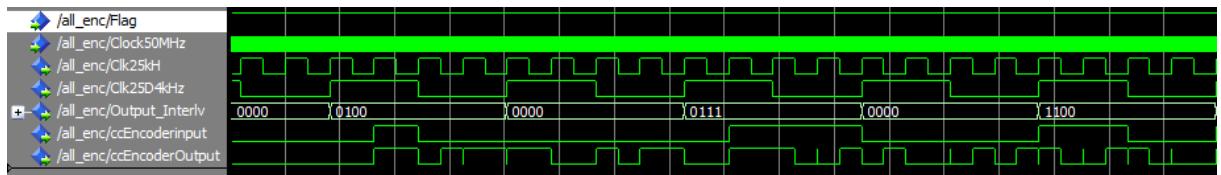
Ainda existe outro módulo que insere uma cadeia de 6 *bits* zeros entre o cabeçalho e a mensagem. Isso reinicia o Convolucional, garantindo que o estado inicial seja "000000",

estado conhecido posterior decodificação. A implementação é simples: processa as 30 primeiras palavras e depois envia os zeros via um registrador de deslocamento.

### Puncture

Para realizar o *puncture* de  $\frac{1}{4}$  o valor A é mantido pelo tempo equivalente a dois *clocks* do próximo módulo e o valor de B por mais dois *clocks*. Segundo o especificado na metodologia. A simulação de todas operações descritas encontra-se abaixo:

Figura 42 – Simulação do Codificador Convolucional. Nota-se a forma que ele serializa os dados do *Interleaver* em ccEncoderInput antes de utilizá-los, para gerar a saída em ccEncoderOutput.

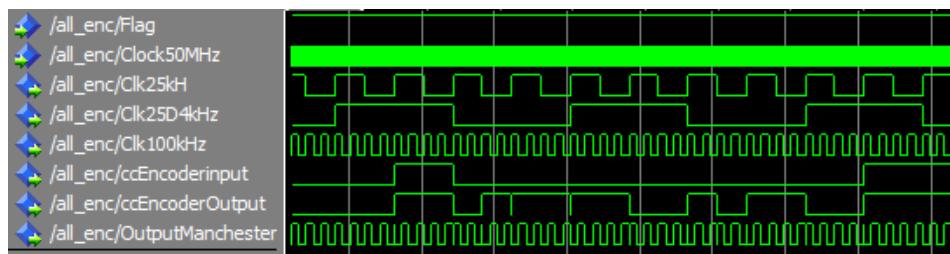


Fonte: Autores.

### 3.2.4 Codificador Manchester

A arquitetura da codificação de Manchester segue o mesmo modelo explicado na Metologia. Duas entradas: *clock* e mensagem, recebem a operação lógica XOR e retornam o código. A simulação do componente está abaixo:

Figura 43 – Simulação do Codificador Manchester.

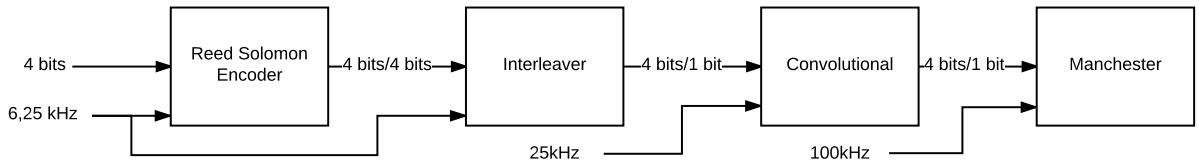


Fonte: Autores.

### 3.2.5 Integração do Codificador

Os codificadores foram integrados de modo a não existir nenhum *buffer* entre os componentes, com exceção entre a UART e o Reed Solomon. Para a saída ser na velocidade de 200kHz como foi especificado na norma, os *clocks* dos componentes têm frequências diferentes. Isso se deve ao número de *bits* de entrada diferir com o número de *bits* de saída dos componentes. Essa relação de entrada e saída de *bits* e suas frequências, é mostrada na figura abaixo.

Figura 44 – Relação entre vazão de *bits* entre cada módulo do Codificador.



Fonte: Autores.

### 3.3 Decodificador Digital

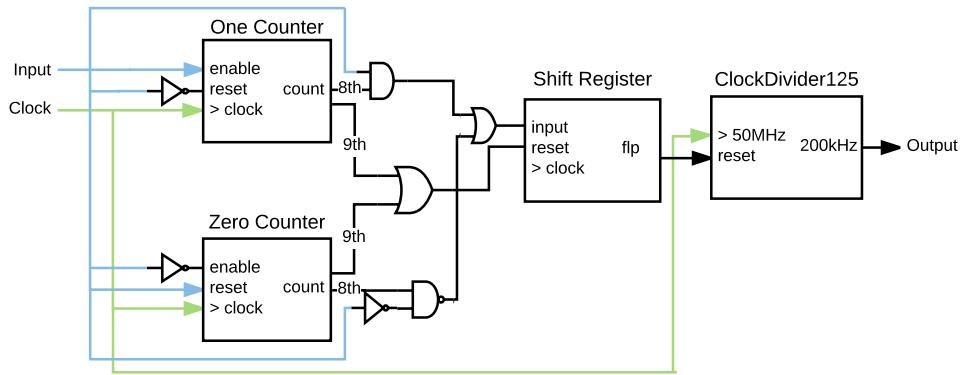
#### 3.3.1 Sync

Módulo que sincroniza o *clock* do transmissor com o do receptor, para que os dados recebidos sejam os mesmos que os enviados.

Inicialmente foi utilizado uma propriedade intelectual da Altera chamada PLL (*Phase-Locked Loop*). Um estudo mais detalhado demonstrou que sua entrada recebe apenas sinais de ordem de grandeza de MHz. É portanto incompatível com a frequência definida pela camada PHY I.

O trabalho propõe uma solução para esse problema desenvolvida pelos próprios autores, e é dividida em quatro módulos: *oversampling*, decisor de limite de tempo, detector de FLP e gerador de *clock*. O diagrama abaixo (Figura 45) ilustra as conexões entre esses módulos.

Figura 45 – Diagrama esquemático do Sync. O *clock* do registrador de deslocamentos foi omitido para simplificar a representação.



Fonte: Autores.

#### *Oversampling*

Componente que realiza a verificação do tempo de permanência do sinal para cada *bit*. Possui dois contadores, um para o sinal lógico zero e outro para um. O próprio sinal

de entrada habilita (*enable*) o contador de zeros quando está em *high*, e analogamente o contador de uns quando está em *low*. O funcionamento do sinal de *reset* é inverso ao de *enable* aos dois contadores. Portanto o módulo contador de zeros conta por quantos *clocks* de 50MHz a entrada ficou em zero. Caso a entrada seja um, o contador zera e ele é desabilitado. Analogamente para o módulo contador de uns, que conta por quantos *clocks* a entrada ficou em um, zerando.

### Decisão de Limite de Tempo

Esse subsistema decide se o sinal analisado pelo módulo de *Oversampling* possui pelo menos 250 amostras iguais (zeros ou uns). Ele parte do princípio que a entrada será uma onda quadrada de período 10  $\mu\text{s}$ , garantido pelo envio do FLP, não possuindo dois uns nem dois zeros seguidos.

Com intuito de facilitar a sincronização, o decisor de limite de tempo verifica se o *Oversampling* está dentro de um intervalo de 128 a 255 ciclos do *clock* interno da FPGA de 50 MHz. Esse processo ocorre verificando apenas os *bits* mais significativos do contador. Caso o oitavo *bit* seja um, significa que o módulo anterior contou pelo menos até 128, caracterizando um possível um ou zero. Verificando o nono *bit* é possível concluir se ele não excedeu o tempo de permanência no mesmo *bit* (além de 255).

Para o módulo sincronizador, dois uns ou dois zeros seguidos fogem da norma, portanto tais mensagens são descartadas.

### Detector de FLP

A detecção de FLP é realizada com um registrador de deslocamento de 200 *bits*. Se pelo menos 200 *bits* recebidos preenchem o registrador com zeros e uns alternados, o módulo gera um sinal lógico um para o gerador de *clock*.

O registrador é reiniciado quando o nono *bit* do Decisor de limite de tempo é um, pois caracteriza um FLP inválido. Enquanto apenas o oitavo *bit* for detectado, o registrador grava o valor e desloca seus valores à direita.

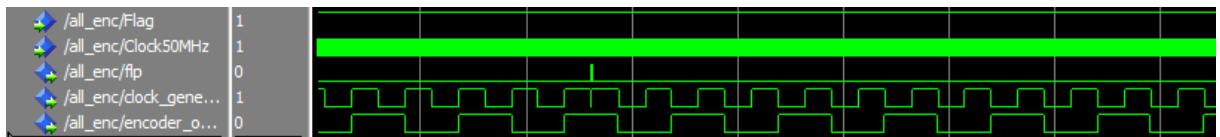
### Gerador de *clock*

Módulo que gera um *clock* de 200kHz estável para o decodificador inteiro. Utiliza um divisor de frequência do *clock* interno da FPGA.

