Lucas Moura Bastos

# Estágio Integrado na Idea! Sistemas Eletrônicos

Campina Grande, Brasil

20 de dezembro de 2018

Lucas Moura Bastos

# Estágio Integrado na Idea! Sistemas Eletrônicos

Relatório de Estágio Integrado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Bacharel em Ciências no domínio da Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Área de Concentração: Microeletrônica

Orientador: Gutemberg Gonçalves dos Santos Júnior, D.Sc.

Supervisor: Jacklyn Dias Reis, PhD

Campina Grande, Brasil

20 de dezembro de 2018

Lucas Moura Bastos

# Estágio Integrado na Idea! Sistemas Eletrônicos

Relatório de Estágio Integrado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Bacharel em Ciências no domínio da Engenharia Elétrica.

Aprovado em: /12/2018

---

**Gutemberg Gonçalves dos Santos Júnior, D.Sc.**
Orientador

---

**Marcos Ricardo Alcântara Morais, D.Sc.**
Convidado

Campina Grande, Brasil
20 de dezembro de 2018

*Dedico este trabalho aos meus pais, Marcus e Cláudia, ao meu irmão, Danilo, e aos Pseudomitos.*

# Agradecimentos

Primeiramente, agradeço aos meus pais, Marcus e Cláudia, e ao meu irmão, Danilo, pelo apoio e suporte incondicional em todas as minhas escolhas.

Aos meus amigos de faculdade, em especial aos *Pseudomitos*, que desde 2012 vêm se ajudando e suprindo as dificuldades acadêmicas em grupos de estudo. Sem eles essa jornada seria, certamente, impossível.

Aos companheiros e professores do laboratório XMEN, em especial ao meu orientador Gutemberg, essenciais para a minha capacitação na área de microeletrônica.

Aos colegas da Idea!, em especial Daniel, Tomaz, Eric e Leandro, com quem trabalho diretamente e me orientam em todas as dúvidas que surgem no decorrer do projeto.

*"and this bird you cannot change",*
*Lynyrd Skynyrd*

# Resumo

Neste relatório, descreve-se as atividades que foram realizadas durante o período de estágio integrado na área de Microeletrônica da Idea! Sistemas Eletrônicos. O foco deste trabalho foi na compreensão do funcionamento de um Processador de Sinais Digitais (DSP) voltado para a aplicação em sistemas ópticos coerentes, para que a partir do conhecimento adquirido fosse possível realizar a descrição de *hardware* de blocos internos a um DSP.

**Palavras-chaves**: Idea; DSP; Sistemas Ópticos Coerentes.

# Abstract

In this report, the activites performed during my internship at Idea's Microelectronics team described. The main goal of this work was to understand the operation of a DSP used on an application to process the signal that passes through a coherent optical system. With this acquired knowledge, it was possible to describe some of the internal blocks, using hardware language, of the DSP.

**Key-words**: Idea; DSP; Coherent Optical Systems.

# Lista de ilustrações

# Lista de abreviaturas e siglas

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| API | Application Programming Interface |
| ASE | Amplified Spontaneous Emission |
| CD | Chromatic Dispersion |
| CI | Circuito Integrado |
| CMA | Constant Modulus Algorithm |
| DAC | Digital-to-Analog Converter |
| DPI | Direct Programming Interface |
| DSP | Digital Signal Processor |
| FIR | Finite Impulse Response |
| FIFO | First in, first out |
| HDL | Hardware Description Language |
| ISI | Inter Symbol Interference |
| LMS | Least Mean Square |
| LO | Local Oscillator |
| NCO | Number-Controlled Oscillator |
| PMD | Polarization Mode Dispersion |
| RTL | Register Transfer Level |

# Sumário

# 1 Introdução

Este relatório tem como objetivo descrever as atividades desenvolvidas pelo estudante Lucas Moura Bastos durante o período de estágio integrado realizado na Idea! Sistemas Eletrônicos do dia 16/08/2018 ao dia 14/12/2018 totalizando uma carga horária de 691 horas.

O estágio em questão foi realizado no grupo de Microeletrônica da Idea e ao estagiário foram atribuídas as seguintes atividades:

- Estudo da documentação do fluxo de referência da Idea! para o projeto lógico dos circuitos digitais;
- Estudo dos principais artigos da área de sistemas ópticos com detecção coerente e algoritmos de processamento de sinais relevantes para os blocos a serem desenvolvidos;
- Modelagem em alto nível para validação funcional dos algoritmos;
- Desenvolvimento dos modelos arquiteturais para validação funcional e estrutural;
- Projeto lógico dos circuitos digitais seguindo a linguagem de descrição de máquina de referência usado na empresa, i.e. SystemVerilog.

## 1.1 Idea! Sitemas Eletrônicos

A Idea!, localizada em Campinas - SP, é uma empresa de Pesquisa e Desenvolvimento (P&D) de alta tecnologia que trabalha no estado da arte nas áreas de microeletrônica e fotônica (IDEA, 2018).

A empresa vem alcançando as condições para atingir suas metas investindo em um time de engenheiros altamente capacitados, Figura 1, construindo parcerias com empresas de alto nível, Figura 2 , e investindo em salas limpas, equipamentos e ferramentas de desenvolvimento.

Seus projetos, focados na transmissão digital de informação, colaboram para suprir a crescente demanda no tráfego de dados digitais. Estes avanços estimulam o que há de mais inovador no campo de processamento de sinais.

Com as comunicações ópticas como principal motivador, a empresa desenvolve tecnologias como circuitos integrados para aplicações específicas (ASICs - *Application Specific Integrated Circuits*), lasers sintonizáveis e amplificadores ópticos.

Figura 1 – Time Idea!

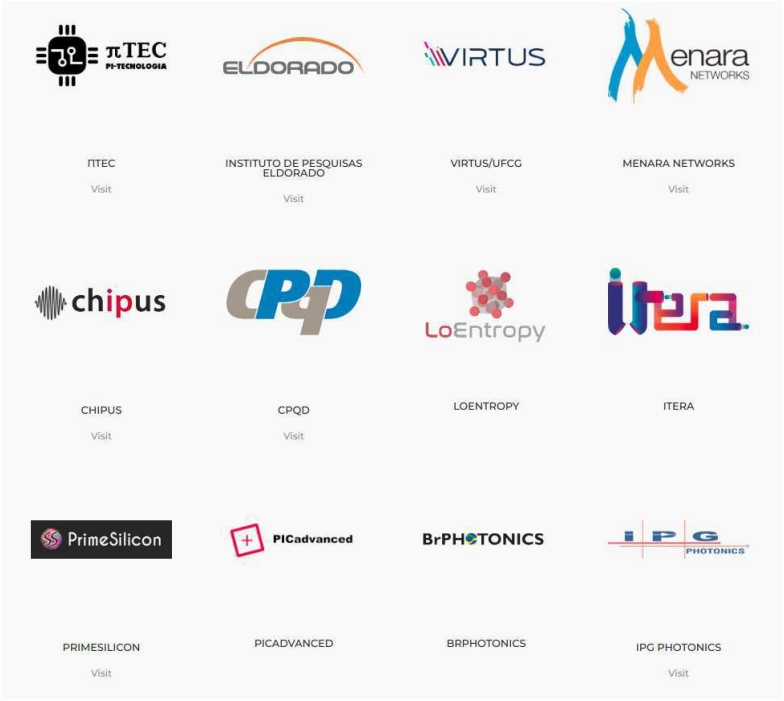


Figura 2 – Empresas parceiras da Idea!.

