UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE ENGENHARIA ELÉTRICA

Leonardo de Andrade Santos

# Projeto de um circuito integrado de um pré-distorcedor digital baseado em polinômio de memória

**Curitiba**
**2024**

Leonardo de Andrade Santos

# Projeto de um circuito integrado de um pré-distorcedor digital baseado em polinômio de memória

Trabalho de conclusão de curso do Curso de Graduação em Engenharia Elétrica da Universidade Federal do Paraná, como exigência parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientadora: Sibilla Batista da Luz França

**Curitiba**
**2024**

# Lista de abreviaturas e siglas

| | |
|---|---|
| DPD | Pré-Distorcedor Digital |
| FPGA | Matriz de Portas Programáveis em Campo |
| PA | Amplificador de Potência |
| RF | Rádio Frequência |
| PARF | Amplificador de Potência de Rádio Frequência |
| HDL | Linguagem de Descrição de software |
| VHSIC | Circuito integrado de Velocidade Muito Elevada |
| VHDL | VHSIC Hardware Description Language |
| LUT | Look Up Table |
| SOP | Soma de Produtos |
| LAB | Logic Array Block |
| ALM | Adaptive Logic Module |
| LE | Logic Element |
| HEMT | transistor de efeito de campo de heterojunção |
| VSA | analisador de sinal vetorial |
| NMSE | Erro Médio Quadrado Normalizado |

# Resumo

A evolução dos sistemas de comunicação sem fio acarretou na implementação de diversas aplicações móveis e sem fio como desenvolvimento web, aplicação IoT, entre outros. Neste cenário, melhorar a eficiência energética se torna uma alternativa desejável tanto para os dispositivos móveis que buscam melhorar a autonomia das suas baterias, quanto para as estações de rádio base, que buscam reduzir sues desperdício em perdas de calor. No entanto, uma melhor eficiência energética implica em uma menor linearidade nos sistemas de amplificação de sinais, presentes nos sistemas transmissores de sinais de rádio. Isto é importante de ser ressaltado, pois a banda reservada para aplicações móveis é reduzida, de forma que para se alcançar maiores taxas de transmissão é necessário alternar estratégias de modulação tanto da fase, quanto da amplitude da onda portadora. E essas duas condições são conflitosas, já que a modulação AM é sensível a linearidade de forma que quanto mais linear um sistema ocorrem menos erros de transmissão. Sendo assim, uma alternativa para contornar esse obstáculo, que é implementar um sistema, eficiente energeticamente e linear é a implementação de um DPD em cascata com um PA. Portanto, o objetivo deste trabalho de conclusão de curso é o design de um circuito integrado dedicado de um DPD. Para atingir esse objetivo esse projeto foi divido em quatro etapas, o estudo e modelagem dos DPDs, modelagem do DPD em software, implementação do DPD em FPGA e finalmente o design do circuito integrado do DPD. Para a modelagem do DPD foi utilizada a métrica do NMSE, nela quanto menor o NMSE encontrado mais fiel e o modelo com a realidade. Sendo assim, na etapa de modelagem do PA alcançou-se um NMSE de -23.57 db. Em seguida, foi feita o levantamento do número de bits necessários para a realização desses cálculos de forma a minimizar o NMSE, para isso foi verificado que com apenas 8 bits de resolução do sinal já foi possível alcançar um NMSE próximo do valor alcançado em virgula flutuante. Portanto, ainda restam as etapas de implementação do DPD em FPGA e o design do circuito integrado.

**Palavras-chave**: VHDL, FPGA, DPD .

# Lista de ilustrações

# Sumário

# 1 Introdução

A evolução dos sistemas de comunicação móveis, impulsionada pela crescente demanda por comunicações mais rápidas e eficientes, tem levado à implementação de uma variedade de serviços, incluindo aplicações multimídia, desenvolvimento web e aplicações IoT (1). No entanto, essa evolução também trouxe desafios significativos, como a necessidade de melhorar a eficiência energética, tanto para dispositivos móveis, visando aumentar a autonomia da bateria, quanto para estações de rádio base, visando reduzir o consumo de energia devido às perdas de calor. Para atender a essas demandas, estratégias de modulação que alteram tanto a fase quanto a amplitude de ondas portadoras em radiofrequência se tornaram essenciais (2). Além disso, a modulação na amplitude requer linearidade na transmissão para evitar erros e interferências na comunicação entre usuários vizinhos (3). Essa complexa tarefa recai sobre o projetista do PARF (Amplificador de Potência de Rádio Frequência), que enfrenta o desafio de desenvolver um hardware eficiente em termos energéticos e linear ao mesmo tempo, uma vez que esses dois objetivos podem entrar em conflito (4). Uma solução para contornar esse desafio é a implementação de um pré-distorcedor de Sinais Digital em Banda Base, que visa compensar a distorção causada pelo PARF (3). O DPD (Pre-distorcedor Digital) é conectado em cascata ao PARF e requer um modelo de alta precisão e baixa complexidade computacional para representar as características de transferência direta e inversa do PARF. Existem duas abordagens para modelar o PARF: modelos físicos, que são detalhadas e computacionalmente complexos, e modelos empíricos, que se baseiam em medições de entrada e saída do PARF, com menor complexidade computacional, mas com uma possível diminuição da precisão. Devido às exigências rigorosas de frequência de operação, a paralelização das operações torna-se essencial, e as FPGAs (Matriz de Portas Programáveis em Campo) emergem como uma alternativa viável para a implementação de circuitos pré-distorcedores (5). As FPGAs são dispositivos lógicos programáveis que permitem a reconfiguração física de componentes de eletrônica digital, acelerando processos e suportando operações paralelas e sequenciais. Neste contexto esse projeto foi planejado com os seguintes objetivos geral e específicos:

## 1.1 Objetivo Geral

O desenvolvimento de um circuito integrado dedicado para um pré-distorcedor digital na tecnologia BiCMOS 130 nm 8HP.