Quando recebe o sinal de detecção de FLP, reajusta seu *clock* de 200kHz forma que a borda de subida fique no centro do *bit* de entrada. Essa característica do gerador de *clock* protege o decodificador do fenômeno de *clock drift*, ou da perda de sincronia entre os *clocks* devido ao comprimento da mensagem.

A simulação resultante desse componente se encontra abaixo:

Figura 46 – Simulação do comportamento do Sincronizador. Quando o sinal FLP é habilitado, o gerador de clock reinicia.



Fonte: Autores.

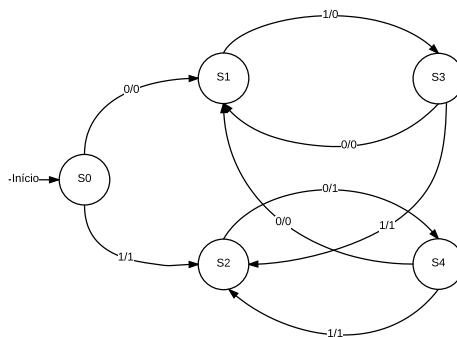
Seu RTL se encontra no Apêndice, na [Figura 65](#).

### 3.3.2 Decodificador Manchester

O decodificador funciona, conforme explicado anteriormente, de acordo com a [Tabela 5](#).

Neste projeto, o decodificador de Manchester é inicializado quando a combinação de TDPs e não TDPs é identificado. Seu circuito foi desenvolvido com uma máquina de estados que tem estado inicial S0. Essa máquina funciona de modo que quando existe uma transição de 0 para 1, a saída é um *bit* 0 e quando a transição é de 1 para 0, a saída é um *bit* 1 para 0. Isso é demonstrado na figura [Figura 47](#).

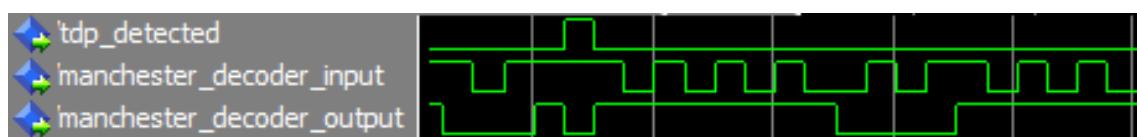
Figura 47 – Visão de máquina de estados do decodificador Manchester.



Fonte: Autores.

A simulação do componente encontra-se abaixo na [Figura 48](#), e a arquitetura implementada no programa Altera Quartus no Apêndice na [Figura 66](#).

Figura 48 – Simulação do Decodificador Manchester.



Fonte: Autores.

### 3.3.3 Decodificador Viterbi Fangled

O Decodificador Viterbi tem o mesmo problema do Codificador Convolucional: as vazões de entrada e saída são diferentes. Deve-se então converter dados seriais para paralelos utilizando um módulo SIPO (Serial Input Parallel Output). Os *clocks* também devem ser ajustados de acordo. Mais detalhes com relação a integração entre componentes do decodificador na [subseção 3.3.6](#).

Como especificado na metodologia ([seção 2.2.2](#)), a implementação Viterbi utilizada é simplificada e se chama *Fangled*. Consome menos memória e processamento e é mais veloz, porém tem capacidade de detecção e correção de erro muito menor. O *Fangled* Viterbi corrige no mínimo um *bit* a cada quatro recebidos, devido à sua arquitetura de taxa  $\frac{1}{4}$ .

#### *Branch Metrics*

A entrada é composta por AABB. Calcula-se a distância entre o primeiro A com o primeiro B e todas as possíveis combinações de bits (00, 01, 10, 11). A distância é calculada do mesmo modo para o segundo A e o segundo B recebidos. A operação feita entre os dados é XOR. As distâncias de uma mesma combinação de bits é posteriormente somada.

#### Tabela Trellis

A tabela foi implementada através de uma LUT (*Look Up Table*) com dois valores possíveis de entrada para o Viterbi. Possui duas saídas para cada entrada: uma quando o *bit* é um e outra quando o *bit* é zero.

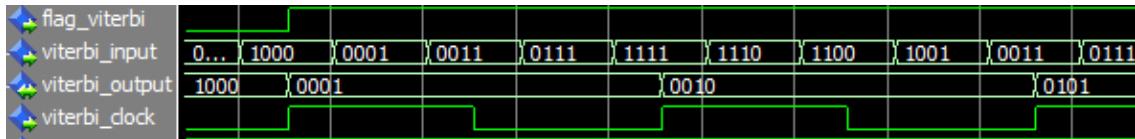
#### *Path Metrics*

O PM (*Path Metrics*) recebe a informação de quais são as duas possíveis entradas da Tabela Trellis e quais as distâncias do *Branch Metrics*. Com esses dados o PM identifica qual o bit mais provável de ter sido codificado. Definindo assim, qual o próximo estado e a saída do decodificador Viterbi. O valor do próximo estado é armazenado para ser posteriormente utilizado pela Tabela Trellis.

Caso as distâncias consideradas forem iguais, o valor da saída é escolhido como sendo zero e o próximo estado é definido de acordo. No caso do *Fangled* Viterbi, essa escolha deverá ser realizada ao acaso.

A simulação do módulo de decodificação Viterbi *Fangled* se encontra na [Figura 49](#) abaixo. A arquitetura do componente RTL encontra-se no Apêndice, na [Figura 67](#).

Figura 49 – Simulação do funcionamento do Decodificador Viterbi *Fangled*. Observa-se que o componente apenas conclui qual o símbolo recebido após quatro ciclos.



Fonte: Autores.

### 3.3.4 Decodificador Reed Solomon

A arquitetura do decodificador segue o diagrama funcional da figura X do capítulo de Metologia. Desta forma, o desenvolvimento pode ser separado em quatro partes principais, apresentadas a seguir.

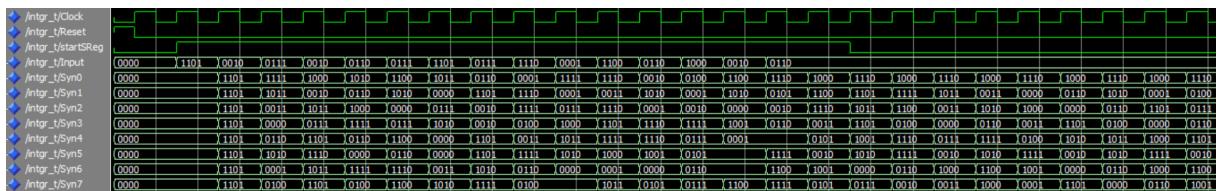
#### Cálculo das Síndromes

O cálculo das síndromes é feito de forma iterativa, durando tantos ciclos quanto forem os símbolos da mensagem. O cálculo de cada síndrome é independente dos cálculos das demais, e depende apenas dos símbolos de entrada e de cada uma das raízes do polinômio codificador, de alpha0 a alpha7 neste caso. O arranjo dos componentes pode se exemplificar através da arquitetura do módulo, representada na [Figura 60](#), no Apêndice.

Os elementos indicados por `static` são constantes, guardando os valores dos símbolos de cada umas raízes citadas. Estes valores são enviados para multiplicadores de GF(16), representados por `gf_mult`. Como pode-se observar, o cálculo é iterativo e portanto os multiplicadores recebem também o valor dos registradores que são representados por `register4b`. Os somadores de Campos de Galois são representados por `gf_sum`.

A simulação a seguir representa o caso em que existem erros na mensagem codificada, o que é indicado por síndromes diferentes de 0000.

Figura 50 – Simulação do cálculo das síndromes.



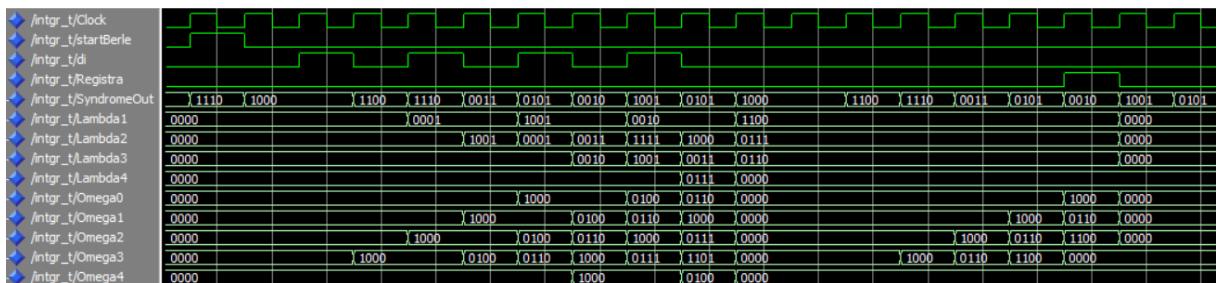
#### Módulo de Berlekamp-Massey

O módulo de Berlekamp-Massey foi implementado de forma a primeiramente calcular as localizações dos erros, ou seja, os coeficientes do polinômio localizador de erros,

representados por Lambda, e guardados em registradores, para o posterior cálculo dos valores dos erros. Os valores de erros são calculados, então, com as síndromes e localizações de erro. A arquitetura desse circuito pode ser encontrada no Apêndice, na [Figura 61](#).

