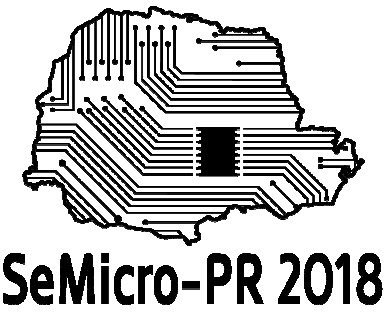

Pré-distorcedores Digitais Baseados em Perceptrons e em Vírgula Fixa

Juliana C. L. Pereira, Felipe A. Schoulten, Caio P. Mizerkowski, Sibilla B. L. França, Eduardo G. Lima
Grupo de Circuitos e Sistemas Integrados (GICS) – Departamento de Engenharia Elétrica
Universidade Federal do Paraná, Curitiba, Brazil

***Resumo*—Este artigo compara dois modelos comportamentais de pré-distorcedores digitais (DPDs) para linearização de amplificadores de potência (PAs). As descrições em aritmética de vírgula fixa para operação em tempo real são detalhadas para ambas topologias que utilizam uma arquitetura de rede neural conhecida como perceptron de três camadas. Simulações nos softwares MATLAB e ISE Design Suite indicam que um modelo otimiza a complexidade computacional, enquanto que o outro minimiza o erro de modelagem.**

Palavras-chave: Amplificador de potência, FPGA, pré-distorcedor digital, vírgula fixa, VHDL.

I. INTRODUÇÃO

Em um sistema de telecomunicação, o amplificador de potência (PA, do inglês *Power Amplifier*) é um dos dispositivos mais importantes [1]. Nessas aplicações, o PA necessita transmitir o máximo possível de potência, o que muitas vezes acaba se tornando um obstáculo. Devido às suas características não lineares, para altas potências, muitas vezes ocorrem distorções indesejadas na saída do amplificador. Uma das formas de contornar esse problema é a utilização de pré-distorcedores digitais (DPDs, do inglês *Digital Baseband Predistorters*) [2], que se tratam de um sistema digital colocado em cascata com o PA e que distorce o sinal a ser amplificado. A distorção causada pelo DPD, quando somada à distorção gerada pelo PA, acaba por linearizar o sinal de saída.

A topologia do DPD necessita de um modelo que represente as características de transferência inversas de um PA. *Perceptrons* de três camadas (TLPs, do inglês *Three-Layer Perceptrons*) são boas alternativas para essa representação [3]. Os resultados dessa operação, porém, são dados em vírgula flutuante, o que não é aplicável para uma implementação em FPGA. Para que isso seja realizado, é necessário transformar esses resultados utilizando aritmética de vírgula fixa.

Para a realização dessa conversão, é necessário levar alguns pontos em consideração. O primeiro ponto é que a realização de operações complexas como raiz quadrada e função recíproca se torna consideravelmente mais complexa. Além disso, ao trabalhar com números binários, quando é realizada a multiplicação entre dois números, o número de *bits* é duplicado no resultado. Para manter a quantidade original de *bits*, é necessário reduzir o número de *bits* no resultado pela metade. Considerando que a aplicação final deve ser capaz de trabalhar em altas frequências, na ordem de algumas dezenas de MHz, é necessário atentar que cada operação realizada não pode exceder a ordem de alguns nanossegundos. Para tornar isso possível, é necessário utilizar as características paralelas inerentes à FPGA, dividindo o código VHDL em vários blocos de operação que podem ser processados simultaneamente. O objetivo deste trabalho é comparar dois modelos baseados em TLPs apropriados para o processamento em tempo real e em aritmética de vírgula fixa.

II. PRÉ-DISTORCEDOR DIGITAL COM ARQUITETURA TLP

Uma alternativa que vem sendo bastante discutida em trabalhos recentes na tentativa de aumentar a eficiência dos PAs é a utilização da técnica de DPD. O seu objetivo é propositalmente distorcer o sinal de entrada do PA, de forma que o sinal em sua saída seja uma cópia linear do sinal de entrada [1].