O mercado de comunicações por fibra óptica revolucionou a indústria de telecomunicações principalmente por proporcionar transmissões de dados a uma elevada taxa, por distâncias bem maiores e com menos perdas.

# 2 Fundamentação Teórica

Nos sistemas ópticos, a implementação de um DSP visa a recuperação do sinal transmitido e, para isso, faz-se necessária a compensação dos diversos efeitos insertos pela fibra óptica, bem como pelos demais processos necessários para a transmissão desse sinal.

Existem dois principais esquemas de detecção utilizados para converter um sinal óptico para o domínio elétrico: detecção direta e coerente. No primeiro, apenas a amplitude do sinal óptico é convertido. Enquanto no segundo, a amplitude, fase e polarização podem ser convertidas para o domínio elétrico (SEIMETZ, 2009).

A grande vantagem desse último esquema de detecção é a capacidade de detectar toda a informação do campo óptico, possibilitando formatos de modulação avançados nas quatro dimensões disponíveis na transmissão óptica, ou seja, as componentes em fase e quadratura das polarizações X e Y.

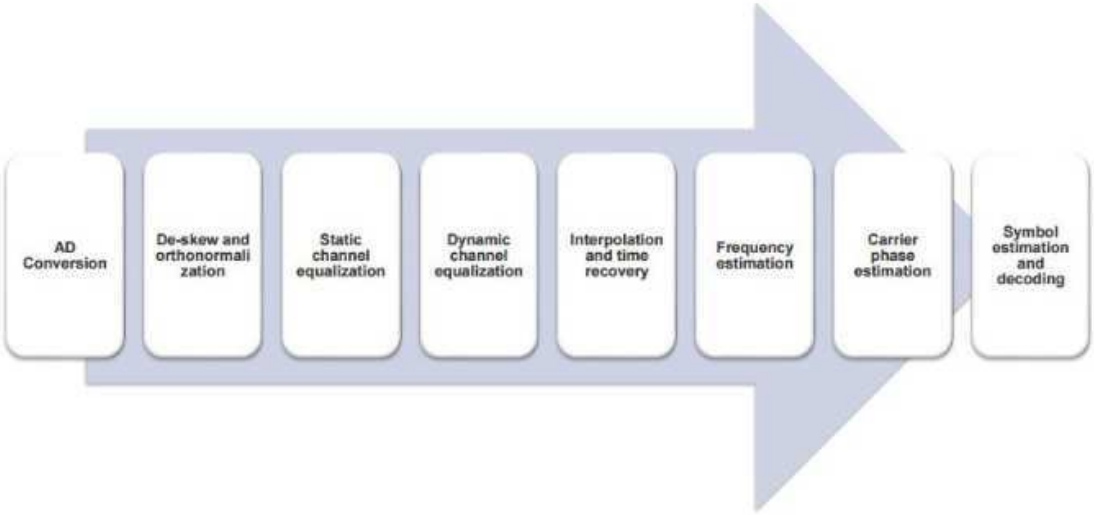


Figura 3 – Esquema de fluxo de dados do receptor de um DSP para sistemas ópticos coerentes (FORMIGA, 2017).

O sinal analógico recebido pelo DSP é convertido para o domínio elétrico (*front-end* óptico)e passa, primeiramente, por um conversor analógico-digital (ADC) responsável pela amostragem e quantização do sinal. Posteriormente, técnicas de processamento digital de sinais avançadas podem ser utilizadas para compensar as diversas distorções que afetam o sinal óptico como atenuação, ruído de amplificação óptica (ASE – Amplified Spontaneous Emission), dispersão cromática e dispersão do modo de polarização (PMD - *Polarization Mode Dispersion*).

Como ilustrado na Figura 3, um processador digital típico inclui um bloco deskew,

para compensar atrasos nos tempos de amostragem dos ADCs; um bloco de equalização estática, para compensar o atraso de grupo causado pela dispersão cromática introduzida pela fibra óptica, cujos coeficientes podem ser calculados com base em parâmetros de transmissão, como comprimento de onda da luz e taxa de amostragem do sinal; um bloco de equalização dinâmica, para compensar efeitos inseridos pelo canal de transmissão, como multiplexação dos sinais polarizados, ou seja, o sinal de uma polarização começa a se misturar com o da outra; um bloco de recuperação de desvio de frequência e de fase da portadora (KIKUCHI, 2016).

Uma representação do sinal após as principais etapas de processamento pode ser vista na Figura 4.

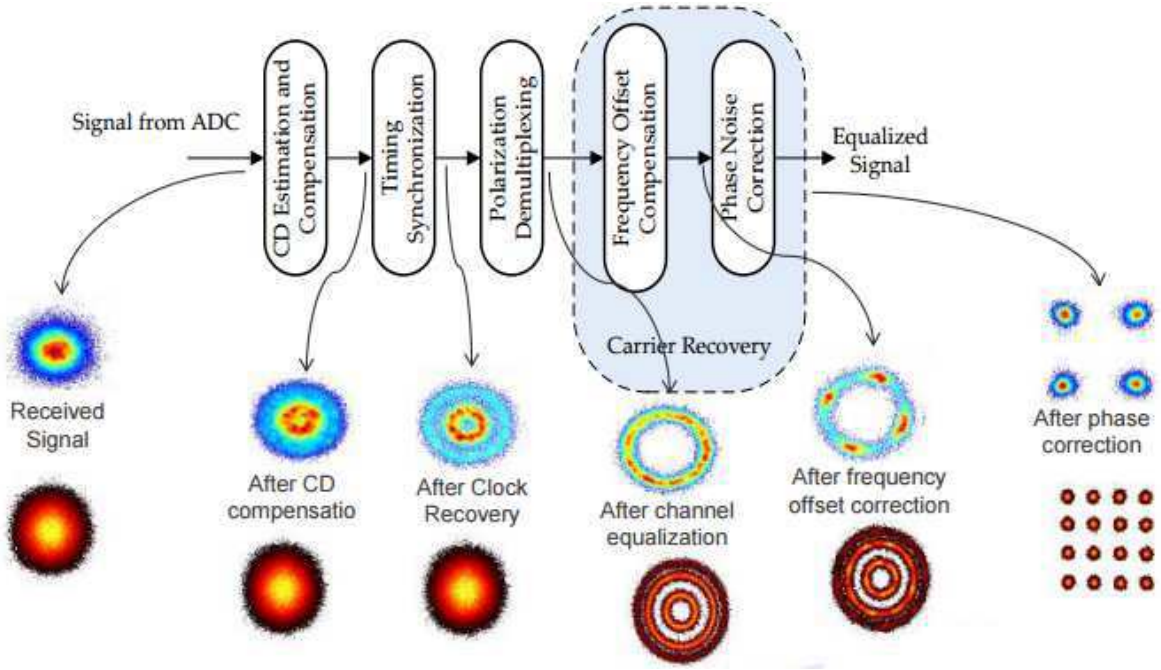


Figura 4 – Representação do processamento do sinal no DSP-RX (FORMIGA, 2017)

# 3 Atividades Desenvolvidas

Um projeto de microeletrônica pode ser dividido em duas principais etapas: *front-end* e *back-end*. O conjunto de etapas relativo à implementação em descrição de hardware, bem como sua verificação, recebe o nome de *frontend*. A implementação diz respeito à transformação de uma ideia e um conjunto de especificações para representações lógicas e físicas. Enquanto isso, a verificação é responsável por garantir a funcionalidade da implementação no decorrer do projeto.

