RECONSTRUÇÃO *ONLINE* PARA CALORÍMETROS OPERANDO EM CONDIÇÕES DE ALTAS LUMINOSIDADES

Marcos Vinícius Teixeira*, Luciano M. de A. Filho*, Bernardo S. Peralva*

*UFJF – Universidade Federal de Juiz de Fora
Faculdade de Engenharia – Programa de Pós–Graduação em Engenharia Elétrica
Campus Universitário, Plataforma 5 – Martelos – Juiz de Fora/MG – CEP:36036-030

Emails: teixeira.marcosv@gmail.com, luciano.ma.filho@gmail.com, bernardo.peralva@gmail.br

Abstract— This paper presents an online energy reconstruction algorithm for calorimeters operating in high-luminosity environments and its FPGA implementation. Based on the Gradient Descent algorithm, the implemented method has its properties defined through the COF (Constrained Optmal Filter) method. The COF aims at designing a luminosity independent signal amplitude estimator. Furthermore, COF recovers the amplitude of the central signal and the amplitude of each superposed component. However, the method results in a matrix inversion in the work proposed here. In order to avoid such computation, the Gradient Descent algorithm is applied on the COF method, resulting in sum and product operations, which are suitable in FPGA for online applications. For this end, it was developed a dedicated RISC (Reduced Instruction Set Computer) processor with pipelined architecture, in order to optimize the FPGA resources. It was also developed a C compiler to facilitate the development of the Gradient Descent algorithm subroutines.

Keywords— Amplitude Estimation, Best Linear Unbiased Estimator, Optimal Filter, Gradient Descent

Resumo— Este trabalho apresenta um algoritmo para reconstrução *online* de energia em calorímertros operando em ambiente de alta luminosidade e sua respectiva implementação em FPGA. Baseado no algoritmo Gradiente Descendente, o método implementado tem suas propriedades definidas através do método **COF** (do inglês, *Constrained Optmal Filter*). O **COF** visa estimar a amplitude de sinais empilhados independentemente das informações de luminosidade, recuperando, além do sinal de interesse, os sinais sobrepostos. Porém o método resulta em inversão de matrizes para a estimativa da amplitude dos sinais. Para evitar a inversão de matrizes, o algoritmo Gradiente Descendente é aplicado no método **COF** resultando em operações de soma e produto, facilitando sua implementação em FPGA para reconstrução *online*. Para este fim, foi desenvolvido um processador RISC (do inglês, *Reduced Instruction Set Computer*) dedicado com arquitetura *pipeline*, permitindo a otimização de recursos da FPGA. Também foi desenvolvido um compilador C para facilitar o desenvolvimento das subrotinas do algoritmo Gradiente Descendente.

Palavras-chave— Estimação de Amplitude, Melhor Estimador Linear Imparcial, Filtro Ótimo, Gradiente Descendente.

1 Introdução

O LHC (do inglês, *Large Hadron Collider*) (Evans, 2008), é o maior acelerador de partículas do mundo. O experimento opera acelerando feixes de prótons em sentidos opostos, onde cada feixe é formado por milhares de pacotes de prótons. Os pacotes, então, colidem a uma taxa constante de 40 MHz no ponto de interesse dos detectores posicionados ao longo do acelerador.

O ATLAS (do inglês, *A Toroidal LHC ApparatuS*), apresentado na Figura 1, é o maior detector do LHC. Seus sub-detectores construídos em forma cilíndrica, envolvem o feixe de partículas em um ponto de colisão (Collaboration, 2008). O detector de trajetórias (Ros, 2003), que está na camada mais interna, envolvendo a linha de feixe de partículas, é responsável por medir o momento e a posição do vértice das partículas incidentes. O sistema de detecção de múons (Palestini, 2003) forma a camada mais externa do ATLAS e é responsável pela identificação de múons não absorvidos pelos calorímetros.