As síndromes são enviadas sequencialmente para o módulo de Berlekamp-Massey a cada ciclo de *clock*. A atualização do polinômio localizador, representado por 4 registradores do diagrama, ocorre quando ele não possibilita gerar uma síndrome específica. Nesse caso, são feitos os cálculos no módulo inversor, com a atualização do polinômio no próximo ciclo. Esse processo é exemplificado pela [Figura 51](#).

Figura 51 – Simulação do módulo de Berlekamp-Massey.



Fonte: Autores.

Uma vez calculados os valores dos coeficientes do polinômio localizador, equivalentes a Lambda na simulação, é necessário calcular os coeficientes do polinômio de valor dos erros. Novamente, as síndromes são enviadas sequencialmente para o circuito. O novo cálculo ocorre com as síndromes e o polinômio localizador de erros. O resultado são os coeficientes do polinômio de valores de erro, representados por Omega na simulação, obtidos em sequência em cada ciclo de *clock*.

O componente de Berlekamp-Massey possui um circuito de controle próprio. Por ele é recebido um sinal de início de cálculo, vindo do controlador do estágio anterior, de cálculo de síndromes. Sendo assim, ele deve manipular registradores, para, por exemplo, registrar os valores de Lambda e Omega para cálculo posterior, no módulo seguinte.

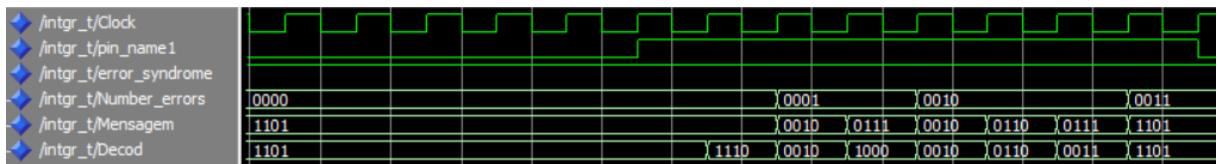
### Módulos de Busca de Chien e Forney

O módulo de busca de Chien é na verdade composto por dois módulos: busca de localização e de valores dos erros. Ambos trabalham paralelamente e têm como entrada os coeficientes  $\Lambda$  e  $\Omega$ .

Pode-se observar que ambos funcionam com cálculo incremental. A cada ciclo, o valor de registradores é atualizado com o uso das raízes do polinômio codificador e dos coeficientes já apontados. As suas saídas vão diretamente para o módulo de Forney, uma delas indicando se deve-se corrigir o símbolo de mensagem, e outra indicando qual o valor da correção, que somado ao valor da mensagem corrigiu quaisquer erros. É importante

apontar que, caso haja mais do que 4 símbolos com erros, nenhuma correção será feita, o que configura a condição para descarte do bloco recebido. Nesse circuito, é possível observar esse processo com a identificação de síndromes diferentes de zero, com o sinal `error_syndrome` na [Figura 52](#), no Apêndice. Considerando que o sinal está na posição alta, significa que pelo menos um *bit* de alguma das síndromes é diferente de 0, e portanto a mensagem deve ser corrigida.

Figura 52 – Simulação do módulo de Forney.



Fonte: Autores.

Por outro lado, caso haja mais de 4 símbolos com erro, nenhuma correção será feita, portanto contabiliza-se o número de correções feitas até então. Se for identificado um erro com as síndromes e não for contada nenhuma correção no módulo de Forney, temos duas condições suficientes para afirmar que o bloco deve ser descartado e este não pode ser corrigido através do método de Reed Solomon. No exemplo anterior, o número de erros corrigidos é inferior a 5, portanto a mensagem final está correta e pode sair da camada física PHY I. Caso a mensagem seja descartada, é necessário que se envie novamente pelo transmissor.

### 3.3.5 UART

UART, ou *Universal Asynchronous Receiver/Transmitter* é o componente utilizado para fazer comunicação entre dispositivos. A transmissão de dados é feita a partir de pacotes compostos por: *bit* de início, *data frame*, *bit* de paridade e *bit* de fim.

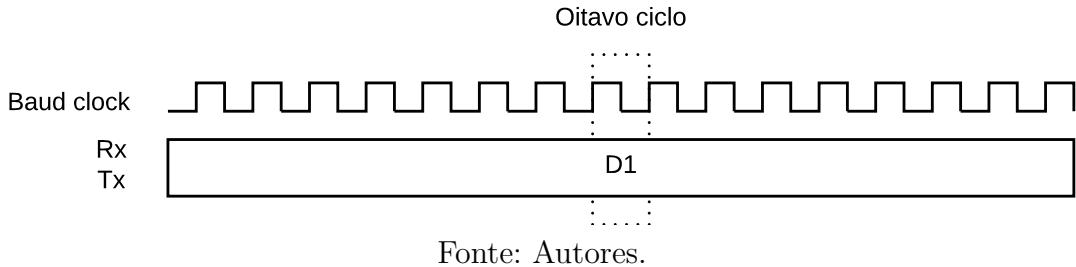
Figura 53 – Quadro de mensagem da UART utilizada no projeto.



Fonte: Autores.

Gerador de Baud: recebe um *clock* de entrada e o divide para gerar o *baud clock*. A frequência do *baud clock* é dezesseis vezes o *baud rate*. Quando a UART está recebendo o *bit* de entrada é amostrado no oitavo ciclo do baud clock para o esquema de *oversampling* dezesseis vezes.

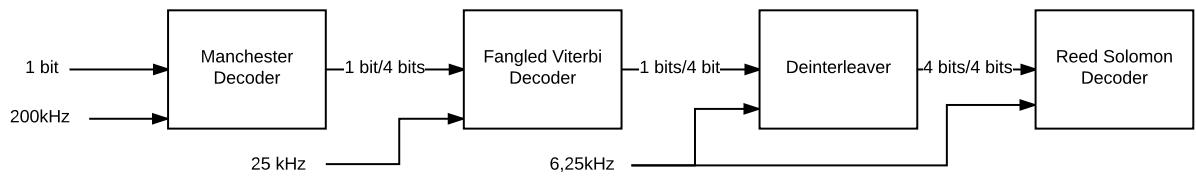
Gráfico 7 – Oversampling com baud clock.



Fonte: Autores.

### 3.3.6 Integração do Decodificador

A integração dos decodificadores foi realizada de modo análogo ao da integração dos codificadores, não existindo nenhum *buffer* entre eles. Recebe-se dados a 200kHz como especificado na norma, e os *clocks* dos decodificadores foram ajustados de modo a receber esses dados e decodificá-los. A relação de entradas e saídas de *bits* e suas frequências, é ilustrada na [Figura 54](#).

Figura 54 – Relação entre vazão de *bits* entre cada módulo do Decodificador.

Fonte: Autores.

## 3.4 Transmissor Analógico

A seção abaixo discorrerá sobre a aplicação dos métodos estudados nos capítulos anteriores para o transmissor analógico. Antes de iniciar a execução do trabalho, foram estabelecidas algumas decisões de projeto:

- Única fonte de alimentação de 5V.
- Implementação de um circuito apenas para a camada PHY I da norma, fixando a frequência de operação a 200kHz.

Essas decisões persistirão para o projeto analógico como um todo, para manter consistência no projeto.

### 3.4.1 Conversor Digital-Analógico

Com os estudos feitos na [seção 2.3.1](#), é possível especificar os requisitos reais para o transistor de potência:

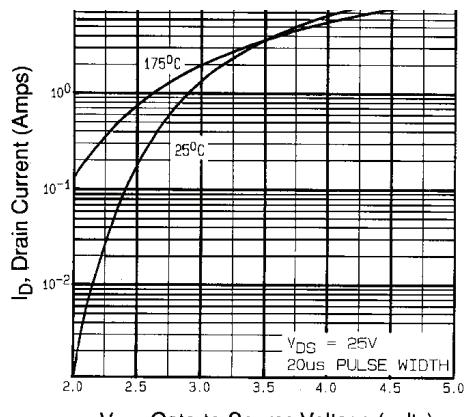
- Tensão de base/*gate* compatível com 3.3V;
- Corrente de saída compatível com LED de alta potência, no mínimo 750mA;
- Resposta de base/*gate* a  $V_{on(FPGA)}$  de no máximo 1us;

Alguns dos parâmetros são decisões de projeto, como a utilização de níveis TTL (Transistor Transistor Logic) para chaveamento do transistor (3.3V ou 5V), e outros são requisitos da norma IEEE, como frequência de operação a 200kHz. Especialmente no segundo caso, é importante escolher um transistor com  $T_{rise}$  e  $T_{fall}$  de no mínimo 10-100 vezes menor que o período da onda transmitida - no caso  $5\mu s/100 = 50ns$ . Se o transistor não chavear rápido suficiente, é possível que a onda seja alterada. Em alguns casos é possível que fique semelhante a uma onda "dente de serra".