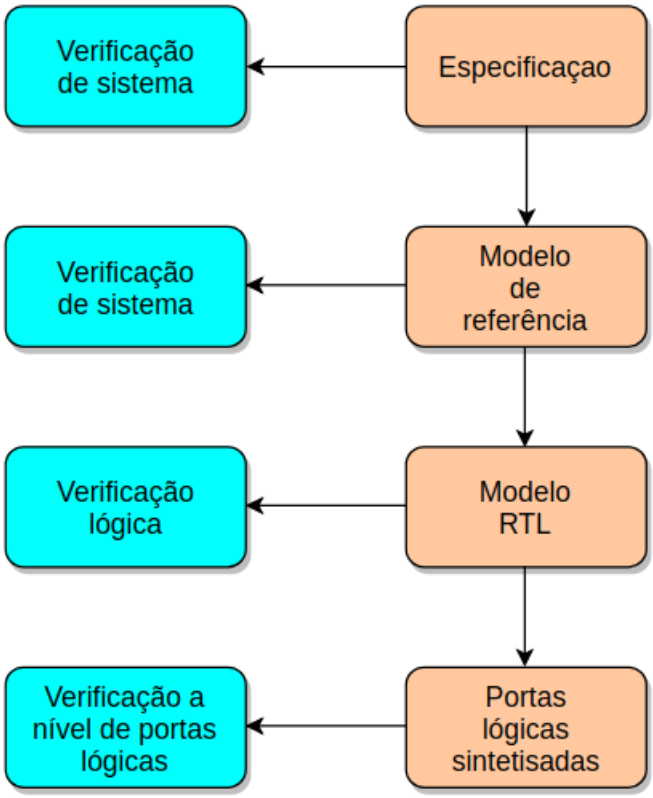


Figura 5 – Fluxo geral da etapa frontend de um projeto de microeletrônica.

A Figura 5 ilustra como dá-se o fluxo nas etapas de implementação e verificação. Ela pode ser, inicialmente, subdividida em duas para um melhor entendimento. A coluna à direita refere-se à implementação (*design*) do *hardware* e a coluna à esquerda, à verificação.

A etapa de especificação é onde acorda-se o comportamento geral e objetivos do *design* de forma textual ou gráfica. Definido isto, um estudo da microarquitetura é feito a fim de obter-se uma representação por *software* do comportamento especificado. O modelo de referência, implementado em linguagem de alto nivel, deve ser validado a nível comportamental ou sistêmico para, então, dá-se início ao processo de implementação do circuito em uma linguagem de *hardware* (HDL - *Hardware Description Language*). Uma

vez obtendo-se um modelo funcional, ou estável, em HDL, o processo de verificação a nível de RTL (*Register Transfer Level*) pode ser iniciado por meio de testbenches que comparam as saídas dos modelos.

O back-end corresponde ao design físico. Nesta etapa, as representações de circuito dos componentes são convertidas em representações geométricas (layout) de formas que, quando fabricadas nas correspondentes camadas de materiais, garantirão o funcionamento necessário dos componentes (PEIXOTO, 2018).

A fim de facilitar o fluxo do projeto como um todo, as etapas de processamento são divididas em diferentes blocos chamados de Propriedades Intelectuais (IPs – Intelectual Properties), que são implementados em linguagem de descrição de hardware (HDL – Hardware Description Language), no caso da Idea, SystemVerilog.

## 3.1 Tradução de Módulos para Python e Estudo da API Python/C

Como explicado anteriormente, no fluxo de projeto na área de microeletrônica, faz-se necessária a modelagem do projeto como um todo a fim de verificar se o projeto final atende às expectativas funcionais do que foi requerido pelo cliente. Visando um melhor aproveitamento dos recursos necessários e tempo gasto no projeto, principalmente na etapa de verificação, a primeira atividade consistiu na tradução de módulos programados em Matlab para a linguagem Python e no estudo da Interface de Programação de Aplicação (API - Application Programming Interface) que serve para instanciar módulos Python em C/C++ (PYTHON, 2018).

O Matlab é certamente uma das ferramentas de modelagem mais utilizadas. Isto deve-se ao fato de existirem diversos pacotes e *toolboxes* com inúmeros recursos matemáticos muito utilizados no processamento de sinais. De forma geral, estes pacotes disponibilizam funções para analisar, pré-processar e extrair características dos sinais.

Via de regra, um ambiente de verificação, no qual o modelo de referência é feito em Matlab, possui uma estrutura básica semelhante à ilustrada na Figura 6. Um código gerador de dados fornece as entradas necessárias ao modelo de referência, ao passo que salva estes dados em arquivos. À medida que o modelo de referência recebe estas entradas, o mesmo gera as saídas esperadas que servirão para uma posterior comparação entre referência e modelo RTL.

Ao se comparar o *Matlab* ao *Python*, as principais vantagens e desvantagens devem ser ressaltadas levando-se em conta a troca de dados entre os diferentes modelos, o consumo de memória, custo, desempenho e poder da linguagem.

Com o *Matlab* a troca de dados é feita via escrita e leitura de arquivos, que são tarefas que demandam tempo e um alto consumo de memória, piorando seu desempenho

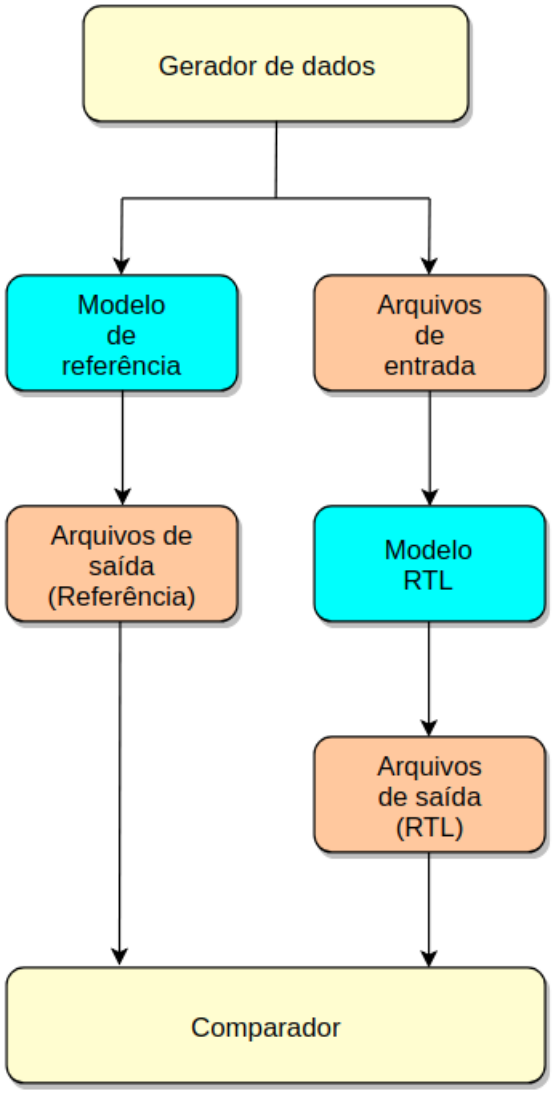


Figura 6 – Estrutura usual de um testbench.

