Implementação Eficiente de Filtros Volterra Explorando Abordagens de Posto Reduzido e Características de Hardware

Felipe Dennis de Resende Oliveira, Eduardo Luiz Ortiz Batista e Rui Seara

***Resumo*— Este trabalho apresenta uma nova abordagem para implementação eficiente de filtros Volterra. Tal abordagem baseia-se na exploração dos diferentes graus de significância dos ramos de uma estrutura Volterra de posto reduzido, visando definir tamanhos de palavra individuais a serem utilizados para implementação de cada ramo. Com isso, torna-se possível a obtenção de significativas reduções de custo computacional, especialmente em implementações realizadas em FPGAs (*field-programmable gate arrays*) ou ASICs (*application-specific integrated circuits*). Resultados obtidos em um estudo de caso são apresentados, demonstrando a eficácia da abordagem proposta.**

***Palavras-Chave*— Filtros Volterra, implementações de posto reduzido.**

***Abstract*— This paper presents a new approach for obtaining efficient implementations of Volterra filters. Such an approach is based on exploiting the different significance levels of the branches of reduced-rank Volterra structures, aiming to define individual word sizes for the implementation of each branch. In this way, significant reductions of computational cost can be obtained, especially for implementations carried out using field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). Results obtained from a case study are presented, demonstrating the effectiveness of the proposed approach.**

***Keywords*— Reduced-rank implementation, Volterra filters.**

I. Introdução

Diversos problemas na área de processamento digital de sinais são resolvidos adequadamente utilizando sistemas e filtros lineares. Algumas das características que motivam o uso desse tipo de sistema são a simplicidade de implementação e o custo computacional relativamente baixo em comparação com as soluções não lineares. Contudo, em muitas aplicações práticas, fenômenos não lineares importantes estão envolvidos na geração ou no processamento dos sinais, o que frequentemente torna o uso de sistemas não lineares indispensável para alcançar os níveis de desempenho pretendidos [1]. Alguns exemplos dessas aplicações são o controle ativo de ruído [2], o cancelamento de eco acústico [3], e a compensação das não linearidades existentes em sistemas de controle e comunicação digital [4].

A complexidade matemática associada a sistemas com características não lineares dificulta a elaboração de uma teoria geral que contemple os mais diferentes tipos de sistemas não lineares [5]. Para contornar esse problema, o estudo de sistemas não lineares é realizado separando os sistemas em classes definidas a partir de características específicas de suas não linearidades. Assim, em aplicações práticas, um importante passo é escolher uma classe de sistemas não lineares que represente apropriadamente as características do problema em questão. Nesse contexto, os filtros Volterra têm surgido como uma opção atrativa, uma vez que eles compõem uma classe de sistemas não lineares com capacidade universal de representação para sistemas discretos, causais e de memória finita [5], [6].

Uma característica indesejada dos filtros Volterra, que advém de sua universalidade de representação, é seu elevado custo computacional de implementação. Como consequência, nas últimas décadas, um grande esforço de pesquisa vem sendo direcionado ao desenvolvimento de diferentes abordagens para implementação de filtros Volterra com custo computacional reduzido [1]. Nesse contexto, as abordagens denominadas de posto reduzido vêm suscitando grande interesse [7]–[12]. Essas implementações são em geral baseadas na aplicação de decomposições matriciais ou tensoriais em representações estruturadas (na forma de matrizes ou tensores) dos coeficientes do filtro Volterra. Como resultado, estruturas de implementação compostas por ramos dispostos em paralelo são usualmente obtidas, sendo que cada um dos ramos dessas estruturas tem um grau de significância diferente relacionado a um dos valores singulares da matriz ou tensor de coeficientes. Dessa forma, a partir da remoção dos ramos menos significativos (relacionados aos menores valores singulares), torna-se possível obter implementações eficientes de filtros Volterra que apresentem reduzido custo computacional.