Neste trabalho, será utilizada uma arquitetura de *perceptrons* com três camadas. Redes neurais artificiais (ANNs, do inglês *Artificial Neural Networks*) são utilizadas em diversos campos da ciência e tecnologia para aproximar funções contínuas em um intervalo arbitrário, como é o caso do *perceptron* de múltiplas camadas (MLP, do inglês *Multi-Layer Network*) [4]. O *perceptron* nada mais é que um modelo matemático baseado no neurônio que recebe uma entrada de *n*-dimensões e oferece como saída em qual lado do hiperplano, que é definido pelos coeficientes do *perceptron*, a saída está. Essa operação é realizada através da multiplicação da entrada pelos coeficientes do *perceptron*, conhecidos como pesos, e da soma desse resultado com outro valor conhecido como *bias*. O resultado dessa operação é transferido para uma função de ativação (normalmente a função de Heaviside para um *perceptron*) e o resultado dessa função indica em que lado do hiperplano a entrada está.

III. ARQUITETURA DO TLP EM VÍRGULA FIXA

Para esse trabalho, foram implementados dois modelos. O primeiro modelo, mais complexo, necessita de cálculos envolvendo as funções raiz quadrada, recíproca e tangente hiperbólica. Já o segundo, com exceção da utilização da função tangente hiperbólica em apenas uma situação, não utiliza as funções raiz quadrada e recíproca. Os diagramas de blocos apresentados nas Figuras 1 e 2 representam as arquiteturas em vírgula fixa dos modelos baseados em TLP, onde os valores de entrada e saída são representados, respectivamente, por x e s. O instante presente é $[n]$ e as amostras passadas são $[n-1]$ e $[n-2]$. Apenas o modelo 1 foi detalhado em [5].

Os diagramas de blocos podem ser divididos em três partes. A primeira calcula os setes sinais que são usados como entrada para ambos os TLPs. A segunda calcula a saída dos TLPs que correspondem às partes real e imaginária de um número complexo $y[n]$, a partir do qual a saída da rede neural é obtida.

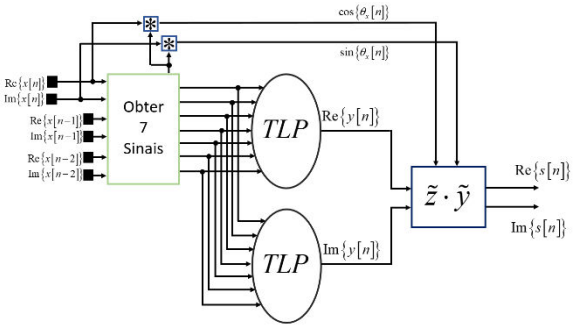

Fig. 1. *Modelo 1 da rede neural*

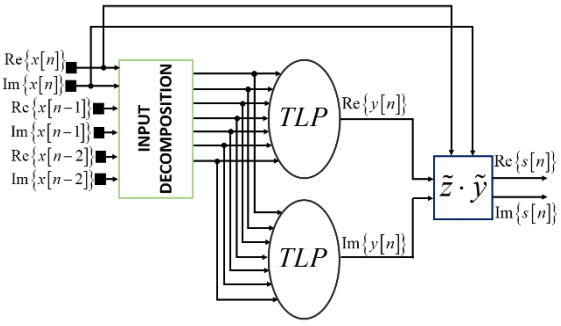

Fig. 2. *Modelo 2 da rede neural*

Para a implementação em uma FPGA, o primeiro passo é a conversão dos sinais de vírgula flutuante para vírgula fixa. Esse processo é realizado da seguinte forma. Inicialmente todos os números complexos, que representam os sinais de entrada e saída de um PA, são multiplicados por um ganho real e constante para garantir que as partes real e imaginária estejam em um intervalo de -1 a 1. Então, é considerado que o número decimal 1 é representado por 2 elevado ao número de *bits* de resolução. Mesmo com a limitação existente para os sinais de entrada e saída, ainda é possível que números com módulo maior que 1 apareçam em algum momento dentro da rede neural. Para representar esses números, *bits* extras são adicionados à esquerda para evitar *overflow* (por exemplo, se 5 é considerado como o maior número, três *bits* extras são necessários, já que $2^3 > 5$ e $2^2 < 5$). Portanto, a representação dos números em vírgula fixa consiste no número de *bits* de resolução mais os *bits* extras utilizados para evitar *overflow* mais um *bit* de sinal.

Todas as operações matemáticas dentro da rede neural são realizadas utilizando multiplicadores, somadores ou tabelas de busca (LUTs, do inglês *Look Up Tables*), sendo que ambos os multiplicadores e somadores possuem duas entradas e uma saída. Existem operações mais complexas dentro da rede neural, mais especificamente, as funções recíproca, raiz quadrada e tangente hiperbólica. Essas operações são calculadas utilizando LUTs e interpolação linear. Como criar uma LUT que realize o endereçamento de todas as possibilidades em uma mensagem com elevado número de *bits* é uma tarefa que exige muito tempo de processamento, o número de *bits* utilizado no endereçamento é reduzido. Para cada posição endereçável da LUT, existem dois coeficientes correspondentes, a (angular) e b (linear), que são utilizados para calcular uma função linear no formato $au + b$, onde u é o valor de entrada. Portanto, o cálculo de cada função é realizado através da leitura de dois valores armazenados em uma LUT, uma multiplicação e uma soma.

