UNIVERSIDADE FEDERAL DO PARANÁ

JEFFERSON RODRIGO SCHUERTZ

# PROJETO DE UM DPD EM FPGA BASEADO EM POLINÔMIOS DE MEMÓRIA DESCRITO EM VHDL

CURITIBA

2024

JEFFERSON RODRIGO SCHUERTZ

# PROJETO DE UM DPD EM FPGA BASEADO EM POLINÔMIOS DE MEMÓRIA DESCRITO EM VHDL

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia Elétrica – Ênfase em Sistemas Eletrônicos Embarcados, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Orientadora: Profa. Dra. Sibilla Batista da Luz França

Coorientador: Prof. Dr. Eduardo Gonçalves de Lima

CURITIBA

2024

## AGRADECIMENTOS

Quero expressar meu agradecimento pela valorosa oportunidade de completar esta graduação. Reforço meu reconhecimento aos esforços de meus familiares, em especial ao de meu Pai e ao corpo docente do Departamento de Engenharia Elétrica, particularmente a dedicação e profissionalismo demonstrado pela orientadora e coorientador deste trabalho que acompanharam meu amadurecimento e desenvolvimento como profissional e indivíduo desde o início na Academia. Sou grato também a todos que dividiram um capítulo de suas vidas nas dependências do GICS (*Group of Integrated Circuits and Systems*) e do departamento, bem como ao grupo PET (*Programa Educação Tutorial*) e seu tutor e orientador de meu primeiro estágio que permitiu meu ingresso profissional. Que o aprendizado, os desafios e as circunstâncias reservadas pela Divina Providência ao longo deste curso guiem-me a um propósito ainda em vida.

## RESUMO

Este trabalho apresenta o desenvolvimento e implementação de um Digital Pre-Distorter (DPD) em FPGA, utilizando polinômios de memória descritos em VHDL. O objetivo principal foi mitigar as não linearidades de amplificadores de potência (PAs), otimizando sua eficiência energética e melhorando a qualidade do sinal transmitido. A abordagem proposta começou com a modelagem comportamental em software, utilizando simulações em Python para validar o uso de polinômios de memória (MP) como base para a linearização. Nesse contexto, foram incorporadas Look-Up Tables (LUTs) para reduzir a complexidade computacional, possibilitando cálculos rápidos e precisos por meio de valores pré-computados e armazenados. O modelo foi descrito em VHDL e sintetizado na FPGA Virtex5 XC5VLX20T. A arquitetura projetada destacou-se por adotar uma segmentação em pipeline, distribuindo as operações em oito ciclos de clock. Essa estrutura permitiu o processamento contínuo dos dados e otimizou o *throughput* do sistema, mesmo com a latência inicial necessária para a primeira saída. Para compatibilizar o modelo ao hardware, a representação de ponto flutuante foi substituída por inteiros, exigindo ajustes nos cálculos e escalonamento de valores para manter a precisão esperada. Os resultados obtidos demonstraram a eficácia da abordagem. As LUTs foram fundamentais para garantir um balanceamento entre precisão e eficiência, alcançando um NMSE de -10,17 dB com discretização de 3 bits. A análise também evidenciou o uso eficiente dos recursos da FPGA, com apenas 12% das LUTs, 15% dos blocos RAM e 3% dos flip-flops consumidos, enquanto os 24 DSPs disponíveis foram integralmente utilizados para atender às operações matemáticas intensivas. Esses resultados comprovam a viabilidade do modelo em sistemas de alta complexidade, como comunicações de rádio frequência. A contribuição deste trabalho inclui o avanço no entendimento de técnicas de linearização digital, bem como a demonstração de sua aplicabilidade em sistemas de comunicação eficientes, com potencial para implementação em circuitos integrados futuros.



Palavras-chave: 1. Digital Pre-Distorter (DPD) 2. Polinômios de Memória 3. FPGA 4. Look-Up Tables (LUTs) 5. Linearização de Amplificadores de Potência

## ABSTRACT

This work presents the development and implementation of a Digital Pre-Distorter (DPD) on FPGA, using memory polynomials described in VHDL. The main objective was to mitigate the nonlinearities of power amplifiers (PAs), optimizing their efficiency and improving the quality of the transmitted signal. The proposed approach began with behavioral modeling in software, using Python simulations to validate the use of memory polynomials (MP) as a basis for linearization. In this context, Look-Up Tables (LUTs) were incorporated to reduce computational complexity, enabling fast and accurate calculations through pre-computed and stored values. During the hardware implementation phase, the model was described in VHDL and synthesized on the Virtex5 XC5VLX20T FPGA. The designed architecture stood out by adopting a pipelined segmentation, distributing operations over eight clock cycles. This structure allowed continuous data processing and optimized system throughput, despite the initial latency required for the first output. To adapt the model to hardware, floating-point representation was replaced with integers, requiring adjustments in calculations and value scaling to maintain the expected accuracy. The results demonstrated the effectiveness of the proposed approach. The LUTs were fundamental in balancing accuracy and efficiency, achieving an NMSE of -10.17 dB with 3-bit discretization. The analysis also highlighted the efficient use of FPGA resources, with only 12% of LUTs, 15% of RAM blocks, and 3% of flip-flops consumed, while all 24 available DSPs were fully utilized to handle intensive mathematical operations. These results confirm the feasibility of the model in high-complexity systems such as radio frequency communications. This study not only validated the application of DPDs on FPGA but also established a promising path for the use of modular and optimized architectures. The contribution of this work includes advancements in the understanding of digital linearization techniques and the demonstration of their applicability in efficient communication systems, with potential for future implementation in integrated circuits.



**Keywords**: 1. Digital Pre-Distorter (DPD) 2. Memory Polynomials 3. FPGA 4. Look-Up Tables (LUTs) 5. Power Amplifier Linearization

## LISTA DE FIGURAS

LISTA DE TABELAS

# LISTA DE ABREVIATURAS

PA – AMPLIFICADOR DE POTÊNCIA

DPD – DIGITAL PRE-DISTORTER

LUT – LOOK-UP TABLE

VHDL – VHSIC HARDWARE DESCRIPTION LANGUAGE

CI – CIRCUITO INTEGRADO

RF – RADIOFREQUÊNCIA

ISE – INTEGRATED SOFTWARE ENVIRONMENT

ISIM – INTEGRATED SIMULATOR

NMSE – NORMALIZED MEAN SQUARE ERROR

MP – MEMORY POLYNOMIAL

FPGA – FIELD-PROGRAMMABLE GATE ARRAY

DSP – DIGITAL SIGNAL PROCESSOR

PM-TO-AM – PHASE MODULATION TO AMPLITUDE MODULATION

PM-TO-PM – PHASE MODULATION TO PHASE MODULATION

ASIC – APPLICATION-SPECIFIC INTEGRATED CIRCUIT

**SUMÁRIO**

# 1 INTRODUÇÃO

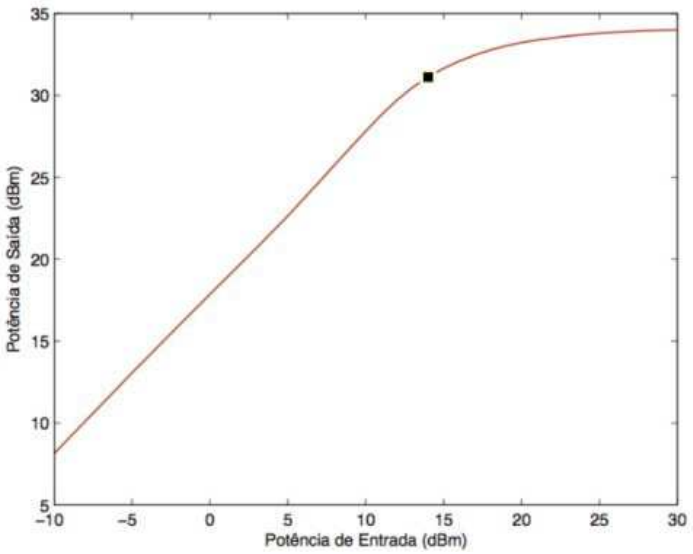
## 1.1 Contextualização e Motivação

Nos últimos anos, a rápida evolução e massificação dos dispositivos móveis gerou aparelhos com altas velocidades de transferência de dados, conectividade constante, uso intensivo de aplicações multimídia, prevalência de telas maiores e brilhantes, além de outras diversas funcionalidades. Esse cenário tem exigido dos dispositivos móveis um aumento significativo do consumo de energia [1-3].

Em resposta a esses desafios, a indústria tem se concentrado no desenvolvimento de tecnologias energeticamente mais eficientes, especialmente na otimização dos esquemas de modulação e na redução da não-linearidade dos Amplificadores de Potência ou *Power Amplifier* (PA). Estes esforços visam não apenas melhorar a eficiência energética, mas também otimizar o desempenho geral do dispositivo, garantindo uma experiência de usuário aprimorada em termos de qualidade de sinal e confiabilidade.

Entre os diversos componentes eletrônicos presentes nos dispositivos, os PAs desempenham um papel crucial. A transmissão de informações digitais para redes ou outros dispositivos demanda uma considerável quantidade de potência [1]. Em virtude disso, os PAs têm sido objeto frequente de estudo na literatura [1-4]. A Figura 1 apresenta um gráfico onde no eixo horizontal tem-se a potência de entrada e na vertical a potência de saída de um PA. Conclui-se que para uma alta potência de entrada a relação entrada-saída não é mais linear.

