INSTITUTO FEDERAL DA PARAÍBA - IFPB
CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

ICARO MENDES DE ALCÂNTARA BEZERRA

DESENVOLVIMENTO DE UM SOFT IP CORE PARA EQUALIZAÇÃO CEGA MMA COM A METODOLOGIA MODEL-BASED DESIGN

CAMPINA GRANDE
2025

ICARO MENDES DE ALCÂNTARA BEZERRA

DESENVOLVIMENTO DE UM SOFT IP CORE PARA EQUALIZAÇÃO CEGA MMA COM A METODOLOGIA MODEL-BASED DESIGN

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Moacy P. da Silva

Coorientador: Prof. Dr. Marcos R. Alcântara Morais

CAMPINA GRANDE

2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa – CRB 15/479

B574d Bezerra, Ícaro Mendes de Alcântara
Desenvolvimento de um soft IP core para equalização cega MMA com a metodologia model-based design / Ícaro Mendes de Alcântara Bezerra. - Campina Grande, 2025.
10 f.: il.

Trabalho de Conclusão de Curso - artigo (Graduação em Bacharelado em Engenharia de Computação) - Instituto Federal da Paraíba, 2025.
Orientador: Prof. Dr. Moacy P. da Silva
Coorientador: Prof. Dr. Marcos R. Alcântara Morais

1. Engenharia de computação. 2. Comunicações digitais. 3. Multimodulus algorithm. I. Silva, Moacy P. da. II Morais, Marcos R Alcântara III. Título.

CDU 621.39

Desenvolvimento de um Soft IP Core para Equalização Cega MMA com a Metodologia Model-Based Design

Icaro M. de Alcântara Bezerra
Orientador: Prof. Dr. Moacy P. da Silva
Co-Orientador: Prof. Dr. Marcos R. Alcântara Morais
Instituto Federal de Ciência e Tecnologia da Paraíba
Curso de Engenharia de Computação - Campus Campina Grande

***Resumo*—Este trabalho apresenta o desenvolvimento de um *Intellectual Property* (IP) Core para equalização adaptativa cega, empregando o algoritmo Multimódulo (MMA), voltado para sistemas de comunicação digital baseados em 64-QAM sujeitos à Interferência Intersimbólica (ISI). A metodologia adotada foi o *Model-Based Design* (MBD), permitindo a modelagem, simulação e validação do sistema em ambiente de alto nível (Python) antes da implementação em hardware. O projeto abrange desde a análise de desempenho em ponto flutuante até a conversão para aritmética de ponto fixo e descrição de hardware (SystemVerilog). O IP Core foi verificado e sintetizado para FPGA, demonstrando uma aderência rigorosa ao modelo de referência com degradação mínima devido à quantização. Os resultados comprovam a eficácia do equalizador na recuperação de sinais 64-QAM em canais ruidosos e a eficiência da arquitetura proposta.**
[1]

***Palavras chave*—Equalização Cega, Algoritmo MMA, Model-Based Design, FPGA, IP Core, Processamento Digital de Sinais.**

***Abstract*—This work presents the development of an Intellectual Property (IP) Core for blind adaptive equalization, employing the Multimodulus Algorithm (MMA), aimed at 64-QAM based digital communication systems subject to Inter-Symbol Interference (ISI). The methodology adopted was Model-Based Design (MBD), allowing for system modeling, simulation, and validation in a high-level environment (Python) prior to hardware implementation. The project covers everything from floating-point performance analysis to fixed-point arithmetic conversion and hardware description (SystemVerilog). The IP Core was verified and synthesized for FPGA, demonstrating strict adherence to the reference model with minimal degradation due to quantization. The results confirm the effectiveness of the equalizer in recovering 64-QAM signals over noisy channels and the efficiency of the proposed architecture.**

***Index Terms*—Blind Equalization, MMA Algorithm, Model-Based Design, FPGA, IP Core, Digital Signal Processing.**

I. Introdução

A crescente demanda por maiores taxas de transmissão e maior confiabilidade em sistemas de comunicação digital, como redes Wi-Fi de nova geração e enlaces de fibra óptica, impulsiona a necessidade de técnicas avançadas de processamento de sinais. A capacidade de transmitir dados com o mínimo de erros em canais não ideais é um fator crítico para a viabilidade tecnológica e comercial desses sistemas. Um dos principais obstáculos para sistemas de comunicações é a distorção do canal, que causa a Interferência Intersimbólica (Inter-Symbolic Interference - ISI). Este fenômeno ocorre quando a energia de um símbolo "vaza" e interfere com os símbolos adjacentes, degradando a qualidade do sinal recebido e, consequentemente, aumentando a Taxa de Erro de Bit (Bit Error Rate - BER). Sem um tratamento adequado, a ISI pode inviabilizar completamente a comunicação. A solução proposta neste trabalho para mitigar os efeitos da ISI é o uso de equalizadores adaptativos. Estes são filtros digitais cujos coeficientes não são fixos, mas sim ajustados continuamente em tempo real para modelar e compensar a distorção introduzida pelo canal de comunicação. Através de algoritmos de equalização treinada como LMS (*Least Mean Squares*), RLS (*Recursive Least Squares*), ou cega como CMA (*Constant Modulus Algorithm*) e MMA (*Multi Modulus Algorithm*), o equalizador reduz as características do canal sobre o sinal, ou seja, atua para restaurar a integridade do sinal original. Apesar de sua eficácia teórica, a implementação de algoritmos adaptativos diretamente em hardware através de Hardware Description Languages - HDL é uma tarefa complexa, demorada e sujeita a erros. O ciclo tradicional de projeto em Register Transfer Level - RTL dificulta a verificação e a validação do comportamento dinâmico desses algoritmos. Para superar essa lacuna entre o modelo matemático e a implementação física, este trabalho adota a metodologia de Design Baseado em Modelo (Model-Based Design - MBD). Esta abordagem permite a modelagem, simulação e validação do sistema em um ambiente de alto nível (Python), utilizando o modelo validado como uma especificação executável para guiar e estruturar o desenvolvimento do código HDL. Nesse contexto, este trabalho se propõe a realizar o fluxo de projeto, implementação e validação de um equalizador adaptativo, utilizando o MBD como pilar metodológico para assegurar a corretude e a eficiência do design final em hardware.

II. Fundamentação Teórica

Neste capítulo, são apresentados os conceitos fundamentais que sustentam o desenvolvimento deste trabalho. Inicia-se com uma descrição dos sistemas de comunicação digital e o

[1]Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação no segundo semestre de 2025, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

problema da Interferência Intersimbólica (ISI). Em seguida, são detalhadas as técnicas de equalização adaptativa cega, com foco nos algoritmos CMA e MMA, incluindo a derivação de suas funções de custo e erro. Por fim, são abordados a metodologia de *Model-Based Design* (MBD) e os conceitos de implementação de hardware em FPGA através de um *IP Core*.