Quando uma multiplicação binária é realizada, o número de *bits* é duplicado. Uma vez que cada número é formado por um *bit* de sinal, *bits* de resolução e *bits* extras para evitar *overflow*, para manter todos os números com a mesma quantidade de *bits*, é necessário arredondar os mesmos descartando alguns dos *bits* de resolução menos significativos e dos *bits* de *overflow* mais significativos.

A execução de cada uma dessas operações exige tempo. Não tem muito que pode ser realizado em apenas um ciclo de *clock* de apenas alguns nanossegundos. Apenas uma operação pode ser executada de cada vez, e, portanto, todas as operações devem ser executadas simultaneamente. Isso justifica o uso de um projeto de *hardware* dedicado descrito em VHDL. Isso permite que a lógica seja dividida em diversos blocos, cada um deles responsável por uma operação, o que consequentemente permite que todas sejam processadas ao mesmo tempo. Dado que essa separação é feita, se torna possível a criação de uma aplicação que opere em altas frequências.

Como ilustrado anteriormente, o processo inicia com a decomposição das entradas da rede neural para que elas sejam utilizadas nos *perceptrons*, como ilustrado nas Figuras 1 e 2. Na Figura 1, que corresponde ao modelo 1, a decomposição desses valores é realizada através do cálculo do módulo das entradas $[n]$, $[n-1]$ e $[n-2]$, e do seno e cosseno da diferença entre os instantes $[n]-[n-1]$ e $[n]-[n-2]$, obtidos, respectivamente, por

$$|x[n]| = \sqrt{Re\{x[n]\}^2 + Im\{x[n]\}^2} \quad (1)$$

$$\cos(\theta_n - \theta_{n-1}) = Re\left\{\frac{x(n).x^*(n-1)}{|x(n)|.|x(n-1)|}\right\} \quad (2)$$

$$\text{sen}\,(\theta_n - \theta_{n-1}) = Im\left\{\frac{x(n).x^*(n-1)}{|x(n)|.|x(n-1)|}\right\} \quad (3)$$

A Figura 3 ilustra o diagrama de blocos do equivalente dessas operações no modelo 1. A realização dos cálculos de função recíproca e raiz quadrada é executada através do uso de LUTs com interpolação linear.

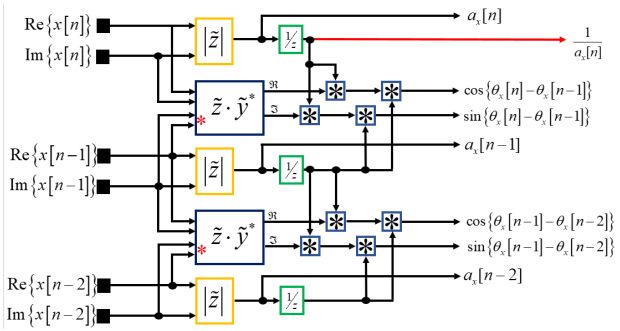

Fig. 3. Decomposição das entradas para o modelo 1

Para o modelo 2, conforme ilustra a Figura 4, o uso dessas funções foi descartado. Por isso, foi necessário também simplificar as operações realizadas. Neste caso, a função raiz quadrada foi substituída pelo módulo ao quadrado das entradas, e os operadores seno e cosseno foram substituídos pela multiplicação complexa dos instantes atuais e passados, conforme

$$|x[n]| \Leftrightarrow Re\{x[n]\}^2 + Im\{x[n]\}^2 \quad (4)$$

$$\cos\,(\theta_n - \theta_{n-1}) \Leftrightarrow Re\{x(n).x^*(n-1)\} \quad (5)$$

$$\text{sen}\,(\theta_n - \theta_{n-1}) \Leftrightarrow Im\{x(n).x^*(n-1)\} \quad (6)$$

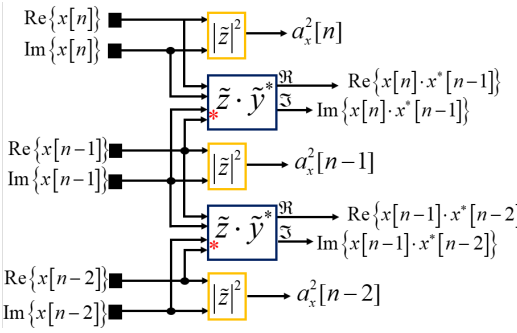

Fig. 4. Decomposição das entradas para o modelo 2

O processo de cálculo de uma função utilizando a LUT com interpolação linear consiste na leitura dos coeficientes a e b, na multiplicação do coeficiente a pela entrada da LUT e pela soma deste resultado com o coeficiente b. Esse processo utiliza três ciclos de *clock* para ser executado.