O componente escolhido foi um MOSFET de Potência, mais especificamente o IRLZ14, pois é um transistor de nível lógico e tem o *gate* compatível com voltagens de microcontrolador.

De acordo com função de transferência da [8](#), a  $V_{GS}=3.3V$  é permitida uma corrente de dreno de 2A. A *datasheet* também especifica parâmetros de resposta dinâmica do circuito, como seu  $T_{rise}$ ,  $T_{fall}$ , que estão disponibilizados na [Tabela 11](#).

Gráfico 8 – Características de transferência do MOSFET de potência IRLZ14.



Fonte: ([VISHAY](#), 2011)

### 3.4.2 Transmissão de Luz

Para realizar a transmissão de luz a distâncias de pelo menos um metro, foi necessária a utilização de um LED de potência. Esse LED deve atender a requisitos de

Tabela 11 – Características Dinâmicas do MOSFET IRLZ14.

Parâmetro	Valor
$T_{rise}$	110 ns
$T_{fall}$	26 ns

Fonte: ([VISHAY, 2011](#))

altas frequências e resposta luminosa de acordo com seu chaveamento. Procurando satisfazer tais parâmetros, o componente escolhido foi o LUW W5-AM, fabricado pela OSRAM.

Utilizando o circuito polarizador do LED da [Figura 29](#), é necessário calcular o valor da resistência de  $R_{LIMIT}$ . O LED permite no máximo 1000mA de corrente de polarização ([OSRAM, 2015](#)), mas não é desejável trabalhar na região limite de corrente, portanto o circuito será projetado para funcionar a 500mA. Como a voltagem de operação é de 5V, utilizando a Lei de Ohm:

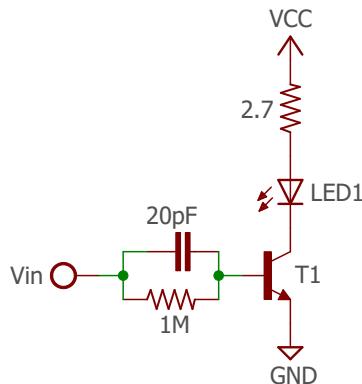
$$R_{LIMIT} = 5V \cdot 500mA = 2.5\Omega \quad (3.2)$$

### 3.4.3 Versões Anteriores

Devido à falta de conhecimento do comportamento de transistores e da resposta de todos os componentes a frequência de 200kHz, foram projetados vários circuitos que não satisfaziam o formato de onda desejado. Abaixo são exemplificadas algumas versões implementadas, juntamente com os motivos de terem sido abandonadas.

#### Filtro da Ponta de Prova

Figura 55 – Circuito de transmissão com filtro de ponta de prova.



Fonte: Autores.

O primeiro problema que o projeto do transmissor encontrou foi o comportamento de subamortecimento nos terminais do LED. Ele utiliza um circuito semelhante ao utilizado

em pontas de provas para filtrar ruídos e é conectado em série com a entrada do transmissor, como visto na [Figura 55](#).

Gráfico 9 – Comportamento do circuito de transmissão com filtro de ponta de prova em série com a entrada. A onda azul é saída do gerador de funções enquanto a onda amarela é a tensão submetida ao LED. Observa-se o comportamento de subamortecimento em ambas.



Fonte: Autores.

A saída dessa implementação é mostrada no Gráfico 9 e não foi satisfatória. O primeiro ponto a se notar é que a frequência de operação era baixa, no caso 80kHz - ainda 120kHz abaixo da especificação da velocidade da camada PHY I. Além dessa frequência, a forma de onda se distorce demais. O segundo ponto observado foi o fato de que houve uma resposta de subamortecimento tanto na entrada quanto na saída. Atribuiu-se esses efeitos a capacitâncias parasitas no LED e no transistor de potência.

O transistor de potência possui capacitância devida à separação de suas placas. Seu valor é significativo e é até especificado na *datasheet*. Para o IRLZ14, a capacitância de entrada é 400pF e de saída 170pF. Esses valores devem ser levados em conta ao polarizar tal componente.

No caso do LED, esse comportamento é mais complexo. A altas frequências, seu chaveamento cria um capacitor entre seus terminais, devido a sua arquitetura semicondutora. É possível observar esse comportamento capacitivo tanto em um LED de baixa potência quanto de alta. Na prática, a tensão sob o LED não diminui acompanhando a tensão submetida a ele, e isso causa o comportamento indesejável visto, como os picos na onda amarela.

A forma de onda gerada por esse circuito é muito boa, porém os picos de voltagem de até o dobro de  $2 \cdot V_{CC} = 10V$  com certeza danificam os componentes. Essa versão foi descartada por esse motivo (como pode-se observar em no osciloscópio - **Maximum 10.1V**).

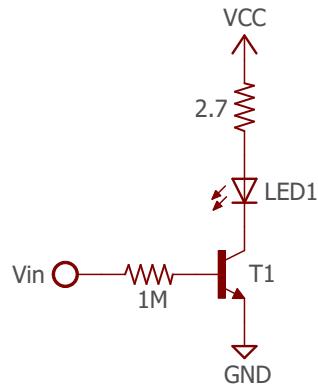
Em um teste unitário, foi observado o comportamento do chaveamento de um

resistor  $R_L$  a 200kHz, removendo completamente o LED do sistema. A saída observada era exatamente igual à entrada. Colocar um LED em série com o resistor adicionava o comportamento subamortecido, portanto conclui-se que a anomalia é atribuída ao LED.

### Aumento da Frequência

O aumento da frequência de operação a  $f_{OP} = 200\text{kHz}$  causa um subamortecimento substancialmente maior. Observa-se no Gráfico 10 que provém de circuito sem o filtro da ponta de prova na Figura 56. Nesse caso é muito mais evidente o amortecimento

Figura 56 – Circuito de transmissão sem o filtro de ponta de prova.



Fonte: Autores.

Gráfico 10 – Operação de circuito transmissor sem ponta de prova. A onda amarela representa voltagem no LED sem filtro de ponta de prova em frequências mais altas. O gerador de funções é medido e gera a forma de onda verde.



Fonte: Autores.

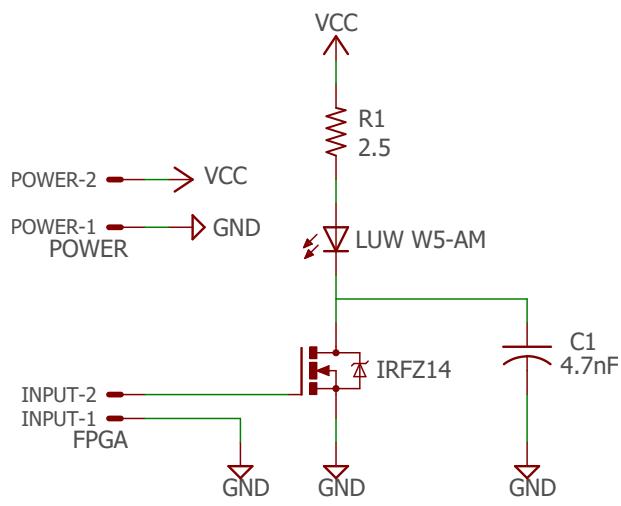
visto na onda. Atribui-se esse comportamento à componente capacitiva ao chavear o LED. No entanto, observa-se que não há *feedback* do circuito na saída do gerador de funções, fato observado na última versão do circuito. O capacitor colocado adicionava um nível de complexidade desnecessário ao circuito, pois se carregava com oscilação da entrada.

Além disso, ele não preserva a forma de onda da entrada. Ainda é possível refinar mais a solução.

### Final - Filtro Passa-Altas

Por fim, realiza-se a correção dessa componente subamortecida utilizando um capacitor entre os terminais da *source* do MOSFET e GND (observado em [Figura 57](#)).

Figura 57 – Circuito final de transmissão de dados LiCy, utilizando um filtro passa-altas.



Fonte: Autores.

Gráfico 11 – Forma de onda após adicionar um capacitor que age como filtro passa-altas. Está defasada em 180°.



