Contribuições ao Algoritmo *A-SPADE* Visando a Implementação de um *Declipper* em FPGA

Daniel Gomes de Pinho Zanco, Natan Votre e Walter Antônio Gontijo

***Resumo*— Este artigo apresenta contribuições ao algoritmo *A-SPADE*, visando a implementação de um *Declipper* de áudio em FPGA. É proposta uma abordagem que acelera o *A-SPADE*, originando o algoritmo *ATA-SPADE*. Tal abordagem mostrou que o *ATA-SPADE* é significativamente mais rápido que o *A-SPADE*, apresentando um desempenho equivalente. Também é proposto um algoritmo para estimar o nível de *clipping*. Os algoritmos apresentados constituem os principais componentes de um *Declipper*. O *Declipper* proposto é implementado em FPGA e tem seu desempenho comprovado em um *kit* de desenvolvimento.**

***Palavras-Chave*— A-SPADE, ATA-SPADE, Declipper, FPGA.**

***Abstract*— This paper presents contributions to the A-SPADE algorithm, for the purpose of implementing a audio Declipper in FPGA. An approach that accelerates A-SPADE is proposed, originating the ATA-SPADE algorithm. Such an approach showed that ATA-SPADE is much faster than A-SPADE, showing equivalent performance. A clipping level estimation algorithm is also proposed. The presented algorithms constitute the main components of a Declipper. The proposed Declipper is implemented in FPGA and has its performance proven in a development kit.**

***Keywords*— A-SPADE, ATA-SPADE, Declipper, FPGA.**

I. Introdução

Clipping é a saturação que ocorre quando um sinal ultrapassa os limites de amplitude de um sistema, levando à perda de informações. No caso de sinais de áudio, o seu efeito na percepção auditiva é desagradável, dependendo do conteúdo do áudio e da severidade do *clipping*. Esse problema ocorre principalmente em conversões A/D, reproduções e, também, em gravações profissionais de áudio [1].

Declipping é o processo que restaura sinais contaminados pela saturação. Diferentes algoritmos vêm sendo propostos [2], [3], [4], [5], [6] para realizar essa restauração. Um dos algoritmos que apresenta melhores resultados é o *Analysis SParse Audio DEclipper* (*A-SPADE*), proposto por [6]. Entretanto, tal algoritmo possui alta complexidade computacional, impedindo sua execução em um sistema embarcado, em tempo real.

Neste artigo, são apresentadas contribuições ao *A-SPADE* [6], originando o algoritmo *Accelerated Thresholding A-SPADE* (*ATA-SPADE*) de forma a permitir sua execução em tempo real. Também é proposto um algoritmo para estimar o nível de *clipping*. Os algoritmos propostos constituem as principais funções de um *Declipper*. O *Declipper* proposto é implementado em FPGA para validar seu desempenho.

Daniel Gomes de Pinho Zanco, Natan Votre e Walter Antônio Gontijo, LINSE - Laboratório de Circuitos e Processamento de Sinais, Departamento de Engenharia Elétrica e Eletrônica, Universidade Federal de Santa Catarina (UFSC), Florianópolis, SC, Brasil, e-mails: zanco@linse.ufsc.br, natan@linse.ufsc.br e walter@linse.ufsc.br.

A. Clipping

Seja um vetor $\boldsymbol{x} \in \mathbb{R}^n$ que, ao ser saturado (*hard-clipping*), resulta em $\boldsymbol{y'} \in \mathbb{R}^n$ de intervalo dinâmico dado por $[-\tau, \tau]$, em que τ é o nível de *clipping*. Pode-se definir o *hard-clipping* como

$$y'_i = \begin{cases} x_i & \text{se } |x_i| \leq \tau \\ \text{sign}(x_i)\tau & \text{se } |x_i| > \tau. \end{cases} \quad (1)$$

Busca-se obter uma estimativa $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ que melhor aproxime o vetor $\boldsymbol{x}$, dado que apenas $\boldsymbol{y'}$ é conhecido.

O diagrama de blocos de *Clipping* e o posterior *Declipper* é mostrado na Fig. 1. O vetor $\boldsymbol{x}$ é a entrada do bloco *Clipping*. Considerando que o processamento é em tempo real, o sinal saturado é segmentado em *frames*. O valor de τ é estimado para cada novo vetor $\boldsymbol{y'}$ através de um estimador de nível de *clipping* (*Clipping Level Estimator*). Em seguida, é utilizado o método de *Overlap Add* para que cada *frame* seja processado pelo *Declipping* e posteriormente sobreposto.