O objetivo principal do presente trabalho de pesquisa é desenvolver implementações eficientes de filtros Volterra. Nesse contexto, uma nova abordagem de implementação baseada nas estratégias de posto reduzido é discutida. A ideia central de tal abordagem é explorar os diferentes graus de significância dos ramos de uma estrutura de posto reduzido, não apenas para remover os ramos menos significativos, mas também para a especificação do *hardware* usado para implementação de cada ramo. Para tal, a estrutura de posto reduzido é modificada de forma a possibilitar o uso de diferentes tamanhos de palavra na implementação dos diferentes ramos em função do grau de significância de cada ramo. Como resultado, implementações eficientes voltadas a plataformas de *hardware* específicas, tais como FPGAs (*field-programmable gate arrays*) e ASICs (*application-specific integrated circuits*), são obtidas. A abordagem proposta é validada a partir de um estudo de caso envolvendo a implementação de filtros Volterra utilizando um FPGA. É importante mencionar ainda que, por simplicidade, o foco deste trabalho está nas implementações de filtros Volterra de segunda ordem. Apesar disso, a abordagem proposta pode ser facilmente estendida para implementações de maior ordem.

O restante desse artigo está organizado conforme descrito a seguir. Na Seção II, são revisadas diferentes estruturas de implementação do filtro Volterra, dentre elas destacam-se a estrutura padrão, a triangular e as de posto reduzido. A abor-

Felipe D. R. Oliveira, Eduardo L. O. Batista e Rui Seara, LINSE - Laboratório de Circuitos e Processamento de Sinais, Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis-SC, Brasil, E-mails: {felipedennis, seara}@linse.ufsc.br, ebatista@ieee.org.

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

dagem de projeto proposta neste artigo é apresentada na Seção III. Na Seção IV, a redução de custo de implementação obtida usando a abordagem proposta é avaliada em comparação com outras implementações usuais da literatura. Finalmente, a Seção V apresenta as conclusões deste trabalho de pesquisa.

II. Filtros Volterra e Implementações com Posto Reduzido

Esta seção apresenta a fundamentação teórica necessária para o desenvolvimento do presente trabalho de pesquisa. Nesse contexto, o filtro Volterra padrão é primeiramente apresentado, seguido de uma discussão acerca da filosofia que serve de base para as implementações de posto reduzido.

A. Filtro Volterra Padrão

A caracterização de um sistema não-linear a partir da expansão em série de Volterra é realizada por meio de um conjunto de *kernels* (núcleos), cada um relacionado a uma certa ordem de não linearidade polinomial [1]. A saída $y(n)$ do filtro Volterra é então obtida somando as saídas dos *kernels* de diferentes ordens, isto é,

$$y(n) = \sum_{p=1}^{P} y_p(n) \tag{1}$$

com P representando a ordem do filtro Volterra e $y_p(n)$, a saída do *kernel* de ordem p. Considerando *kernels* com tamanho de memória N, a saída do *kernel* de ordem p é dada por

$$y_p(n) = \sum_{m_1=0}^{N-1} \cdots \sum_{m_p=0}^{N-1} h_p(m_1, ..., m_p) \prod_{k=1}^{p} x(n - m_k) \tag{2}$$

onde $x(n)$ representa o sinal de entrada e $h_p(m_1, ..., m_p)$, os coeficientes de ordem p correspondentes. A partir de (2), é possível notar que o *kernel* de primeira ordem (isto é, com $p = 1$) é equivalente a um filtro FIR e, portanto, trata-se de um *kernel* linear. Por outro lado, os demais *kernels* (com $p \geq 2$) são os *kernels* não lineares do filtro Volterra.

Um dos principais problemas para o uso de filtros Volterra em aplicações práticas é a alta complexidade computacional requerida para a sua implementação. Essa complexidade é resultante do elevado número de coeficientes presente especialmente nos *kernels* não lineares. Tal característica fica evidente observando, a partir de (2), que o número de coeficientes em um *kernel* de ordem p é dado por

$$N_p = N^p. \tag{3}$$

A relação exponencial, evidenciada em (3), entre o número de coeficientes de um dado *kernel* e o seu tamanho de memória é um obstáculo que frequentemente restringe a implementação prática de filtros Volterra.