Figura 1 – Potência de entrada e saída em dBm em um PA.




FONTE: MACHADO (2016).

Para otimizar o uso do PA, é importante tornar mais eficiente a operação em sua zona não linear. Contudo, isso gera desafios relacionados à linearidade do sinal. Para resolver isso, várias técnicas de linearização, como *feed-forward*, *feed-back* e pré-distorção, têm sido exploradas. Entre elas, a Pré-Distorção Digital (*Digital Pre-Distorter* - DPD) se destaca [7]. O DPD atua em conjunto com o PA, aplicando uma característica não linear inversa a do PA. Isso faz com que o sinal de entrada seja pré-distorcido de forma oposta à não linearidade do PA, resultando em um sinal de saída linear enquanto mantém o PA na zona de alta eficiência. Assim, o DPD oferece um equilíbrio eficaz entre a linearidade do sinal e a eficiência energética do PA.

## 1.2 Objetivos

Este trabalho tem como objetivo aplicar conceitos de polinômios de memória e suas derivações no desenvolvimento de filtros digitais para pré-distorção digital, com foco na implementação em *hardware*. Os objetivos específicos incluem a seleção de um modelo polinomial apropriado para o dispositivo de pré-distorção, realização de simulações pertinentes, uso de LUTs (*Look-up Tables*) para simplificação de cálculos, abstração da técnica escolhida em VHDL (*VHSIC Hardware Description Language*) em uma FPGA (*Field-Programable Gate Array)*. Embora o desenvolvimento de um Circuito Integrado (CI) dedicado não seja atingido neste projeto, a abordagem adotada visa viabilizar a concepção de um CI em trabalhos futuros, consolidando as bases técnicas e metodológicas para tal avanço.

Este estudo visa beneficiar tanto a comunidade acadêmica, especialmente pesquisadores em microeletrônica com foco em técnicas de pré-distorção digital para amplificadores de potência, quanto profissionais da indústria buscando aprimorar a eficiência em circuitos de RF (Radiofrequência).

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 Amplificadores de Potência

Os PAs são dispositivos analógicos vitais em sistemas de transmissão sem fio, projetados para fornecer a maior potência possível às cargas [3]. Usados em equipamentos de rádio frequência (RF), eles desempenham um papel crucial na transmissão de grandes volumes de dados. O principal componente do PA RF é o transistor, que tem a função de amplificar a potência do sinal de entrada. O processo envolve a conversão de energia de corrente contínua (CC) em energia de corrente alternada (CA), e a eficiência do PA é medida pela relação entre a potência entregue à carga e a potência fornecida pela fonte de alimentação CC, buscando sempre a máxima eficiência possível. O rendimento pode ser determinado pela equação 1:

$$\eta = \frac{Potência\ Entregue}{Potência\ Fornecida}, \qquad (1)$$

Devido à natureza não ideal dos componentes que compõem o PA e do ambiente em que ele é instalado, a eficiência ideal de 100% nunca é alcançada. Além do transistor, o circuito do PA inclui redes de casamento de impedância na entrada e saída, que ajudam a modelar a característica de transferência do PA, e um circuito de polarização de corrente contínua (CC), que fornece a corrente e tensão necessárias para o transistor.

A eficiência de um PA está diretamente ligada ao aumento da potência de saída, ou seja, quanto maior a potência de saída, maior o rendimento do PA [4]. A potência não utilizada na amplificação é dissipada como calor, gerando custos adicionais de projeto e podendo prejudicar o funcionamento dos dispositivos eletrônicos. Por isso, é crucial para redes de telecomunicações ter um PA RF eficiente. Além disso, a característica de transferência do PA, que mostra a relação entre potência de saída e de entrada, é importante para avaliar seu comportamento. Esta característica não é linear, apresentando um ponto de inflexão conhecido como ponto de compressão de 1 dB, onde o ganho do amplificador diminui. Após esse ponto, o aumento da potência de saída desacelera até o ponto de saturação do transistor.

## 2.2 Modelos Comportamentais de Amplificadores de Potência

Os PAs são dispositivos caracterizados por sua não linearidade e efeitos de memória, os quais são resultados da exposição aos sinais de banda larga e das características internas do circuito do amplificador. Esses sistemas podem ser descritos matematicamente por meio da Série de Volterra, que fornece um modelo teórico para representar a complexa relação entre entrada e saída em sistemas não lineares com memória. Tal série é representada na equação 2.

$$\tilde{y}(t) = \sum_{p=1}^{P} \sum_{m_1=0}^{M} \sum_{m_2=m_1}^{M} \cdots \sum_{m_p=m_{p-1}}^{M} \sum_{m_{p+1}=0}^{M} \sum_{m_{p+2}=m_{p+1}}^{M} \cdots \sum_{m_{2p-1}=m_{2p-2}}^{M} \times \tilde{h}_{2p-1}(m_1, m_2, \cdots, m_{2p-1}) \prod_{i_1=1}^{p} \tilde{x}(n-m_{i_1}) \prod_{i_2=p+1}^{2p-1} \tilde{x}^*(n-m_{i_2}) \quad (2)$$

Onde $h_{2p1}(m_1, m_2, \cdots, m_{2p-1})$ são os parâmetros do modelo e *n* corresponde a amostra discreta.

A série de Volterra é notável por sua linearidade nos parâmetros, isso é benéfico quando se almeja identificar tais parâmetros do polinômio através de sistemas de identificação linear, como o Método dos Mínimos Quadrados (MMQ). Contudo, o número de parâmetros aumenta significativamente com a ordem do polinômio e a quantidade de termos de memória considerados, resultando em modelos computacionais mais complexos. Para contornar isso, mesmo em polinômios de alta ordem e com muitos termos de memória, a literatura sugere simplificações da série de Volterra, como o modelo comportamental Memory Polynomial (MP) [8].

O modelo MP é uma variação do modelo polinomial que incorpora termos de memória por meio de uma versão simplificada da série de Volterra. No modelo MP, apenas os produtos dos valores capturados no mesmo instante de tempo são levados em consideração. Isso permite uma representação matemática mais focada e gerenciável do modelo, sua representação assume a forma da equação 3.

$$\tilde{y}(n) = \sum_{p=1}^{P} \sum_{m=0}^{M} \tilde{b}_{mp} \tilde{x}(n-m) \mid \tilde{x}(n-m) \mid^{(p-1)} \quad (3)$$

Onde $b_{m,p}$ são os coeficientes lineares, $x(n - m)$ a entrada em função da amostra discreta *n* e da memória *m*. Já *p* indica a ordem do polinômio.

## 2.3 Linearização de Amplificadores de Potência

Como destacado na seção 2.1, uma baixa eficiência em PAs resulta em grande dissipação de calor, afetando negativamente a vida útil da bateria de dispositivos móveis e aumentando os custos de operação de estações rádio base. Sendo assim, uma melhor eficiência é, portanto, economicamente necessária. Além disso, a linearidade é crucial para a fidelidade na transmissão de dados, exigindo que o PA reproduza o sinal de entrada com maior potência e precisão.

A arquitetura do circuito também tem influência na linearidade do PA. Embora circuitos mais complexos possam aumentar a linearidade, eles também acrescentam complexidade ao *design*. Arquiteturas inovadoras, como o amplificador de Doherty e o *Envelope Tracking* [7-9], são exemplos de soluções encontradas na literatura que buscam otimizar o equilíbrio entre linearidade e eficiência. Estas soluções visam aprimorar o desempenho geral do PA, mantendo a qualidade do sinal e reduzindo a dissipação de calor.

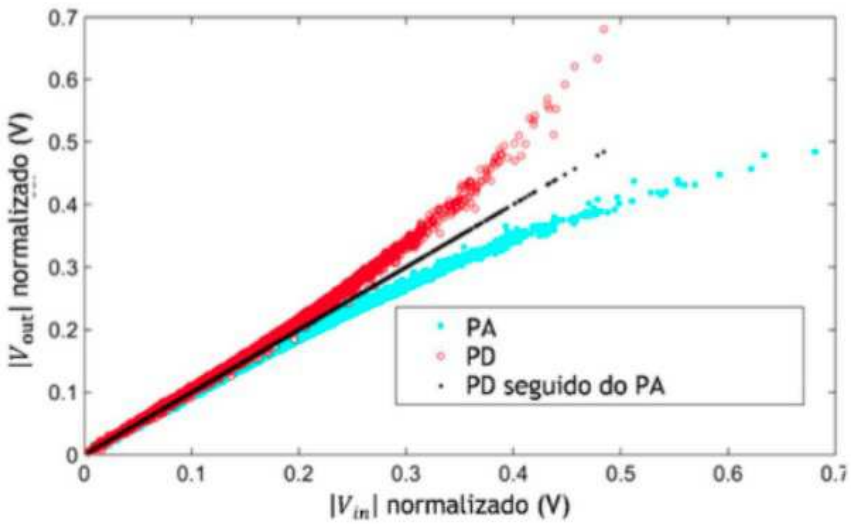
Para alcançar alta eficiência e linearidade em PAs, a inclusão de técnicas de linearização, como a pré-distorção, é uma estratégia eficaz. Esta abordagem envolve a adição de um circuito, seja analógico ou digital, com uma função de transferência inversa a do PA. Assim, o sinal de saída permanece linear, mesmo quando o PA opera fora da sua região linear. O sinal de entrada é pré-distorcido pelo circuito de pré-distorção, compensando as não linearidades do PA e permitindo maior eficiência no circuito.