drasticamente. Em termos de poder da linguagem, o Matlab tem um desempenho matricial muito bom, mas que depende do "estilo"de codificação para fazer uso dela e, apesar de ter um bom número de funções específicas como mencionado anteriormente, alguns componentes de propósito geral como *FIFOs* e bibliotecas de *multithreading/multiprocessing* não são de fácil implementação. Além do fato dessa ferramenta ser código fechado o que muitas vezes pode levar à necessidade do uso do *Octave* por problemas de licenciamento. *Software*, este, que tem desempenho muito pior quando comparado com o *Matlab*.

Com relação aos quesitos mencionados, a utilização do *Python* traz as vantagens de que a troca de dados entre o *SystemVerilog* e C é feita via DPI (*Direct Programming Interface*) e entre C e Python, via a API, o que torna possível a implementação de uma nova estrutura de verificação, Figura 7, e o consumo de memória muito mais eficiente. Ao mesmo tempo de que não há necessidade de licenciamento visto que o *Python* é uma

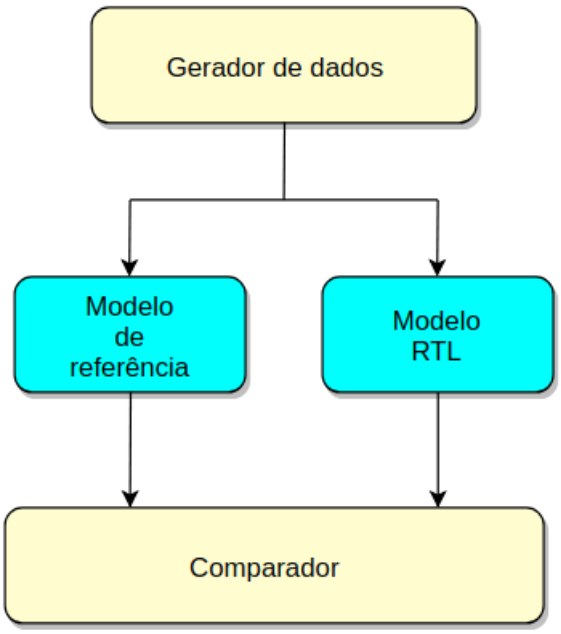


Figura 7 – Estrutura sugerida de um testbench.

linguagem de código aberto, a utilização de bibliotecas como *Numpy* possibilita uma codificação com alto desempenho e com funções essenciais para processamento de sinais análogas às *toolboxes* do *Matlab*. Além de um suporte a soluções de *software* (*third-party tools*) extremamente amplo e um bom número de componentes de propósito geral (desde *FIFOs* até bibliotecas *multithreading*/*multitasking*).

## 3.2 Equalizador Adaptativo - Atualizador de Coeficientes

As distorções causadas no canal de transmissão podem ser minimizadas utilizando-se equalizadores digitais, que têm o objetivo de diminuir a interferência intersimbólica (ISI - *Inter Symbol Interference*). A configuração de um equalizador é baseada em um filtro linear, geralmente implementado com a estrutura de um filtro com resposta ao impulso finita (FIR - *Finite Impulse Response*). Como a resposta do canal é de difícil descrição, pois varia com o tempo, o equalizador precisa ser atualizado constantemente. A isto, dá-se o nome de equalização adaptativa.

A Figura 10 ilustra o comportamento esperado de convergência dos coeficientes de um equalizador adaptativo com relação ao tempo.

Algumas técnicas de equalização adaptativa, como o mínima média quadrática (LMS - *Least Mean Square*), requerem que uma sequência de treinamento, previamente conhecida pelo receptor, seja enviada periodicamente para que o equalizador adapte a saída para o que se espera. A inclusão desses sinais na transmissão sacrifica parte da

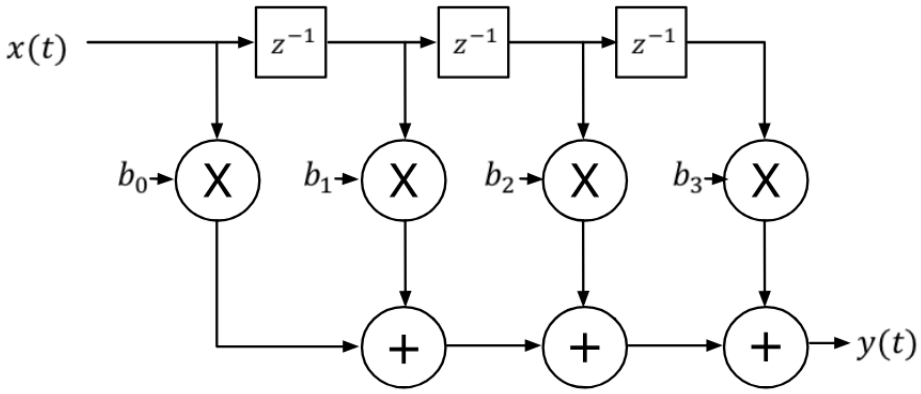


Figura 8 – Estrutura geral de um filtro FIR

capacidade do canal. Portanto torna-se preferível a utilização de técnicas que não precisam da sequência de treinamento, chamadas de adaptações cegas.

Um dos algoritmos mais estudados para essa finalidade é o do módulo constante (CMA - *Constant Modulus Algorithm*) proposto por D. Godard em 1980, que visa minimizar os desvios na magnitude do sinal equalizado em relação a um valor fixo. Para encontrar esse mínimo, o algoritmo avança pela superfície da função de custo em passos na direção do negativo do gradiente (ZHOU et al., 2010). Os coeficientes são atualizados de acordo com a fórmula 3.1 (FAN et al., 2010):

$$H[k] = H[k-1] + \mu.e_{CMA}[k].X^*[k] \tag{3.1}$$

Onde H[k] e H[k-1] correspondem aos coeficientes atuais e passados, $e_{CMA}$ é o erro calculado, $\mu$ é uma costante que define a velocidade e precisão da convergência e X*[k] é a entrada atual do módulo.

Uma vez obtendo-se um modelo RTL funcional, começa-se um estudo na tentativa de economia de potência sem que haja perda significativa de desempenho. As principais técnicas aplicadas para isso são truncamento e/ou arredondamento de *bits*, aumento do *pipeline* e aplicações de lógicas mais otimizadas. O projetista deve, inicialmente, fazer uso de gráficos e histogramas para comparar a versão otimizada com a original *full-precision*.