Os calorímetros, localizados entre os detectores de trajetórias e múons, são

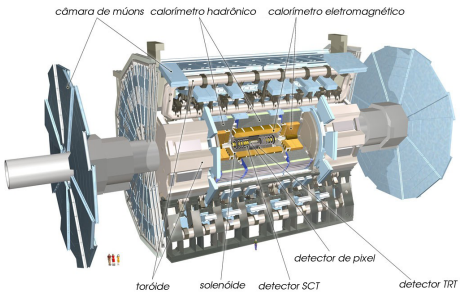

Figura 1: Detector ATLAS.

responsáveis por absorver e amostrar, com precisão, a energia das partículas incidentes (Wigmans, 2000). Eles se dividem em duas partes, o calorímetro eletromagnético (LAr) (ATLAS Group, 1996a) e calorímetro hadrônico (TileCal) (ATLAS Group, 1996b; Adragna, June 2006), sendo o último, o ambiente de pesquisa deste trabalho.

O TileCal utiliza o aço como material absorvedor passivo e telhas cintilantes como amostradores de energia. Com a passagem das partículas, as telhas se excitam,

produzindo fótons, que são coletados por células fotomultiplicadoras (PMT, do inglês *PhotoMultiplier Tube*), cuja função é converter a luz em um sinal elétrico. O sinal gerado na PMT passa por um circuito conformador de pulsos, visando tornar o sinal um pulso padrão (pulso de referência do TileCal), cuja amplitude é diretamente proporcional à energia depositada pela partícula. Cada PMT recebe sinais luminosos de certo número de telhas cintilantes formando as células do calorímetro. Cada célula é formada por duas PMTs, resultando em duas leituras independentes. O TileCal é segmentado ao longo da profundidade do detector em camadas e cada camada apresenta um certo número de canais de leituras (PMTs). No total, o TileCal apresenta, aproximadamente, 10.000 canais de leitura a serem lidos pelo sistema de aquisição. Cada canal de leitura amostra um pulso de referência na taxa de 40 MHz resultando em 7 amostras discretas separadas de 25 ns umas das outras, compondo uma janela de leitura de 150 ns.

Atualmente, o método de reconstrução *online* de energia para o TileCal tem sua formulação matemática baseada na estimativa da amplitude de apenas um pulso de referência. Na cadeia eletrônica do TileCal, o tempo de processamento para o segundo nível de filtragem (Conde et al., 2008), correspondente a reconstrução de energia, não pode exceder 10 us. O mesmo é implementado basicamente por uma estrutura de filtro **FIR** (do inglês, *Finite Impulse Response*). A Figura 2 ilustra o pulso característico de um canal do TileCal com seus parâmetros de amplitude, largura, pedestal (linha de base) e fase.

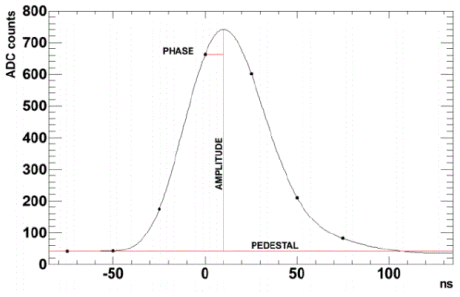

Figura 2: Pulso característico do TileCal.

Este método, chamado de **OF** (Fullana, 2006)(do inglês, *Optmal Filter*), é otimizado para estimar a amplitude de um sinal determinístico em ruído gaussiano. Entretanto, devido ao aumento do número de colisões prótons-prótons, ocorre o fenômeno de *pileup* (Cleland and Stern, 1994), onde os sinais são acumulados em uma mesma janela de 150 ns. Conseqüentemente, a sobreposição de sinais em condição de alta luminosidade degrada o pulso original comprometendo a estimativa da energia atual. No método **OF**, os sinais acumulados são tratados como ruído e a estatística de segunda ordem do mesmo é usada no projeto do filtro de forma a minimizar o efeito do *pileup*. Porém, o ruído deixa de ser Gaussiano nestas condições, o que torna o filtro sub-ótimo. Além disso, o projeto do filtro torna-se dependente da luminosidade, que varia durante uma tomada de dados.