A técnica utilizada em um DPD envolve aplicar uma função de transferência não linear, inversamente proporcional a do PA, ao sinal de entrada [1]. Isso permite que os circuitos do DPD e do PA trabalhem conjuntamente para produzir um ganho global linear. O design do DPD é intrinsecamente relacionado ao comportamento do PA, o que requer uma modelagem comportamental detalhada e precisa do circuito do PA. Esta modelagem é crucial para garantir que o DPD compense efetivamente as não linearidades do PA, otimizando assim a qualidade do sinal de saída e aumentando a eficiência do sistema. A precisão da modelagem do PA é fundamental para o sucesso da implementação do DPD, já que quaisquer imprecisões na modelagem podem levar a uma pré-distorção inadequada, resultando em uma performance subótima do sistema.

## 2.4 Técnica da Pré-Distorção Digital

A técnica de pré-distorção digital permite mitigar os efeitos de não linearidade em PAs, melhorando assim a eficiência do dispositivo. O princípio do DPD envolve distorcer o sinal de entrada de maneira oposta às não linearidades do PA. Isso compensa as distorções causadas pelo circuito do amplificador, resultando em um ganho linear no sinal de saída. A Figura 2 demonstra o efeito da mitigação da não-linearidade, nela evidencia-se, em vermelho, a curva do sinal após o DPD e, em azul, a curva de resposta de um PA. Destaca-se que entre as duas curvas há, em preto, o sinal de saída de um DPD combinado com um PA, nota-se um sinal com comportamento linear.

Figura 2 – Mitigação da não linearidade quando DPD é disposto antes do PA [1]

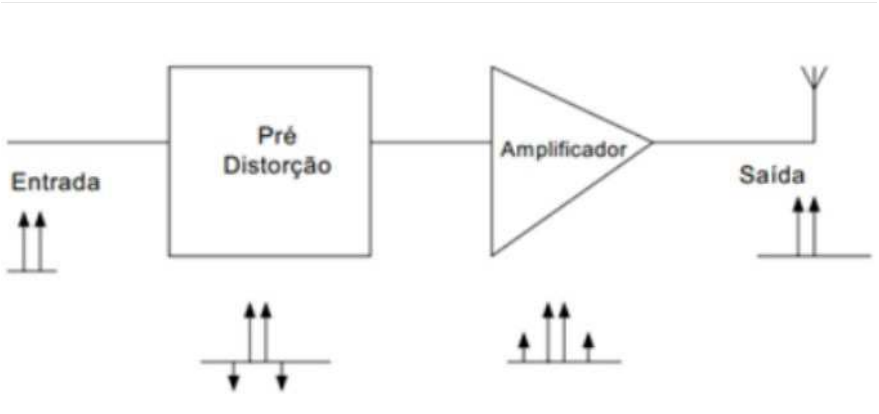



FONTE: MACHADO (2024).

Algoritmos corretivos devem ser empregues em um DPD para corrigir distorções causadas pelo PA, relacionando os sinais de entrada e saída. A escolha do modelo de DPD depende do conhecimento do sinal de entrada e das características do PA. Modelos comuns incluem aqueles baseados em polinômios com memória, algoritmos de Wiener e Hammerstein, e redes neurais artificiais. Geralmente, o DPD é posicionado antes do PA, já que a potência de entrada é menor que a de saída. A função do DPD é inversa a do PA, e seus parâmetros são derivados da modelagem do PA. Alternativamente, o método de aprendizagem indireta pode ser usado, onde um circuito auxiliar (PoD) é implementado após o PA

para otimizar o desempenho do DPD. Então, tal como na representação cartesiana da Figura 2, a Figura 3 ilustra o diagrama da disposição em cascata de um DPD e um PA.

Figura 3 – Exemplo prático da implementação da técnica de pré-distorção em um sistema global.




FONTE: MACHADO (2024).

# 3 METODOLOGIA

Para o desenvolvimento deste trabalho, a primeira etapa realizada consistiu em uma revisão do estado da arte, cujo objetivo foi explorar as técnicas de pré-distorção digital e a existência de modelos que poderiam ser adaptados à criação de filtros digitais. Concluída essa etapa, foram definidas 5 tarefas para a conclusão do DPD:

1. Seleção de um modelo polinomial;
2. Redução da complexidade computacional com tabelas de busca;
3. Implementação em Python;
4. Descrição em *VHDL* e implementação em FPGA.

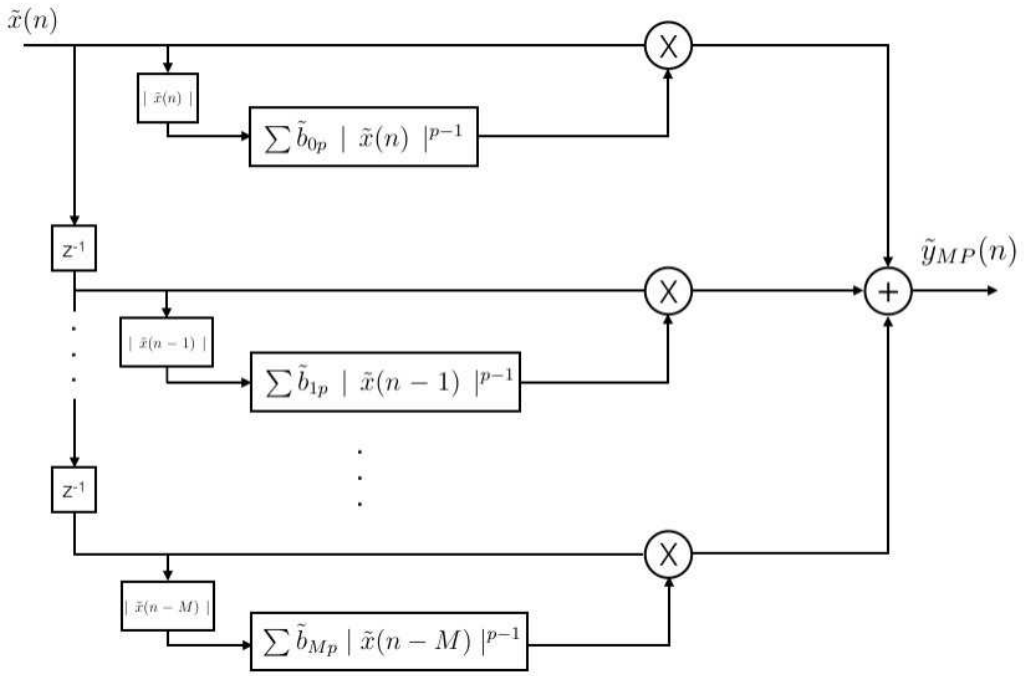
## 3.1 Seleção de um modelo polinomial

Um modelo capaz de modelar o comportamento de um amplificador de potência foi escolhido, o Polinômio de Memória (MP). Esta decisão foi motivada por vantagens distintas, destacando-se a linearidade de seus parâmetros. A obtenção desses parâmetros ocorre por meio de um processo de aprendizagem linear, como o método dos mínimos quadrados (LS), reconhecido como um algoritmo eficiente em termos de custo-benefício, considerando a relação entre complexidade computacional e acurácia [10].

O MP se destaca por realizar multiplicações de sinais no mesmo instante de tempo. Além disso, por depender de informações passadas da fase da envoltória do sinal de entrada, ele é capaz de modelar fenômenos que envolvem variações de fase, como as conversões de modulação de fase para amplitude (*PM-to-AM*) e de fase para fase (*PM-to-PM*).

O modelo polinomial pode ser abstraído através do diagrama da Figura 4, onde um polinômio de ordem genérica tem sua representação em blocos. O diagrama representa o MP e seus *n* instantes de memória. Ele ilustra a entrada de sinal *x(n)*, que é processada através de múltiplas etapas, cada uma representando um atraso de tempo, indicado por Z-1, e multiplicada por seus respectivos coeficientes polinomiais. Estes coeficientes são somados em cada etapa para formar a saída do modelo, $y_{MP}(n)$, após a aplicação dos termos representados pelo somatório do produto dos coeficientes pelo módulo da entrada complexa, havendo a elevação de toda a expressão a p-1. Este diagrama é fundamental para entender como o modelo captura as dinâmicas temporais e espaciais do sistema que está sendo modelado.

Figura 4 – Diagrama de Blocos do MP [1].




FONTE: MACHADO (2024).

### 3.2 Redução da complexidade computacional com tabelas de busca

Computacionalmente, certas operações matemáticas, especialmente aquelas que envolvem algoritmos complexos ou grandes conjuntos de dados, podem ser extremamente custosas em termos de recursos computacionais. Por exemplo, a multiplicação de matrizes grandes, a inversão de matrizes e fatoração de números grandes são tarefas que exigem um alto grau de poder de processamento e memória.