III. Sistemas de Comunicação e a Interferência Intersimbólica (ISI)

Os sistemas de comunicação digital modernos visam transmitir dados sobre canais de banda limitada. Um sistema de comunicação digital base pode ser modelado por um transmissor, um canal e um receptor [1]. No transmissor, uma sequência de bits é mapeada em símbolos complexos, como os utilizados em modulações por amplitude em quadratura (QAM).

Esses símbolos são então transmitidos através de um canal de comunicação (ex: rádio frequência, fibra ótica, etc.). Idealmente, o canal deveria ser transparente ao sinal. No entanto, canais reais possuem características que causam dispersões no sinal transmitido. Um desses fenômenos, que é conhecido como **Interferência Intersimbólica (ISI)**, faz com que um símbolo "espalhe" sua energia no tempo, interferindo com os símbolos adjacentes (anteriores e posteriores) [2]. A ISI é uma das principais fontes de distorção em sistemas de comunicação de alta velocidade e, se não for mitigada, aumenta drasticamente a taxa de erro de bits (BER) no receptor.

IV. Equalização Adaptativa e o Algoritmo LMS

Para combater os efeitos da ISI, emprega-se no receptor um filtro digital conhecido como **equalizador**. A função do equalizador é compensar a distorção introduzida pelo canal. Como as características do canal podem variar no tempo, o equalizador precisa ser **adaptativo**, ajustando seus coeficientes (vetor de pesos $\mathbf{w}(n)$) de forma iterativa para minimizar uma função de custo.

A base para muitos algoritmos adaptativos é o método do gradiente descendente, que atualiza os pesos na direção oposta ao gradiente da função de custo J:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla_{\mathbf{w}} J \tag{1}$$

Onde μ é o passo de adaptação. Em implementações práticas, como no algoritmo LMS, o gradiente exato é substituído por uma estimativa estocástica. Para sinais complexos, a atualização dos pesos do equalizador é dada por [1]:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \cdot e(n) \cdot \mathbf{x}^*(n) \tag{2}$$

Onde $e(n)$ é o sinal de erro e $\mathbf{x}^*(n)$ é o conjugado do vetor de amostras na entrada do equalizador. A principal diferença entre os algoritmos de equalização reside na forma como o sinal de erro $e(n)$ é calculado.

V. Algoritmos de Equalização Cega para Sinais QAM

A equalização cega (*Blind Equalization*) não requer uma sequência de treinamento, adaptando-se com base em propriedades estatísticas do sinal transmitido [1].

A. O Algoritmo de Módulo Constante (CMA)

O CMA [1] alcança a equalização do canal penalizando a dispersão do módulo da saída, $|y(n)|$, em relação à constante γ_C. A função de custo minimizada pelo CMA é definida como:

$$J^{\text{cma}} = \frac{1}{pq} E\left\{ (|y(n)|^p - \gamma_C^p)^q \right\} \tag{3}$$

onde p e q são inteiros positivos, geralmente escolhidos como 1 ou 2. O caso mais típico é o CMA2-2, com $p = q = 2$. A constante de dispersão γ_C^p é definida como:

$$\gamma_C^p = \frac{E\{|s(n)|^{2p}\}}{E\{|s(n)|^p\}} \tag{4}$$

Um algoritmo de ajuste do equalizador por gradiente descendente estocástico que minimiza J^{cma} é definido como:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(-\nabla_{\mathbf{w}} J^{\text{cma}}) \tag{5}$$

Para o caso em que $p = q = 2$, a equação de atualização dos pesos se torna:

$$\begin{aligned}\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(&\underbrace{y_R(n)(\gamma_C^2 - y_R^2(n) - y_I^2(n))}_{e_R^{\text{cma}}(n)} \\ &+ j\underbrace{y_I(n)(\gamma_C^2 - y_R^2(n) - y_I^2(n))}_{e_I^{\text{cma}}(n)})\mathbf{x}^*(n)\end{aligned} \tag{6}$$

B. O Algoritmo Multimódulo (MMA)

O MMA [3, 1] separa a saída do equalizador em componentes em fase $y_R(n)$ e em quadratura $y_I(n)$ e penaliza a dispersão de cada uma em torno de contornos retos separados. A função de custo minimizada pelo MMA é definida como:

$$J^{\text{mma}} = \frac{1}{2p} E\left\{ (y_R^p(n) - \gamma_M^p)^2 \right\} + jE\left\{ (y_I^p(n) - \gamma_M^p)^2 \right\} \tag{7}$$

onde p é um inteiro positivo. Um valor de $p = 2$ é tipicamente escolhido, pois oferece o melhor compromisso entre desempenho e complexidade de implementação e é o valor que será usado nesse trabalho. A constante γ_M^p é definida como:

$$\gamma_M^p = \frac{E\{a^{2p}(n) + b^{2p}(n)\}}{E\{|a(n)|^p + |b(n)|^p\}} = \frac{E\{a^{2p}(n)\}}{E\{|a(n)|^p\}} \tag{8}$$

O valor de γ_M^p para a modulação 64QAM é de aproximadamente 0.880952381, que para fins de simplificação dos cálculos em ponto fixo fora arrendodado para 0.9. A igualdade à direita é válida no caso em que os símbolos $a(n)$ e $b(n)$ têm as mesmas estatísticas. Um algoritmo de gradiente descendente estocástico que minimiza J^{mma} é definido como:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(-\nabla_{\mathbf{w}} J^{\text{mma}}) \tag{9}$$

Para o caso em que $p = 2$, a equação de atualização se torna:

$$\begin{aligned}\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(&\underbrace{y_R(n)(\gamma_M^2 - y_R^2(n))}_{e_R^{\text{mma}}(n)} \\ &+ j\underbrace{y_I(n)(\gamma_M^2 - y_I^2(n))}_{e_I^{\text{mma}}(n)})\mathbf{x}^*(n)\end{aligned} \tag{10}$$

onde $e^{\text{mma}}(n) = e_R^{\text{mma}}(n) + je_I^{\text{mma}}(n)$ é o sinal de erro do MMA.