## 1.2 Objetivos Específicos

Para alcançar o objetivo geral, este trabalho foi desenvolvido com base nos seguintes objetivos específicos:

1. Modelar com precisão o PA em software;
2. Modelar o DPD em software a partir da modelagem do PA;
3. Implementar o DPD em hardware utilizando uma HDL;
4. Desenvolver o design do circuito integrado.

# 2 Revisão de Literatura

A evolução dos sistemas de comunicações sem fio, fomentou a implementação de diversas aplicações móveis. Neste contexto, melhorar a eficiência energética desses sistemas se torna uma característica desejável, tanto para os dispositivos móveis que buscam melhorar a autonomia das baterias quando para os sistemas de comunicação radio base que, tendem a perder energia em forma de calor. Sendo assim, o sistema de comunicação pode ser dividido em 3 sub-sistemas principais: Meio transmissor, Receptor e o Transmissor, conforme argumentado por (6). No entanto, este trabalho foca exclusivamente no sistema transmissor, ilustrado pela figura 1, onde observa-se diversos componentes que compõem o transmissor de sinal entre esses elementos o amplificador de potência é o componente de maior demanda energética, por se tratar do componente que converte a energia da fonte em energia irradiada pela antena de transmissão. Portanto, a eficiência do sistema de transmissão depende diretamente do desempenho do transmissor.

Fonte: (6)

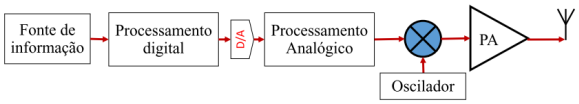


Figura 1 – Sistema de transmissão simplificado

Considerando também que a largura de banda reservada para sistemas de comunicação sem fio é reduzida, torna-se desejável que ela seja utilizada da maneira mais eficiente o possível. Diante desse cenário, segundo (2), só é possível alcançar as maiores taxas utilizando estratégias de modulações, que alterem tanto a fase quanto a amplitude de uma onda portadora em rádio frequência. Ainda segundo (2), a modulação na amplitude, exige linearidade na transmissão afim de evitar erros e interferência na comunicação entre os usuários vizinhos. Ante esse panorama, o projetista do PARF se depara com esse desafio, que é desenvolver um hardware eficiente energeticamente e com uma boa linearidade, o que é um compromisso é conflitante, conforme descrito por (3). Esse comportamento se deve, ao fato que um PARF atua de forma eficiente, ou seja, com baixo consumo de energia, na área próxima à de saturação, que é a região em que opera em regimes não lineares, conforme ilustrado pela figura 2.

A fim de contornar esse obstáculo foi adicionado a cadeia de transmissão um método de equalização de sinais, conforme argumentado por (2). Um exemplo de técnica de linearização de sinais é a implementação de um pré-distorcedor de sinais digitais em banda base, o qual apresenta um melhor custo-benefício (2). Essa técnica consiste em distorcer o sinal de entrada utilizando técnicas de processamento digital, antes que esse module uma

Fonte: (4)

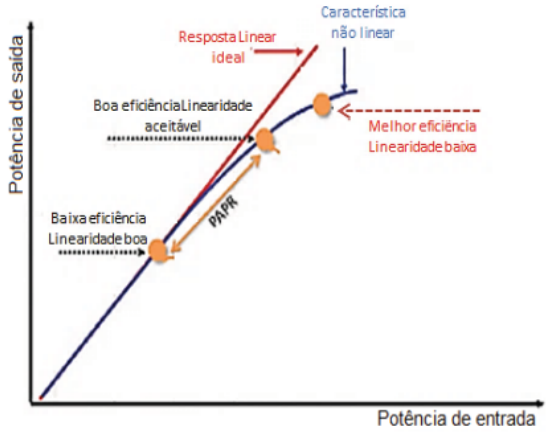


Figura 2 – Curva de saida do amplificador

portadora, de forma compensativa à distorção causada pelo PARF. De maneira sucinta, o DPD é conectado em cascata ao PARF e é projetado de forma que apresenta a função transferência inversa ao PARF. Para isso, é necessário um modelo de alta precisão e baixa complexidade computacional, capaz de representar as características de transferência direta e inversa de um PARF. Isso significa modelar o seu comportamento real utilizando um software. A figura 3 ilustra o processo do um pré-distorcedor digital.

Fonte: (4)

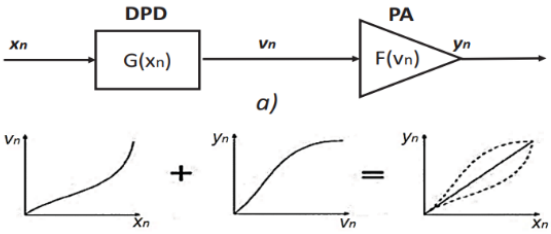


Figura 3 – ilustração do pré-distorcedor em cascata

Segundo (1), existem duas técnicas utilizadas para fazer essa modelagem. Uma consiste na descrição detalhada do PARF, que implica em uma maior complexibilidade computacional, esses modelos são conhecidos como modelos físicos. A outra abordagem é conhecida como modelo empírico, este modelo consiste em coletadas amostradas na entrada e na saída do PARF em domínio temporal, e através destes dados simulam um modelo matemático do sistema. Uma das vantagens desse método é que ele não exige conhecimento prévio da estrutura do PARF e possui baixa complexidade computacional e mesmo se todos os parâmetros fossem conhecidos o equacionamento completo do circuito, uma função inversa poderia ser encontrada, possivelmente muito mais complexa que séries de Volterra. No entanto, sua precisão pode ser ligeiramente afetada pelo modelo adotado. Sendo assim, como a proposta do projeto é a implementação de um DPD em hardware,

se faz necessário que o circuito apresente a menor complexidade possível, torna-se mais viável fazer a implementação utilizando modelagem matemática.