## 3.3 Recuperação de relógio

A sincronização de símbolos, ou recuperação de relógio, é uma das etapas mais importantes em sistemas de comunicação síncronos. O relógio do receptor deve ter sua frequência e fase constantemente ajustada para otimizar os instantes de amostragem do sinal recebido e para compensar desvios de frequência entre os osciladores locais utilizados no transmissor e receptor.

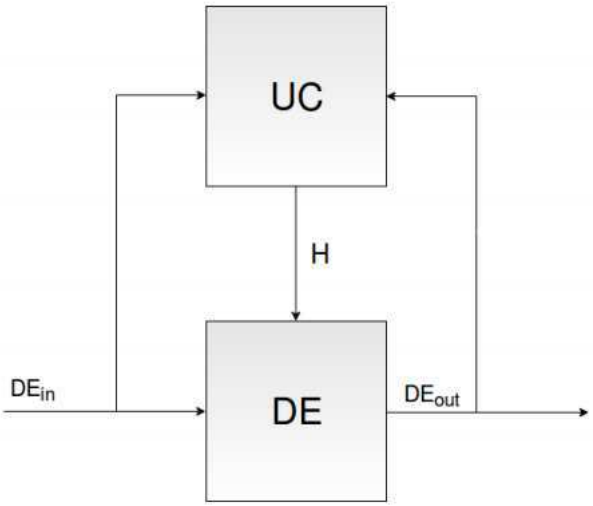


Figura 9 – Estrutura de conexão entre o equalizador dinâmico e o atualizador de coeficientes (Autoria própria).

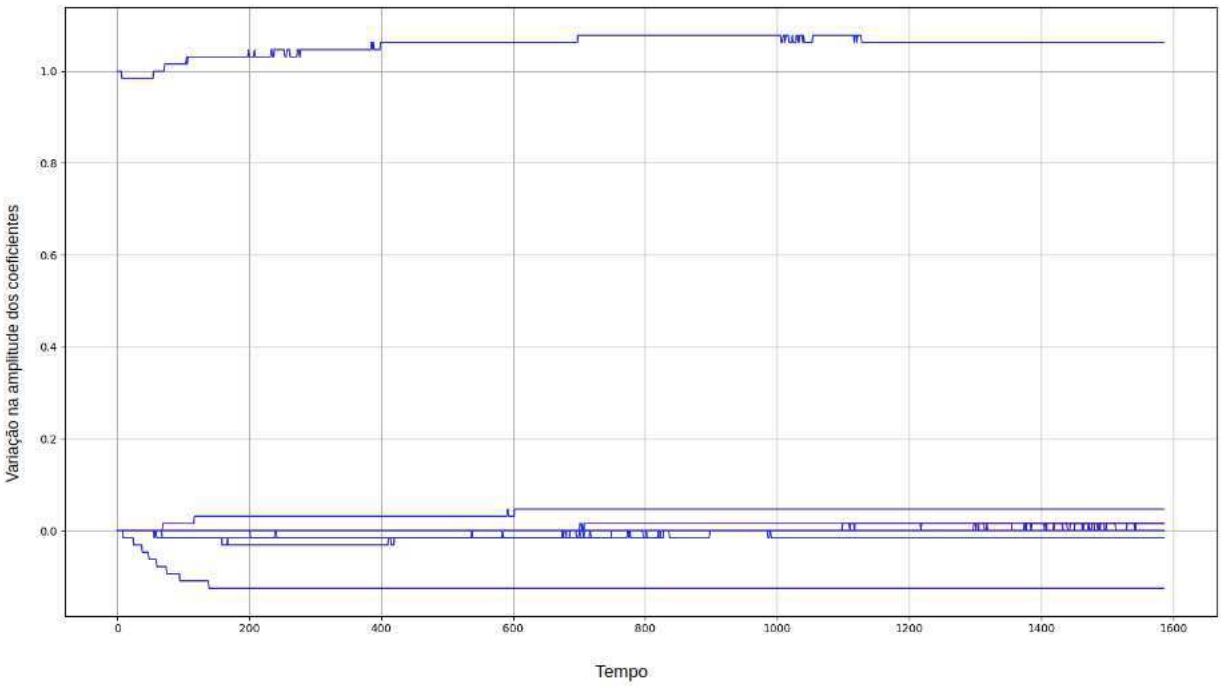


Figura 10 – Convergência dinânimca dos coeficientes do filtro com relação ao tempo.

O erro de fase surge quando a frequência do relógio do receptor é sincronizado com precisão com a do transmissor, mas com defasagem. Enquanto o erro de frequência de relógio resulta de pequenos deslocamentos de frequência entre o transmissor e receptor, geralmente caracterizados em partes por milhão (ppm).

Em alguns casos, o tempo de amostragem pode ser sincronizado com os símbolos de entrada. Em outro, o relógio é independente e a sincronização deverá ser corrigida por

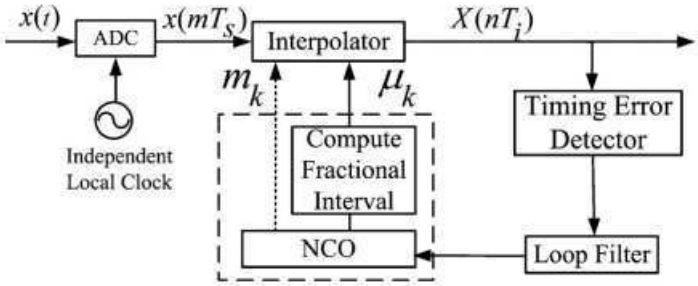


Figura 11 – Estrutura de controle do interpolador com recuperação de relógio (ZHOU et al., 2010).

meio de uma interpolação (Seção 3.4) das amostras assíncronas (Figura 12).