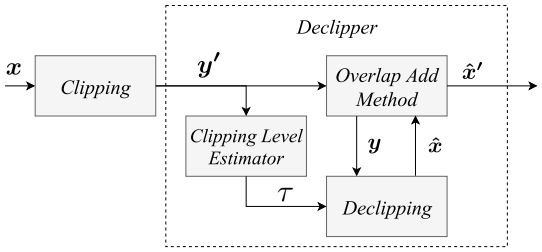

Fig. 1. Diagrama de blocos do *Clipping* e *Declipper*.

A modelagem para o *clipping* dada em (1) permite que, dado um τ conhecido, as amostras não confiáveis de $\boldsymbol{y'}$ sejam identificadas. A partir da identificação, pode-se adicionar restrições nas amostras de uma estimativa $\hat{\boldsymbol{x}}'$ para que esta seja consistente com a informação em $\boldsymbol{y'}$.

Respeitando tal consistência, são obtidas as matrizes de restrição $\boldsymbol{M}_c$, $\boldsymbol{M}_s^+$ e $\boldsymbol{M}_s^-$, que selecionam de $\boldsymbol{y'}$ suas amostras confiáveis, saturadas positivamente ($+\tau$) e negativamente ($-\tau$), respectivamente. Portanto, as seguintes restrições devem ser atendidas para a estimação $\hat{\boldsymbol{x}}'$

$$\boldsymbol{M}_c\hat{\boldsymbol{x}}' = \boldsymbol{M}_c\boldsymbol{y'},\ \boldsymbol{M}_s^+\hat{\boldsymbol{x}}' \geq \boldsymbol{M}_s^+\boldsymbol{y'},\ \boldsymbol{M}_s^-\hat{\boldsymbol{x}}' \leq \boldsymbol{M}_s^-\boldsymbol{y'}. \quad (2)$$

Entretanto, somente respeitar as restrições não garante uma solução única, pois a função definida em (1) admite infinitas inversas. Logo, é preciso regularizar ainda mais o problema inverso. Neste trabalho optou-se pela regularização através de um modelo baseado em representação esparsa.

B. Modelo Baseado em Representação Esparsa

Sinais de áudio usualmente concentram sua energia em um baixo número de frequências ou um pequeno período de tempo [7], ou seja, há esparsidade no domínio tempo-frequência. Também é observado que, devido a sua não-linearidade, a ocorrência de *clipping* causa um aumento na banda do sinal [8], reduzindo a esparsidade. Esse fato pode ser explorado como um critério na recuperação de $\boldsymbol{x}$, isto é, após o processo de *declipping*, $\hat{\boldsymbol{x}}$ deve ter um espectro mais esparso que $\boldsymbol{y}$. Essa característica dos sinais de áudio pode ser formulada sob a perspectiva de síntese esparsa [9]. Outra interpretação é a de análise coesparsa [10], que afirma que $\boldsymbol{Ax}$ é esparso e $\boldsymbol{A}$ é um operador de análise. Neste trabalho é adotada a perspectiva de análise, e o operador $\boldsymbol{A}$ realiza a Transformada Rápida de Fourier (FFT).

Este artigo é organizado como segue: Na Seção II, é apresentado o Algoritmo de referência *A-SPADE*. Na Seção III, são apresentadas as contribuições realizadas. Na Seção IV, é apresentada a Implementação em FPGA. Na Seção V, são apresentados os resultados obtidos. Finalmente, na Seção VI, são apresentadas as conclusões do artigo.

II. Algoritmo A-SPADE

Considerando as restrições definidas em (2) e a esparsidade em frequência existente em sinais de áudio, a seguinte otimização pode ser definida

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimizar}} \quad & ||\boldsymbol{z}||_0 + ||\boldsymbol{Ax} - \boldsymbol{z}||_2^2 \\ \text{sujeito a} \quad & ||\boldsymbol{z}||_0 \leq k \\ & \boldsymbol{M}_c\boldsymbol{x} = \boldsymbol{M}_c\boldsymbol{y} \\ & \boldsymbol{M}_s^+\boldsymbol{x} \geq \boldsymbol{M}_s^+\boldsymbol{y} \\ & \boldsymbol{M}_s^-\boldsymbol{x} \leq \boldsymbol{M}_s^-\boldsymbol{y} \end{aligned} \tag{3}$$

onde $\boldsymbol{z} \in \mathbb{C}^d$ é a estimativa do espectro de $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{C}^{d\times n}$ é o operador de análise, $d \geq n$ e $k \in \mathbb{N}^*$ é o nível de esparsidade requerido para $\boldsymbol{z}$. A pseudo-norma ℓ_0 é calculada pelo termo $||\cdot||_0$ e é definida como $||\boldsymbol{a}||_0 = \#(i|a_i \neq 0)$, que é o número de elementos não-nulos de um vetor $\boldsymbol{a}$.