É possível observar, a partir de (2), que os coeficientes $h_p(m_1, ..., m_p)$ com índices $m_1, ..., m_p$ permutados são multiplicadores de um mesmo produto cruzado de amostras de $x(n)$ para o cálculo de $y_p(n)$. Por exemplo, os coeficientes $h_2(0, 1)$ e $h_2(1, 0)$ do *kernel* de segunda ordem são ambos multiplicadores de $x(n)x(n-1)$ no cálculo de $y_2(n)$. Assim, a partir da combinação de coeficientes com índices permutados em um único coeficiente denotado por $\underline{h}_p(m_1, m_2, \ldots, m_p)$ [por exemplo, $\underline{h}_2(0, 1) = h_2(0, 1)+h_2(1, 0)$], torna-se possível obter uma implementação equivalente do filtro Volterra com número reduzido de coeficientes. No entanto, tal forma de implementação, conhecida como representação triangular [1], usualmente proporciona uma redução de complexidade insuficiente para viabilização do uso do filtro Volterra em certas situações práticas [13]. Como consequência, a aplicação de filtros Volterra muitas vezes é feita a partir de implementações com complexidade reduzida, como as implementações de posto reduzido consideradas neste trabalho. Além disso, conforme mostrado em [14], a representação triangular não é a mais interessante para obter implementações de *kernels* de segunda ordem com posto reduzido. Assim, como o foco deste trabalho está em implementações de segunda ordem, a representação triangular será considerada apenas na Seção IV para fins de comparação de desempenho.

B. Implementações de Posto Reduzido

As implementações de posto reduzido de filtros Volterra têm sido consideradas com sucesso em muitos trabalhos de pesquisa encontrados na literatura [7]–[12]. Essas implementações são em geral baseadas na representação da relação de entrada e saída de um dado *kernel* em função de matrizes (ou tensores) de coeficientes. Então, decomposições matriciais (ou tensoriais), tal como a decomposição em valores singulares (SVD), são aplicadas em tais matrizes (ou tensores), levando a formas alternativas de representação da relação de entrada e saída do *kernel* considerado. Essas formas alternativas em geral resultam em estruturas de implementação compostas por ramos com diferentes graus de significância dispostos em paralelo. Assim, torna-se possível a obtenção de estruturas de implementação com complexidade computacional reduzida a partir da remoção dos ramos com menor grau de significância [7]–[12].

Conforme mencionado anteriormente, o foco deste trabalho de pesquisa está na implementação de *kernels* de segunda ordem. Dessa forma, a estratégia para implementação de *kernels* de segunda ordem com posto reduzido discutida em [9] é aqui considerada, uma vez que ela leva a implementações bastante eficientes. Tal estratégia baseia-se em rescrever a relação de entrada e saída do *kernel* de segunda ordem, dada por (2) com $p = 2$, como

$$y_2(n) = \mathbf{x}^{\mathbf{T}}(n)\mathbf{H}_2\mathbf{x}(n) \tag{4}$$

com

$$\mathbf{x}(n) = [x(n) \quad x(n-1) \quad ... \quad x(n-N+1)]^{\mathrm{T}} \tag{5}$$

e

$$\mathbf{H}_2 = \begin{bmatrix} h_2(0,0) & h_2(0,1) & \cdots & h_2(0,N-1) \\ h_2(1,0) & h_2(1,1) & \cdots & h_2(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ h_2(N-1,0) & h_2(N-1,1) & \cdots & h_2(N-1,N-1) \end{bmatrix}. \tag{6}$$