A escolha pelo algoritmo MMA justifica-se pela sua adequação superior à geometria de sinais QAM em comparação ao CMA, a função de custo do MMA minimiza a dispersão das componentes real e imaginária de forma independente, criando contornos de erro retangulares que se alinham à geometria quadrada da constelação 64-QAM, isso permite que o algoritmo corrija automaticamente a rotação de fase e reduza o erro residual, superando a abordagem "circular"do CMA que é cega à fase que possui a necessidade de um rotador externo em regime permanente. Além disso, simulações demonstram que o MMA atinge um Erro Quadrático Médio (Mean Sqaured Error - MSE) inferior ao CMA para modulações de alta ordem, como 16-QAM e 64-QAM, devido ao alinhamento de sua função de custo com a estrutura quadrada da constelação [2] [1].

VI. Model-Based Design (MBD)

O *Model-Based Design* (MBD) é uma metodologia de desenvolvimento de sistemas em que um modelo matemático e visual do sistema é o elemento central durante todo o processo de projeto, desde a especificação até a implementação e o teste [1].

O fluxo de trabalho típico em MBD inclui as seguintes etapas:

1) **Modelagem:** O sistema é construído e descrito em um ambiente de alto nível de abstração, como Python.
2) **Simulação e Análise:** O comportamento do sistema é simulado sob condições que simulam um uso real para verificar a funcionalidade e otimizar o desempenho do algoritmo.
3) **Conversão para Ponto Fixo:** O modelo, geralmente desenvolvido em ponto flutuante, é convertido para uma representação em ponto fixo, essencial para a implementação em hardware.
4) **Desenvolvimento do Código RTL:** A partir do modelo validado, o código do design RTL (SystemVerilog) que descreve o circuito digital é gerado respeitando as limitações de etapas sequenciais e aproveitando a possibilidade de paralelismo das operações.
5) **Verificação e Validação:** O hardware gerado é verificado em relação ao modelo original para garantir a equivalência funcional.
6) **Síntese:** Após a sua validação é feita a síntese do design RTL para a plataforma escolhida.

O MBD é particularmente vantajoso para o projeto de sistemas complexos de DSP, pois permite a exploração rápida de arquiteturas e a detecção de erros em fases iniciais do projeto, reduzindo significativamente o tempo de desenvolvimento e os riscos associados à tradução manual do algoritmo para o hardware [1].

VII. Implementação em Hardware: FPGA e IP Core

A implementação física de algoritmos de DSP, como o equalizador adaptativo, pode ser realizada em ASICs (*Application-Specific Integrated Circuits*) ou FPGAs (*Field-Programmable Gate Arrays*). Os FPGAs oferecem um meio-termo entre o desempenho do hardware dedicado e a flexibilidade do software, permitindo o remapeamento de circuitos lógicos para se adequar a novas funcionalidades ou correções.

Um **IP Core** (*Intellectual Property Core*) é um bloco de lógica reutilizável, pré-projetado e pré-verificado, que desempenha uma função específica. A criação de um equalizador na forma de um IP Core permite que ele seja facilmente integrado em projetos maiores de *System-on-Chip* (SoC), como por exemplo um demodulador QAM completo para um sistema de comunicação sem fio [1]. No contexto deste trabalho, o equalizador desenvolvido é classificado especificamente como um **Soft IP**. Esta classificação justifica-se pois o módulo é entregue na forma de descrição de hardware sintetizável (RTL em SystemVerilog), sem estar atrelado a um layout físico ou processo de fabricação específico. Diferente de um *Hard IP*, que possui uma implementação fixa em silício, a abordagem de *Soft IP* confere ao projeto flexibilidade para ser mapeado nos recursos lógicos genéricos (LUTs, Flip-Flops e DSP Slices) de diversas famílias de FPGAs ou ASICs.

O objetivo deste projeto de IP Core é fornecer uma solução parametrizável e robusta. Para isso, o uso de Model-Based Design reduz riscos ao permitir a validação matemática e a definição de precisão (ponto fixo) em alto nível antes da codificação, evitando que erros conceituais avancem para a fase de hardware. Além disso, ele fornece uma referência executável fidedigna (Golden Model) para a geração automática de vetores de teste, garantindo que a implementação RTL corresponda exatamente à especificação funcional e matemática.

VIII. Desenvolvimento

Este capítulo descreve a metodologia e a implementação prática do sistema de comunicação e do equalizador cego MMA. A abordagem segue os princípios de MBD, onde um modelo de simulação é construído e validado em um ambiente de software de alto nível. Primeiramente, são detalhadas as ferramentas utilizadas. Em seguida, a arquitetura do sistema é apresentada em blocos funcionais. Por fim, a implementação de cada bloco em Python é descrita, juntamente com a configuração dos experimentos realizados para avaliar o desempenho do equalizador.

IX. Ambiente de Simulação e Ferramentas

O desenvolvimento, a simulação e a análise do sistema foram realizados integralmente na linguagem de programação Python, versão 3.12. A escolha desta plataforma se deu pela sua flexibilidade, vasta gama de bibliotecas científicas e facilidade para prototipagem rápida de algoritmos complexos de Processamento Digital de Sinais (DSP). As principais bibliotecas utilizadas foram:

- **NumPy:** Utilizada como base para toda a computação numérica, incluindo a criação e manipulação de sinais como vetores de números complexos, implementação de filtros FIR através de produtos escalares e a geração de ruído gaussiano.
- **Matplotlib:** Empregada para a visualização gráfica dos resultados, sendo fundamental para a geração dos diagramas de constelação e das curvas de convergência do MSE, que são as métricas de desempenho analisadas neste trabalho.

A implementação do algoritmo de equalização seguiu as formulações matemáticas apresentadas na Fundamentação Teórica, especificamente a equação de atualização de pesos base-

ada no erro do MMA, conforme detalhado nas equações (10) (8).

X. Arquitetura do Sistema Baseada em Modelos (MBD)

Seguindo a metodologia MBD, o sistema de comunicação foi modelado como uma cascata de blocos funcionais onde cada bloco representa uma etapa do processo de transmissão e recepção, permitindo a simulação isolada e integrada de cada componente.