Além disso, a razão $\left(\frac{d}{n}\right)$ pode ser vista como um fator de redundância. Como o operador de análise é a FFT, esse fator corresponde a realizar um *zero-padding*, ou seja, adicionar $(d-n)$ zeros ao final do vetor $\boldsymbol{x}$ e, então, aplicar a transformação linear, resultando em $\boldsymbol{z}$.

Na função objetivo, $||\boldsymbol{z}||_0$ força uma solução esparsa em frequência, enquanto o termo $||\boldsymbol{Ax} - \boldsymbol{z}||_2^2$ e o conjunto de restrições (2) fornecem garantias quanto à consistência do sinal recuperado.

No Algoritmo 1 (*A-SPADE*), desenvolvido por [6], os autores utilizam uma heurística de busca do valor de k. Como $\boldsymbol{x}$ é desconhecido, não se sabe o nível de esparsidade em que $\boldsymbol{z}$ deve ser otimizado. Logo, a cada iteração, a esparsidade de $\boldsymbol{z}$ é diminuída até que $\|\boldsymbol{A}\hat{\boldsymbol{x}}^{(i)} - \boldsymbol{z}^{(i)}\|_2^2$ seja menor do que um $\epsilon > 0$, de forma que o número k é acrescido de s a cada r iterações, onde $s, r \in \mathbb{N}^*$.

O operador de *hard-thresholding* $\mathcal{H}_k(\boldsymbol{a})$, mostrado no *A-SPADE*, seleciona os k maiores termos de um vetor $\boldsymbol{a}$, forçando os restantes para zero. A projeção $\mathcal{P}_c$ força que as restrições (2) sejam satisfeitas para cada estimativa $\hat{\boldsymbol{x}}^{(i)}$.

Algoritmo 1 Analysis SParse Audio DEclipper (A-SPADE)

Require:
$\hat{\boldsymbol{x}}^{(0)} = \boldsymbol{y}, \boldsymbol{u}^{(0)} = \boldsymbol{0}, i = 1, k = s$
1: **loop**
2: $\quad \boldsymbol{z}^{(i)} = \mathcal{H}_k(\boldsymbol{A}\hat{\boldsymbol{x}}^{(i-1)} + \boldsymbol{u}^{(i-1)})$
3: $\quad \hat{\boldsymbol{x}}^{(i)} = \mathcal{P}_c(\boldsymbol{A}^H(\boldsymbol{z}^{(i)} - \boldsymbol{u}^{(i-1)}))$
4: $\quad$ **if** $\|\boldsymbol{A}\hat{\boldsymbol{x}}^{(i)} - \boldsymbol{z}^{(i)}\|_2^2 \leq \epsilon$ **then**
5: $\quad\quad$ **return** $\hat{\boldsymbol{x}}^{(i)}$
6: $\quad$ **else**
7: $\quad\quad \boldsymbol{u}^{(i)} = \boldsymbol{u}^{(i-1)} + \boldsymbol{A}\hat{\boldsymbol{x}}^{(i)} - \boldsymbol{z}^{(i)}$
8: $\quad\quad i = i + 1$
9: $\quad\quad$ **if** $i \bmod r = 0$ **then**
10: $\quad\quad\quad k = k + s$

O término de execução do *A-SPADE* é garantido para $i = \frac{dr}{s} + 1$ iterações [6]. Apesar de geralmente terminar com menos iterações, o algoritmo é ainda muito exigente para aplicações em tempo real. Além disso, para uma mesma escolha de parâmetros (r, s, ϵ), o número de iterações não é determinístico e varia com a esparsidade do sinal analisado.

Para um k fixo (linhas 2, 3 e 7 do Algoritmo 1), *A-SPADE* utiliza o *Alternating Direction Method of Multipliers* (ADMM) para resolver a otimização (3), visto a sua similaridade com a seleção de regressor (*regression selector*) [11].

A heurística simples de atualização de k resulta em um grande número de iterações do *A-SPADE*. Além disso, não é trivial definir uma regra de atualização para k que simultaneamente acelere a convergência e mantenha a qualidade da restauração. Por exemplo, aumentando o valor de s, o algoritmo irá convergir mais rápido, entretanto, tende a selecionar componentes de frequências incorretas. Porém, diminuindo s, a convergência se torna lenta.