Considerando que $\mathbf{H}_2$ pode ser organizada na forma de uma

matriz simétrica[1] e aplicando a SVD a tal matriz, tem-se

$$\mathbf{H}_2 = \sum_{k=0}^{N-1} \lambda_k \mathbf{h}_{2,k} \mathbf{h}_{2,k}^{\mathrm{T}} \tag{7}$$

onde λ_k e $\mathbf{h}_{2,k}$ representam o k-ésimo valor singular e o k-ésimo vetor singular de $\mathbf{H}_2$, respectivamente. Agora, substituindo (7) em (4), obtém-se [5]

$$y_2(n) = \sum_{k=0}^{N-1} \lambda_k \left[\mathbf{x}^{\mathrm{T}}(n)\mathbf{h}_{2,k}\right]^2. \tag{8}$$

Levando em conta que $\mathbf{x}^{\mathrm{T}}(n)\mathbf{h}_{2,k}$ corresponde à relação de entrada e saída de um filtro FIR com vetor de coeficientes $\mathbf{h}_{2,k}$, é possível concluir, a partir de (8), que o *kernel* Volterra de segunda ordem pode ser implementado na forma da estrutura com ramos paralelos ilustrada na Fig. 1. Cada ramo dessa estrutura é composto por um filtro FIR cuja saída é elevada ao quadrado e, então, multiplicada por um dos valores singulares de $\mathbf{H}_2$. Os vetores de coeficientes dos filtros FIR (dados por $\mathbf{h}_{2,k}$ com $k = 0, \ldots, N-1$) são os vetores singulares de $\mathbf{H}_2$ e, portanto, eles possuem norma unitária. Consequentemente, o grau de significância de cada ramo da estrutura da Fig. 1 é determinado pela magnitude do valor singular λ_k correspondente. Então, desconsiderando os ramos relacionados aos valores singulares de menor magnitude, torna-se possívela obtenção de uma implementação eficiente com redução de custo computacional.

As demais abordagens de implementação com posto reduzido de filtros Volterra encontradas na literatura são obtidas associando diferentes formas de representação da relação de entrada e saída de um *kernel* com diferentes tipos de decomposições matriciais ou tensoriais [7]–[12]. De maneira geral, tais abordagens também resultam em estruturas de implementação compostas por ramos paralelos que apresentam diferentes graus de significância.

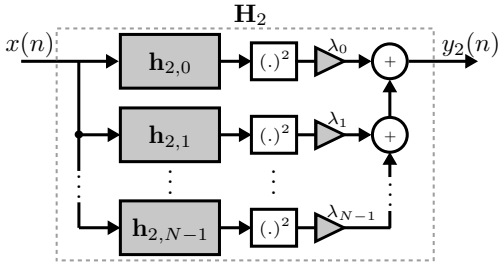

Fig. 1. Diagrama de blocos de uma implementação baseada na SVD de um *kernel* Volterra de segunda ordem.

III. Abordagem Proposta

Nesta seção, uma nova abordagem para a implementação de filtros Volterra com custo computacional reduzido é apresentada. Tal abordagem baseia-se em explorar os diferentes graus de significância dos ramos de uma implementação de posto reduzido tanto para remoção de ramos não significativos quanto para a escolha das características do *hardware* usado para implementar cada ramo. Nesse contexto, algumas hipóteses iniciais são consideradas com relação às características do *hardware* que será utilizado para implementação do filtro Volterra:

- A plataforma digital alvo para implementação do filtro utiliza representação em ponto fixo.
- O tamanho de palavra disponível nessa plataforma é de n_{b} bits, sendo $(n_{\mathrm{b}} - 1)$ bits dedicados à mantissa e 1 bit de sinal.
- A faixa dinâmica de representação de valores numéricos é $[1, -1]$.
- Amostras do sinal fora da faixa dinâmica são truncadas ao valor representável mais próximo (isto é, em caso de *overflow*, ocorre saturação).
- Em função da faixa de representação utilizada, os valores singulares da matriz de coeficientes tem módulo menor ou igual a 1.