Tendo em vista os inconvenientes e no intuito de solucioná-los, a colaboração brasileira propôs o método **OF** com restrições adicionais, chamado de **COF** (L. Filho, 2012). A proposta visa estimar, além da amplitude do pulso de interesse, a amplitude de cada componente sobreposta o que caracteriza as restrições adicionais se comparado ao método **OF**. Nestas condições o ruído é caracterizado somente pelo ruído eletrônico que é Gaussiano, mantendo a característica ótima do estimador. Entretanto, o método **COF** resulta em uma inversão de matrizes, dificultando a implementação em *hardware* para a estimação *online*. Atualmente o **COF** está sendo utilizado para estimação *offline* e vem obtendo excelentes resultados. Porém, sua implementação para processamento *online* não é otimizada pois se faz necessário a utilização de bancos de filtros para cada combinação de *pileup*.

Sendo assim, o presente trabalho propõem a implementação do processo de minimização do **COF**, através de um processo iterativo, baseado em Gradiente Descendente (GD) (Haykin, 1996) para a estimação *online*. O método resulta em operações de soma e produto, possibilitando sua implementação em dispositivos FPGA. Para a execução destas operações, foi desenvolvido um processador dedicado, baseado em arquitetura **RISC** para otmização de recursos em FPGA. O processador tem como núcleo de execução uma ALU (do inglês, *Arithmetic Logic Unit*), capaz de efetuar as operações de soma e multiplicação em ponto flutuante. O processador apresenta, como técnica de implementação, uma arquitetura em *pipeline*, capaz de executar uma instrução por ciclo de clock. O conjunto de instruções *assembly*, armazenado na memória principal do processador, são geradas por subrotinas em linguagem C, através de um compilador, também desenvolvido na presente proposta. Apesar de o **GD** ser uma técnica bem difundida em processos de minimização, sua utilização na medida de múltiplas amplitudes, em um processador dedicado, visando a reconstrução de energia em calorímetros é algo novo.

Este trabalho está organizado da seguinte forma. Na próxima seção são apresentados os métodos de reconstrução de energia visando descrever a coerência para o método proposto. Na Seção III apresenta-se o método proposto. A Seção IV descreve a implementação do algoritmo do Gradiente Descendente e o núcleo do processador para execução do algoritmo

em FPGA. Na Seção V são apresentadas as conclusões.

2 Métodos de Reconstrução de Energia

A seguir são apresentadas as formulações para os métodos de reconstrução de energia. A descrição dos mesmos visa o embasamento teórico para o método proposto. Inicialmente descreve-se, através do método **BLUE** (do inglês, *Best Linear Unbiased Estimator*) (Kay, 1993) para um vetor de parâmetros, a estimativa da amplitude de um modelo genérico. Posteriormente, o método **COF** para reconstrução de sinais no ambiente do TileCal é apresentado.

2.1 BLUE para um vetor de parâmetros

O método **BLUE** tem como objetivo restringir-se a um estimador linear, de modo a determinar os pesos ótimos de um filtro que minimiza a variância da estimativa. Para compreensão da equação geral do método **BLUE**, um modelo, para estimação das amplitudes de sinais determinísticos sobrepostos (Kay, 1993), é apresentado. Neste, as amostras do sinal de entrada digitalizado são dadas pela Equação (1):

$$\mathbf{r} \;=\; \mathbf{Ga} \;+\; \mathbf{w} \tag{1}$$

onde $\mathbf{a}$ é o vetor coluna que contém as amplitudes dos sinais sobrepostos. Cada coluna da matriz $\mathbf{G}$ representa as amostras dos sinais de referência, cujas amplitudes estão contidas no vetor $\mathbf{a}$. A componente $\mathbf{w}$ trata-se do ruído aditivo WG (do inglês, *White-Gaussian*) com média zero. Sendo assim, a equação geral do método **BLUE** pode ser definida como o produto interno entre os pesos do filtro e as amostras do sinal de entrada, conforme a Equação (2):