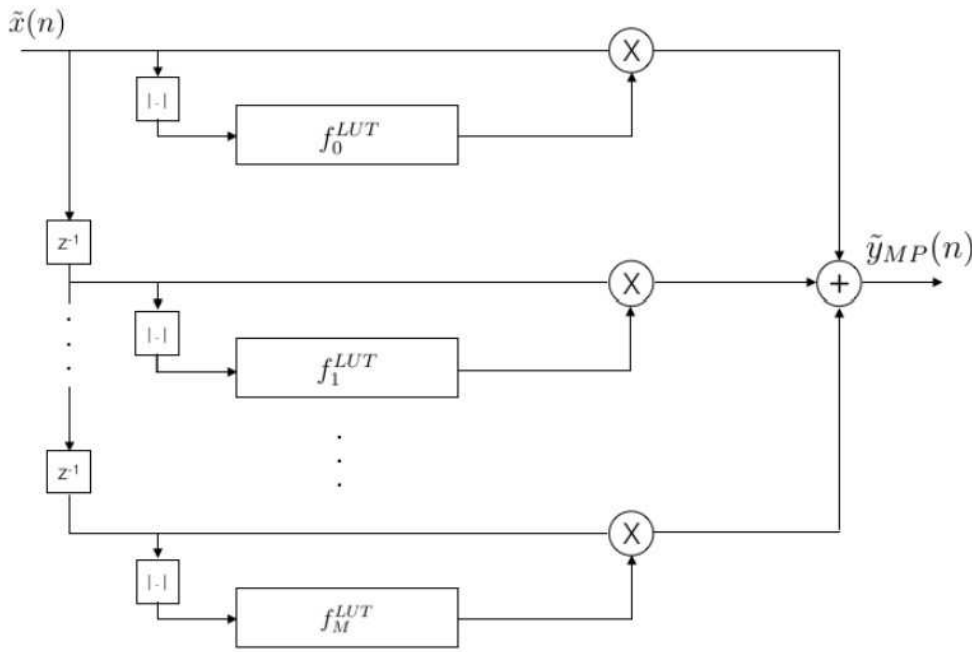
Operações complexas podem exigir ou muitos ciclos de execução, quando implementadas em *software*, ou ainda, no escopo desse trabalho, quando implementadas em *hardware*. Tarefas custosas computacionalmente pode exigir circuitos relativamente grandes, além de tais circuitos acabarem por demandar maior área de silício e consumo de energia. No intuito de contornar esse problema, buscar alternativas aproximadas ou heurísticas para reduzir a carga computacional é do interesse do projetista.

O uso de tabelas de busca ou, como na literatura, *Look-up Tables* (LUT), são elementos onde valores pré-calculados estão armazenados. Esta estratégia permite evitar que tais valores sejam recorrentemente calculados, pois eles são consultados

diretamente em uma LUT. Ao adotar o conceito das tabelas de busca é possível alterar o diagrama da Figura 4 em um novo diagrama (Figura 5). Nela, cada LUT mapeia um valor de entrada específico para uma saída correspondente, servindo como uma função de transferência não linear.

O sinal de entrada *x(n)* passa por sucessivos atrasos representados por Z-1, onde cada versão atrasada do sinal é então modificada por uma LUT diferente. As saídas das LUTs são multiplicadas por uma versão modificada do sinal de entrada original e somadas juntas para produzir o sinal de saída MP(*n*), que é a versão pré-distorcida do sinal. Este processo ajuda a compensar as não linearidades inerentes ao sistema de amplificação para o qual o sinal está sendo preparado.

Figura 5 – Diagrama de Blocos do MP aplicado ao conceito das tabelas de busca. [1]




FONTE: MACHADO (2024).

Como apresentado na Figura 5, há agora LUTs no lugar onde havia a seguinte função, descrita na equação 4:

$$f_{NL}(in) = \sum_{p=0}^{P-1} h_{p,m} \left(\sqrt{in}\right)^p \quad (4),$$

Onde *in* é o valor absoluto da entrada complexa em um determinado instante, $h_{p,m}$

corresponde aos coeficientes e p a ordem do polinômio.

### 3.3 Implementação em Python

Linguagens de alto nível são ferramentas fundamentais para validar modelos e desenvolver soluções eficientes em projetos complexos. Elas possibilitam a prototipagem rápida de algoritmos, a análise precisa de resultados e a preparação de dados para implementações futuras. Neste trabalho, Python foi amplamente utilizado para realizar essas tarefas, facilitando o progresso e prototipação da abordagem.

O Python foi utilizado em vários situações, inicialmente para implementar o modelo de regressão linear, aplicando métodos numéricos que minimizassem os erros entre valores previstos e observados. Além diisso, o MP foi primeiramente implementado nessa linguagem, o que permitiu identificar os parâmetros por aprendizado linear. Além disso, foi usado na preparação e normalização dos dados, bem como na prototipação da integração do MP com as LUTs.

### 3.4 Descrição em VHDL e implementação em FPGA.

No fluxo desse projeto, após a conclusão da implementação em *software* do algoritmo do DPD, é necessário que se aplique a mesma lógica a uma linguagem específica para descrição de *hardware*, isto é, suas entradas saídas e arquitetura.

O VHDL é uma linguagem de descrição de hardware que permite a modelagem de circuitos eletrônicos de maneira eficiente e sistemática. Sendo uma das linguagens de descrição de *hardware* mais populares, o VHDL destaca-se por sua capacidade de descrever a estrutura e o comportamento de sistemas eletrônicos digitais em vários níveis de abstração [6].

Seu uso permite descrever a arquitetura de um circuito digital bem com simular o funcionamento de um circuito antes mesmo de sua fabricação física, possibilitando a correção de erros e otimização de desempenho de forma virtual.

No âmbito deste trabalho, após a descrição do algoritmo de pré-distorção nessa linguagem, foram realizadas sucessivas rodadas de simulação a partir de *testbenches* elaborados também em VHDL. A plataforma ISE Design Suite, cuja empresa proprietária, Xilinx, foi utilizada em todo o fluxo apresentado nessa seção. A FPGAs (*Field-Programmable Gate Array*) utilizada foi a Virtex5 XC5VLX20T em uma frequência de operação de 100 MHz.

# 4 RESULTADOS E DISCUSSÃO

## 4.1. Implementação do método dos mínimos quadrados para otimização

A primeira etapa consistiu na aplicação do método dos mínimos quadrados ordinários (OLS - Ordinary Least Squares) para ajustar um modelo de regressão linear a um conjunto de dados de uma reta qualquer. Esse método visa determinar os coeficientes que minimizam o erro quadrático médio entre os valores preditos pelo modelo e os valores observados nos dados.

A fórmula utilizada, expressa pela Equação 5, é dada por:

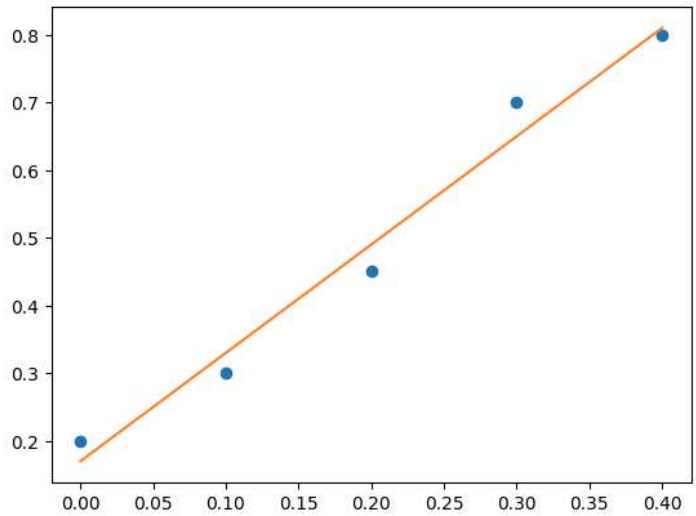
$$COEF = ((XX.T) \times XX)^{-1} \times XX.T \times Y, \qquad (5)$$

Onde,

- COEF é o vetor coluna contendo os coeficientes estimados do modelo de regressão linear. Cada elemento de COEF representa o peso associado a uma variável preditora na matriz XX.
- XX é a matriz de design (ou matriz de preditores), onde cada linha corresponde a uma observação e cada coluna a uma variável independente. Se houver *n* observações e *p* variáveis independentes, XX será de dimensão $n \times p$.
- $XX^T$ é a transposta da matriz XX, onde as linhas e colunas são invertidas, resultando em uma matriz de dimensão $p \times n$.
- Y é o vetor coluna dos valores observados da variável dependente (ou resposta), com dimensão n × 1.
- $(XX^T\ XX)^{-1}$ representa a inversa da matriz produto $XX^T\ XX$. Esta matriz quadrada, de dimensão $p \times p$, deve ser invertível, o que exige que $XX^T\ XX$ seja de posto completo (ou seja, todas as variáveis independentes são linearmente independentes).

Os coeficientes obtidos por esta operação, denotados por COEF, determinam a linha de melhor ajuste para os dados, minimizando o erro quadrático entre as predições e as observações reais. A Figura 6 apresentada ilustra a aplicação desta técnica, mostrando os pontos de dados em um gráfico de dispersão e a respectiva linha de regressão resultante. A concordância entre a linha ajustada e os pontos de dados evidencia a eficácia do modelo linear em capturar a relação subjacente entre as variáveis consideradas.

Figura 6 – Regressão realizada após identificação dos coeficientes após a otimização (Dados Adimensionais)




FONTE: O Autor, (2024).