## 2.1 Modelagens Matemáticas

### Séries de Volterra

Segundo (7) série de Volterra pode ser vista como uma extensão multidimensional da série de Taylor para sistemas dinâmicos. A modelagem começa com a representação do sistema através de uma série infinita de integrais convolucionais, onde cada termo da série corresponde a uma ordem de não linearidade e memória.

A saída $y(t)$ de um sistema pode ser expressa pela equação 2.1:

$$y(t) = h_0 + \sum_{n=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \tau_2, \ldots, \tau_n) \prod_{i=1}^{n} x(t - \tau_i)\, d\tau_i \tag{2.1}$$

onde $h_n$ são os núcleos de Volterra, que caracterizam a resposta do sistema para a $n$-ésima ordem de não linearidade e $x(t)$ é a entrada do sistema.

Os núcleos de Volterra $h_n$ são funções de várias variáveis que capturam a dinâmica do sistema em diferentes ordens. Para a maioria das aplicações práticas, a série é truncada para incluir apenas um número finito de termos, já que a identificação de todos os núcleos de uma série infinita é impraticável.

### Polinômio de memória

Um modelo simples, utilizado na modelagem comportamental simplificada das séries de Volterra considerando apenas componentes unidimensionais[1] é o MP, que é um modelo compacto, de baixo custo computacional e linear em seus parâmetros. O MP gera baixo erro quando aplicado à PAs que apresentam pouco efeito de memória. O DPD e pós distorsor apresentam característica inversa a do PA (6), portanto o mesmo modelo pode ser utilizado. A equação 2.2 apresenta o MP conforme apresenta (6):

$$y(n) = \sum_{p=1}^{P} \sum_{m=0}^{M} h_{p,m} x(n-m)\, |x(n-m)|^{p-1} \tag{2.2}$$

Como a proposta do trabalho é a implementação em hardware desse modelo, torna-se necessário paralelizar operações aritméticas de forma a alcançar uma taxa de operação que satisfaça a norma regulamentadora. Nesse contexto, as FPGAs apresentam-se como uma alternativa viável para a implementação de circuitos pré-distorcedores

[1] Cada termo do somatório é composto por amostras no mesmo instante, por exemplo: $x(n)|x(n)|, x(n-1)|x(n-1)|$; termos bidimensionais são compostos por amostras em instantes de tempos distintos, como por exemplo: $x(n)|x(n-1)|$

## 2.2 FPGAs

Como descrito em (5), FPGAs são uma classe de dispositivos lógicos programáveis que permitem a reconfiguração física de seus componentes de eletrônica digital por meio de uma linguagem de descrição de hardware. Basicamente, as FPGAs consistem em um conjunto de subcircuitos digitais interconectados, capazes de realizar diversas funções comuns enquanto oferecem um alto nível de flexibilidade. Devido a essas características, FPGAs podem ser utilizadas para aplicações como processamento de imagem em tempo real e aprendizado de máquina.

FPGAs têm a capacidade de sintetizar arquiteturas complexas de eletrônica digital, resultando em um funcionamento altamente paralelizado que permite um processamento rápido com várias portas de entrada e saída. Além disso, elas também suportam o desenvolvimento de códigos sequenciais.

A estrutura interna de uma FPGA é composta fundamentalmente por blocos lógicos interligados, organizados em uma matriz. Cada bloco é formado por diversos sub-blocos, que por sua vez contêm os componentes mais básicos da hierarquia. FPGAs da Intel e da Xilinx possuem nomenclaturas e organizações diferentes para esses blocos e sub-blocos. Isso é ilustrado na Figura 4 e na Figura 5, que mostram as FPGAs Intel Stratix X e Xilinx Ultrascale+, respectivamente. Embora as arquiteturas sejam fundamentalmente semelhantes, com a disposição em matriz dos blocos e funcionalidades dos componentes fundamentais sendo universais, os blocos lógicos são denominados LAB nas FPGAs da Intel e CLB nas FPGAs da Xilinx. Os sub-blocos são chamados de ALM ou LE, dependendo da FPGA da Intel, e de Slices nas FPGAs da Xilinx.

Fonte: (5)

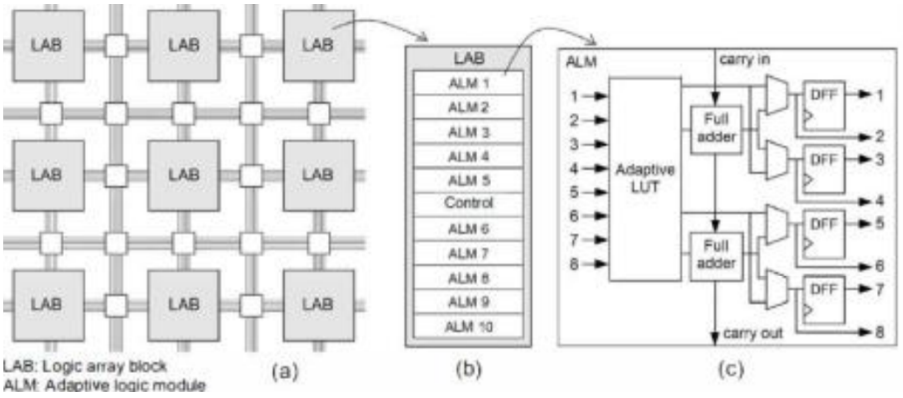


Figura 4 – Estrutura Interna da FPGA Stratix X da Intel

Os sub-blocos das FPGAs são compostos por LUTs e registradores. As LUTs são compostas por uma árvore binária de multiplexadores 2:1, permitindo o armazenamento de uma função lógica na forma de SOP. Os registradores são os componentes síncronos dos sub-blocos. Além da estrutura mencionada, FPGAs comumente possuem diversos