$$\hat{\mathbf{a}} = \mathbf{H^T}\,\mathbf{r} \tag{2}$$

onde $\hat{\mathbf{a}}$ é a estimativa resultante do vetor de amplitudes e $\mathbf{H}$ representa a matriz de coeficientes do banco de filtros lineares a ser determinado. Para que os pesos do **BLUE** sejam encontrados, é imposta a imparcialidade na estimativa de $\hat{\mathbf{a}}$. Um estimador é dito imparcial, ou não-tendencioso (do inglês, *unbiased*), se o valor esperado da estimativa for igual ao valor real. Portanto, para um estimador linear não-tendencioso, temos que:

$$E\{\hat{\mathbf{a}}\} = \mathbf{H^T}\,E(\mathbf{r}) = \mathbf{a} \tag{3}$$

Substituindo a Equação (1) na Equação (3), o valor esperado da estimativa resultante do vetor de amplitudes $E\{\hat{\mathbf{a}}\}$ poderá ser reescrita conforme a Equação (4):

$$E\{\hat{\mathbf{a}}\} = \mathbf{H^T}\,E\{\mathbf{Ga} \;+\; \mathbf{w}\} = \mathbf{a} \tag{4}$$

Seja o vetor $\mathbf{w}$ um ruído aditivo com média zero, então a Equação (4) para o valor esperado da estimativa resultante do vetor de amplitudes $E\{\hat{\mathbf{a}}\}$ pode ser reescrita, como mostra a Equação (5):

$$E\{\hat{\mathbf{a}}\} = \mathbf{H^T}\,\mathbf{Ga} = \mathbf{a} \tag{5}$$

No método **BLUE**, seja N o número de sinais sobrepostos, então N restrições são inseridas no processo de minimização de cada componente. Logo, para que $E\{\hat{\mathbf{a}}\}$ seja igual a $\mathbf{a}$, o produto entre a matriz dos pesos do **BLUE** $\mathbf{H}$ e a matriz do pulso de referência $\mathbf{G}$ deve ser igual a matriz identidade $\mathbf{I}$, conforme é apresentado pela Equação (6).

$$\mathbf{H^T}\,\mathbf{G} = \mathbf{I} \tag{6}$$

Desta forma, o valor esperado da estimativa resultante do vetor de amplitudes $E\{\hat{\mathbf{a}}\}$ será igual ao próprio vetor de constantes $\mathbf{a}$ e, dado que o vetor $\mathbf{a}$ representa a amplitude real das amostras do sinal de entrada $\mathbf{r}$, a propriedade de imparcialidade apresentada pela Equação (6) é atendida. Uma vez determinada a restrição para pesos do **BLUE**, o objetivo passa a ser encontrar a variância dos estimadores, a partir da Equação (7):

$$var\{\hat{\mathbf{a}}\} \;=\; E\left\{(\hat{\mathbf{a}} - E\{\hat{\mathbf{a}}\})^2\right\} \tag{7}$$

Resolvendo a Equação (7) encontramos a Equação (8).

$$var\{\hat{\mathbf{a}}\} = \mathbf{H^T C\,H} \tag{8}$$

onde $\mathbf{C}$ é a matriz de covariância do ruído. A etapa seguinte consiste em aplicar o método dos multiplicadores de Lagrange (Bertsekas, 1996), de forma a minimizar a Equação (8) dado as restrições (6). Como N restrições são inseridas no processo de minimização de cada componente, então N^2 multiplicadores são utilizados no método. O método de Lagrange, formado por uma combinação linear, determina o ponto mínimo da variância da estimativa da amplitude condicionada à restrição imposta pela Equação (6). Isto é feito derivando a função de Lagrange e igualando-a a zero. Em seguida, as etapas da formulação matemática consistem em isolar variáveis e substituí-las para se determinar os pesos do **BLUE** e, conseqüentemente, a estimativa resultante ótima do vetor de amplitudes. Este procedimento é

descrito em (Kay, 1993). Logo, os pesos ótimos do **BLUE** para um vetor de parâmetros são encontrados através da Equação (9):

$$\mathbf{H_{otimo}} = \left(\mathbf{G^T\,C^{-1}G}\right)^{-1}\mathbf{C^{-1}G} \tag{9}$$