Fonte: Autores

Esse capacitor atua como filtro passa-altas e remove a componente AC da saída para o LED. O comportamento não fica exatamente igual à entrada, mas é satisfatório, pois o período é muito similar, chaveando o LED corretamente. A forma de onda resultante pode ser observada no gráfico 11. Está defasada em 180°.

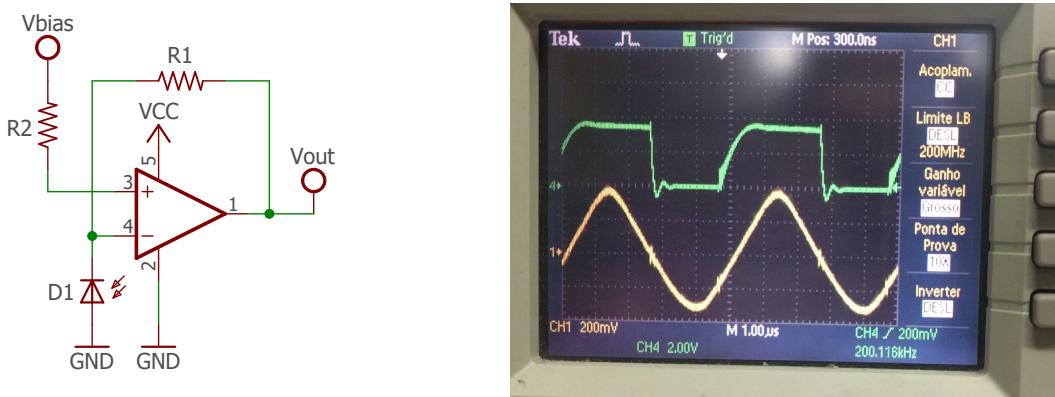
## 3.5 Receptor Analógico

A recepção será feita utilizando os conceitos apresentados nas seções 2.3.2, 2.3.2, 2.3.2, 2.3.2 e 2.3.2.

### 3.5.1 Recepção Luminosa

Para o recebimento luminoso, foi utilizado um circuito da *Application Notes* da (INSTRUMENTS, 2007), simplificado e esquematizado na Figura 58, diferente do utilizado na Metodologia (Figura 31). Com  $V_{bias} = 2.5V$  polarizando reversamente o fotodiodo, uma resposta mais confiável era recebida pelo componente. O valor escolhido de R1 e R2 foram  $1M\Omega$  e  $100k\Omega$ . O fotodiodo escolhido foi o SFH2701 (OSRAM, 2008) .

Figura 58 – Circuito e saída de um amplificador de impedâncias. No circuito, o fotodiodo é reversamente polarizado por  $V_{bias}$ . No osciloscópio, a saída está marcada em amarelo enquanto a saída do LED está em verde.



Fonte: Autores.

No osciloscópio, a componente DC da saída do amplificador de transimpedância é zero. No entanto, esse valor é apenas zero quando o padrão da oscilação luminosa se estabiliza. Mais especificamente: nesse e em todos casos de teste, o receptor está sendo submetido a uma onda quadrada de  $f = 200kHz$ . Visto que a norma define Manchester como código de RLL, é providenciado balanço DC ao receptor (são transmitidos a mesma quantidade de zeros e uns). Então, o fotodiodo se desestabilizará quando há movimentação dos módulos ou quando há mudança na iluminação do ambiente.

Para remover esses fatores de incerteza, o circuito de passas baixas com acomplamento DC  $V_{ref} = 1V$  foi projetado a partir da 5. Para obter uma voltagem de referência de 1V, foi utilizado um divisor resistivo de 12k e 47k.

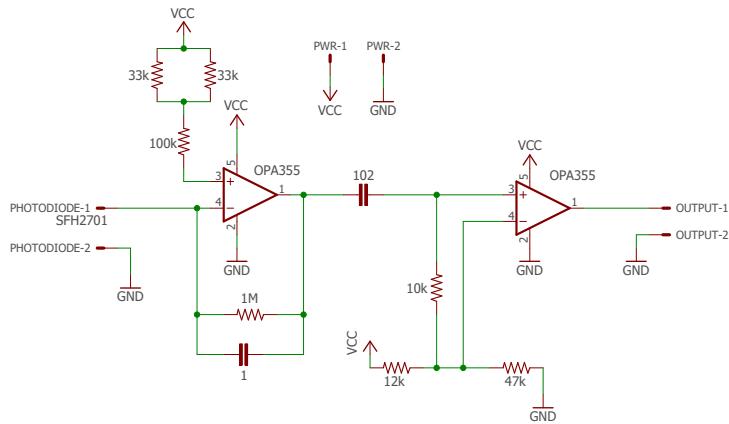
$$V_{out} = V_{in} \cdot \frac{12k}{12k + 47k} = V_{in} \cdot \frac{12k}{59k} \approx \frac{V_{in}}{5} = \frac{V_{CC}}{5} \approx 1V \quad (3.3)$$

### 3.5.2 Conversor Analógico-Digital

Com a onda em  $V_{DC} = 1V$ , é possível utilizar o comparador definido na Figura 32 para converter a onda senoidal em uma onda quadrada, com níveis compatíveis com a FPGA.

A voltagem de referência do comparador deve ser a mesma que a voltagem  $V_{ref}$  do filtro passa-baixas, provenientes do divisor resistivo. O circuito final integrado do receptor é detalhado na Figura 59.

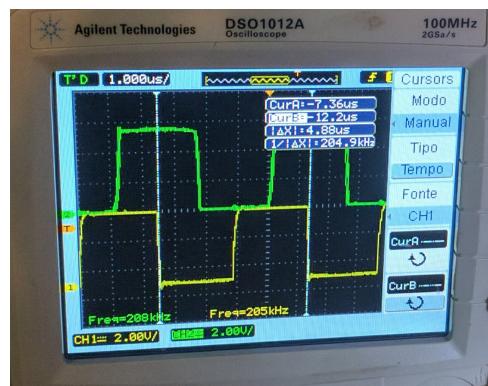
Figura 59 – Circuito esquemático final de recepção de dados LiCy.



Fonte: Autores.

Os testes realizados foram bem sucedidos e resultaram na onda do gráfico 12. Nela é possível observar a medição do período da onda resultante convertida, que foi 4.88us, muito próximo de 5us, período para 200kHz.

Gráfico 12 – Saída do transmissor em verde e saída digital convertida do receptor em amarelo. É possível observar uma defasagem de 90° em relação às ondas.



Fonte: Autores.

Em seguida será realizada a integração entre a parte analógica e digital do trabalho.

## 3.6 Integração Completa

### 3.6.1 Etapas da Integração

O processo de integração seguiu o seguinte método: primeiro testaram-se unitariamente todos os componentes digitais. Uma vez concluídos os testes unitários feitos por meio de testbenches do Quartus, existiram duas frentes de integração: codificação e decodificação. Ambas são análogas, com exceção da presença do circuito de sincronização do módulo decodificador. É importante relatar que alguns subcomponentes foram testados unitariamente na própria FPGA, a fim de garantir que não houvesse comportamentos não previstos quando da integração.

O próximo passo da integração foi gravar o decodificador e o codificador na FPGA e testá-los separadamente. Neste caso, utilizou-se a ferramenta Signal Tap, onde se conecta a memória gravada a uma interface USB, controlam-se sinais de entrada e se observam as saídas. Uma vez que os testes foram satisfatórios, a integração de hardware digital pôde se concluir. Nesta última etapa, foram primeiramente integrados codificador e decodificador dentro do próprio Quartus. Por fim, o circuito integrado foi gravado na FPGA, para mais testes, que antecederam os testes digitais finais, onde o transmissor e o receptor operaram cada um numa memória diferente, com *clocks* diferentes e um cabo como interface.

Uma vez que se obteve os dois circuitos digitais principais operantes, cada um foi integrado com seu respectivo hardware analógico. Antes disso, o receptor e o transmissor analógico já haviam sido testados, com auxílio de uma fonte de sinais digitais na entrada do transmissor e um osciloscópio na saída do receptor.

## 3.7 Testes

A realização dos testes foi feita utilizando apenas uma máquina para enviar e receber os dados, conectada a duas FPGAs não conectadas por fios. Os resultados obtidos estão listados abaixo:

- Velocidade de download: 480 bits/s;
- Distância de transmissão: 16cm;
- BER: ele não recebe bits errados, e sim perde pacotes;
- Perda de pacotes: para mensagens maiores que 10 caracteres, 30%.

A velocidade de download reduzida pode ser atribuída ao *delay* inserido para enviar os dados corretamente para a UART, diminuindo de 11.67kbits/s especificado pela norma

IEEE 802.15.7 para 4.8kbits/s. Também pode-se atribuir a essa redução ao *overhead* introduzido pelo pipeline de codificação e decodificação.