A arquitetura é composta pelos seguintes módulos principais:

1) **Fonte de Sinais (64-QAM):** Este bloco é responsável por gerar a sequência de símbolos a ser transmitida. Ele cria uma série de bits aleatórios e os mapeia para os pontos correspondentes da constelação 64-QAM.
2) **Modelo do Canal:** Simula os efeitos do meio de transmissão. Neste trabalho, um canal LTI (Linear e Invariante no Tempo) foi implementado para introduzir a Interferência Intersimbólica (ISI) e o Ruído Branco Aditivo Gaussiano (AWGN). A ISI é modelada matematicamente através da resposta ao impulso discreta do canal, representada por um vetor de 6 coeficientes (*taps*). A operação de convolução entre os símbolos transmitidos e estes coeficientes simula o espalhamento temporal característico de canais com multipercurso, fazendo com que réplicas atrasadas dos símbolos anteriores interfiram no símbolo atual.
3) **ACG (Automatic Gain Control):** Responsável por ajustar a potência do sinal recebido na entrada do equalizador. Como o canal introduz atenuação e ruído, o nível de potência do sinal pode variar significativamente. O AGC normaliza a potência média do sinal em blocos, garantindo que o equalizador opere dentro da sua faixa dinâmica ideal e prevenindo instabilidades na adaptação dos coeficientes.
4) **Equalizador Cego MMA:** O núcleo deste projeto. É um filtro FIR adaptativo que processa o sinal recebido e distorcido, ajustando seus coeficientes iterativamente para minimizar a ISI, sem o uso de uma sequência de treinamento.
5) **Módulo de Análise de Desempenho:** Coleta os dados da simulação para calcular e visualizar as métricas de desempenho, permitindo a validação do equalizador.

XI. Implementação do Módulo de Equalização em Python

Cada bloco da arquitetura foi implementado como uma função ou classe em Python, garantindo modularidade e reutilização.

A. Fonte do Sinal 64-QAM

O equalizador MMA foi implementado na função `mma_equalizer`, que encapsula toda a lógica de filtragem e adaptação. A inicialização dos pesos do filtro FIR foi realizada com a técnica *center-spike*, onde o coeficiente central real é inicializado com o valor 1 e os demais com 0. Este método é comum em equalização cega, pois garante uma resposta inicial que se aproxima de um impulso. A ordem do filtro (N) foi determinada experimentalmente, seguindo a metodologia de saturação de desempenho[1]. A análise iterativa demonstrou que incrementos acima de 16 coeficientes não resultavam em redução significativa do MSE em regime permanente. Dessa forma, fixou-se $N = 18$, tal como no trabalho apresentado por *Banovic*[1] para garantir a convergência robusta sem sobredimensionar o sistema. Essa escolha otimiza a relação custo-benefício, evitando o consumo desnecessário de recursos de hardware (multiplicadores, somadores e memória) que não traria ganhos práticos de equalização.

O processo de adaptação ocorre em um loop iterativo, onde, para cada amostra recebida, os seguintes passos são executados:

1) O vetor de amostras de entrada do filtro (regressor) é atualizado.
2) A saída do equalizador, $y(n)$, é calculada através do produto escalar entre o vetor de pesos atual, $\mathbf{w}(n)$, e o regressor.
3) O sinal de erro complexo, $e^{\text{mma}}(n)$, é calculado utilizando a equação do MMA.
4) O vetor de pesos é atualizado para a próxima iteração, $\mathbf{w}(n+1)$, de acordo com a regra de atualização do gradiente descendente estocástico, utilizando o erro calculado, o passo de adaptação μ e o regressor.

XII. Métricas de Desempenho, Configuração dos Experimentos e Resultados

Para validar o funcionamento do equalizador e analisar seu desempenho, foram utilizadas duas métricas principais, uma qualitativa e outra quantitativa, sob diferentes condições de canal.

A. Métricas de Avaliação

- **Diagrama de Constelação:** Utilizado como métrica visual para inspecionar a qualidade do sinal na saída do equalizador. A redução da ISI e do ruído é evidenciada pela formação de agrupamentos (clusters) de pontos nítidos em torno das posições originais da constelação 64-QAM.
- **Curva de MSE vs. Iterações:** Uma métrica quantitativa que plota o Erro Quadrático Médio (MSE), em dB, ao longo das iterações do algoritmo. Esta curva permite analisar a velocidade de convergência e o erro residual em regime permanente, indicando a precisão do equalizador.

B. Configuração dos Experimentos

O modelo completo foi simulado sob diferentes condições de canal para avaliar a robustez do algoritmo MMA. Os experimentos foram configurados com os seguintes parâmetros:

- **Relação Sinal-Ruído (SNR):** Foram testados os valores de 45, 40, 35 e 30 dB.
- **Comprimento do Sinal:** 50.000 amostras para cada simulação.
- **Parâmetros do Equalizador:** 18 coeficientes complexos (taps) e um passo de adaptação $\mu = 2^{-10}$.
- **Número de Simulações:** Para cada nível de ruído, foram realizadas 150 simulações independentes para obter uma média estatisticamente relevante do MSE.

C. Resultados e Análises

Os resultados obtidos a partir das simulações são apresentados a seguir, focando nos cenários mais representativos para validar o desempenho do equalizador MMA. A análise combina a inspeção visual dos diagramas de constelação com a análise quantitativa da curva de convergência do MSE.

c.1) Análise Visual por Diagramas de Constelação

A eficácia do equalizador é melhor demonstrada pela comparação visual do sinal antes e depois do seu processamento, e pelo impacto do ruído no resultado final.

A Figura 1 mostra o sinal na entrada do equalizador para uma SNR de 40 dB. Devido à ISI e ao ruído, os símbolos estão completamente dispersos, impossibilitando a demodulação. Após a aplicação do equalizador MMA, a Figura 2 revela que os símbolos foram agrupados em 64 *clusters* nítidos, validando a capacidade do algoritmo de mitigar a distorção do canal.

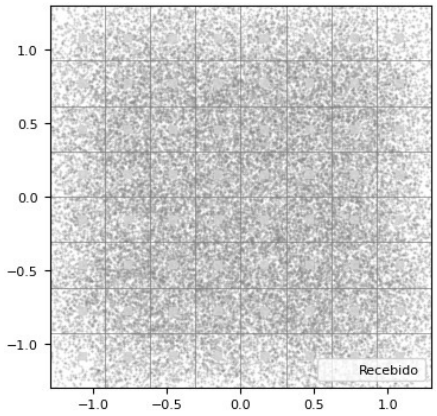

Fig. 1. Sinal Recebido Não-Equalizado (SNR 40 dB).

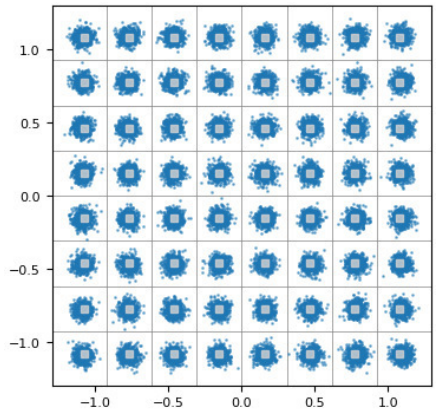

Fig. 2. Sinal Equalizado (SNR 40 dB).