Este método de aprendizado linear, baseado nos mínimos quadrados ordinários (OLS), foi utilizado como base para a obtenção dos parâmetros do modelo polinomial escolhido. A abordagem permite determinar, de forma eficiente, os coeficientes que melhor ajustam o modelo às observações, minimizando o erro quadrático médio entre os valores preditos e os dados reais. A simplicidade e o rigor matemático desta técnica tornam-na ideal para estimar os parâmetros do modelo polinomial de pré-distorção, garantindo uma implementação precisa e consistente, fundamental para as etapas subsequentes de validação e síntese em hardware.

### 4.2 Implementação do polinômio de memória e ajuste dos coeficientes por métodos algébricos

Posteriormente, dotado de um novo conjunto de valores formado por valores igualmente espaçados de uma senoide e já tendo aplicado o métodos dos mínimos quadrados, o MP foi implementado em Python. Montou-se a matriz de autorregressão com os valores e fez-se a redução, porém desta vez de modo mais prático através da expressão da equação 6:

$$\beta' = min(|Y - XX \cdot \beta|)^2, \qquad (6)$$

onde,

- β’ é o vetor coluna que contém os coeficientes estimados do modelo.
- Y representa o vetor com os valores observados da variável dependente

(saída).

- XX é a matriz de autorregressão, composta pelos valores das variáveis preditoras e, opcionalmente, por colunas adicionais, como um termo constante para o intercepto.
- β é o vetor de coeficientes do modelo, ajustado para minimizar o erro quadrático médio.
- $\|\cdot\|^2$ denota a norma euclidiana ao quadrado, que mede a soma dos quadrados dos resíduos. corresponde aos coeficientes lineares, Y a matriz com os valores de saída e XX a matriz de autorregressão.

A solução desta equação resulta na estimativa dos coeficientes que proporcionam a melhor aproximação linear entre os valores preditos e observados.

Com os coeficientes estimados β', foi possível calcular os valores estimados Y' da variável dependente utilizando a equação (7):

$$Y' = XX \times \beta, \qquad (7)$$

onde,

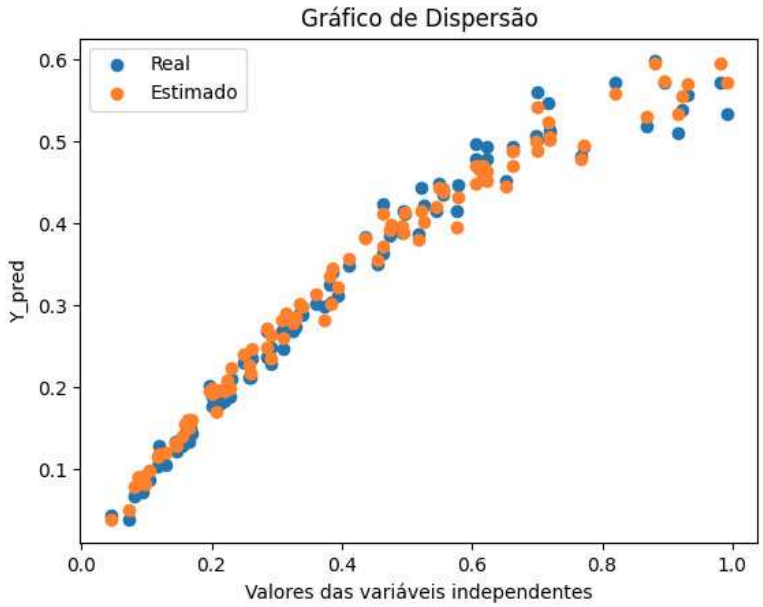
- Y' é o vetor de valores estimados da variável dependente.
- XX continua sendo a matriz de autorregressão.
- β' é o vetor dos coeficientes ajustados.

Dessa forma, os valores observados (Y) foram projetados no espaço dos valores estimados (Y'), permitindo uma comparação gráfica entre ambos. A Figura 7 ilustra os valores originais, representados em azul, e os valores estimados pelo modelo, em vermelho, ao longo do tempo. Já a Figura 8 apresenta a projeção cartesiana entre os valores de entrada e saída, onde os valores reais são indicados em azul, e os estimados, em vermelho.

Figura 7 – Curva das amostras originais e estimadas ao longo do tempo (Dados Adimensionais)


FONTE: O Autor, (2024).

Figura 8 – Gráfico de dispersão das entradas e saídas originais (Azul) e estimadas (Laranjado) (Dados Adimensionais).




FONTE: O Autor, (2024).

### 4.3 Treinamento com valores reais e predição com valores para teste

Nos itens 4.1 e 4.2, foram validados, respectivamente, o método dos mínimos quadrados e o polinômio de memória, demonstrando a eficiência de ambos para modelagem de sistemas dinâmicos. Nesta etapa do estudo, buscou-se estender a aplicação para um cenário mais realista, utilizando um conjunto de dados experimentais. Para isso, empregaram-se dados complexos, oriundos da resposta

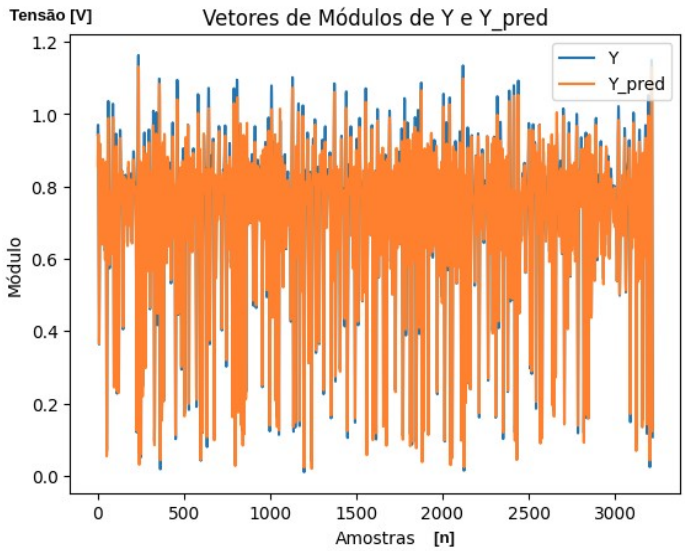
de um PA, obtidos por meio de simulações detalhadas. Esse conjunto de dados já estava previamente particionado em dois subconjuntos: um para treinamento e outro para validação.

Aplicando-se o método descrito na Equação 7, os coeficientes β foram estimados utilizando os dados do conjunto de treinamento. Posteriormente, os valores de β foram aplicados ao conjunto de validação, seguindo o procedimento descrito na Equação 8, para realizar a predição. Esse processo permitiu avaliar o desempenho do modelo em um contexto de generalização, crucial para validar a robustez do método proposto.

Para avaliar graficamente a eficácia das predições em relação aos valores originais do conjunto de validação, foram geradas as Figuras 9 e 10. A Figura 9 apresenta uma comparação direta entre os módulos dos valores originais de validação (em azul) e os valores preditos pelo modelo (em laranja). Esse gráfico ilustra o alinhamento entre as curvas, destacando o comportamento consistente do modelo em estimar os valores esperados.

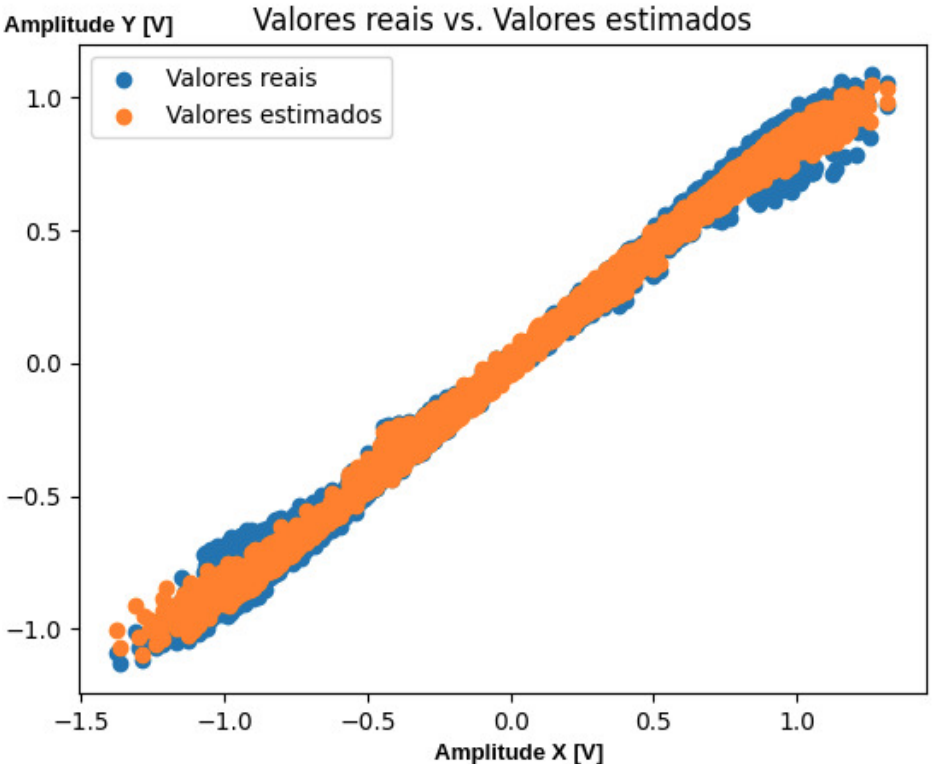
A Figura 10, por outro lado, fornece uma análise mais detalhada em uma projeção bidimensional, onde os valores originais e preditos são apresentados com base em suas respectivas entradas e saídas. Nesse gráfico de dispersão, o eixo horizontal representa as entradas, enquanto o eixo vertical retrata as saídas. A concentração dos pontos próximos à diagonal principal evidencia a alta precisão do modelo ao prever os valores do conjunto de validação.