## 4 Conclusão

Os principais desafios encontrados podem ser divididos entre as duas frentes de desenvolvimento. No caso do hardware digital, podem-se apontar como desafios significativos a compreensão e implantação dos algoritmos teóricos de correção de erros em VHDL, bem como manter sincronicidade entre diferentes subcomponentes da FPGA. Uma dificuldade especialmente relevante foi integrar os componentes dentro de um pipeline, com a presença de circuitos de controle para tomar decisões sobre um circuito de fluxo de dados, já que a depuração é dificultada pelas ferramentas de desenvolvimento. Por conseguinte, pode-se ressaltar ainda que o funcionamento teórico de circuitos no Quartus não garante automaticamente o mesmo comportamento dentro de uma memória de fato. Portanto, a transição entre os dois ambientes levou um tempo significativo para depuração e ajustes. Finalmente, possíveis aperfeiçoamentos identificados incluem otimizações nos circuitos digitais para aumentar a vazão de dados efetiva, já que a própria camada física PHY I limita a taxa de transmissão para valores relativamente baixos se observadas as camadas PHY II e PHY III, o que pode igualmente limitar as aplicações finais do Li-Fi.

Por outro lado, o hardware analógico também ocasionou demandas muito expressivas dentro do projeto. Conforme o que se discutiu na sessão de execução, houve demasiados incrementos nos circuitos para lidar com eliminação de sinais parasitários, filtragem da luz ambiente e o trabalho adequado na frequência de operação indicada pela norma para a camada implementada. Com efeito, ainda existem possíveis melhorias caso este estudo seja levado à frente. Atualmente, o transmissor e receptor devem estar a até uma distância máxima para que a comunicação se estabeleça. É desejável que se aumente essa distância máxima através de uma abordagem analógica.

Apesar de ter uma taxa de comunicação reduzida, a camada PHY I mostrou-se muito eficaz para manter a integridade dos dados, sendo recomendada para aplicações onde esse fator é importante. Sendo assim, a norma IEEE 802.15.7 foi um fator decisivo para mitigar a existência de erros e possibilitar uma comunicação estável entre dois pontos, dadas certas condições como distância máxima e a ausência de obstáculos entre transmissor e receptor. Portanto, este projeto é um protótipo indicativo de que futuramente a disruptividade da comunicação por luz visível tem potencial para ultrapassar do campo teórico e tornar-se realidade em nível de manufatura e produto.

### 4.1 Trabalhos Futuros

A partir da análise anterior e dos resultados citados, podem-se determinar alguns passos para trabalhos futuros. Entre eles, faz-se necessária a integração com camadas

mais altas de comunicação, ressaltada a MAC, indicada pela norma. Uma alternativa para a implantação desta camada, que não foi discutida neste estudo, seria o uso de um microcontrolador, de modo a ser um intermediário entre a camada física e camadas mais altas, não especificadas na norma. O uso das camadas físicas mais avançadas, PHY II e PHY III também é desejável, já que elas aumentam a vazão dos dados e diminuem possíveis interferências, estendendo a gama de aplicações para a comunicação Li-Fi.

Ademais, é ainda importante para a continuidade do projeto encontrar aplicações pertinentes ao usuário do produto. Possivelmente, um uso interessante seria acoplar o módulo receptor a um celular ou dispositivo móvel, o que demandaria usar uma interface USB e desenvolver um aplicativo para realizar requisições e mostrar informações. Finalmente, num cenário futuro, seria possível conectar um celular à internet usando a tecnologia Li-Fi, portanto com o estabelecimento de bilateralidade na comunicação.

## Referências

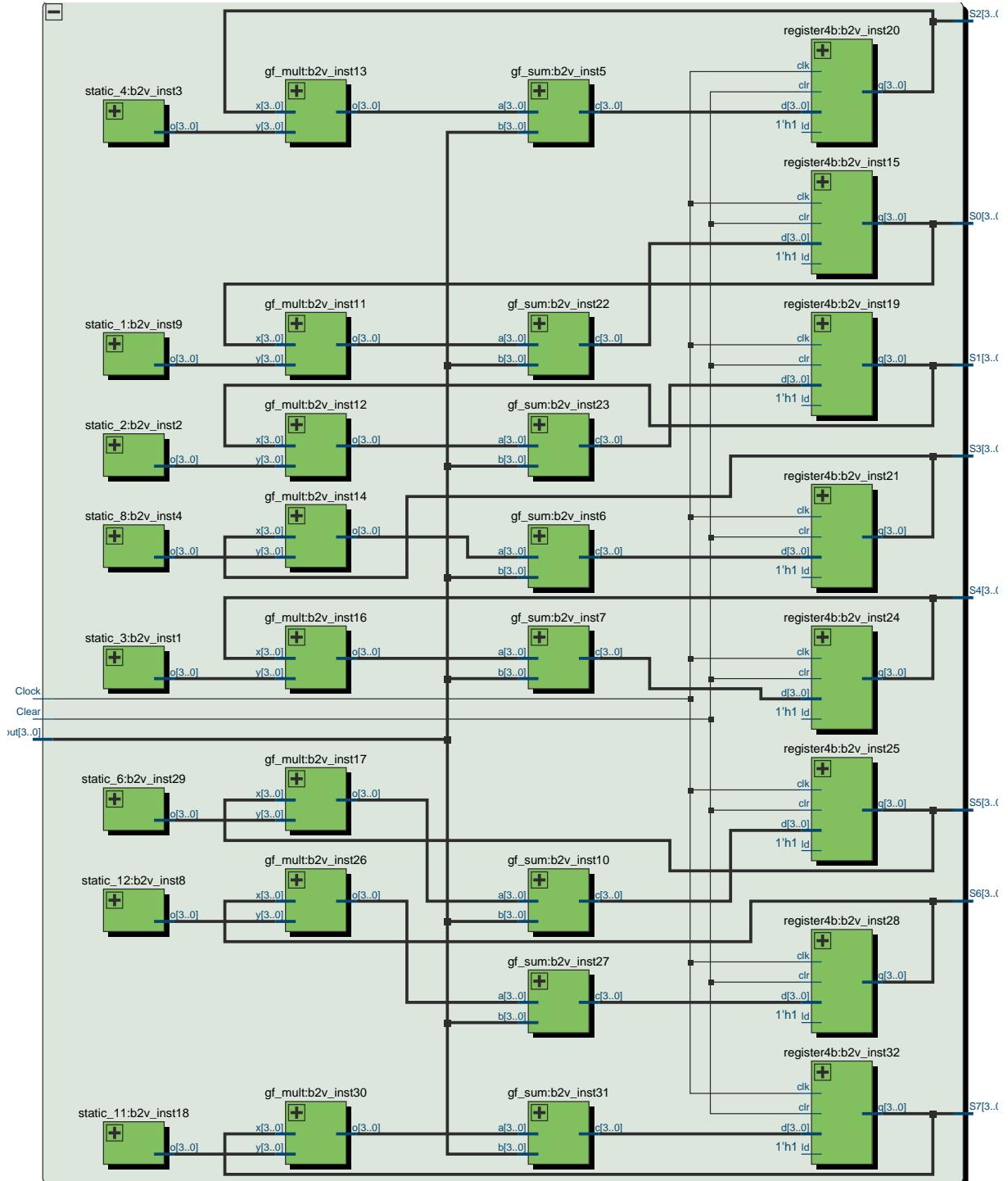
- ANATEL. Atribuição da faixa de frequências no brasil. 2006. Disponível em: <<http://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?numeroPublicacao=98580&assuntoPublicacao=Quadro%20de%20Atribui%C3%A7%C3%A3o%20&caminhoRel=null&filtro=1&documentoPath=radiofrequencia/qaff.pdf>>. Acesso em: 6 mar 2016. Citado na página 17.
- BUSER, M. I. P. A. *Vision*. London, England: MIT Press, 1992. Disponível em: <<https://books.google.com/books?id=NSZvt8Ld2-8C&pg=PA50>>. Acesso em: 6 mar 2016. Citado na página 17.
- CHANIOTAKIS; CORY. *Operational Amplifier Circuits Comparators and Positive Feedback*. 2006. Disponível em: <[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-071j-introduction-to-electronics-signals-and-measurement-spring-2006/lecture-notes/24\\_op\\_amps3.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-071j-introduction-to-electronics-signals-and-measurement-spring-2006/lecture-notes/24_op_amps3.pdf)>. Acesso em: 16 nov 2016. Citado na página 53.
- CISCO. Cisco visual networking index: Global mobile data traffic forecast update. 2016. Disponível em: <<http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf>>. Acesso em: 6 mar 2016. Citado na página 16.
- CLARKE, C. Reed-solomon error correction. *BBC Research and Development*, 2002. Disponível em: <<http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP031.pdf>>. Acesso em: 20 jul 2016. Citado na página 29.
- COSSU WAJAHAT ALI, R. C. E. C. G. 3.4 gbit/s visible optical wireless transmission based on rgb led. *Optics Express*, v. 20, 12 2012. Disponível em: <<https://www.osapublishing.org/oe/fulltext.cfm?uri=oe-20-26-B501&id=246734>>. Acesso em: 27 dez 2016. Citado 2 vezes nas páginas 20 e 21.
- ERRICSON. Ericsson mobility report june 2015. 2015. Disponível em: <<http://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf>>. Acesso em: 6 mar 2016. Citado na página 16.
- GUPTA, S. Research on li-fi technology& comparison of li-fi/wi-fi. 2015. Disponível em: <[http://www.ijarcsse.com/docs/papers/Volume\\_5/6\\_June2015/V5I6-0175.pdf](http://www.ijarcsse.com/docs/papers/Volume_5/6_June2015/V5I6-0175.pdf)>. Acesso em: 6 mar 2016. Citado na página 17.
- HAAS, H. What is lifi. *Journal of Lightwave Technology*, IEEE, v. 34, 2016. Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7360112>>. Acesso em: 22 abr 2016. Citado na página 16.
- IEEE. *802.15.7-2011 - IEEE Standard for Local and Metropolitan Area Networks–Part 15.7: Short-Range Wireless Optical Communication Using Visible Light*. [S.l.], 2011. Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6016195>>. Acesso em: 6 mar 2016. Citado na página 16.