Para avaliar a robustez em um cenário com maior nível de ruído, a Figura 3 exibe o resultado para uma SNR de 25 dB. Nota-se que a dispersão dos pontos dentro de cada cluster é visivelmente maior. No entanto, o algoritmo ainda consegue manter os 64 clusters distintos, o que demonstra seu funcionamento robusto mesmo em condições de canal menos favoráveis.

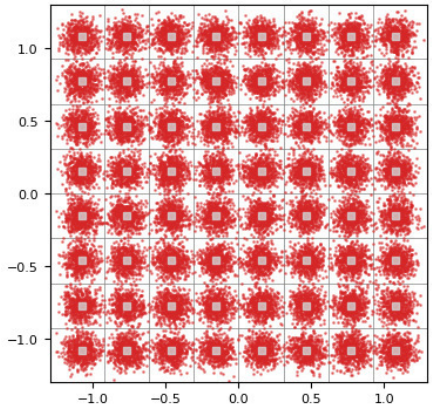

Fig. 3. Sinal Equalizado (SNR 25 dB).

c.2) Análise Quantitativa da Convergência

A Figura 4 apresenta as curvas de aprendizado do MSE médio, confirmando a análise visual com dados quantitativos. O gráfico mostra que o algoritmo converge rapidamente em todos os cenários. Fica evidente que o erro de estado estacionário é dependente do nível de ruído — quanto maior a SNR, menor o erro residual, o que corresponde diretamente à menor dispersão observada nos diagramas de constelação.

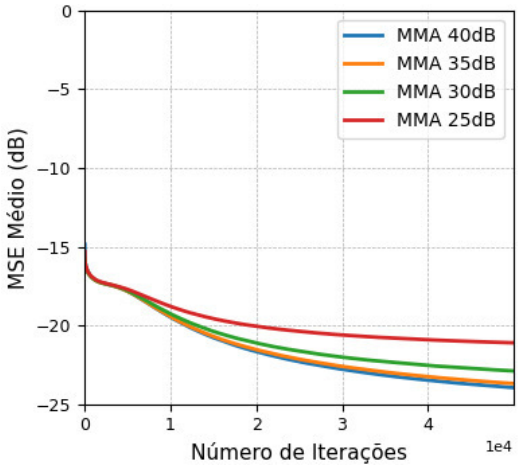

Fig. 4. Curvas de aprendizado do MSE médio para o equalizador MMA em diferentes SNRs.

D. Definição e Análise do Formato de Ponto Fixo

Para viabilizar a implementação em FPGA, foi necessária a conversão do modelo de ponto flutuante (utilizado nas simulações em Python) para aritmética de ponto fixo. A escolha das larguras de banda (*bit-width*) envolveu um compromisso entre a precisão numérica, necessária para manter a integridade do algoritmo MMA e a estabilidade da convergência e o consumo de recursos. Utilizou-se a notação $Q(m.n)$, onde m representa os bits inteiros (incluindo o bit de sinal) e n os bits fracionários [4]. A definição das larguras de bits foi realizada com base nas variações, valores máximos e mínimos do modelo. As larguras e formatos definidos para os principais sinais do *datapath* são apresentadas na Tabela I.

Observa-se que os coeficientes do filtro ($w[n]$) receberam maior precisão fracionária (18 bits) em relação aos dados

TABELA I
DEFINIÇÃO DOS FORMATOS DE PONTO FIXO

Sinal	Largura (bits)	Fracionário (bits)	Q(m.n)
Entrada ($x[n]$)	14	12	Q(2.12)
Saída ($y[n]$)	14	12	Q(2.12)
Erro ($e[n]$)	16	12	Q(4.12)
Coeficientes ($w[n]$)	20	18	Q(2.18)
Passo (μ)	12	10	Q(2.10)

de entrada e saída. Essa decisão de projeto foi tomada para garantir a suavidade na atualização dos pesos, permitindo ajustes finos e evitando o congelamento da adaptação (*stalling*) quando o gradiente do erro se torna muito pequeno[4].

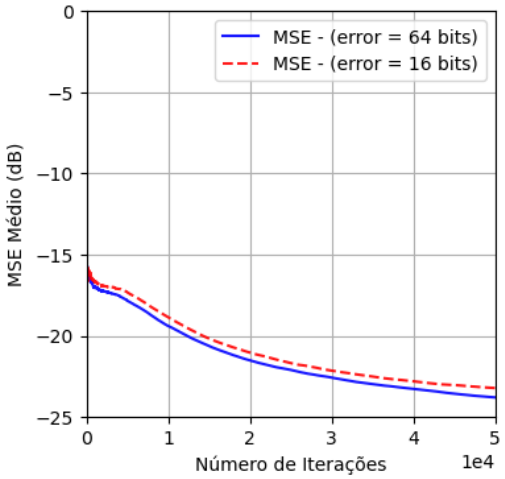

Fig. 5. Curvas de aprendizado do MSE médio calculado a apartir do $e^{mma}(n)$ para o equalizador que usa o formato númerico float e um outro que usa o formato ponto fixo proposto.

As curvas de aprendizado apresentadas na Figura 5 mostram que o modelo em ponto fixo proposto fica muito próximo do modelo que utiliza 64 bits para a representação númerica não apenas do $e^{mma}(n)$ usado para realizar o cálculo do MSE, mas também para as amostras de entrada, saída e coeficientes, o que mostra a eficiência dos formatos escolhidos.

XIII. IMPLEMENTAÇÃO EM HARDWARE DO EQUALIZADOR

A etapa final do processo de *Model-Based Design* consiste na tradução do modelo validado em Python para uma descrição de hardware (RTL) sintetizável. O objetivo é criar um *IP Core* parametrizável e eficiente, capaz de executar o algoritmo MMA em tempo real. A linguagem escolhida foi SystemVerilog, permitindo uma modelagem modular e hierárquica. A implementação foi estruturada em módulos discretos para garantir clareza e reutilização. A seguir, detalham-se a arquitetura geral, a unidade de controle e o caminho de dados otimizado.

A. Arquitetura Geral

A arquitetura de alto nível, mostrada na Figura 6, definida no módulo `mma_top.sv`, adota a separação clássica entre controle e dados. O módulo instancia e interconecta duas unidades principais:

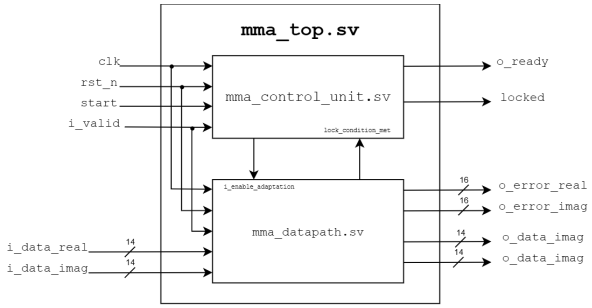

Fig. 6. Diagrama de representação do módulo `mma_top.sv`.