Substituindo a Equação (9) na Equação (2), a estimativa ótima da amplitude do sinal é determinada por meio da Equação (10):

$$\mathbf{\hat{a}} = \left(\mathbf{G^T\,C^{-1}G}\right)^{-1}\mathbf{G^TC^{-1}r} \tag{10}$$

2.2 OF com restrições adicionais - COF

Para o caso em que se deseja estimar apenas a informação relevante para o sistema de *trigger* do TileCal, ou seja, apenas o pulso central da janela, os pesos de um filtro linear seriam dados pela Equação (9), substituindo a matriz **G** pelo vetor **g** com as amostras do sinal de referência do TileCal. Portanto, na ausência de *pileup*, e assumindo que o ruído eletrônico é gaussiano branco (WG), a Equação (9) pode ser reescrita conforme a Equação (11)

$$\mathbf{h_{otimo}} = \frac{\mathbf{g}}{\|\mathbf{g}\|^{\mathbf{2}}} \tag{11}$$

Para o caso particular em que se observa *pileup* em algum BC (do inglês, *Bunch-Crossings*) e assumindo que o ruído eletrônico é WG, a Equação (9) pode ser simplificada, conforme é apresentado pela Equação (12):

$$\mathbf{H_{otimo}} = \left(\mathbf{G^T\,G}\right)^{-1}\mathbf{G} \tag{12}$$

onde **G** contém os sinais de referência relacionados aos BC onde o *pile-up* foi observado. O processo de detecção de *pileup*, para os 7 BC centrais, também pode ser executado com a ajuda da Equação (10). No caso especial em que se deseja estimar todas as sete componentes sobrepostas, por ser uma matriz quadrada, podemos reescrever a Equação (12) reduzindo-a, como mostra a Equação (13). Desta forma, a inversa da matriz $\mathbf{G^T}$ é, portanto, chamada de matriz de deconvolução **(DM)**.

$$\mathbf{H_{otimo}} = \left(\mathbf{G^T}\right)^{-1} \tag{13}$$

O processo de deconvolução indica a existência de informações de *pile-up*. Estas informações são então comparadas com um limiar. Se as informações de *pile-up* forem maiores do que este limiar, então o **COF** será capaz de reconstruir a energia de cada componente detectada com o mínimo de restrições. As figuras a seguir ilustram o diagrama o método **COF**.

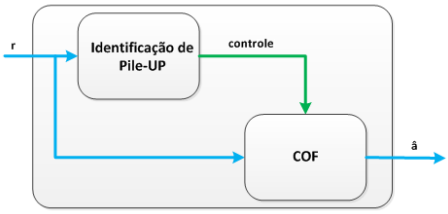

Figura 3: Diagrama Geral do método COF.

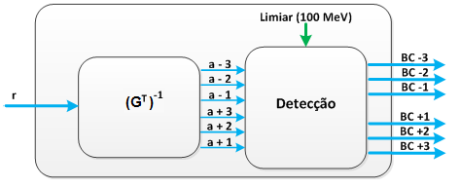

Figura 4: Identificação de PileUp e Reconstrução da Energia.

3 Método Proposto

De forma a implementar o **COF** iterativamente, utilizando somente operações de soma e produto, partimos da função custo para a minimização do erro médio quadrático entre o sinal recebido **r** e a projeção das amplitudes estimadas $\mathbf{\hat{a}}$ na matriz dada pelos sinais de referência **G**, como mostra a Equação (14):

$$J\left(\hat{a}\right) \;=\; \|\mathbf{G\hat{a}} \;-\; \mathbf{r}\|^{\mathbf{2}} \tag{14}$$