Fonte: (5)

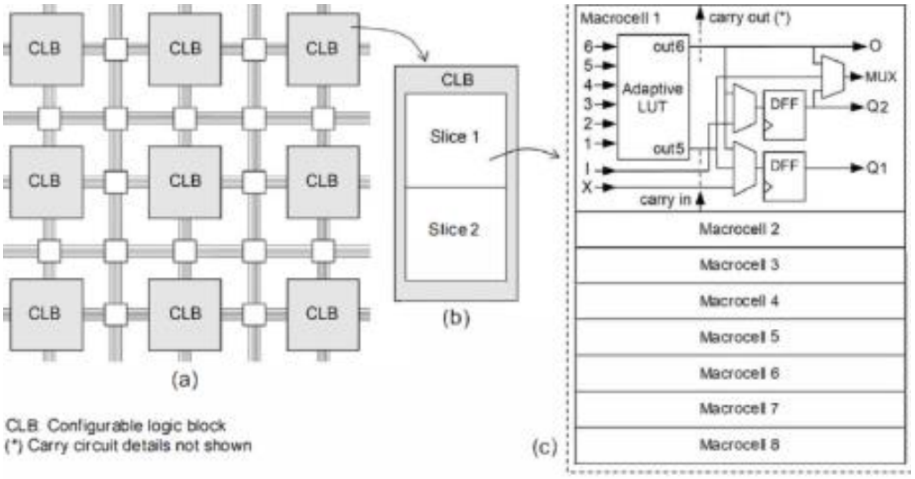


Figura 5 – Estrutura Interna da FPGA Ultrascale+

módulos integrados, como CPUs, DSPs, memória Flash, PLLs, que aumentam ainda mais as capacidades do FPGA.

As FPGAs são programadas utilizando uma linguagem de descrição de hardware, sendo o VHDL uma das mais comuns para a síntese de circuitos integrados de alta velocidade. Criada por uma iniciativa financiada pelo Departamento de Defesa dos Estados Unidos em meados dos anos 80, o VHDL foi a primeira linguagem de descrição de hardware padronizada pela IEEE.

A estrutura de um código VHDL consiste em três partes principais: declaração de bibliotecas/pacotes, entidade e arquitetura. Na primeira parte, são listadas as bibliotecas e pacotes necessários para o projeto. As bibliotecas padrão incluem a ‘std‘ e a ‘work‘. A entidade, que é a interface do sistema, descreve as entradas e saídas e é dividida em duas partes: parâmetros e conexões. Os parâmetros são valores constantes, como a largura de um barramento, que são declarados como genéricos. As conexões, por sua vez, definem a transferência de informações e correspondem aos pinos de entrada e saída do circuito. Já a arquitetura é a parte principal do sistema, onde o circuito é descrito. Nessa seção, são definidas as atribuições, operações lógicas e aritméticas, comparações, entre outros. Há também uma parte declarativa da sintaxe, que apresenta uma ampla variedade de declarações possíveis.

Dessa forma, circuitos digitais para processamento de sinais em tempo real são muito utilizados em sistemas de comunicações sem fio. Um exemplo de aplicação são os DPDs para transmissores sem fio. Os DPDs são baseados em operações matemáticas que envolvem uma grande quantidade de somas, produtos e tabelas de busca. Devido às rigorosas exigências de frequência de operação, torna-se fundamental a paralelização das operações necessárias. Nesse contexto, as FPGAs são uma alternativa viável para a implementação de circuitos DPDs.

# 3 Material e Métodos

Como mencionado anteriormente, este trabalho tem como objetivo desenvolver o design de um circuito integrado para um DPD, partindo de um modelo previamente validado tanto em software quanto em hardware, especificamente em FPGA. O projeto foi dividido em quatro etapas principais:

- Estudo do DPD;
- Implementação em software;
- Implementação em FPGA;
- Design e validação do circuito integrado.

## 3.1 Estudo dos DPDs

A etapa consistiu no estudo dos DPDs, conforme apresentado no Capítulo 2, onde foi feito todo o levantamento sobre os tipos de modelagem dos DPDs. O objetivo deste estudo é entender as diferentes abordagens de modelagem, avaliar seus desempenhos e identificar as mais adequadas para a aplicação em amplificadores de potência.

## 3.2 Implementação em software

Nesta etapa, foi realizada a implementação do modelo DPD em software, utilizando a linguagem de programação Python. Esta linguagem amigável e amplamente difundida na comunidade acadêmica.