- **Unidade de Controle (`mma_control_unit`):** Responsável pelo gerenciamento da máquina de estados e sinais de *handshake*.
- **Caminho de Dados (`mma_datapath`):** Onde o processamento aritmético é executado. Este módulo recebe parâmetros genéricos como `N_TAPS` (definido como 18) e as larguras de bits para dados e coeficientes.

B. Unidade de Controle (FSM)

A unidade de controle (`mma_control_unit.sv`) foi implementada como uma Máquina de Estados Finitos (FSM) com quatro estados principais, garantindo a sincronização do fluxo de dados:

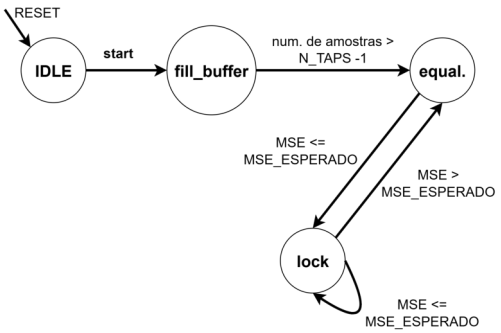

Fig. 7. Diagrama de representação da FSM presente no módulo `mma_control_unit.sv`.

1) **IDLE:** Estado de reinício, onde o sistema aguarda o sinal `start`.
2) **FILL_BUFFER:** O sistema aguarda o preenchimento da linha de atraso (*Tapped Delay Line* - TDL) com as amostras iniciais. Um contador monitora a entrada de dados até que `N_TAPS` amostras válidas tenham sido recebidas.
3) **EQUALIZATION:** O estado de operação principal. O sinal `enable_adaptation_out` é ativado, permitindo que o *datapath* atualize os coeficientes do filtro a cada nova amostra.
4) **LOCKED:** Atingido quando o erro quadrático médio instantâneo cai abaixo de um limiar (`ERROR_THRESHOLD`) definido como 0.02 por um tempo mínimo determinado pelo contador `lock_timer`. Este estado sinaliza a convergência do equalizador.

C. Caminho de Dados (Datapath)

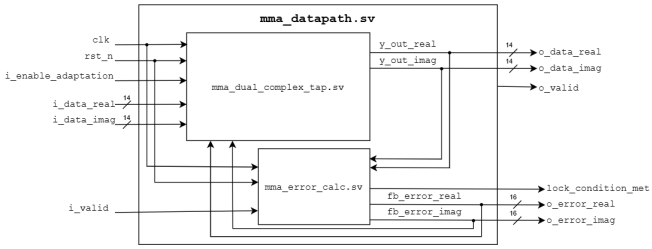

Fig. 8. Diagrama de representação do módulo `mma_datapath.sv`.

O *datapath* (`mma_datapath.sv`) implementa a filtragem FIR e o algoritmo de adaptação. Uma característica fundamental desta implementação é o uso de multiplexação no tempo para redução de área. O sistema opera com um clock (`clk`) na frequência dobrada em relação à taxa de amostragem, utilizando um sinal de controle interno (`phase_sel`) para alternar o processamento:

- **Fase 0 (Phase Sel = 0):** Processamento dos coeficientes e dados de índice par.
- **Fase 1 (Phase Sel = 1):** Processamento dos coeficientes e dados de índice ímpar e finalização da soma.

Essa estratégia reduz pela metade o número de multiplicadores complexos necessários no filtro FIR. O *datapath* é composto por dois sub-módulos, como mostrado na Figura 8 e uma árvore de soma:

c.1) Célula de Tap Duplo Complexo

O módulo `mma_dual_complex_tap.sv` é a unidade básica de processamento. Cada instância deste módulo é responsável por gerenciar **dois** coeficientes do filtro (um par e um ímpar). Ele contém:

- Memória para armazenar os pesos w_{par} e w_{impar}.
- Um único multiplicador complexo compartilhado, que calcula $w_{par} \cdot x_{par}$ ou $w_{impar} \cdot x_{impar}$ dependendo da fase do clock.
- Lógica de atualização dos coeficientes ($w_{new} = w_{old} + \mu \cdot e \cdot x^*$) com funções de arredondamento e saturação para evitar *overflow* na aritmética de ponto fixo.

c.2) Cálculo do Erro Pipelined

O cálculo do erro MMA, descrito em `mma_error_calc.sv`, implementa a função não-linear $e(n) = y(n) \cdot (\gamma^2 - |y(n)|^2)$. Devido à complexidade aritmética, este módulo foi projetado em um *pipeline* de dois estágios:

1) Cálculo dos módulos ao quadrado ($|y_R|^2$ e $|y_I|^2$).
2) Subtração da constante de dispersão ($\gamma^2 - |y|^2$).

O uso de *pipeline* permite que o equalizador opere em frequências de clock elevadas através da redução do tempo de propagação dos sinais, além de garantir que o cálculo do erro não se torne um gargalo, processando os dados na mesma taxa de vazão (*throughput*) do equalizador principal.

c.3) Árvore de Soma e Monitoramento

As saídas parciais dos Taps Duplos são somadas sequencialmente. Um acumulador armazena o resultado da fase par e o soma ao resultado da fase ímpar para gerar a saída final $y[n]$. Além disso, o *datapath* calcula a magnitude do erro instantâneo e a compara com o limiar de convergência, enviando o sinal de feedback `lock_condition_met` para a unidade de controle.

XIV. Resultados da Verificação Funcional e Síntese

A. Verificação Funcional e Co-Simulação

A validação da arquitetura proposta seguiu uma metodologia de *Verificação Funcional* , essencial para garantir a equivalência entre o modelo matemático abstrato e a implementação física em nível de transferência de registros (RTL). A Figura 9 apresenta o ambiente de teste utilizado, onde adotou-se uma abordagem de **Co-Simulação com Modelo de Referência** (*Golden Model Verification*), onde o modelo em ponto fixo desenvolvido em Python serviu como o padrão de ouro para a validação do *Device Under Test* (DUT) em SystemVerilog.