A próxima etapa para o desenvolvimento da abordagem proposta consiste em reorganizar a estrutura da Fig. 1 com a finalidade de facilitar a exploração dos diferentes graus de significância dos diferentes ramos. Para tal, (6) é reescrita como

$$y_2(n) = \sum_{k=0}^{N-1} \mathrm{sign}(\lambda_k) \left[\sqrt{|\lambda_k|}\mathbf{x}^{\mathrm{T}}(n)\mathbf{h}_{2,k}\right]^2 \tag{9}$$

onde

$$\mathrm{sign}(\lambda_k) \begin{cases} 1 & \mathrm{se} \lambda_k \geq 0 \\ -1 & \mathrm{se} \lambda_k < 0. \end{cases} \tag{10}$$

A partir de (9), nota-se que a estrutura da Fig. 1 pode ser reorganizada na forma ilustrada na Fig. 2. Observe que, na entrada do ramo de índice k da estrutura da Fig. 2, o sinal de entrada $x(n)$ é multiplicado pela raiz quadrada do valor singular λ_k. O sinal resultante, dado por

$$x_k(n) = \sqrt{|\lambda_k|} \cdot x(n), \tag{11}$$

possui potência tipicamente menor do que a potência de $x(n)$, sendo essa redução de potência mais expressiva nos ramos menos significantes da estrutura (relacionados aos menores valores de λ_k). Como consequência, torna-se possível reduzir a faixa dinâmica utilizada para implementação de certos ramos sem qualquer impacto na capacidade de representação do sistema. No caso de implementações em FPGAs ou ASICs, tal redução de faixa dinâmica possibilita uma diminuição do tamanho de palavra utilizado, reduzindo assim a complexidade do *hardware*. Nesse contexto, a abordagem aqui proposta consiste em reduzir a faixa dinâmica (e consequentemente o tamanho de palavra) de cada um dos ramos individualmente, usando a probabilidade de *overflow* de $x_k(n)$ como critério para tal. Mais especificamente, a faixa dinâmica é reduzida mantendo a probabilidade de *overflow* de $x_k(n)$ menor ou igual à probabilidade de *overflow* de $x(n)$, considerando este último como um sinal de entrada branco gaussiano.

Para uma faixa dinâmica de $[1, -1]$, a probabilidade de *overflow* de um sinal $x(n)$ branco gaussiano com variância

[1]Note que a suposição de a matriz $\mathbf{H}_2$ ser simétrica pode ser feita sem perda de generalidade, visto que é possível alterar os coeficientes de um *kernel* qualquer visando obter uma representação simétrica sem modificar a relação de entrada e saída do filtro [1].

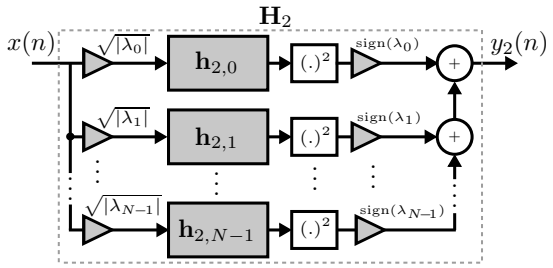

Fig. 2. Diagrama de blocos de uma forma alternativa de implementação da estrutura da Fig. 1.

σ_x^2 é dada por [15]

$$P[|x(n)| > 1] = \int_{-\infty}^{-1} \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{\frac{1}{2\sigma_x^2}x^2(n)} dx + \int_{1}^{\infty} \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{\frac{1}{2\sigma_x^2}x^2(n)} dx. \quad (12)$$

A partir de (12), obtém-se [15]

$$P[|x(n)| > 1] = 2Q\left(\frac{1}{\sigma_x}\right) \quad (13)$$

onde $Q(\cdot)$ representa a bem conhecida Função Q [15].