- INSTRUMENTS, T. *Precision, High Speed Transimpedance Amplifier*. [S.l.], 2007. Disponível em: <<http://www.ti.com/lit/ds/symlink/opa380.pdf>>. Acesso em: 16 nov 2016. Citado na página 81.
- MIT. *Convolutional Encoding*. [S.l.], 2010. Disponível em: <<http://web.mit.edu/6.02/www/f2010/handouts/lectures/L8.pdf>>. Acesso em: 16 nov 2016. Citado na página 38.
- OSRAM. *High Speed PIN Photodiode*. [S.l.], 2008. Disponível em: <[http://www.osram-os.com/Graphics/XPic9/00083297\\_0.pdf](http://www.osram-os.com/Graphics/XPic9/00083297_0.pdf)>. Acesso em: 16 nov 2016. Citado na página 81.
- OSRAM. *LUW W5AM*. [S.l.], 2015. Disponível em: <[http://www.osram-os.com/Graphics/XPic6/00089171\\_0.pdf](http://www.osram-os.com/Graphics/XPic6/00089171_0.pdf)> LUW%20W5AM.pdf. Acesso em: 16 nov 2016. Citado na página 77.
- SEDRA/SMITH. *Microelectronics Circuits*. 7. ed. [S.l.]: Oxford, 2014. ISBN 0199339139. Citado 2 vezes nas páginas 47 e 50.
- SHIH, I. R. M. Vlsi design of inverse-free berlekamp-massey. *IEEE Computer and Digital Techniques*, v. 138, n. 5, Setembro 1992. Citado na página 39.
- SOLOMON, I. R. G. Polynomial codes over certain finite fields. *SIAM Journal of Applied Math*, v. 8, Junho 1960. Citado na página 29.
- VISHAY. *IRLZ14, SiHLZ14 Datasheet*. [S.l.], 2011. Disponível em: <<http://www.vishay.com/docs/91325/sihlz14.pdf>>. Acesso em: 16 nov 2016. Citado 2 vezes nas páginas 76 e 77.
- VUCIC CHRISTOPH KOTTKE, S. N. K.-D. L. J.; WALEWSKI, J. W. 513 mbit/s visible light communications link based on dmt-modulation of a white led. *JOURNAL OF LIGHTWAVE TECHNOLOGY*, v. 28, 12 2010. Disponível em: <<http://ieeexplore.ieee.org/document/5608481/>>. Acesso em: 27 dez 2016. Citado 2 vezes nas páginas 19 e 20.
- WANG XINGXING HUANG, L. T. J. S. N. C. Y. 4.5-gb/s rgb-led based wdm visible light communication system employing cap modulation and rls based adaptive equalization. *Optics Express*, v. 23, 05 2015. Disponível em: <<https://www.osapublishing.org/oe/abstract.cfm?uri=oe-23-10-13626>>. Acesso em: 27 dez 2016. Citado 2 vezes nas páginas 21 e 22.
- WANG, Y. Dynamic load balancing with handover in hybrid li-fi and wi-fi networks. *Journal of Lightwave Technology*, IEEE, Edinburgh, v. 33, 2015. Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7274270>>. Acesso em: 6 mar 2016. Citado na página 16.

# Apêndices

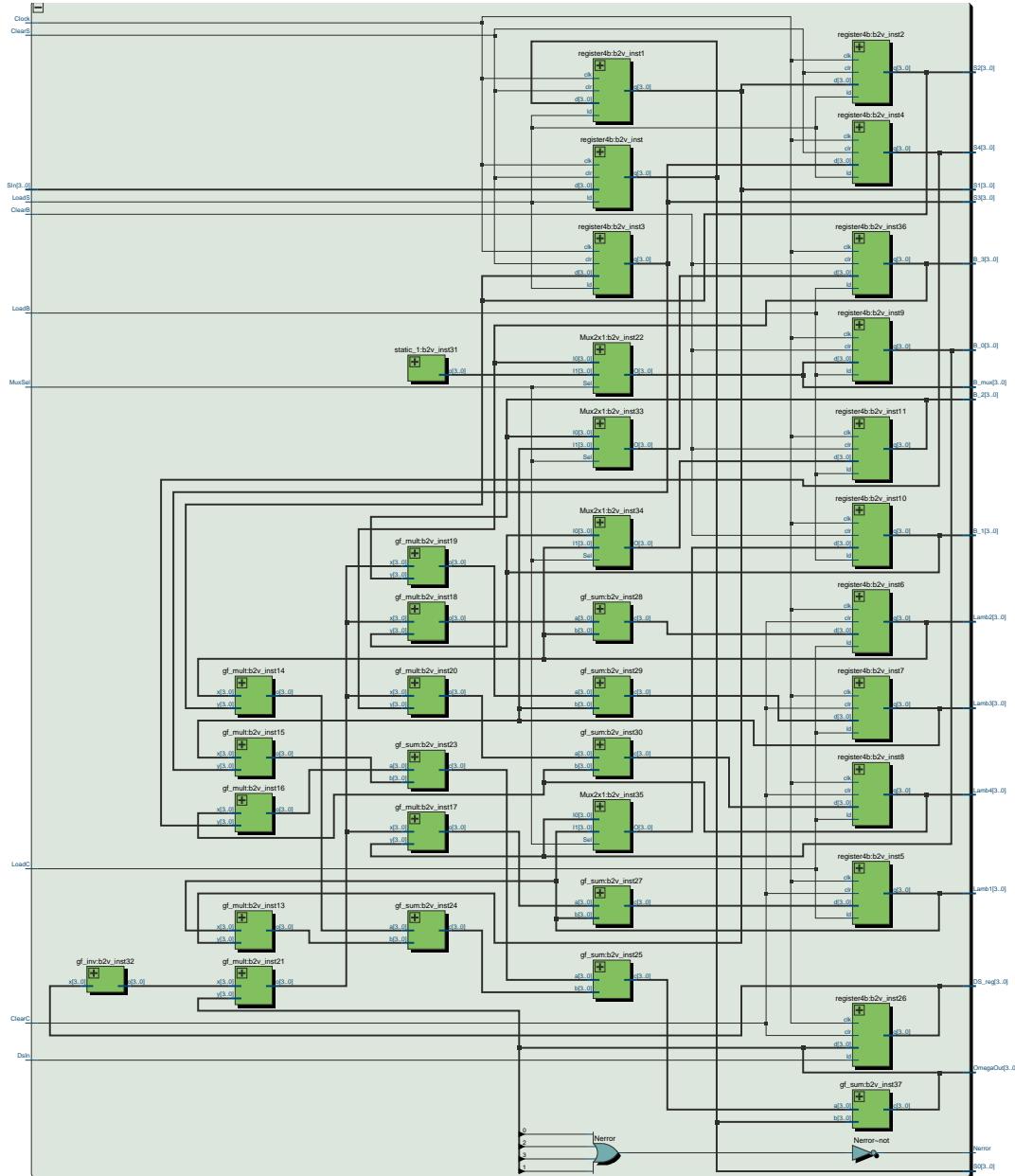
# APÊNDICE A – Diagramas da Arquitetura

Figura 60 – Arquitetura do módulo de cálculo das síndromes.