No modelo 1, para a realização do cálculo dos senos e cossenos, é necessário o valor do módulo e de uma divisão (como visto em (2) e (3)). Ambas as funções são dependentes da utilização de LUTs linearmente interpoladas e, por isso, foram descartadas na implementação do modelo 2. Com a simplificação dessas operações, percebe-se que o tempo de execução da saída é reduzido em 6 ciclos de *clock*.

Com as entradas dos *perceptrons* calculadas, o passo seguinte consiste em realizar os cálculos para a obtenção da saída dos mesmos. As Figuras 5 e 6 mostram a topologia do TLP para os dois modelos implementados.

Como é possível observar, as duas primeiras parcelas de ambos os TLPs consistem na multiplicação de todas as entradas por vetores de peso w_1 e w_2 e das somas desses valores com os *bias* b_1 e b_2. O resultado dessas operações então é aplicado a uma função tangente hiperbólica. Vale ressaltar que, para ambos os modelos, a tangente hiperbólica é a única função calculada através de uma LUT linearmente interpolada.

Após realizado o cálculo da tangente hiperbólica, os resultados originados dessas funções são multiplicados pelos vetores de peso h_1 e h_2, e o resultado dessas multiplicações é somado a um *bias* de saída b_o. Após o resultado dessa soma encontra-se a diferença significativa entre as duas topologias do TLP. No modelo 1, o resultado da soma realizada anteriormente é aplicado mais uma vez a mais uma função tangente hiperbólica. Esse cálculo não é realizado no modelo 2, onde a saída do TLP é a própria soma. Com essa simplificação, o tempo de execução total da rede neural é reduzido em mais três ciclos de *clock*.

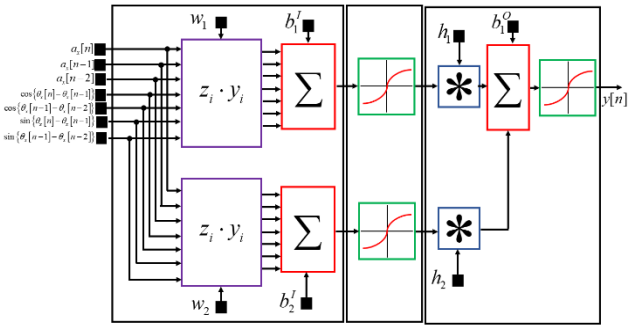

Fig. 5. Arquitetura do TLP para o modelo 1

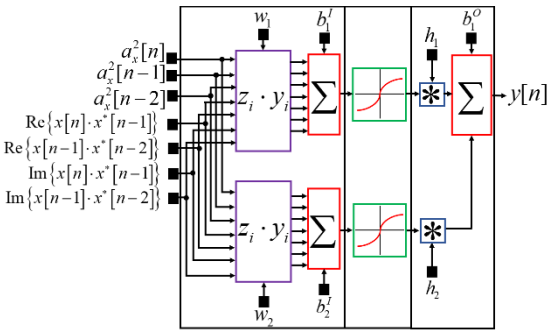

Fig. 6 – Arquitetura do TLP para o modelo 2

Com os valores obtidos nas saídas dos TLP, resta apenas calcular o valor da saída da rede neural. Observando as Figuras 1 e 2, percebe-se que a grande diferença na realização desse cálculo é que, no modelo 1, a saída é realizada através de uma multiplicação complexa entre o seno e cosseno da diferença com as saídas dos TLPs, enquanto no modelo 2 é realizada a multiplicação complexa entre a entrada $x[n]$ e as saídas dos TLPs. Apesar dessa diferença, essa simplificação não apresenta

nenhum impacto significativo na quantidade de ciclos de *clock* necessários para o cálculo da saída. Como o cálculo do seno e cosseno da entrada $x[n]$ no modelo 1 não depende de nenhum resultado calculado no TLP, esse valor é calculado anteriormente e armazenado para ser usado no instante em que for necessário.