Para essa modelagem, foram coletados sinais de entrada e saída de um amplificador de potência classe AB, que utiliza um HEMT fabricado com tecnologia GaN. O amplificador foi excitado por um sinal portador de frequência de 900 MHz, modulado por um sinal de envelope WCDMA 3GPP com aproximadamente 3,84 MHz de largura de banda. Os dados de entrada e saída do amplificador de potência foram medidos usando um VSA Rohde & Schwarz FSQ com uma taxa de amostragem de 61,44 MHz, conforme disponível em (8). Claro! Aqui está o parágrafo reescrito:

Em seguida, realizou-se o cálculo da estimativa do sinal utilizando números com vírgula fixa. Para verificar a precisão dessa estimativa em relação ao sinal original, calculou-se o NMSE. Para essa validação, os dados foram inicialmente divididos em conjuntos de extração e validação. A matriz de confusão foi calculada com os dados de extração, utilizando o código disponível no anexo 5. Esse cálculo é essencial para a extração dos

coeficientes do polinômio de memória. Após a extração dos coeficientes, calculou-se o modelo do PA, que foi então validado com os dados de validação. O NMSE obtido para um polinômio de 2° grau com uma amostra memorizada foi de -23,57 dB.

Em seguida, o algoritmo foi ajustado para operar com números em vírgula fixa e o número total de bits foi reajustado para atingir a menor resolução possível, buscando o menor NMSE simulado, conforme ilustrado pelo anexo 5. Por ser tratar de um cálculo em virgula fixa, fez-se necessário uma readequação do resultado obtido entre cada multiplicação de forma a manter a resolução inicial.

## 3.3 Implementação em FPGA

Essa etapa consiste na implementação do DPD em FPGA. Para isso, é necessário realizar paralelizações nas operações aritméticas. A Figura 6 ilustra como esse processo está dividido entre cada ciclo de clock. Aqui está o trecho reescrito de forma mais clara: A cada ciclo, duas operações são realizadas em paralelo: o sinal atual é elevado ao quadrado e registrado, enquanto ocorre o somatório do produto entre os sinais do mesmo instante de tempo e seus respectivos coeficientes. Esse processo ocorre P vezes para os $P$ graus do polinômio de memória. Portanto, a saída do DPD é incompleta para os primeiros $P$ períodos de clock, pois, nesses primeiros ciclos, realiza-se o cálculo com base em entradas de sinais anteriores que ainda não ocorreram, resultando em uma saída incompleta.

Fonte: Autor

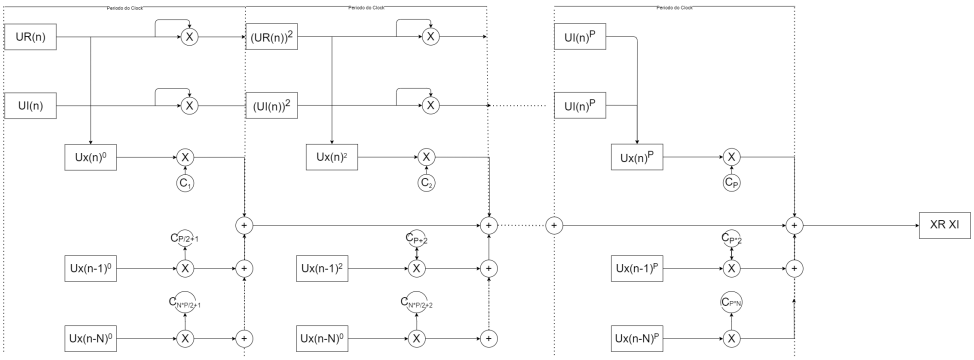


Figura 6 – Processo de cálculo da saída

## 3.4 Design e validação

Finalmente, na última etapa, realiza-se o processo de concepção do circuito integrado do DPD como um circuito dedicado integrado na tecnologia BiCMOS 130 nm 8HP, utilizando as ferramentas específicas de desing de circuito integrado. O fluxo de projeto VLSI para design de um circuito integrado de aplicação específica, inclui a descrição do circuito em VHDL, síntese lógica utilizando as células padrão da tecnologia, PAR (*place*

*and route*) e simulações comportamentais e temporais. O diagrama do fluxo VLSI pode ser ilustrado pela figura 7.

Fonte: (9)

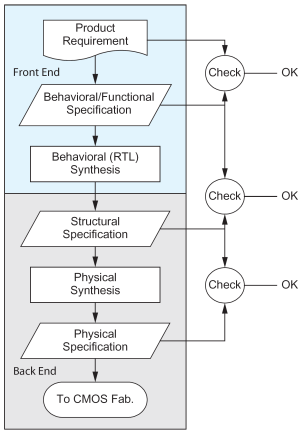


Figura 7 – Fluxo de projeto VLSI.

No processo de desenvolvimento do circuito várias etapas são executadas. Primeiro, há a simulação comportamental para verificar se o circuito descrito em VHDL atende às expectativas, utilizando um *testbench* em VHDL e a ferramenta Cadence NCLaunch. Em seguida, ocorre a síntese lógica, onde a partir do modelo comportamental, utiliza-se a ferramenta Genus para criar um modelo RTL com células padrão de tecnologia específica, considerando restrições de área, frequência e consumo de energia. A síntese gera dois arquivos: um com componentes e conexões, em Verilog, e outro com informações de atraso no formato SDF. A simulação pós-síntese é realizada para validar o netlist gerado usando o mesmo *testbench* da simulação comportamental. Em seguida, na etapa de PAR, o layout é criado posicionando as células e realizando as conexões entre essas células, utilizando a ferramenta Innovus. Por fim, na simulação pós-PAR, o circuito é simulado considerando as resistências e capacitâncias parasitas. Cada etapa é fundamental para garantir o correto funcionamento do circuito.