A. Método de Overlap Add

Ocasionalmente amostras saturadas podem estar posicionadas nas bordas de *frames*, portanto, se o processo for sequencial (sem sobreposição), as amostras do sinal restaurado poderão sofrer descontinuidades. O uso do método de *Overlap Add* reduz a descontinuidade, aumentando o desempenho do *A-SPADE*. Para esse método, deve-se considerar alguns parâmetros na escolha da janela. O percentual de *overlap* não deve ser muito alto, visto que aumenta o custo computacional. Além disso, o maior lóbulo lateral (*sidelobe*) [12] do espectro da janela deve ser baixo, aumentando a esparsidade do sinal. Neste trabalho, foi utilizada a janela de *Hanning* com *overlap* de 50%.

III. Contribuições

Pretende-se obter um *Declipper* capaz de realizar processamento em tempo real e que, ao mesmo tempo, apresente um bom desempenho. Entretanto, o uso do *A-SPADE* em sistemas embarcados não é adequado, devido à sua complexidade elevada.

As modificações realizadas no *A-SPADE* consistem em utilizar um *threshold* $\lambda \in \mathbb{R}$ independente de k no operador $\mathcal{H}_k$ e, também, definir uma regra de atualização de λ que acelere a convergência. Tais modificações dão origem ao *ATA-SPADE* apresentado no Algoritmo 2.

Propõe-se utilizar uma formulação do operador de *hard-thresholding* $\mathcal{H}_k$ independente de k

$$\mathcal{H}(a_i, \lambda) = \begin{cases} a_i & \text{se } |a_i| \geq \lambda \\ 0 & \text{se } |a_i| < \lambda. \end{cases} \tag{4}$$

Assim como k aumenta a cada iteração no *A-SPADE*, λ diminui, visto que a esparsidade de $\boldsymbol{z}$ deve aumentar. Essa formulação permite que o perfil de comportamento de λ possa ser representado por uma função de decaimento $\mathcal{D}(i)$, escolhida experimentalmente

$$\mathcal{D}(i) = \exp\left(\frac{-(i+\mu)^2}{\sigma^2}\right) \tag{5}$$

onde $\mu, \sigma \in \mathbb{R}$ e influenciam a taxa de convergência do algoritmo.

Algoritmo 2 Accelerated Thresholding A-SPADE (ATA-SPADE)

Require:
$\hat{\boldsymbol{x}}^{(0)} = \boldsymbol{y}, \boldsymbol{z}^{(0)} = \mathbf{A}\boldsymbol{y}, \boldsymbol{w}^{(0)} = \boldsymbol{z}^{(0)}$
$\boldsymbol{u}^{(0)} = \mathbf{0}, \lambda^{(0)} = \max(|\boldsymbol{A}\boldsymbol{y}|), i = 1$
loop
  $\boldsymbol{z}^{(i)} = \mathcal{H}(\boldsymbol{w}^{(i-1)}, \lambda^{(i-1)})$
  $\hat{\boldsymbol{x}}^{(i)} = \mathcal{P}_c(\boldsymbol{A}^H(\boldsymbol{z}^{(i)} - \boldsymbol{u}^{(i-1)}))$
  $\boldsymbol{b}^{(i)} = \boldsymbol{A}\hat{\boldsymbol{x}}^{(i)}$
  if $\|\boldsymbol{b}^{(i)} - \boldsymbol{z}^{(i)}\|_2^2 < \epsilon$ **or** $i = i_{max}$ **then**
    return $\hat{\boldsymbol{x}}^{(i)}$
  else
    $\boldsymbol{u}^{(i)} = \boldsymbol{u}^{(i-1)} + \boldsymbol{b}^{(i)} - \boldsymbol{z}^{(i)}$
    $\boldsymbol{w}^{(i)} = \boldsymbol{b}^{(i)} + \boldsymbol{u}^{(i)}$
    $\mathcal{D}(i) = \exp(\frac{-(i+\mu)^2}{\sigma^2})$
    $\lambda^{(i)} = \min(\max(|\boldsymbol{b}|)\mathcal{D}(i), \lambda^{(i-1)})$
    $i = i + 1$

O diagrama em blocos do *ATA-SPADE* é mostrado na Fig. 2. Inicialmente, é calculada a FFT de $\boldsymbol{y}$ e, em seguida, o *hard-thresholding* de $\boldsymbol{A}\boldsymbol{y}$ como uma aproximação inicial para $\boldsymbol{z}$. A transformação inversa de $\boldsymbol{z}$ é então forçada a admitir as condições (2) da projeção $\mathcal{P}_c$, considerando que a variável $\boldsymbol{u}$ é inicializada em zero. Quando $||\boldsymbol{b} - \boldsymbol{z}||_2^2$ for menor do que um ϵ especificado ou o número máximo de iterações i_{max} for alcançado, termina a execução, conforme mostrado no Algoritmo 2. Caso contrário, as variáveis $\boldsymbol{w}$, $\boldsymbol{u}$ e λ são atualizadas. O valor de λ é estritamente decrescente e modificado de acordo com a função $\mathcal{D}(i)$.