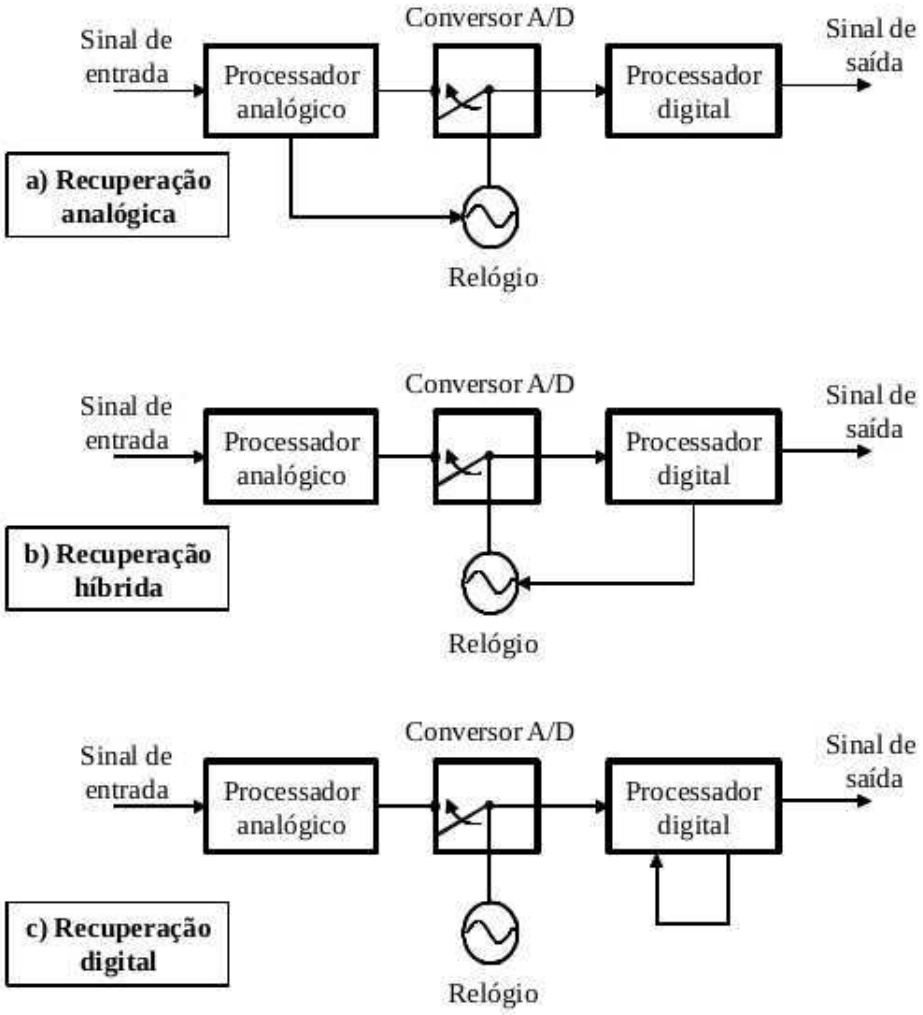


Figura 12 – Métodos de recuperação de relógio (GARDNER, 1993).

O detector de erro de temporização (TED - *Timing Error Detector*) fornece uma tendência do erro, e não o valor absoluto. Em sistemas ópticos coerentes, o algoritmo

de Gardner é muito utilizado devido à sua simplicidade computacional e imunidade a variações de fase (Equação 3.2), (FERREIRA et al., 2015).

$$e_n = y_{2k-1}.(y_{2k} - y_{2k-2}), n = 2k \tag{3.2}$$

Mesmo quando calculada a média utilizando várias amostras, o TED apresenta um comportamento ruidoso devido a, por exemplo, o *jitter*. Para contornar essa adversidade, um controlador PI é utilizado seguindo a estrutura da Figura 13.

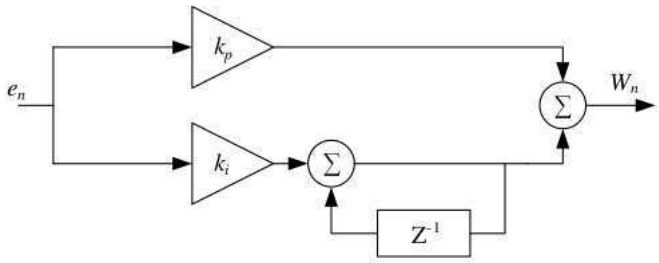


Figura 13 – Estrutura do controlador PI para recuperação de relógio (PORTELA, 2012).

O sinal $W_n$ é, então, encaminhado para um oscilador controlado numericamente (NCO - Number-Controlled Oscillator), que opera de acordo com a Equação 3.3.

$$NCO(n) = [NCO(n-1) - W(n-1)]mod1 \tag{3.3}$$

## 3.4 Interpolador FIR com correção de tempo de amostragem

No processamento digital de sinais, a interpolação se faz necessária quando blocos subsequentes utilizam taxas de amostragem diferentes. Para contornar a problemática de uma reamostragem do sinal original, existem várias estruturas de interpolação propostas na literatura, como a linear e polinomial. Uma das estruturas mais utilizadas, tendo em vista sua simplicidade, é o interpolador gaussiano baseado em um filtro FIR no qual os coeficientes podem ter seus valores estimados através de valores de cossenos levantados (OPPENHEIM; SCHAFER, 1989).

$$x_i[n] = \sum_{k=-\infty}^{\infty} x[k]h_i[n-k] \tag{3.4}$$

onde $h_i$ é o filtro passa baixa baseado no cosseno levantado. A fórmula para o filtro projetado para 6 coeficientes torna-se:

$$x_i[n] = \sum_{k=-2}^{3} x[k]h_i[n-k] \tag{3.5}$$

Em um receptor digital, o sinal recebido é amostrado por um *clock* local, que é independente do *clock* do transmissor e tem características não ideais. Há, portanto, diferença de frequência e fase entre eles, o que resulta em amostras assíncronas. Para tanto, um ajuste de tempo deve ser feito por meio de uma interpolação das amostras não sincronizadas para produzir os valores corretos. A interpolação, neste caso, é uma operação de ajuste de tempo no sinal, não no *clock*.

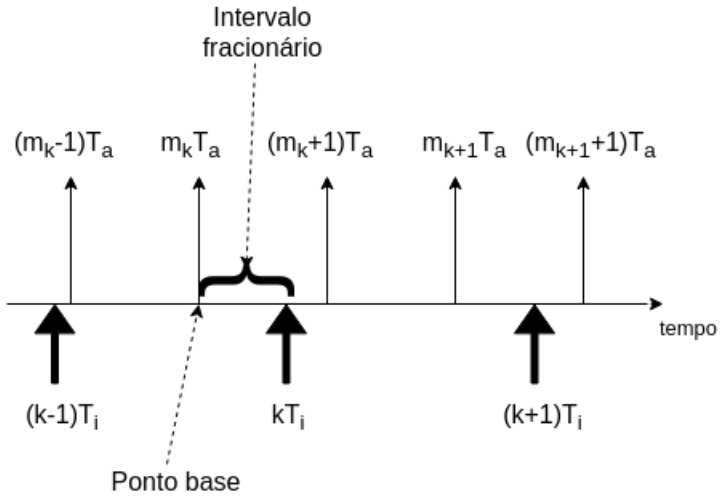


Figura 14 – Relação temporal de amostras (GARDNER, 1993).

A Figura 11 ilustra uma estrutura de um controlador para a interpolação com correção de amostragem. A interpolação é calculada utilizando-se as 6 amostras. O intervalo correto de amostras valores dos coeficientes do filtro a serem utilizadas são identificadas pela amostra definida como índice base, $m_k$, e o intervalo fracional, $\mu_k$. Este controle é provido pelo oscilador controlado numericamente (NCO - Number-Controlled Oscillator).

O resultado da interpolação com oscilador controlado numericamente pode ser visto na Figura 15.

## 3.5 Atividades Auxiliares

No decorrer do estágio, em paralelo com as atividades principais supracitadas, foram atribuídas atividades auxiliares que facilitariam posteriores codificações de novos blocos.

### 3.5.1 Remoção do *Generate for* de módulos RTL

A construção *generate* da linguagem de descrição de hardware *SystemVerilog* é uma construção poderosa no que diz respeito à codificação de módulos configuráveis.