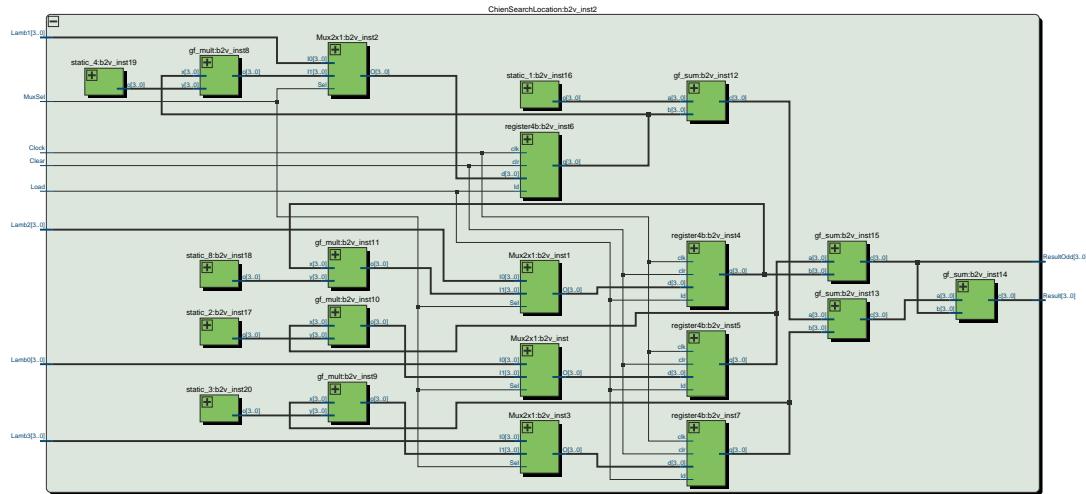
Fonte: Autores.

Figura 61 – Arquitetura do módulo de Berlekamp-Massey.



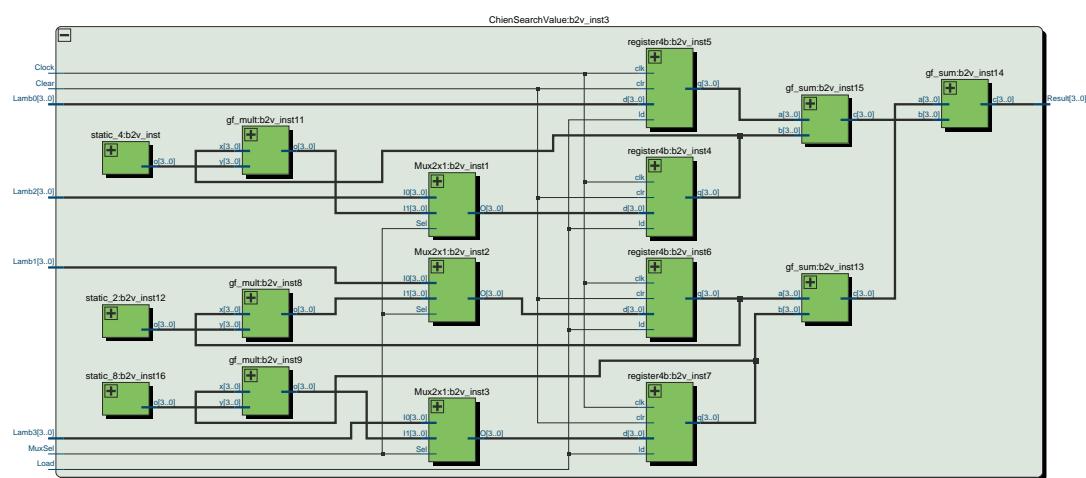
Fonte: Autores.

Figura 62 – Arquitetura do módulo de busca de Chien: localização de erros.

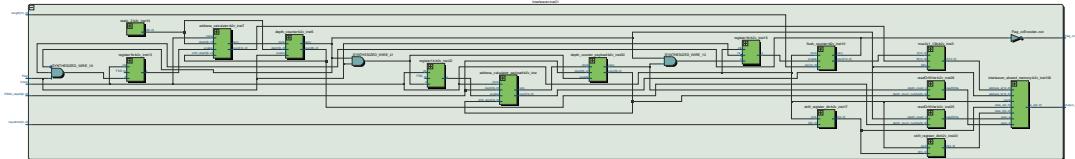


Fonte: Autores.

Figura 63 – Arquitetura do módulo de busca de Chien: valores de erros.

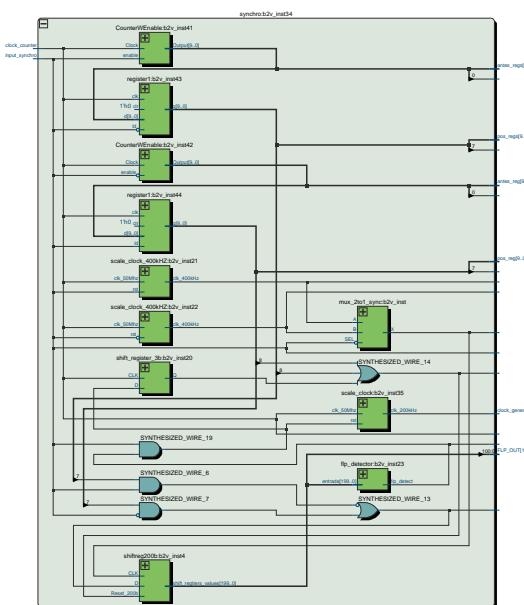


Fonte: Autores.

Figura 64 – Arquitetura do módulo de *Interleaver*.

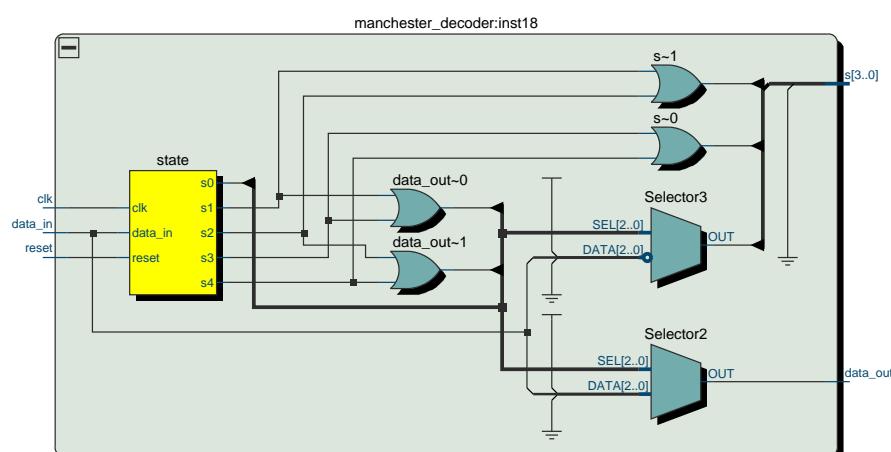
Fonte: Autores.

Figura 65 – Arquitetura do módulo de Sincronização.



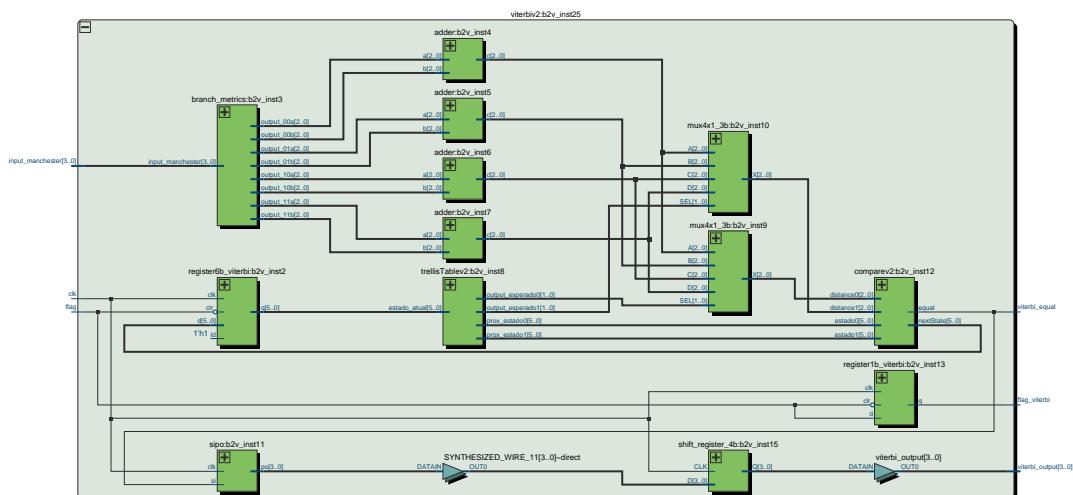
Fonte: Autores.

Figura 66 – Arquitetura do módulo de Decodificação Manchester.



Fonte: Autores.

Figura 67 – Arquitetura do módulo de Decodificação Viterbi.



Fonte: Autores.