A. Estimador de Clipping

Nos algoritmos *A-SPADE* e *ATA-SPADE*, assume-se que o valor de τ é conhecido, o que não ocorre no caso de processamento em tempo real. Logo, para que as matrizes de restrição (2) sejam elaboradas é necessário estimar o nível de

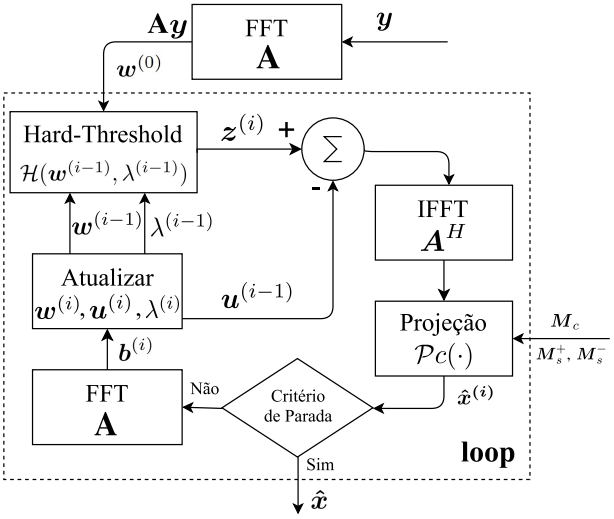

Fig. 2. Diagrama em blocos de execução do *ATA-SPADE*.

clipping. Neste artigo, é proposto o Algoritmo 3, que estima um valor de τ baseado no histograma de $\boldsymbol{y'}$.

Assume-se que a amplitude de um sinal de áudio apresenta distribuição normal, e que a distribuição de um equivalente saturado contém picos em 0 e $\pm\tau$ [13]. Como a distribuição normal é par e decrescente para números positivos, a amplitude mais frequente de $|\boldsymbol{y'}|$ é τ, desconsiderando amostras de baixa amplitude. O operador $|\cdot|$ realiza o módulo de cada elemento de um vetor.

Na presença de ruído branco, os picos em $\pm\tau$ deixam de ser impulsos e as amostras saturadas assumem uma simetria centrada em $\pm\tau$. Para amostras positivas, pode-se mostrar que a menor amplitude submetida a saturação é dada por $2\tau - \max(\boldsymbol{y'})$.

A função Histogram($\boldsymbol{a}$) retorna os vetores $\boldsymbol{c}$ e $\boldsymbol{h}$, em que $\boldsymbol{h}$ corresponde aos intervalos e $\boldsymbol{c}$ é o número de elementos de $\boldsymbol{a}$ por intervalo. O valor h_i é a estimativa de τ sem a presença de ruído. Os valores de $p, q \in [0, 1]$ foram obtidos experimentalmente.

Algoritmo 3 Clipping level estimator

$[\boldsymbol{c}, \boldsymbol{h}] = \text{Histogram}(|\boldsymbol{y'}|)$
Zerar elementos de índice j de $\boldsymbol{c}$ se $h_j < q \cdot \max(|\boldsymbol{y'}|)$
$i = \{i : c_i \geq c_j \forall i \neq j\}$
if $h_i > p \cdot max(|\boldsymbol{y'}|)$ **then**
  $\tau = 2h_i - max(|\boldsymbol{y'}|)$
else
  $\tau = \infty$
return τ

IV. Implementação em FPGA

Como o algoritmo *ATA-SPADE* possui um elevado custo computacional, optou-se por utilizar um FPGA para sua implementação. Tal escolha se justifica pela capacidade de processamento em paralelo e uso de técnicas de *pipeline* disponíveis no FPGA.

A Fig. 3 mostra o diagrama em blocos da implementação do *Declipper* em FPGA. Os blocos *Window in* e *Window*

out/Overlap Add da Fig. 3 correspondem ao *Overlap Add Method*, mostrado na Fig. 1. O *Cliping Level Estimator*, mostrado na Fig. 3, retorna o vetor binário $\boldsymbol{V}_{clip}$, que é equivalente às matrizes $\boldsymbol{M}_c$, $\boldsymbol{M}_s^+$ e $\boldsymbol{M}_s^-$ apresentadas na Seção I, para uma entrada $\boldsymbol{y'}$. Em seguida, o bloco *Window in* realiza o janelamento do vetor $\boldsymbol{y'}$, retornando $\boldsymbol{y}$. No bloco *Declipping*, é executado o *ATA-SPADE* considerando a entrada $\boldsymbol{y}$ e as condições $\boldsymbol{M}_c$, $\boldsymbol{M}_s^+$ e $\boldsymbol{M}_s^-$. A saída do *Declipping* ($\hat{\boldsymbol{x}}$) é enviada para o *Window out/Overlap Add* retornando o vetor $\hat{\boldsymbol{x}}'$, que é a saída da implementação. Os vetores considerados têm tamanho $d = 2048$.