Deseja-se agora definir uma faixa dinâmica $[-\alpha_k, \alpha_k]$ para o ramo de índice k de tal forma que a probabilidade de *overflow* de $x_k(n)$ seja menor ou igual à dada em (13), ou seja,

$$P[|x_k(n)| > \alpha_k] \leq 2Q\left(\frac{1}{\sigma_x}\right). \quad (14)$$

Fazendo uma análise similar a (12)-(13) para obter $P[|x_k(n)| > \alpha_k]$ e substituindo o valor resultante em (14), obtém-se

$$Q\left(\frac{\alpha_k}{\sqrt{|\lambda_k|}\sigma_x}\right) \leq Q\left(\frac{1}{\sigma_x}\right). \quad (15)$$

Considerando então (15) e ainda que a Função Q é uma função decrescente, é possível concluir que (14) é satisfeita somente se

$$\alpha_k \geq \sqrt{|\lambda_k|}. \quad (16)$$

Em outras palavras, a faixa dinâmica considerada dentro de um ramo precisa ser de pelo menos $[-\sqrt{|\lambda_k|}, \sqrt{|\lambda_k|}]$ para a probabilidade de *overflow* de $x_k(n)$ seja menor ou igual do que aquela para $x(n)$ com uma faixa dinâmica de $[-1, 1]$.

A ideia agora é usar o limite estabelecido por (16) para escolha do tamanho de palavra a ser usado na implementação de cada ramo. Nesse contexto, é bem conhecido que, para cada bit removido do tamanho de palavra original, tem-se uma redução pela metade (divisão por 2) nos limites da faixa dinâmica obtida. Como o desejado é reduzir o limite da faixa dinâmica de 1 para α_k visando atender (16) na igualdade, tem-se uma razão de redução de faixa dinâmica dada por $\alpha_k/1$ e, assim, o número de bits a ser removido pode ser obtido resolvendo a seguinte equação:

$$2^{-r_k} = \alpha_k/1 \quad (17)$$

com r_k representando o número de bits a ser removido do ramo de índice k. Considerando (16) e ainda que r_k só pode assumir valores inteiros, a seguinte expressão é obtida para r_k a partir de (17):

$$r_k = \lfloor \log_2(1/\alpha_k) \rfloor \quad (18)$$

com $\lfloor \cdot \rfloor$ representando a operação de truncamento. Finalmente, no contexto da abordagem proposta, o número de bits a ser utilizado para implementação do ramo de índice k da estrutura da Fig. 2 é dado por

$$n_{\mathrm{b},k} = n_{\mathrm{b}} - r_k. \quad (19)$$

Nota-se, a partir de (18), que r_k é uma função decrescente de α_k. Assim, quanto menor o grau de significância de um dado ramo (isto é, quanto menor o valor de $\sqrt{|\lambda_k|}$ e α_k), maior é o número de bits removido e menor é a quantidade de bits usada na implementação de tal ramo. Além disso, como consequência da operação de truncamento presente em (18), observa-se que diferentes valores de α_k irão resultar em uma mesma redução de número de bits. Mais especificamente, qualquer valor de α_k no intervalo $[2^{-r_k}, 2^{-(r_k-1)})$ irá resultar em um mesmo valor r_k. Por exemplo, para qualquer α_k no intervalo $[0,5; 0,25)$, tem-se $r_k = 1$. Assim, visando simplificar ainda mais o *hardware* usado para implementação da estrutura da Fig. 2, a proposta neste trabalho é substituir o ganho $\sqrt{|\lambda_k|}$ de entrada do ramo de índice k por 2^{-r_k}, substituindo também o vetor de coeficientes do filtro FIR por

$$\mathbf{h}'_{2,k} = \frac{\sqrt{|\lambda_k|}}{2^{-r_k}} \mathbf{h}_{2,k}. \quad (20)$$

Como resultado, o multiplicador da entrada de tal ramo pode ser substituído por uma operação (mais simples) de deslocamento, reduzindo o custo computacional de implementação sem alterar as caraterísticas de transferência do ramo. É importante ressaltar que essa alteração resulta em uma probabilidade de *overflow* de $x_k(n)$ igual a $Q(1/\sigma_x)$, ou seja, igual à probabilidade de *overflow* de $x(n)$. O diagrama de blocos da forma de implementação proposta é ilustrado na Fig. 3, com $>> r_k$ representando a operação de deslocamento de r_k bits para a direita. O número de bits utilizado para implementação de cada parte da estrutura também é indicado na Fig. 3. Note que, em função da operação de deslocamento à direita, os r_k bits menos significativos do sinal de entrada são desprezados, podendo resultar em alguma perda de precisão por arredondamento. A análise do impacto de tal perda está fora do escopo deste trabalho.