O cronograma para o desenvolvimento dessas atividades está disponível na tabela 1.

| | mar/24 | abr/24 | mai/24 | jun/24 | jul/24 | ago/24 | set/24 | out/24 | nov/24 |
|---|---|---|---|---|---|---|---|---|---|
| Etapa 1 | | | | | | | | | |
| Etapa 2 | | | | | | | | | |
| Etapa 3 | | | | | | | | | |
| Etapa 4 | | | | | | | | | |

Tabela 1 – Cronograma de Etapas

# 4 Resultados e Discussão

Conforme dito no capítulo 3, o trabalho foi dividido em 4 etapas, nas quais a primeira etapa consistiu no estudo dos DPDs e nos métodos de modelagem deles, a segunda etapa consistiu na implementação desta modelagem em software, a qual utilizou-se o python, a etapa 3 que consiste na implementação do modelo de DPD escolhido em hardware utilizando a linguagem VHDL e finalmente na quarta etapa é feita o design do circuito integrado. Neste capítulo sãi exibidos os resultados da etapa 2 já que a etapa 1 consiste na pesquisa bibliográfica e as etapas 3 e 4 serão desenvolvidas na segunda etapa do projeto.

## 4.1 Modelagem do PA

Para fazer a modelagem em software foi utilizada a linguagem de programação Python. Para isso, separou-se os dados citados na seção 3.2 do capítulo 3, em dados de extração e dados de validação, os quais são utilizados para extração dos coeficientes do modelo do MP e validação do modelo encontrado, respectivamente. Para fazer a validação do modelo utilizou-se a métrica do NMSE, que consiste em calcular o erro médio quadrado do valor medido pelo VSA para o valor calculado pelo modelo. Portanto, quanto menor o NMSE mais fiel é o modelo do PA. Nesta etapa obteve-se um NMSE de -23.57 dB, para cálculos em virgula flutuante, onde o resultado está presente no gráfico da figura 8.

## 4.2 levantamento do número de bits

Após concluída a modelagem matemática, foi feita a modelagem do DPD para então ser feito o levantamento da quantidade de bits necessários para a implementação do DPD em hardware minimizando os erros de quantização. Para isso foi necessário refazer a extração dos coeficientes, mas desta vez com os dados normalizados para valores de 0 a $2^{bits}$. O resultado desse levantamento está presente no gráfico na figura 9.

Neste gráfico observa-se duas curvas, a curva em azul apresenta a quantidade total de bits total contando com os bits de overflow necessárias para as operações de multiplicação, enquanto a curva em vermelho representa a quantidade de bits de resolução do sinal. Analisando este gráfico observou-se que não existem ganhos significativos no erro a partir de 7 bits, portanto foi feita a modelagem do PA utilizando uma resolução de 8 bits o resultado alcançado está ilustrado pela figura 10.