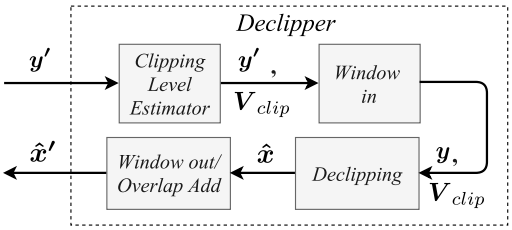

Fig. 3. Diagrama em blocos da implementação em FPGA.

A. Banco de Memórias

Observa-se no *ATA-SPADE* que suas operações matemáticas são realizadas com vetores de tamanho elevado (2048). Portanto, para uma implementação eficiente, deve-se elaborar uma estratégia de transmissão de dados entre os blocos. Propõe-se utilizar o banco de memórias RAM apresentado na Fig. 4, Os blocos do *Declipper*, mostrado na Fig. 3, podem acessar simultaneamente o banco de memórias.

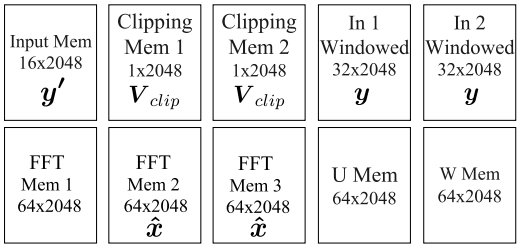

Fig. 4. Banco de memórias para armazenamento de vetores.

Conforme apresentado na Fig. 4, é utilizada uma memória de 16x2048 bits para armazenar o vetor $\boldsymbol{y'}$, duas de 1x2048 para o indicador de *clipping* $\boldsymbol{V}_{clip}$, duas de 32x2048 para o vetor $\boldsymbol{y}$, três de 64x2048 para os cálculos da FFT/IFFT e duas memórias de 64x2048 para os vetores $\boldsymbol{u}^{(i)}$ e $\boldsymbol{w}^{(i)}$.

A entrada $\boldsymbol{y'}$ é um vetor com amostras de 16 bits e precisão de ponto fixo. Já os vetores utilizados pelo *Declipping* são de 32 bits para números reais e 64 bits para complexos, a fim de reduzir os erros nas operações do *ATA-SPADE*.

B. Paralelismo entre Blocos

Na implementação realizada em FPGA, foi utilizada uma estratégia de duplicação dos vetores $\boldsymbol{V}_{clip}$, $\boldsymbol{y}$ e $\hat{\boldsymbol{x}}$ para permitir que os blocos *Window in*, *Declipping* e *Window out/Overlap Add* operem em paralelo. Considerando tal estratégia, o tempo de processamento da implementação é limitado apenas pela execução do *Declipping*, que é o mais complexo computacionalmente.

C. Bloco Declipping

Neste artigo, o algoritmo *ATA-SPADE* é utilizado no *Declipping*, que é o principal bloco do *Declipper*. A FFT ($\boldsymbol{A}$) e sua inversa ($\boldsymbol{A}^H$) são as operações de maior complexidade computacional do *Declipping*. Tais operações foram otimizadas a fim de obter o menor número possível de ciclos de *clock*. As otimizações correspondem à utilização de estratégias de acesso às memórias, de paralelismo e de *pipeline*. O resultado obtido para o cálculo da FFT (ou sua inversa) é de 11407 ciclos para uma entrada de 2048 amostras. Já para uma iteração da implementação do *ATA-SPADE* obteve-se 24839 ciclos de *clock*. Analisando esses resultados, observa-se que o cálculo da FFT/IFFT possui grande impacto no desempenho da implementação.

V. Resultados

Inicialmente, é apresentada uma comparação de desempenho entre os algoritmos *A-SPADE* e *ATA-SPADE*. Na sequência, são discutidos os resultados da implementação em FPGA.