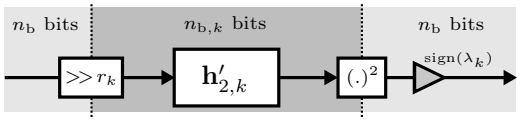

Fig. 3. Diagrama de blocos de um dos ramos da estrutura proposta.

É importante destacar ainda que, apesar de a abordagem proposta ser desenvolvida a partir de uma estrutura de posto reduzido de segunda ordem (ilustrada Fig. 1), ela pode ser facilmente aplicada a *kernels* com ordens superiores. Para tal, a decomposição em *kernels* de segunda ordem apresentada em [14] é utilizada e a abordagem aqui proposta é aplicada a cada um dos *kernels* de segunda ordem resultantes.

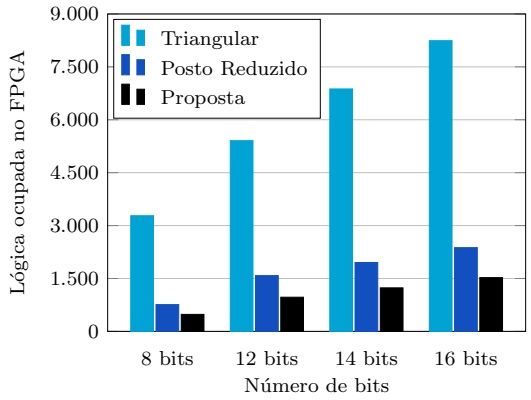

Fig. 4. Custo de implementação de um filtro Volterra de segunda ordem considerando diferentes estruturas de implementação.

TABELA I
CUSTO DE IMPLEMENTAÇÃO EM FPGA DE UM FILTRO VOLTERRA DE SEGUNDA ORDEM PARA DIFERENTES ESTRUTURAS DE IMPLEMENTAÇÃO

Estrutura	Tam. de Palavra	Ocupação Lógica	Ocupação Relativa	
Triangular	8	3279	100%	
	12	5407	100%	
	14	6874	100%	
	16	8242	100%	
Posto Reduzido	8	759	23%	100%
	12	1581	29%	100%
	14	1953	28%	100%
	16	2374	28%	100%
Proposto	8	482	15%	64%
	12	969	18%	61%
	14	1234	18%	63%
	16	1524	18%	63%

IV. RESULTADOS

Nesta seção, um estudo de caso envolvendo a implementação em FPGA de um *kernel* Volterra de segunda ordem é apresentado. Tal estudo de caso tem como objetivo avaliar o custo de implementação obtido com a abordagem proposta em comparação com os custos obtidos usando a representação triangular e a implementação de posto reduzido de [9] (veja Fig. 1). Como métrica de comparação de complexidade, considera-se a área ocupada no FPGA em termos de número de blocos lógicos básicos utilizados. O FPGA escolhido foi o 5CGXFC7C7F23C8 da família Cyclone V da Altera Corporation e a plataforma de projeto utilizada foi o Software Quartus II. De maneira geral, as estruturas foram implementadas buscando a menor ocupação de área possível nos FPGAs. Como consequência, as operações de multiplicação internas aos filtros FIR com coeficientes fixos (filtros esses presentes em todas as implementações consideradas) são realizadas via operações de soma e deslocamento. As demais multiplicações são realizadas usando blocos multiplicadores dedicados presentes no FPGA. O *kernel* de segunda ordem a ser implementado é o mesmo considerado em [7]. Os coeficientes de tal *kernel* são dados por

$$h_2(i,j) = \frac{1,5}{2\pi(1,5^2 + (i-10)^2 + j - 10^2)^2} \qquad (21)$$

com $i, j = 0, 1, ..., 20$. Conforme descrito em [7], implementações eficientes de (21) são obtidas considerando apenas os cinco ramos de maior significância da estrutura de posto reduzido. Para esses cinco ramos, a abordagem proposta leva a uma redução de 0, 1, 2, 3 e 4 bits, do ramo de maior para o de menor significância. Os resultados obtidos para implementações com 8, 12, 14 e 16 bits de tamanho original de palavra estão ilustrados na Fig. 4 e detalhados na Tabela I. A partir de tais resultados, observa-se que a abordagem proposta é capaz de proporcionar reduções bastante superiores de complexidade em relação tanto à implementação tringular, quanto à implementação de posto reduzido de [9].