Figura 9 – Curva das amostras originais e estimadas ao longo do tempo.




FONTE: O Autor, (2024).

Figura 10 – Gráfico de dispersão das entradas e saídas originais (Azul) e estimadas (Laranjado).

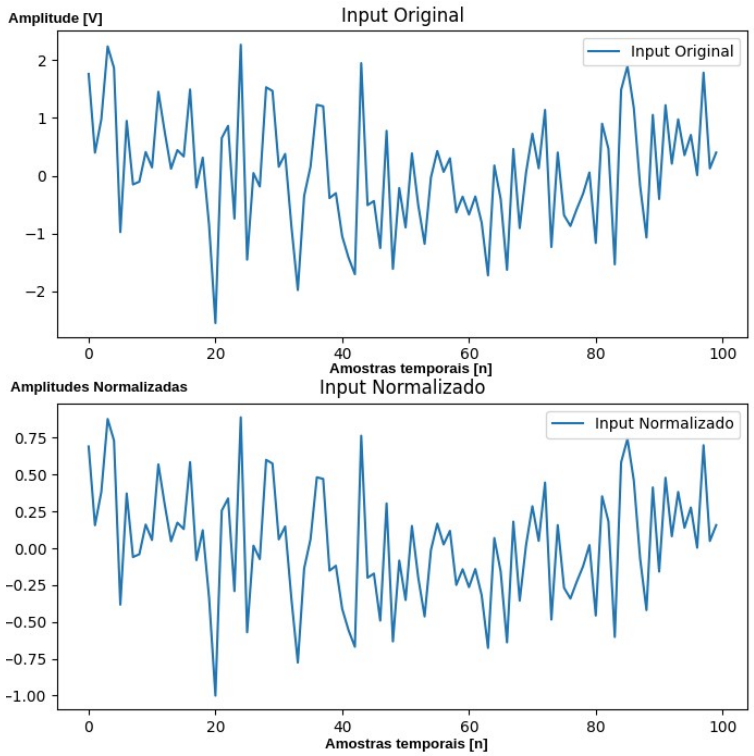



FONTE: O Autor, (2024).

### 4.4 Implementação do MP com LUTs em Python

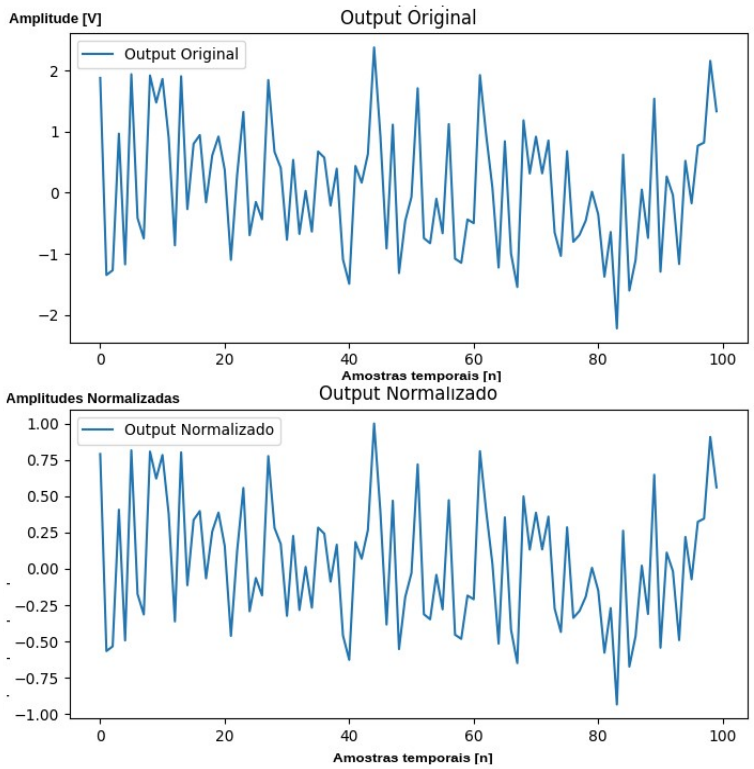
Após a validação dos modelos iniciais, avançou-se para a implementação das LUTs, um passo essencial para otimizar a utilização de recursos computacionais. Para isso, iniciou-se com a normalização dos dados. O processo consistiu em identificar o maior valor absoluto entre todos os módulos dos valores complexos do conjunto de dados e dividir cada valor por este máximo. Esse procedimento garantiu que os valores fossem redistribuídos em um intervalo entre 0 e 1, como ilustrado nas Figuras 11 e 12. Essas figuras apresentam, respectivamente, a transformação dos dados de entrada e saída, comparando seus estados originais e normalizados.

Figura 11 – Dados da entrada antes e após a normalização.




FONTE: O Autor, (2024).

Figura 12 – Dados da saída antes e após a normalização.




FONTE: O Autor, (2024).

Com os dados normalizados, foi gerado computacionalmente um conjunto de pontos uniformemente distribuídos no intervalo [0, 1]. A cada ponto desse conjunto foi atribuído um valor correspondente, calculado como a saída do polinômio de memória ao aplicar o ponto ao modelo. Cada termo do polinômio de memória gerou uma LUT independente, possibilitando uma implementação modular e eficiente.

A implementação das LUTs foi feita utilizando dicionários, uma estrutura de dados eficiente para armazenamento e recuperação de pares de valores. Essa abordagem permitiu que as LUTs fossem acessadas de forma dinâmica, com base no número de bits utilizados na discretização dos endereços.

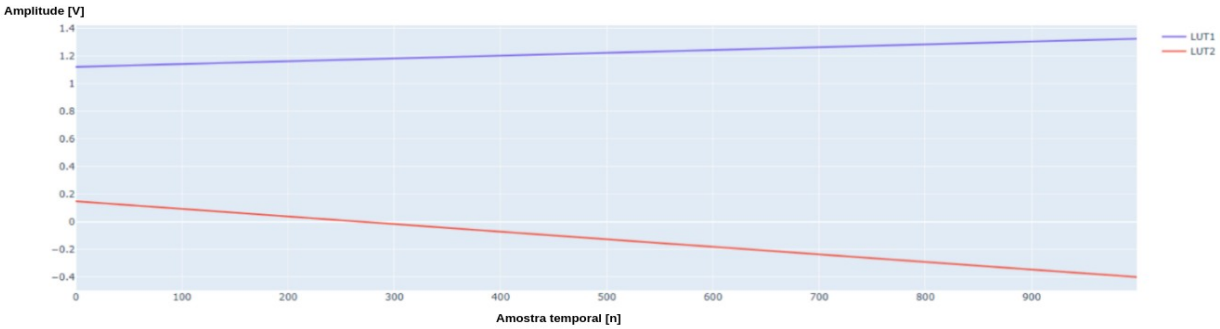
O número de bits definiu o número máximo de endereços nas LUTs. Por exemplo:

- Com 1 bit, os endereços possíveis são limitados a {0,1}{0,1}.
- Com 2 bits, os endereços expandiram para {00,01,10,11}{00,01,10,11}.
- Com 3 bits, os endereços são {000,001,010,011,100,101,110,111} {000,001,010,011,100,101,110,111}.

Esse processo de discretização gerou LUTs que armazenavam pares de valores correspondentes a cada intervalo. Em cada endereço da LUT foram armazenados dois coeficientes, *a* e *b*, que representavam uma reta que aproximava os valores do intervalo correspondente. Essa aproximação linear foi adotada para reduzir ainda mais a complexidade computacional no momento da consulta.

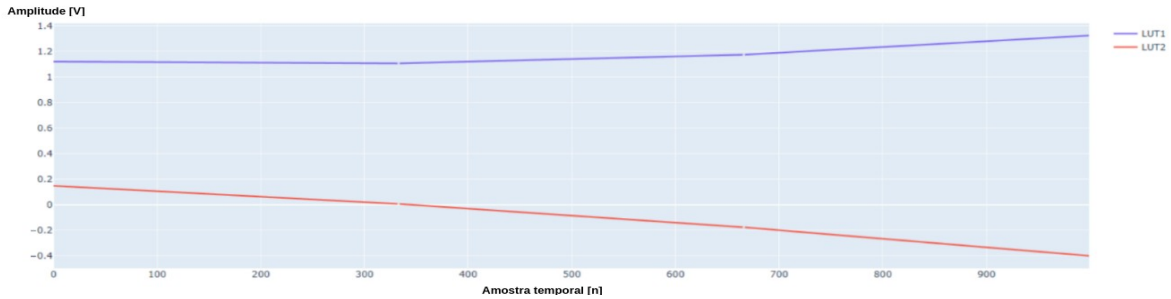
Durante a operação, os valores de magnitude eram comparados com os endereços da LUT. Para valores que não correspondiam exatamente a um endereço, aplicou-se a interpolação linear utilizando os coeficientes aa e bb previamente armazenados. A Figura 13 representa o comportamento das LUTs 1 e 2 com N=1, ou seja, com apenas um bit. Nesse caso, os valores interpolados mostram uma maior discrepância em relação à curva ideal, indicando que a granularidade limitada de 1 bit não permite capturar detalhes mais precisos da função representada. Por outro lado, a Figura 14 demonstra os resultados com N=3, ou seja, utilizando 3 bits. Com essa configuração, observa-se uma melhoria significativa na aproximação das LUTs em relação à curva original. A maior granularidade permite que os valores interpolados estejam mais próximos dos pontos reais, reduzindo as discrepâncias e aprimorando a precisão do modelo.