A. Comparação entre A-SPADE e ATA-SPADE

O algoritmo proposto (*ATA-SPADE*) é comparado quantitativamente (*MATLAB*) ao *A-SPADE*. Assim como em [6], foi utilizada a diferença entre as razões sinal-distorção entre a entrada saturada ($\text{SDR}_{\boldsymbol{y'}}$) e a saída ($\text{SDR}_{\hat{\boldsymbol{x}}'}$), em que

$$\text{SDR}_{\boldsymbol{a}} = 20\log_{10}\frac{||\begin{bmatrix}\boldsymbol{M}_s^+\\ \boldsymbol{M}_s^-\end{bmatrix}\boldsymbol{x}||_2}{||\begin{bmatrix}\boldsymbol{M}_s^+\\ \boldsymbol{M}_s^-\end{bmatrix}\boldsymbol{x} - \begin{bmatrix}\boldsymbol{M}_s^+\\ \boldsymbol{M}_s^-\end{bmatrix}\boldsymbol{a}||_2} \tag{6}$$

onde $\boldsymbol{a}$ corresponde a $\hat{\boldsymbol{x}}'$ ou $\boldsymbol{y'}$ no cálculo do respectivo SDR.

O experimento avaliou o desempenho dos algoritmos *ATA-SPADE* e *A-SPADE* para sinais de áudio saturados artificialmente com diferentes valores de $\text{SDR}_{\boldsymbol{y'}}$ e τ conhecido. Os exemplos de áudio utilizados consistem de um conjunto com dez trechos de músicas de 10 segundos com diferentes conteúdos espectrais. A taxa de amostragem é de 16 kHz e a quantização de 16 bits. Foram utilizados os parâmetros de $s = 4$ e $r = 1$ para o *A-SPADE*, $\mu = 1,5$ e $\sigma^2 = 455$ para o *ATA-SPADE* e, para ambos, $\epsilon = 0,05$, fator de redundância $\left(\frac{d}{n}\right)$ de $1,33$, *frames* de $n = 1536$ amostras, *overlap* de 50% e a janela de *Hanning*.

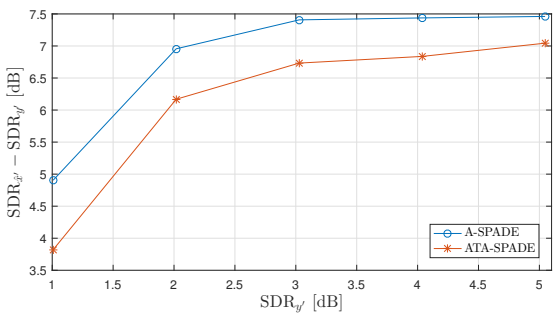

Fig. 5. Comparação de desempenho entre os algoritmos.

Observa-se na Fig. 5 que o algoritmo proposto obteve resultados ligeiramente inferiores ao *A-SPADE*, com uma diferença

média de $0,71$ dB no SDR do sinal recuperado. Entretanto, pôde-se constatar que o *ATA-SPADE* é aproximadamente 8 vezes mais rápido. O número de iterações por *frame* ($n = 1536$ amostras) variou de 222 a 504 para o *A-SPADE*, e entre 54 e 63 para o *ATA-SPADE*.

B. Resultados em FPGA

A implementação do *Declipper* em FPGA é descrita em Verilog no ambiente Quartus II (ALTERA). O FPGA escolhido é o EP4CE115F29C7 da família Cyclone IV. Os testes experimentais foram realizados no *kit* de desenvolvimento DE2-115 disponibilizado pela ALTERA. A frequência máxima de operação obtida foi de 300 MHz e os resultados de ocupação no dispositivo considerado são apresentados na Tabela I.

TABELA I
OCUPAÇÃO DO *Declipper* NO EP4CE115F29C7.

	Usado	Total	Percentual
Total de elementos lógicos	11307	114480	10%
Total de funções combinacionais	10242	114480	9%
Registros lógicos dedicados	7589	114480	7%
Total de bits de memória	1129677	3981312	28%

Observa-se na Tabela I que o recurso mais utilizado na implementação é o número de bits de memória, resultando em, no máximo, 3 instâncias do *Declipper* nesse FPGA.

A implementação do *Declipper* é simulada no *software* Modelsim-Altera e o sinal de saída $\hat{\boldsymbol{x}}'$ é salvo em um arquivo externo. Na simulação, a entrada $\boldsymbol{y}'$ é um sinal de áudio saturado com taxa de amostragem de 48 kHz, o fator de redundância $\left(\frac{d}{n}\right)$ é de $1,33$, *frames* de 1536 amostras e *overlap* de 50%. O número de iterações é fixado em 120, tornando o critério de parada ϵ desnecessário. Para a função $\mathcal{D}(i)$, foram escolhidos os parâmetros $\mu = 1,5$ e $\sigma^2 = 455$. A frequência mínima de operação, obtida em simulação, para o funcionamento em tempo real do *Declipper* é de 187 MHz, consideravelmente menor que a máxima obtida na síntese.