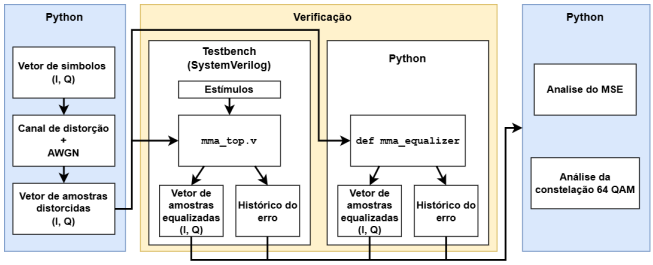

Fig. 9. Diagrama de representação do ambiente de teste.

a.1) Metodologia de Teste

Foi aplicada uma estratégia de teste direcionado, onde cenários específicos de canal e ruído, previamente analisados na etapa de software, foram replicados no ambiente de hardware. O fluxo de verificação consistiu nas seguintes etapas:

1) **Geração de Estímulos:** Vetores de teste contendo sinais 64-QAM distorcidos por ISI e ruído AWGN foram gerados pelo modelo em Python e exportados para arquivos de texto.
2) **Injeção no DUT:** Um *testbench* em SystemVerilog foi desenvolvido para ler esses vetores e injetá-los nas portas de entrada do processador MMA, ciclo a ciclo.
3) **Monitoramento de Caixa Branca (*White Box*):** Além de capturar a saída equalizada, o *testbench* inicialmente monitorou sinais internos críticos, especificamente a evolução dos coeficientes do filtro adaptativo ($\mathbf{w}$) e o sinal de erro ($e[n]$). Isso permitiu verificar não apenas o resultado final, mas a dinâmica de convergência do algoritmo.

A Figura 10 apresenta a comparação direta entre as curvas de aprendizado (MSE) do modelo de referência e da simulação RTL. A sobreposição das curvas valida a precisão da aritmética de ponto fixo implementada e confirma que o hardware reproduz fielmente o comportamento do algoritmo MMA. Para validar a integridade do sinal na saída, a Figura 11 exibe o

diagrama de constelação obtido diretamente dos dados de saída do DUT após a convergência do algoritmo, para simulações com um SNR de 35 bB.

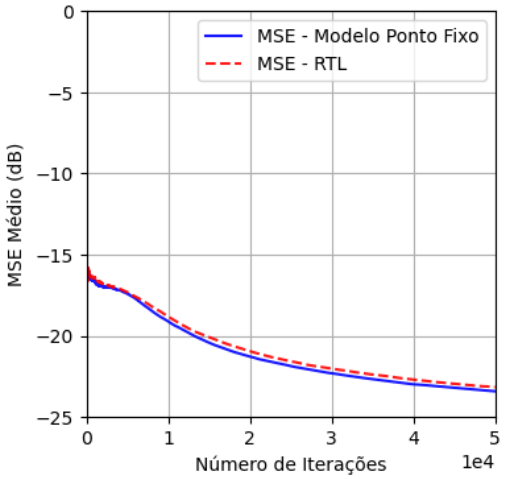

Fig. 10. Comparação da convergência do MSE: Modelo de Ponto Fixo vs. Simulação RTL (35 dB).

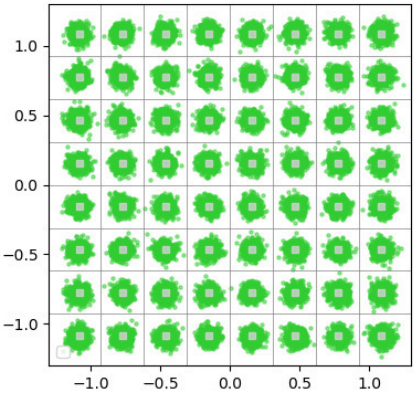

Fig. 11. Constelação de saída 64-QAM obtida via simulação RTL após convergência (35 dB).

A formação nítida dos 64 aglomerados (*clusters*) demonstra que o equalizador em hardware foi capaz de mitigar a Interferência Intersimbólica (ISI) e abrir o diagrama de olho do sinal 64-QAM com eficácia. A simetria e a compactação dos clusters validam a precisão numérica do circuito, assegurando que o IP Core está apto para a etapa de síntese.

B. Análise de Timing e Desempenho

A análise pós-síntese demonstrou que o design atende aos requisitos de velocidade para operação em 50 MHz. O relatório de *timing* apresentou um *Worst Negative Slack* (WNS) de $0.094ns$, além disso, o *Hold Slack* (WHS) registrou um valor de $0.105ns$, confirmando a robustez do circuito contra violações de *hold*. Com base no WNS obtido, a frequência máxima de operação (F_{max}) calculada é de 51.09 MHz, o que valida a capacidade do design de operar com folga na frequência alvo. Esses dados comprovam que a arquitetura multiplexada solucionou os problemas de *setup time*, tornando o design funcional.

C. Utilização de Recursos e Potência

A eficiência da arquitetura proposta é evidenciada pela taxa de ocupação dos recursos do dispositivo FPGA alvo (XC7A35T). A Tabela II resume a utilização pós-implementação.

TABELA II
UTILIZAÇÃO DE RECURSOS DO FPGA E POTÊNCIA

Recurso	Uso	Utilização (%)
DSPs (DSP48E1)	80/90	88.89%
Slice LUTs	5247/20800	25.23%
Pinos I/O (IOB)	73/150	48.67%
Potência Total	0.206 W	-

Destaca-se o uso de 88,89% dos blocos DSP disponíveis. Este valor confirma que a técnica *Time-Multiplexed* foi fundamental para viabilizar o projeto neste dispositivo, limitando a necessidade de unidades de multiplicação para apenas 80, encaixando-se no limite físico do chip.

XV. Conclusões e Trabalhos Futuros

Este trabalho apresentou o ciclo de desenvolvimento de um *Soft Intellectual Property Core* para equalização cega utilizando o algoritmo MMA, fundamentado na metodologia de MBD. A abordagem permitiu transitar desde a modelagem matemática abstrata até a implementação em FPGA.

Os resultados obtidos validaram a eficácia da arquitetura proposta. Na etapa de modelagem em software, as simulações comprovaram a capacidade do algoritmo MMA de recuperar sinais 64-QAM em canais com severa ISI. A análise das curvas de MSE demonstrou que o equalizador converge de forma robusta, com o erro residual de estado estacionário variando proporcionalmente à SNR, atingindo patamares satisfatórios para demodulação mesmo em cenários de 25 dB.

A implementação em hardware, descrita em SystemVerilog e operando em aritmética de ponto fixo, apresentou uma aderência rigorosa ao modelo de referência. A verificação funcional cruzada confirmou que as perdas por quantização foram mínimas e não comprometeram a estabilidade do filtro, onde o mesmo também mostrou-se eficiente para o dispositivo alvo, utilizando 88,89% dos recursos de DSP disponíveis para viabilizar um filtro complexo de 18 taps, atingindo uma frequência máxima de operação superior a 50 MHz, garantindo uma taxa de transmissão de 150 Mbps.