Figura 13 – Plotagem da saída das LUTs 1 e 2 quando N=1.




FONTE: O Autor, (2024).

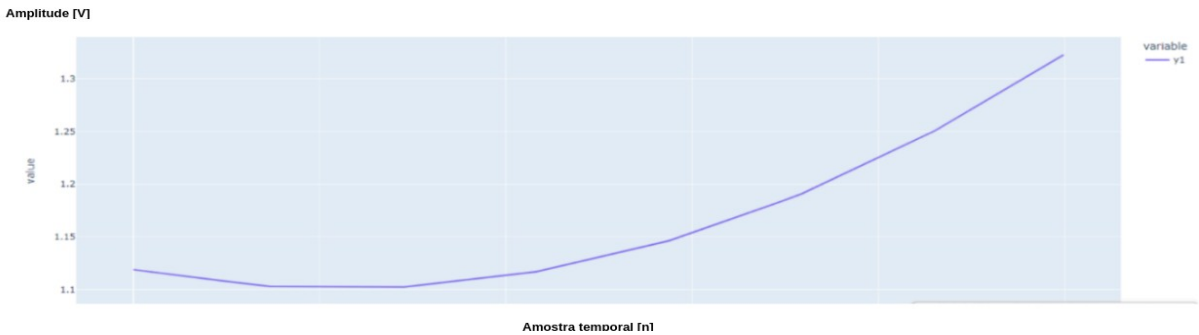
Figura 14 – Plotagem da saída das LUTs 1 e 2 quando N=2.




FONTE: O Autor, (2024).

Com apenas 3 bits, foi possível observar que as LUTs geraram uma aproximação visualmente consistente com a curva original, como demonstrado na Figura 14. Isso evidencia que mesmo com uma granularidade limitada, a resposta obtida já apresenta uma boa fidelidade ao comportamento esperado.

Figura 15 – Plotagem da saída da LUT quando N=3.




FONTE: O Autor, (2024).

A análise dos testes com LUTs variando o número de bits demonstra uma evolução clara na precisão do modelo, evidenciada pela redução progressiva do erro, medido em NMSE (dB). Quando a LUT utiliza apenas 1 bit (N=1), o erro é de -9,93 dB, apresentando uma melhoria significativa para -10,13 dB com 2 bits (N=2).

Ao expandir para 3 bits (N=3), o erro é reduzido ainda mais, alcançando -10,17 dB. Essa tendência mostra que, ao aumentar o número de bits na LUT, a precisão do modelo é aprimorada, refletindo uma maior fidelidade à curva original. Como exemplificado na Tabela 1, observa-se que, embora o ganho marginal entre N=2 e N=3 seja pequeno, ele ainda indica uma melhoria na precisão. Isso sugere que o incremento no número de bits contribui para a minimização do erro, embora os benefícios possam ser marginalmente menores à medida que N aumenta.

Tabela 1 – Erro Médio Quadrático Nornalizado (NMSE) em função do número de bits das LUTs.

| N | NMSE |
|---|---|
| 1 | -9,93 dB |
| 2 | -10,13 dB |
| 3 | - 10,17 dB |

FONTE: O Autor, (2024).

### 4.5 Descrição em VHDL

A transição da abordagem em Python para uma arquitetura de hardware demandou uma reformulação completa na forma como as operações eram organizadas e executadas; outro ponto é que na implementação em VHDL usou-se LUTs de 5 bits. Nesse novo contexto, o uso de sinais foi crucial, pois eles permitem a comunicação e propagação de dados entre as etapas do pipeline, garantindo que cada operação seja realizada em ciclos de clock distintos e de forma sincronizada. Essa abordagem foi essencial, pois as operações envolvem cálculos complexos, como manipulação de números complexos e interpolação via LUTs, que precisam ser divididos em etapas sequenciais para maximizar o desempenho e otimizar os recursos do hardware. Além disso, na implementação realizada em VHDL, houve uma limitação deliberada no formato dos valores. Tanto a parte real quanto a parte imaginária foram representadas apenas com números inteiros, ao contrário da abordagem original em Python, onde os valores eram manipulados com representação em ponto flutuante. Embora muitas operação sejam realizadas de modo concorrente, as tarefas foram organizadas sequencialmente. Estas são apresentadas a seguir a cada ciclo de relógio:

- **Primeiro ciclo: Elevar as entradas complexas ao quadrado**. Nesse estágio, os sinais são utilizados para armazenar os resultados das operações de multiplicação entre os componentes reais e imaginários de cada entrada. Cada operação é mapeada diretamente para hardware combinacional, e os sinais correspondentes são atualizados ao final do ciclo de clock.
- **Segundo ciclo: Somar o quadrado das entradas complexas**. Os sinais gerados no ciclo anterior são somados para calcular o módulo ao quadrado de cada entrada. Essa soma é propagada para sinais específicos, que serão utilizados no próximo ciclo. Essa divisão evita o acúmulo de latência, garantindo que cada etapa opere dentro do tempo alocado para o ciclo de clock.
- **Terceiro ciclo: Extrair os bits mais significativos do resultado do segundo ciclo.** Com os sinais do módulo ao quadrado já disponíveis, uma lógica de extração é implementada para capturar os bits mais significativos. Esses bits são armazenados em novos sinais, que funcionarão como chave para acessar os valores da LUT no próximo ciclo.
- **Quarto ciclo: Endereçar os valores da LUT usando como chave a saída do terceiro ciclo.** Os sinais que contêm os bits mais significativos são usados para indexar a LUT. Essa operação garante que os coeficientes a e b correspondentes ao intervalo do valor calculado sejam recuperados e armazenados em sinais próprios.
- **Quinto ciclo: Realizar o produto dos coeficientes “a” da LUT com a soma do quadrado das entradas.** Nesse estágio, os sinais contendo os coeficientes ‘a’ são multiplicados pelos valores de módulo ao quadrado armazenados no segundo ciclo. O resultado dessa multiplicação é propagado para sinais intermediários.
- **Sexto ciclo: Somar o resultado do quinto ciclo com os coeficientes “b” da LUT.** A interpolação linear é completada aqui somando o produto calculado no quinto ciclo com os coeficientes ‘b’ extraídos da LUT. O resultado é armazenado em sinais que serão utilizados na próxima etapa.
- **Sétimo ciclo: Realizar o produto do resultado do sexto ciclo com as entradas complexas.** Utilizando os sinais das entradas originais e os valores interpolados calculados no sexto ciclo, realiza-se o produto que ajusta os valores interpolados às entradas complexas. O resultado é armazenado em dois sinais distintos: um para a parte real e outro para a imaginária.

- **Oitavo ciclo: Somar isoladamente parte real e parte imaginária.** No ciclo final, os sinais contendo as partes real e imaginária são somados separadamente, consolidando o resultado final. Esses sinais representam a saída do sistema, pronta para ser utilizada ou armazenada.

O uso de sinais para operações síncronas foi necessário porque, em uma arquitetura de hardware, sinais são representados como *wires* ou fios, cujos valores são propagados são atualizados uma única vez ao fim de cada ciclo, isso facilita que os resultados estejam sincronizados e disponíveis para as etapas subsequentes.

### 4.6 Implementação e Simulação em FPGA

A implementação foi realizada utilizando a FPGA Virtex5 XC5VLX20T, cujos recursos permitiram acomodar o design de forma eficiente e funcional. A análise dos recursos utilizados, conforme apresentado na Tabela 2, evidencia uma alocação equilibrada e otimizada dos componentes de hardware disponíveis.

O consumo de 386 flip-flops, correspondente a apenas 3% dos 23.480 disponíveis, demonstra que o projeto requer uma quantidade modesta de lógica sequencial. Isso reflete a eficiência do design, que priorizou o uso de sinais em vez de elementos de armazenamento intermediário, reduzindo a necessidade de flip-flops para a propagação de valores entre os ciclos de clock. Essa escolha também contribuiu para uma utilização mais econômica dos recursos, garantindo que o projeto permanecesse escalável e adaptável a mudanças futuras.

Outro grupo de recursos consumido extensivamente foram os Blocos de Lógica Combinacional; para esses, apresentou-se uma utilização de 12%, totalizando 1.590 das 12.480 disponíveis. Esse resultado é consistente com as características do design, que faz uso significativo de recursos lógicos combinacionais para operações matemáticas, como multiplicações e somas, e para o endereçamento das LUTs de coeficientes. Dividir as tarefas em 8 ciclos permitiu uma configuração em *pipeline*, onde a cada novo ciclo, novas entradas eram processadas.

O consumo de 4 blocos de RAM, equivalente a 15% dos 26 disponíveis, está relacionado ao armazenamento dos coeficientes das LUTs e dos dados intermediários necessários para a interpolação e cálculos subsequentes.

Além disso, foram utilizados 24 DSPs (Digital Signal Processing) disponíveis na FPGA, evidenciando o alto nível de operações realizadas.