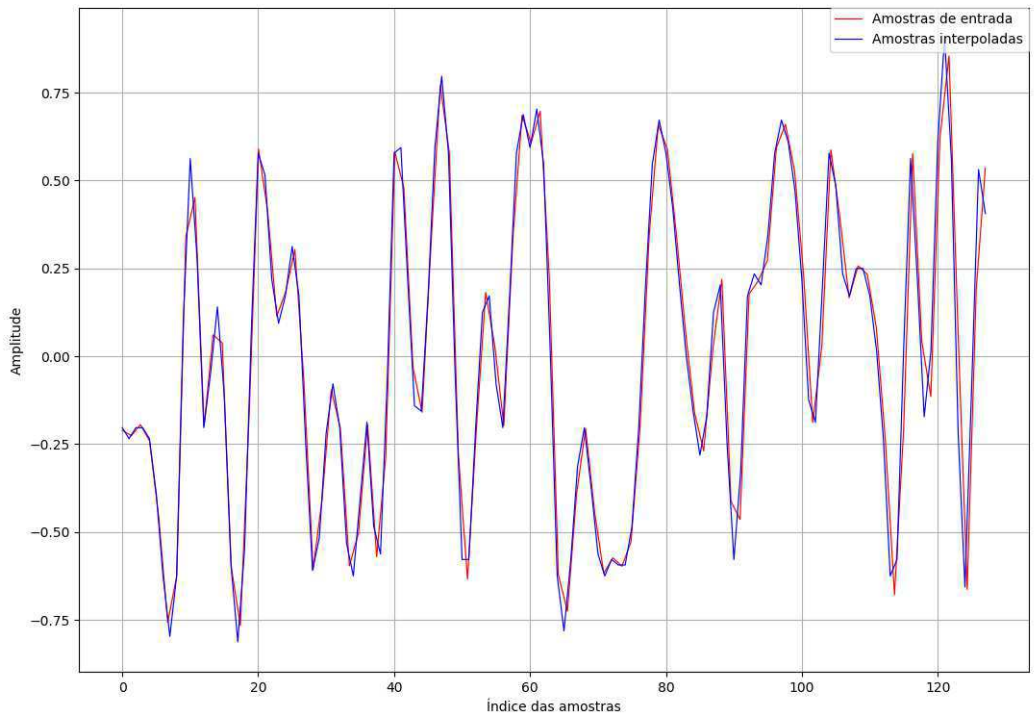


Figura 15 – Resultado da interpolação.

Existem dois tipos de construções generate. A construção em *loop* permite que um bloco seja instanciado múltiplas vezes sendo controlada por um índice do tipo genvar. A construção condicional seleciona um bloco de código entre vários. A construção condicional inclui *if-generate* e *case-generate*.

Essa construção é avaliada no momento da elaboração, que acontece após o momento de análise do HDL e antes da simulação começar. Todas as expressões dentro de um *generate* devem ser constantes, determinadas em tempo de elaboração.

O bloco *generate* cria novos níveis de hierarquia, análogo à instanciação de novos módulos. Isto pode causar uma certa confusão quando da tentativa de escrever uma referência hierarquica para sinais ou módulos dentro do bloco *generate*.

Esta "confusão hierarquica"faz com que o tempo de elaboração seja significativamente maior, dependendo da quantidade de construções utilizadas.

Com base nisso, uma das atividades consistiu na remoção de todas as construções *generate-for* de um dos módulos de equalização. O resultado foi uma redução de aproximadamente 70% no tempo de simulação.

### 3.5.2 Módulo geral de arredondamento e saturação

Como mencionado anteriormente, a estimativa de potência em se tratando de módulos RTL é uma etapa muito importante quando da fabricação de um *chip*, pois este influencia diretamente no custo do mesmo.

Para atingir o objetivo de potência estimado pelo cliente, faz-se necessário cortes de *bits* por meio de arredondamento e/ou truncamento e, muitas vezes, saturação.

Em módulos responsáveis pelo processamento digital de sinais é comum a realização de operações matemáticas entre sinais de diferentes tamanhos (*bits* relativos à porção inteira e fracionária). Para isso, há a necessidade de um certo "tratamento" em algum dos sinais para que a operação seja realizada de forma coerente.

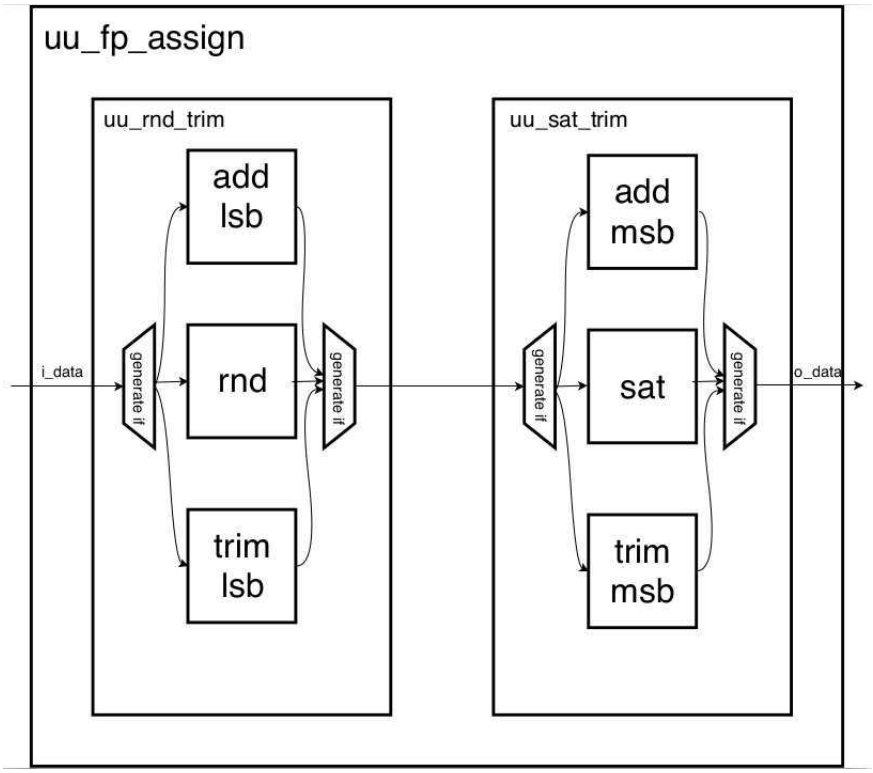


Figura 16 – Diagrama de blocos do módulo de arredondamento e saturação (Autoria própria).

Para facilitar as etapas de cortes de *bits* e preparação de sinais para a realização de operações, um módulo geral de arredondamento e saturação foi implementado a partir de dois submódulos já utilizados pela empresa (Figura 16).

Baseado nos tamanhos das entradas e parâmetros passados quando o módulo é instanciado, ele funcionará com uma ou mais das características abaixo:

- Modo *bypass*;
- Truncamento (Figura 17);

- Arredondamento (Figura 18);
- Arredondamento para infinito (Figura 19);
- Acréscimo de bit na parte fracionária - zeros concatenados;
- Saturação (Figura 20);
- Acréscimo de bit na parte inteira - se a variável for *signed*, uma extensão do sinal é feita. Caso contrário, zeros são concatenados;
- Corte de bit na parte inteira.