Fonte: Autor

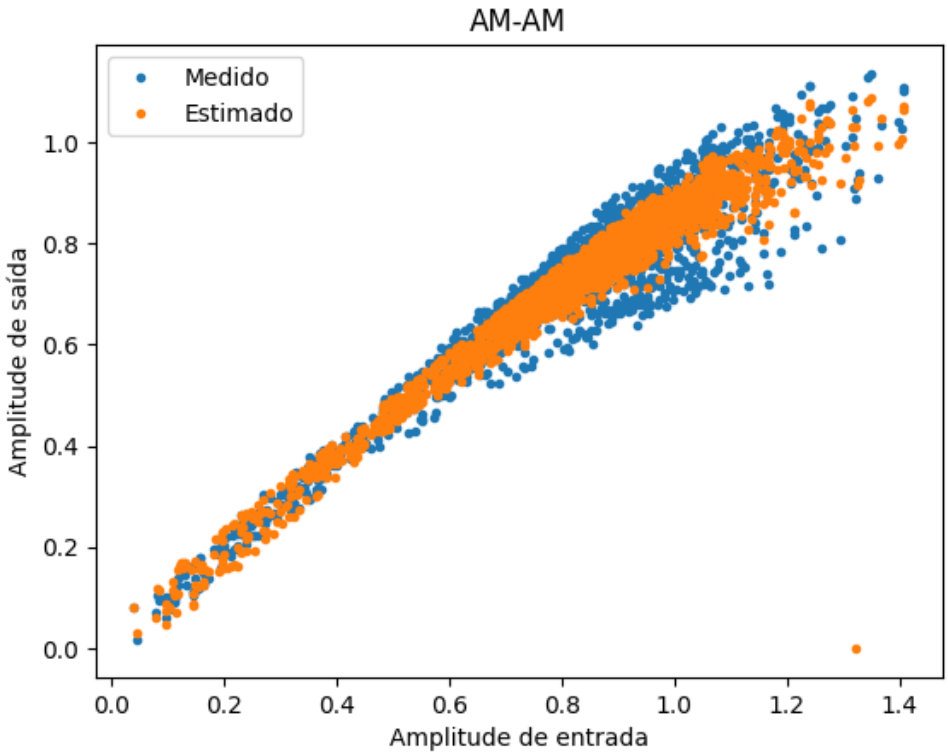


Figura 8 – Modelo do PA em vírgula flutuante

Fonte: Autor

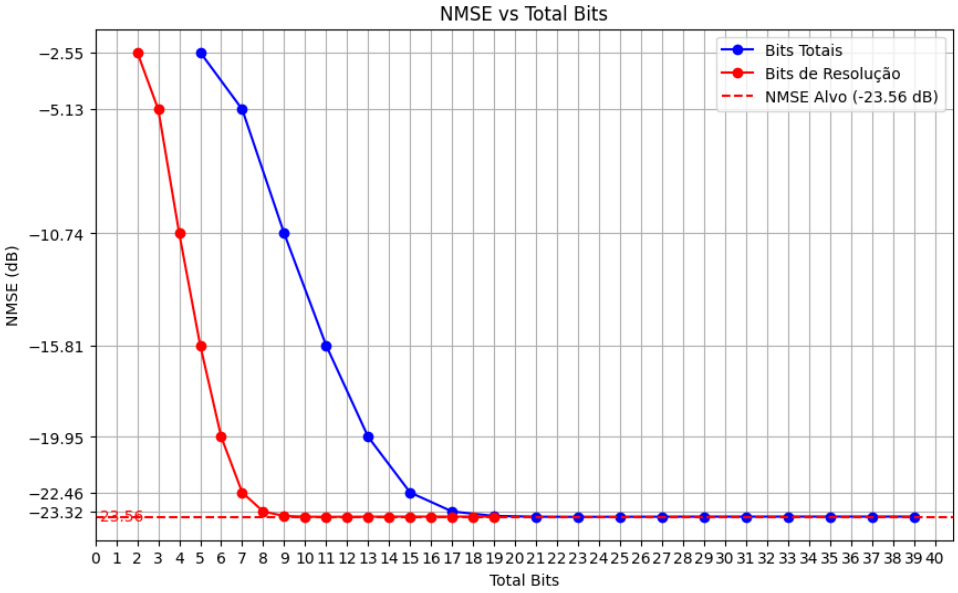


Figura 9 – Gráfico Número de bits x NMSE

## 4.3 Modelagem do DPD

A partir dos resultados obtidos foi possível fazer a modelagem do DPD, para isso foi feito o mesmo processo de modelagem do PA, porém para alcançar a característica de

Fonte: Autor

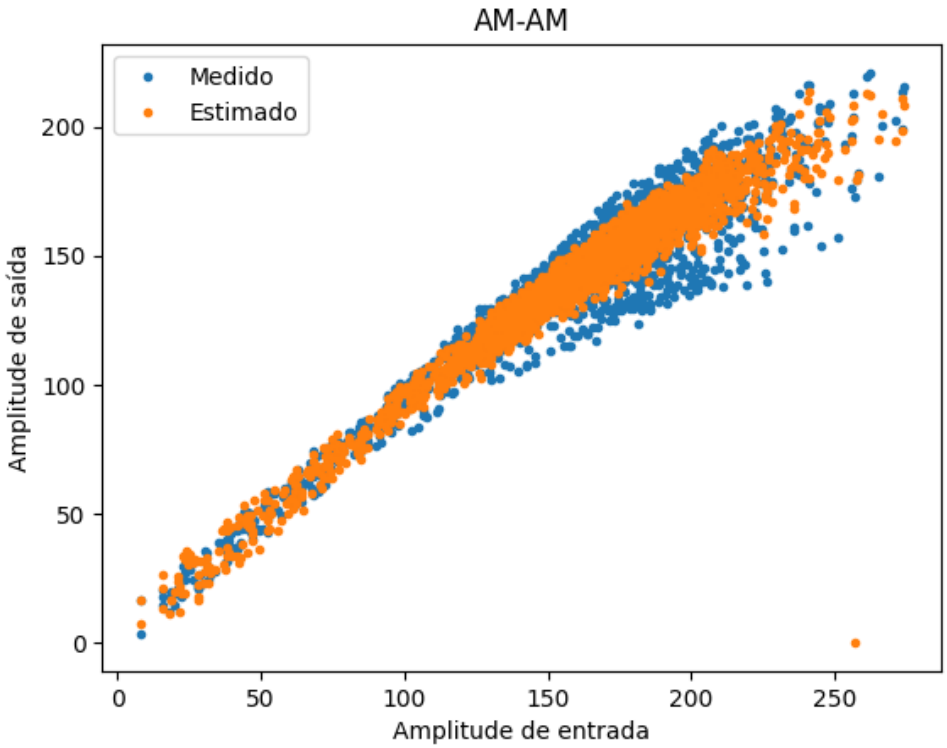


Figura 10 – Modelo do PA em vírgula fixa

transferência inversa PA foi invertido a ordem dos dados de entrada e saída para extração dos coeficientes do DPD. O resultado desta modelagem está ilustrado pela figura 11 a seguir.