Em seguida, foram realizados testes experimentais no *kit* DE2-115, considerando as mesmas condições da simulação. Os sinais de saída obtidos no *kit* foram salvos em um arquivo externo para validar os experimentos. A Fig. 6 mostra um trecho do sinal original $\boldsymbol{x}$, o correspondente saturado $\boldsymbol{y}'$ e a saída recuperada $\hat{\boldsymbol{x}}'$ em um dos testes realizados. Observa-se que a saída se aproxima de $\boldsymbol{x}$ para uma saturação em torno de 50% da amplitude máxima de $\boldsymbol{x}$.

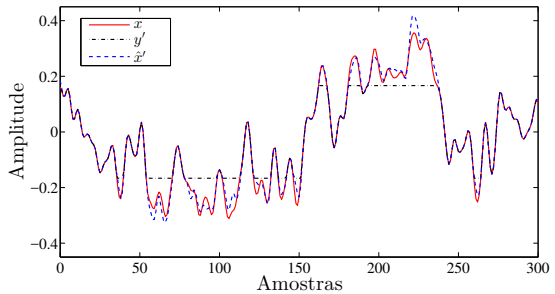

Fig. 6. Trecho de sinal de áudio (vermelho), sua versão saturada (preto) e restaurada (azul).

VI. CONCLUSÕES

Neste artigo, são apresentadas contribuições que reduzem a complexidade computacional do algoritmo *A-SPADE*, visando a execução de um *Declipper* de sinais de áudio em tempo real. As contribuições propostas incluem a aceleração de *A-SPADE* através da utilização de um *threshold* para impor esparsidade e, de uma regra de atualização desse *threshold*. Tais contribuições originaram o algoritmo *ATA-SPADE*, que se mostrou significativamente mais rápido que o *A-SPADE* e com desempenho equivalente. Visando a implementação do *Declipper* em sistemas embarcados, também é proposto um estimador de nível de *clipping* baseado no histograma de sinais de áudio. O *Declipper* proposto é implementado em FPGA e tem seu desempenho em tempo real avaliado em um *kit* de desenvolvimento. Os resultados experimentais obtidos comprovaram o desempenho do *Declipper* proposto.

REFERÊNCIAS

[1] E. Vickers, "The loudness war: Background, speculation, and recommendations," in *Audio Engineering Society Convention 129*, November 2010. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=15598

[2] M. J. Harvilla and R. M. Stern, "Efficient audio declipping using regularized least squares," in *proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Australia, April 2015, pp. 221–225.

[3] A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, and M. D. Plumbley, "A constrained matching pursuit approach to audio declipping," in *proc. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 329–332.

[4] B. Defraene, N. Mansour, S. D. Hertogh, T. van Waterschoot, M. Diehl, and M. Moonen, "Declipping of audio signals using perceptual compressed sensing," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 12, pp. 2627–2637, December 2013.

[5] A. J. Weinstein and M. B. Wakin, "Recovering a clipped signal in sparseland," *CoRR*, vol. abs/1110.5063, 2011. [Online]. Available: http://arxiv.org/abs/1110.5063.

[6] S. Kitic, N. Bertin, and R. Gribonval, "Sparsity and cosparsity for audio declipping: a flexible non-convex approach," *CoRR*, vol. abs/1506.01830, 2015. [Online]. Available: http://arxiv.org/abs/1506.01830.

[7] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, June 2010.

[8] J. O. Smith, *Physical Audio Signal Processing*. http://ccrma.stanford.edu/~jos/pasp/Cubic_Soft_Clipper_Spectrum.html, online book, 2010 edition, [Acessado 21 de Março de 2017].

[9] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, vol. 1, no. 3, 2013.

[10] S. Nam, M. Davies, M. Elad, and R. Gribonval, "The cosparse analysis model and algorithms," *Applied and Computational Harmonic Analysis*, vol. 34, no. 1, pp. 30 – 56, January 2013.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000016.

[12] G. Heinzel, A. Rüdiger, and R. Schilling, "Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new at-top windows," *MPI für Gravitationsphysik - Laser Interferometry & Gravitational Wave Astronomy*, 2002.

[13] M. J. Harvilla and R. M. Stern, "Robust parameter estimation for audio declipping in noise," in *proc. Sixteenth Annual Conference of the International Speech Communication Association*, Dresden, Germany, September 2015, pp. 2459–2463.