Conclui-se que o fluxo de projeto adotado foi bem-sucedido em entregar um módulo de hardware funcional, otimizado e verificado, cumprindo os objetivos propostos de equalização cega para sistemas de alta taxa de transmissão.

A. Trabalhos Futuros

O desenvolvimento deste Soft IP Core abre diversas frentes para investigações futuras e aprimoramentos técnicos. Sugere-se a continuidade da pesquisa nos seguintes tópicos:

B. Aprimoramento do Testbench

Os testes validam o algoritmo, porém uma quantidade maior de teste pode ser feita para reforçar a sua robustez, uma dela é uma melhor exploração dos *corner cases*, onde se exploraria casos mais "problemáticos"para o equalizador, como por exemplo SNR de 15 ou 20 dB.

b.1) Aprimoramento do Canal de Simulação

O ambiente de validação atual utilizou um canal linear invariante no tempo (LTI) com ruído AWGN para a prova de conceito do algoritmo. Um passo natural para a evolução do projeto é a substituição deste modelo por canais mais realistas e dinâmicos.

- **Radiofrequência:** Implementar modelos de desvanecimento multipercurso (como Rayleigh ou Rician) e introduzir deslocamentos de frequência (Doppler) para simular cenários de comunicações móveis ou sem fio.
- **Comunicações Ópticas:** Adaptar o canal para simular efeitos específicos de fibra óptica, como a Dispersão Cromática (CD) e a Dispersão dos Modos de Polarização (PMD), validando a aplicabilidade do equalizador MMA em transceptores ópticos coerentes de alta velocidade.

C. Verificação Avançada Baseada em UVM

A validação funcional apresentada neste trabalho utilizou uma abordagem de testes direcionados (*Directed Testing*), onde vetores de estímulo específicos foram gerados e comparados. Para elevar a confiabilidade do IP Core ao nível exigido por padrões industriais de ASICs, sugere-se a adoção da *Universal Verification Methodology* (UVM).

- **Geração Aleatória Restrita:** Diferente dos testes fixos, a UVM permite a geração automática e aleatória de cenários de teste (variações de SNR, coeficientes de canal extremos, sequências de dados específicas), aumentando a probabilidade de detecção de *bugs* em casos de borda (*corner cases*) não previstos manualmente.
- **Cobertura Funcional:** Implementar métricas de cobertura (*Functional Coverage*) para quantificar matematicamente o quanto do espaço de estados do equalizador foi verificado, garantindo, por exemplo, que todas as transições da máquina de estados e todas as combinações de saturação no *datapath* foram exercitadas.
- **Reutilização do Modelo:** Integrar o modelo de referência em Python desenvolvido neste trabalho diretamente ao ambiente de simulação SystemVerilog através da interface DPI-C (*Direct Programming Interface*). Isso permitiria que o modelo Python atuasse como um *Scoreboard* dinâmico dentro do ambiente UVM, verificando os resultados do hardware em tempo de execução sem a necessidade de arquivos de texto intermediários.

c.1) Otimização da Implementação Física

Embora o design atual atenda aos requisitos temporais, há margem para refinamento na implementação física, explorando recursos avançados da ferramenta de desenvolvimento.

- **Estratégias de Roteamento:** Realizar uma análise detalhada de congestionamento de rotas e aplicar estratégias de *Physical Optimization* (`phys_opt_design`) mais agressivas para tentar elevar a frequência máxima de operação (F_{max}) acima dos 100 MHz.
- **Floorplanning:** Utilizar restrições de área (*Pblocks*) para agrupar logicamente os módulos do *datapath* próximos aos blocos DSP, reduzindo atrasos de interconexão.
- **Consumo de Energia:** Utilizar as ferramentas de análise de potência para identificar os caminhos de maior comutação e aplicar técnicas de *Clock Gating* granular na unidade de controle para reduzir a potência dinâmica.

c.2) Fluxo RTL-to-GDS e ASIC

Visando a aplicação em produtos de consumo em larga escala, sugere-se a migração do projeto de FPGA para um fluxo de design de Circuito Integrado de Aplicação Específica (ASIC).

- **Síntese Lógica e Física:** Utilizar ferramentas de automação de design eletrônico (EDA) para realizar a síntese lógica mapeada em uma biblioteca de células padrão (*Standard Cells*) de uma tecnologia nanométrica..
- **Geração de Layout (GDSII):** Executar o fluxo completo de *Backend* (posicionamento, roteamento, verificação de regras de design - DRC e verificação de layout vs. esquemático - LVS) para gerar o arquivo GDSII final. Isso provaria a viabilidade do IP Core como um bloco funcional integrável em *Systems-on-Chip* (SoC) comerciais de alto desempenho.

REFERÊNCIAS

[1] Kevin Banovic. “Blind adaptive equalization for QAM signals: New algorithms and FPGA implementation”. Master’s thesis. University of Windsor, 2006.

[2] Wen SiYuan. “Blind Equalization of QAM Signals Based On Dual-Mode Multi-modulus Algorithms”. Em: *Advanced in Control Engineering and Information Science*. Vol. 15. 2011, pp. 2434–2438. DOI: 10.1016/j.proeng.2011.08.457.

[3] Jian Yang, Jean-Jacques Werner e Guy A. Dumont. “The multimodulus blind equalization and its generalized algorithms”. Em: *IEEE Journal on Selected Areas in Communications* 20.5 (2002), pp. 997–1015. DOI: 10.1109/JSAC.2002.1007380.

[4] Uwe Meyer-Baese. *Digital Signal Processing with Field Programmable Gate Arrays*. 4ª ed. Signals and Communication Technology. Berlin, Heidelberg: Springer, 2014. ISBN: 978-3-642-45309-0. DOI: 10.1007/978-3-642-45309-0.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Entrega de TCC

Assunto:	Entrega de TCC
Assinado por:	Icaro Alcantara
Tipo do Documento:	Termo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Icaro Mendes de Alcantara Bezerra, ALUNO (201921250010) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 24/02/2026 16:09:26.

Este documento foi armazenado no SUAP em 24/02/2026. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1777226
Código de Autenticação: 4adf15b907

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Monografia do aluno - corrigida

Assunto:	Monografia do aluno - corrigida
Assinado por:	Henrique Nascimento
Tipo do Documento:	Tese
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Documento Original

Documento assinado eletronicamente por:

- **Henrique do Nascimento Cunha**, **COORDENADOR(A) DE CURSO - FUC1 - CCEC-CG**, em 24/02/2026 17:03:50.

Este documento foi armazenado no SUAP em 24/02/2026. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1777412
Código de Autenticação: 999321218b