V. CONCLUSÕES

Neste trabalho, uma nova abordagem para implementação eficiente de filtros Volterra foi apresentada. Tal abordagem explora os diferentes graus de significância dos ramos de uma estrutura de implementação de posto reduzido para definir o tamanho da palavra utilizado na implementação de cada ramo. Essa estratégia permite a obtenção de reduções consideráveis de custo computacional, especialmente em implementações realizadas em FPGAs ou ASICs. Resultados obtidos em um estudo de caso foram apresentados, demonstrando a eficácia da abordagem proposta.

REFERÊNCIAS

[1] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: John Wiley & Sons, Inc., 2000.

[2] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Trans. Signal Process.*, vol. 49, no. 8, pp. 1667–1676, Aug. 2001.

[3] A. Stenger, L. Trautmann, and R. Rabenstein, "Nonlinear acoustic echo cancellation with second-order adaptive Volterra filters," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Phoenix, USA, Mar. 1999, pp. 877–880.

[4] D. Zhou and V. De Brunner, "Novel adaptive nonlinear predistorters based on the direct learning algorithm," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 120–133, Jan. 2007.

[5] E. L. O. Batista and R. Seara, "A novel reduced-rank approach for implementing Volterra filters," in *Proc. Eur. Signal Process. Conf.*, Budapest, Hungary, Aug. 2016, pp. 1778–1782.

[6] A. Carini and G. L. Sicuranza, "Fourier nonlinear filters," *Signal Processing*, vol. 94, pp. 183–194, Jan. 2014.

[7] H.-H. Chiang, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient implementations of quadratic digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 6, pp. 1511–1528, Dec. 1986.

[8] Y. Lou, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient VLSI array processing structures for adaptive quadratic digital filters," *Circuits, Syst. Signal Process.*, vol. 7, no. 2, Jun. 1988.

[9] S. Marsi and G. L. Sicuranza, "On reduced-complexity approximations of quadratic filters," in *Conf. Rec. 27th Asilomar Conf. Signals, Syst. Comput.*, vol. 2, Pacific Grove, CA, Nov. 1993, pp. 1026–1030.

[10] R. D. Nowak and B. D. Van Veen, "Tensor product basis approximations for Volterra filters," *IEEE Trans. Signal Process.*, vol. 44, no. 1, pp. 36–50, Jan. 1996.

[11] T. M. Panicker, V. J. Mathews, and G. L. Sicuranza, "Adaptive parallel-cascade truncated Volterra filters," *IEEE Trans. Signal Process.*, vol. 46, no. 10, pp. 2664–2673, 1998.

[12] G. Favier, A. Y. Kibangou, and T. Bouilloc, "Nonlinear system modeling and identification using Volterra-PARAFAC models," *Int. J. Adapt. Control Signal Process.*, vol. 26, no. 1, pp. 30–53, Jan. 2012.

[13] E. L. O. Batista, O. J. Tobias, and R. Seara, "A sparse-interpolated scheme for implementing adaptive Volterra filters," *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2022–2035, Apr. 2010.

[14] E. L. O. Batista and R. Seara, "A reduced-rank approach for implementing higher-order Volterra filters," *EURASIP Journal on Advances in Signal Process.*, vol. 118, no. 1, pp. 1–8, Nov. 2016.

[15] S. Kay, *Intuitive Probability and Random Processes Using MATLAB*. New York: John Wiley & Sons, Inc., 2006.