Ao final, considerando ambas as topologias, percebe-se uma redução de 9 ciclos de *clock*, de 24 ciclos necessários para o primeiro modelo contra apenas 13 no segundo. A Tabela 1 apresenta alguns valores comparativos em relação às operações realizadas em cada um dos modelos implementados.

TABELA 1. QUANTIDADE DE OPERAÇÕES REALIZADAS NOS MODELOS IMPLEMENTADOS

	MODELO 1	**MODELO 2**
Leituras de LUTs	6	2
Somas	55	46
Multiplicações	72	54
Ciclos de *clock*	24	13

IV. RESULTADOS

Em um estudo comparativo, os modelos 1 e 2 foram treinados em MATLAB tendo por objetivo aproximar características inversas medidas em um PA LDMOS operando em classe AB, alimentado por uma portadora de 2 GHz modulada por um sinal WCDMA de largura de banda igual a 3,84 MHz. Em vírgula flutuante, os modelos foram treinados utilizando mínimos quadrados não lineares baseados no algoritmo Levenberg-Marquardt. A Tabela 2 apresenta o erro quadrático médio normalizado (NMSE, do inglês *Normalized Mean Square Error*), onde o número de neurônios é fixo em dois.

TABELA 2. NMSE PARA OS MODELOS IMPLEMENTADOS

	MODELO 1	**MODELO 2**
NMSE (dB)	-38,1	-27,1

Considerando as mudanças entre os dois modelos, foram necessárias alterações na conversão dos valores de vírgula flutuante para vírgula fixa. A Tabela 3 compara a quantidade de *bits* de *overflow*, endereçamento da LUT tangente hiperbólica e normalização, assim como o NMSE, após a conversão para vírgula fixa. Vale lembrar que o modelo dois não utiliza as funções raiz quadrada e recíproca, portanto estes não foram inclusos na tabela.

A arquitetura em vírgula fixa foi descrita utilizando VHDL. Para validar o circuito, foi realizada uma simulação comportamental no *software* ISE utilizando o simulador ISim para verificar se os valores obtidos através do código VHDL eram equivalentes aos valores obtidos através de simulação do MATLAB. Todos os resultados foram equivalentes, comprovando a viabilidade da implementação dessa aplicação em tempo real.

TABELA 3. VALORES UTILIZADOS EM VÍRGULA FIXA

MODELO	**Número de *bits* de endereçamento da LUT**			**NMSE (dB)**
	Overflow	Endereçamento	Normalização	
1	11	8	14	-37,9
2	21	8	12	-27,0

Foi realizada uma simulação pós roteamento utilizando a FPGA modelo Xilinx Artix7 XC7A200T como placa alvo. Através da simulação foi verificada que a frequência máxima de operação é de 118,033 MHz para o primeiro modelo e 66,820 MHz para o segundo. A Tabela 4 compara as diferenças entre as topologias implementadas em termos de uso de recursos lógicos.

TABELA 4. COMPARAÇÃO DO USO DE RECURSOS LÓGICOS

	MODELO 1	**MODELO 2**
LUTs	3095	1876
Flip-flops	3362	2233

V. CONCLUSÃO

Este trabalho apresentou um estudo comparativo entre 2 arquiteturas para fins de DPD. O modelo 2 permite uma redução muito significativa na complexidade computacional, contudo a sua precisão é muito inferior a do modelo 1. A elevada piora na precisão é devido ao treinamento do modelo 2 ser muito mais susceptível aos chutes iniciais.

AGRADECIMENTOS

Os autores agradecem ao suporte financeiro da Universidade Federal do Paraná, através doPrograma de Iniciação Científica modalidade UFPR/Tesouro Nacional.

REFERÊNCIAS

[1] S. Cripps, *RF Power Amplifiers for Wireless Communications*, 2nd edition. Norwood, MA: Artech House, 2006.

[2] P. B. Kenington, *High Linearity RF Amplifier Design*. Norwood, MA: Artech House, 2000.

[3] J. C. Pedro and S. A. Maas, "A comparative overview of microwave and wireless power-amplifier behavioral modeling approaches," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 4, pp. 1150–1163, Apr. 2005.

[4] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[5] J. C. L. Pereira, F. A. Schoulten, C. P. Mizerkowski, S. B. L. França and E. G. Lima, "Fixed-point Arithmetic Architecture of a Physically-meaningful Perceptron for Digital Pre-distorters", in *18th Microelectronics Students Forum (SFORUM)*, Bento Gonçalves, 2018.