Tabela 2 – Relação dos recursos usados e disponíveis na FPGA utilizada

| Recurso | Usados | Disponível | Utilizável |
|---|---|---|---|
| Flip Flops | 386 | 23480 | 3% |
| Blocos de Lógica Combinacional | 1590 | 12480 | 12% |
| Blocos RAM | 4 | 26 | 15% |
| DSP | 24 | 24 | 100% |

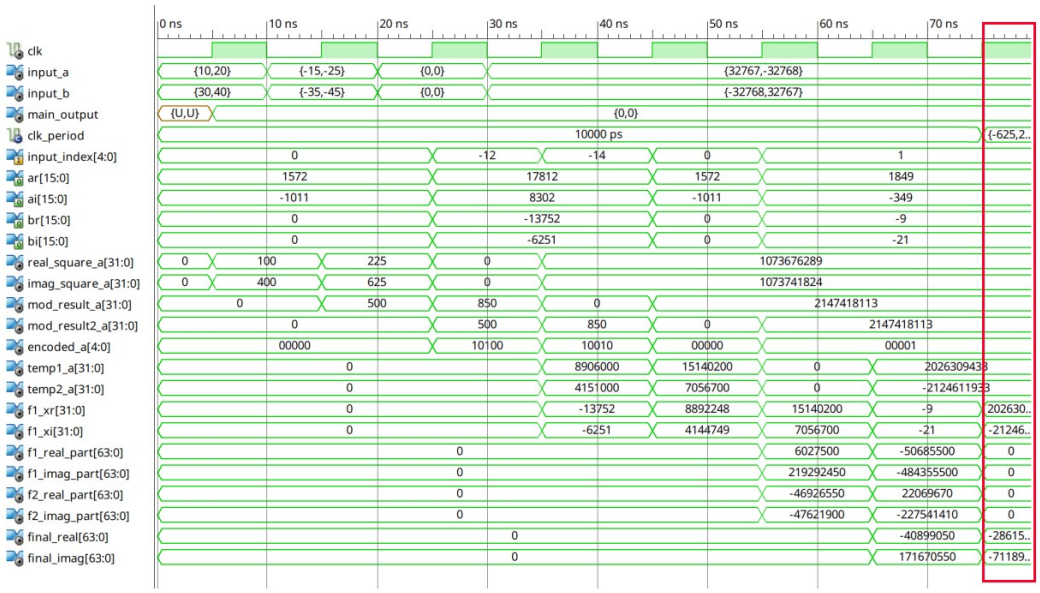
FONTE: O Autor, (2024).

A simulação e implementação do projeto foram realizadas utilizando o ambiente de desenvolvimento Xilinx ISE 14.7, com a simulação sendo conduzida na ferramenta ISim, integrada ao ISE. Esse fluxo de trabalho proporcionou uma análise detalhada do comportamento funcional e temporal do design, assegurando que a arquitetura projetada fosse validada. A ferramenta ISim permitiu validar o funcionamento do pipeline do projeto, estruturado para operar em 8 ciclos de clock. Essa abordagem possibilita que a primeira saída seja disponibilizada após 8 ciclos, enquanto novas entradas já podem ser processadas simultaneamente em ciclos subsequentes. Esse comportamento foi confirmado durante a simulação, na qual as operações de cada estágio do pipeline foram verificadas, garantindo a correta propagação de sinais entre os estágios e a sincronização com o clock. Durante a simulação, diversos estímulos foram aplicados às entradas, incluindo valores positivos, negativos, zero e valores extremos. A análise mostrou que o design comportou-se como esperado em todas as condições. O atraso de 8 ciclos até a geração da primeira saída foi observado conforme planejado, e as saídas subsequentes eram produzidas em ciclos consecutivos, confirmando o bom funcionamento do pipeline na manutenção de um fluxo contínuo de processamento. A implementação do projeto na FPGA validou a integração da arquitetura em hardware.

A Figura 16 apresenta a simulação realizada através do ISim para a FPGA escolhida para uma frequência de clock de 100 MHz; nela pode-se visualizar a propagação dos sinais ao longo dos 8 ciclos, demonstrando o funcionamento do pipeline implementado. Observa-se como os sinais intermediários, desde os valores de entrada até os cálculos finais, são processados em cada estágio do sistema. A região destacada em vermelho indica o momento em que a primeira saída válida é

disponibilizada, representando o término do período inicial de latência necessário para o processamento completo dos primeiros dados. A partir deste ponto, uma nova saída é gerada a cada ciclo de clock, pois o sistema recebe um novo par de entradas em cada ciclo, mantendo a continuidade do processamento.

Figura 16 – Simulação do DPD em FPGA.




FONTE: O Autor, (2024).

## 5. CONCLUSÃO

Este trabalho apresentou o desenvolvimento e implementação de um DPD utilizando polinômios de memória, descritos em VHDL, com foco em otimizar a linearidade e eficiência energética de amplificadores de potência (PAs). O processo de implementação incluiu uma abordagem sistemática que passou pela validação de modelos matemáticos, simplificação computacional por meio de LUTs, abstração em linguagem de hardware e síntese lógica em FPGA. A transição da simulação em Python para a implementação no hardware revelou os desafios intrínsecos à adaptação de operações complexas em ambientes de recursos limitados.

Os resultados obtidos evidenciam o sucesso do modelo proposto em termos de precisão e eficiência. A implementação das LUTs permitiu reduzir a complexidade computacional, enquanto a segmentação em ciclos de clock garantiu uma operação síncrona, organizada em um pipeline de 8 etapas. O uso das LUTs com discretização de 5 bits se mostrou eficaz na aproximação de curvas não lineares, mantendo um equilíbrio entre granularidade e consumo de recursos. Durante os testes, verificou-se uma melhoria progressiva na precisão conforme o número de bits das LUTs aumentava, atingindo um NMSE de -10,17 dB com 3 bits, evidenciando a robustez da técnica.

A implementação em FPGA, realizada no modelo Virtex5 XC5VLX20T, demonstrou um uso adequado dos recursos disponíveis. O consumo de 12% dos blocos de lógica computacional, 15% dos blocos RAM e 3% dos flip-flops destacou um baixo uso dos recursos disponíveis. O uso integral dos 24 DSPs se deu porque durante a síntese as operações aritméticas foram por padrão alocadas nas DSPs.

No entanto, o trabalho enfrentou desafios significativos, especialmente na adaptação do modelo matemático original, que usava ponto flutuante, para uma representação em inteiros. Essa mudança exigiu um cuidado adicional na normalização e manipulação dos valores, além da necessidade de gerenciar operações complexas dentro das limitações do hardware. A integração do pipeline também demandou uma organização síncrona dos sinais.

Em síntese, o projeto demonstrou ser uma solução viável para a linearização de PAs, com potencial aplicação em dispositivos de telecomunicação e sistemas embarcados. Apesar das limitações enfrentadas, os resultados apresentados indicam que o modelo proposto não só alcança os objetivos definidos como oferece possibilidades de avanços futuros, como o desenvolvimento de circuitos integrados personalizados em uma arquitetura que permita representação dos dados de entrada em ponto flutuante.

## REFERÊNCIAS


1. MACHADO, Carolina Luiza Rizental. Modelagem Comportamental de Amplificadores de Potência Usando Soma de Produtos entre Filtros Digitais de Resposta ao Impulso Finita e Tabelas de Busca Unidimensionais. Dissertação (Mestrado em Engenharia Elétrica) - Universidade de Curitiba, Curitiba, 2016.
2. SCHUERTZ, J. R. et al. Circuito digital aplicado a linearização de um amplificador de potência. SeMicro, Curitiba, v. 14, n. 2, p. 45-60, 2021.
3. SCHUARTZ, L.; LIMA, E. G. Comparison among Algorithms for the Identification of Adaptative Memory Polynomial Predistorter Models. 30º Simpósio Sul de Microeletrônica, maio de 2015, p. 1-4.
4. CRIPPS, Steve C. RF Power Amplifiers for Wireless Communications. Artech House, Inc., 2006.
5. KENINGTON, Peter B. High linearity RF amplifier design. Artech House, Inc., 2000.
6. PEDRONI, Volnei. Eletrônica digital moderna e VHDL. São Paulo: Editora GEN LTC, 2010.
7. GIOFRE, Rocco; PIAZZON, Luca; COLANTONIO, Paolo; GIANNINI, Franco. A Doherty architecture with high feasibility and defined bandwidth behavior. IEEE Transactions on Microwave Theory and Techniques, 2013.
8. KIM, J.; KONSTANTINOU, K. Digital predistortion of wideband signals based on power amplifier model with memory. Electronics Letters, 2001.
9. KANG, Daehyun; PARK, Byungjoon; KIM, Dongsu; KIM, Jooseung; CHO, Yunsung; KIM, Bumman. Envelope-tracking CMOS power amplifier module for LTE applications. IEEE Transactions on Microwave Theory and Techniques, 2013.
10. SCHUARTZ, Luis; WOSNIACK, Isabella F.; SCHOULTEN, Felipe A.; SILVEIRA, Émeli L. C.; FRANÇA, Sibilla B. L.; LIMA, Eduardo G. Configuração em aritmética de vírgula fixa para síntese em FPGA dos modelos MP, EMP e CMEMP. Semicro PR, 2018.