# 4 Considerações Finais

O estágio mostrou-se ser uma das etapas mais importantes para a formação de um engenheiro. As atividades designadas durante este período foram realizadas com êxito e também foi possível adquirir conhecimentos nas áreas de Microeletrônica, Comunicações Ópticas e Processamendo Digital de Sinais.

Com profissionais extremamente capacitados, o ambiente de trabalho da Idea! contribui para o crescimento profissional, pois a interação entre os integrantes faz com que haja sempre uma ajuda mútua com foco na elevação do nível da empresa e na entrega do melhor produto para o cliente.

Durante o período de estágio foi possível por em prática e aprofundar conceitos estudados em algumas disciplinas de graduação como Técnicas de Programação, Análise de Sinais e Sistemas, Circuitos Lógicos, Processamento Digital de Sinais, Arquitetura de Sistemas Digitais e Princípios de Comunicações.

# 5 Anexo

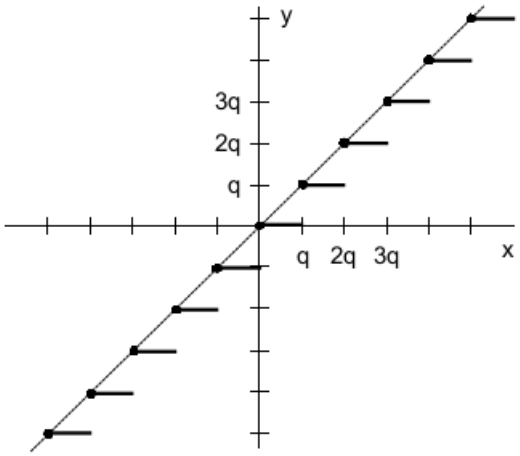


Figura 17 – Truncamento (SYSTEMC, 2003).

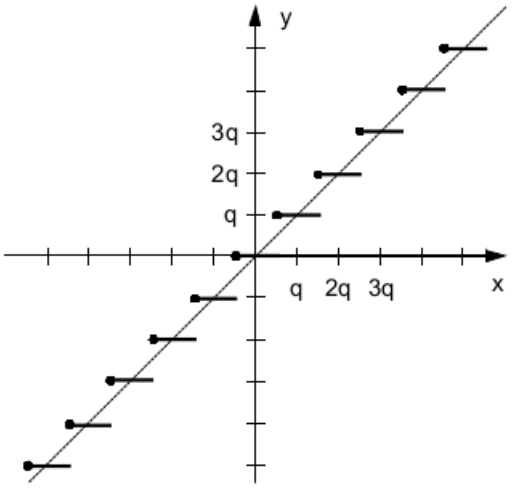


Figura 18 – Arredondamento (SYSTEMC, 2003).

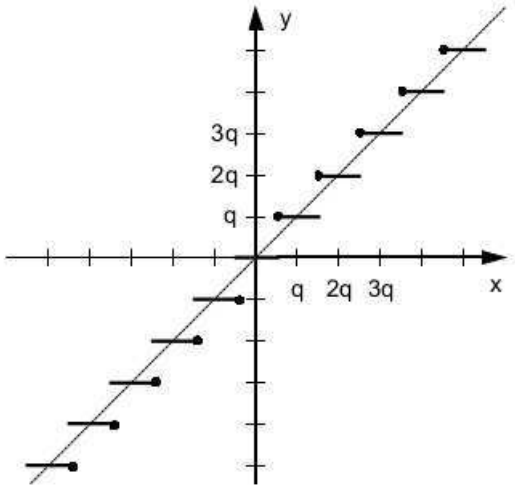


Figura 19 – Arredondamento para infinito (SYSTEMC, 2003).

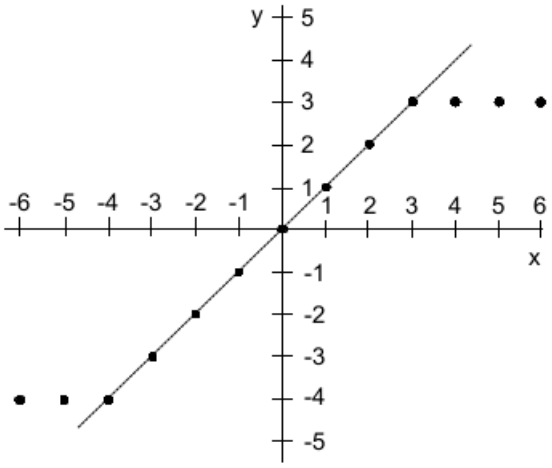


Figura 20 – Saturação (SYSTEMC, 2003).

# Referências


FAN, Y. et al. *The Comparison of CMA and LMS Equalization Algorithms in Opitcal Coherent Receivers*. IEEE, 2010. Citado na página 9.

FERREIRA, R. M. et al. *Performande and Complexity of Digital Clock Recovery for Nyquist UDWDM-PON in Real Time*. IEEE, v. 27, p. 2230–2234, 2015. Citado na página 12.

FORMIGA, D. A. *Estágio integrado na BrPhotonics*. Tese (Graduation dissertation) — Universidade Federal de Campina Grande, 2017. Citado 3 vezes nas páginas 8, 3 e 4.

GARDNER, F. M. *Interpolation in Digital Modems - Part 1: Fundamentals*. IEEE, v. 41, p. 501–507, 1993. Citado 3 vezes nas páginas 8, 11 e 13.

IDEA. 2018. Disponível em: <http://www.idea-ip.com>. Citado na página 1.

KIKUCHI, K. *Fundamentals of Coherent Optical Fiber Communications. Journal of Lightwave Technology*, v. 34, p. 157–179, 2016. Citado na página 4.

OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-Time Signal Processing*. Third edition. [S.l.]: Pearson, 1989. Citado na página 12.

PEIXOTO, J. L. *Design físico de um IP*. Tese (Graduation dissertation) — Universidade Federal de Campina Grande, 2018. Citado na página 6.

PORTELA, T. F. *Técnicas de Recuperação de Relógio para Sistemas DP-QPSK*. Tese (Master dissertation) — Universidade de Brasília, 2012. Citado 2 vezes nas páginas 8 e 12.

PYTHON. *Python/C API Reference Manual*. 2018. Disponível em: <https://docs.python.org/3.6/c-api/index.html>. Citado na página 6.

SEIMETZ, M. *High-order modulation for optical fiber transmission*. [S.l.]: Springer, 2009. v. 143. Citado na página 3.

SYSTEMC. *SystemC 2.0.1 Language Reference Manual*. [S.l.: s.n.], 2003. Citado 3 vezes nas páginas 8, 18 e 19.

ZHOU, X. et al. *All-Digital Timing Recovery And Adaptative Equalization for 112 Gbit/s POLMUX-NZR-DQPSK Optical Coherent Receivers. Journal of Optical Communication Network*, p. 984–991, 2010. Citado 3 vezes nas páginas 8, 9 e 11.