Fonte: Autor

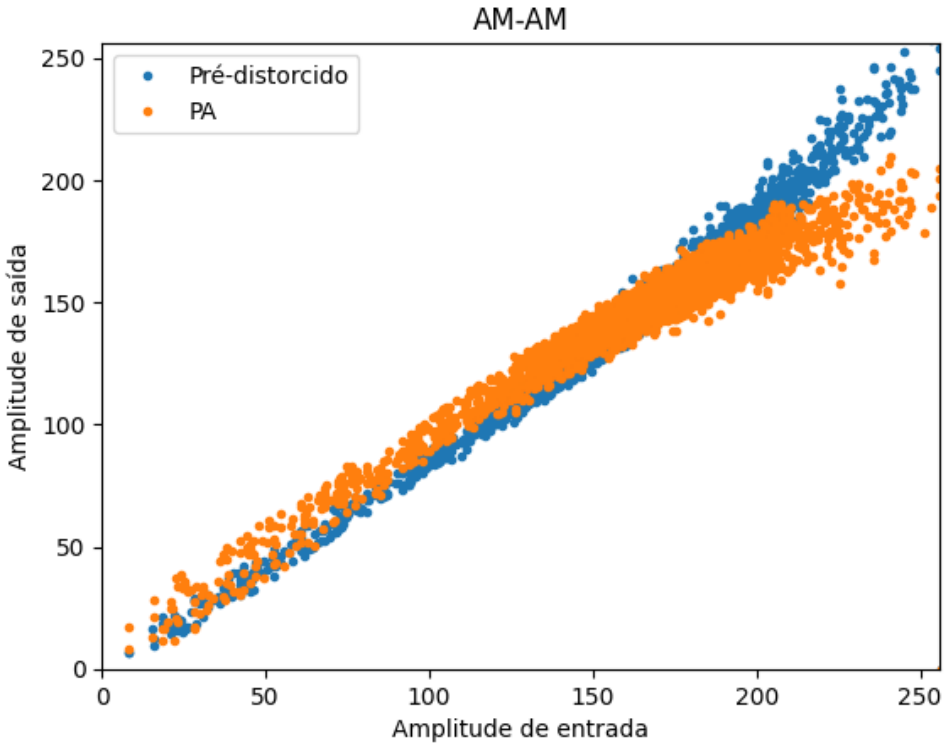


Figura 11 – Modelo do DPD em vírgula fixa

# 5 Conclusão

A evolução dos sistemas de comunicação sem fio tem promovido a implementação de diversos serviços móveis, tornando essencial que esses sistemas operem com máxima eficiência. Nesse cenário, a implementação de um DPD em cascata com o PA surge como uma alternativa de baixo custo e interessante para melhorar o desempenho desses sistemas. O objetivo deste trabalho de conclusão de curso é implementar em hardware um DPD baseado no modelo de Polinômio de Memória. Para isso, o projeto foi dividido em quatro etapas: estudo do DPD e da modelagem matemática, modelagem do DPD em software, implementação do DPD em hardware e, finalmente, design do circuito integrado. Sendo assim a primeira etapa de desenvolvimento do projeto foi a modelagem do PA em virgula flutuante, utilizando o método do MP, para fazer essa modelagem utilizou-se um polinômio de 2° grau com uma amostra de memória, para fazer a validação dessa modelagem uitlizou-se a métrica do NMSE. Nesta etapa obteve-se um NMSE de -23,57 dB, a próxima etapa consiste em otimizar a quantidade de células lógicas utilizadas no processo limitando o número de bits utilizados. Nesta etapa observou-se que a partir de 8 bits, não havia melhora expressiva no NMSE, assim, essa foi a resolução em bits utilizadas para a amostragem de sinais. E por fim foi feito a modelagem do DPD em software o qual apresentou um comportamento inverso em relação o do PA, assim satisfazendo as necessidades. Atualmente, o projeto está na etapa 3, que corresponde à implementação em hardware. Conclui-se, portanto, que o projeto alcançou os resultados esperados, com uma modelagem eficaz do PA e do DPD, satisfazendo os critérios de desempenho estabelecidos nas etapas iniciais.

# Referências

1 JOHN, E. *Modelagem comportamental de amplificadores de potência de radiofrequência usando termos unidimensionais e bidimensionais de séries de Volterra*. 2016.

2 KENINGTON, P. *High Linearity RF Amplifier Design*. [S.l.: s.n.], 2000.

3 CRIPPS, S. *RF Power Amplifiers for Wireless Communications*. [S.l.: s.n.], 2006.

4 CHAVEZ, J. H. *Estudo comparativo entre as arquiteturas de identificação de pré-distorcedores digitais através das aprendizagens direta e indireta*. 2018.

5 PEDRONI, V. *Eletrônica Digital e VHDL*. [S.l.: s.n.], 2010.

6 SCHUARTZ, L.; LIMA, E. *Polinômios com Memória de Complexidade Reduzida e sua Aplicação na Pré-distorção Digital de Amplificadores de Potência*. 2017.

7 LIMA, E. G. de; GHIONE, G. *Behavioral modeling and digital base-band predistortion of RF power amplifiers*. 2009.

8 BONFIM, E. J.; LIMA, E. G. D. *A Modified Two Dimensional Volterra-Based Series for the Low-Pass Equivalent Behavioral Modeling of RF Power Amplifiers*. 2016. 27-35 p.

9 H.E.WESTE, N.; HARRIS, D. M. *CMOS VLSI Design: A Circuits and Systems Perspective (4th Edition)*. [S.l.: s.n.], 2010.

# Anexos

ANEXO A - função que calcula matriz de confusão em virgula flutuante

```
def mp(P, M, xn):
    L = xn.shape
    XX = np.zeros((L[0] - M, P * (M+1)), dtype=np.complex128)
    for l in range(M+1, L[0]):
        for p in range(1, P+1):
            for m in range(0, M+1):
                XX[l-M-1, ((p-1)*(M+1))+m] = (np.abs(xn[l-m])
                    **(2*p-2)*(xn[l-m]))[0]
    return XX

XX_ext = mp(P, M, in_data_ext)
coefficients, _, _, _ = np.linalg.lstsq(XX_ext, out_data_ext[M:],
    rcond=None)
predicted_val = XX_val @ coefficients
```

# ANEXO B - Função que calcula matriz de confusão em virgula fixa

readeq = lambda val, precision: np.floor(val / (2 ** precision))

```
def mp_int(P, M, xn, bits):
L = xn.shape
XX = np.zeros((L[0] - M, P * (M+1)), dtype=np.complex128)
for l in range(M+1, L[0]):
    for p in range(1, P+1):
        for m in range(0, M+1):
                A = np.real(xn[l-m])[0]
                B = np.imag(xn[l-m])[0]
                modulo_power = 2**bits
                modulo_square = readeq(A ** 2 + B ** 2, bits)
                for _ in range(1, p):
                   modulo_power = readeq(modulo_power *
                      modulo_square, bits)
                real_part = readeq(A * modulo_power,bits)
                imag_part = readeq(B * modulo_power,bits)
                XX[l-M-1, ((p-1)*(M+1))+m] = complex(
                   real_part,imag_part)
return XX
```