Atualizando-se os valores em $\mathbf{\hat{a}}$ na direção contrária ao gradiente de J, em pequenos passos, obtém-se os estimadores de amplitude $\mathbf{\hat{a}}$ que miniminizam o erro médio quadrático. Este algoritmo é conhecido como Gradiente Descendente. Assim, derivando J em relação a $\mathbf{\hat{a}}$, temos:

$$\nabla J = 2\mathbf{G}\left(\mathbf{G\hat{a}} \;-\; \mathbf{r}\right) \tag{15}$$

O processo iterativo na determinação de $\mathbf{\hat{a}}$ é implementado da seguinte forma,

$$\mathbf{\hat{a}}[n+1] = \mathbf{\hat{a}}[n] - \mu\mathbf{G}\left(\mathbf{G\hat{a}}[n] - \mathbf{r}\right) \tag{16}$$

.

onde μ e a taxa de evolução do processo iterativo. O valor inicial de $\mathbf{\hat{a}}[\mathbf{0}]$ é obtido com o processo de deconvolução dado pela Equação (13).

Determinada a formulação matemática do método proposto, os passos seguintes consistem em determinar a faixa de valores de μ que permitem a convergência do algoritmo para a estimativa da amplitude de sinais sobrepostos ou mesmo sem a existência do fenômeno de *pile-up*. Da mesma forma busca-se determinar o número

mínimo de iterações em que o desvio entre o **COF** e o **GD** seja inferior a 1%.

Para tal, inicialmente gera-se um vetor linha com 98000 amostras sendo que 10% destas amostras apresentam dados válidos. Ou seja, a ocupância de 10% caracteriza os valores de simulação para as amostras de entrada **r** (sem ruído aditivo) quando o vetor linha com 98000 amostras é dividido 7. Neste caso, cada linha da matriz 14000 x 7 contém um vetor de entrada **r**. Uma vez que trata-se de um processo aleatório, em algumas situações o vetor de entrada **r** não irá conter sinal ou mesmo *pile-up*. Em outros casos, há existência de *pile-up* no BC central com influência de seus BCs adjacentes imediatos o que caracteriza uma situação real no calorímetro. Após determinado os vetores de entrada **r** é adicionado ruído WG.

Em seguida, no processo de identificação, o **DM** é calculado. Para um limiar de 5 ADC *counts*, o **DM** identifica *pile-up* no BC central e em seus adjacentes imediatos, quando há ocorrência. Após obter as informações do **DM** o algoritmo **COF** é executado. De posse dos resultados do **DM** e do **COF** o **MSE** (do inglês, *Mean Squared Error*) é calculado.

O algoritmo **GD** é então executado tendo como valores iniciais de convergência os dados calculados pelo **DM**. Os valores acima do limiar aplicados como valores iniciais para o algoritmo **GD** permitem a convergência de forma eficiente. Além disso, a identificação de *pile-up* antes da execução do algoritmo possibilita a reconstrução do sinal com o mínimo de restrições. Finalmente, de posse dos resultados, o **MSE** entre o **GD** e **COF** é calculado.

Nesta simulação o **GD** foi executado considerando 20 valores para μ dentro faixa $0.01 < \mu < 0.6$. Para o número de interações também foi considerado 20 valores dentro da faixa de $0 <$ iterações < 2000. A Figura (5) mostra a simulação para identificação de valores μ que divergem.

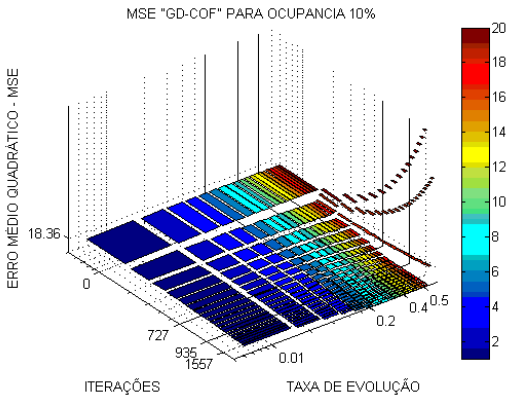

Figura 5: Identificação dos valores de μ em que há divergência.

Através da Figura (5) percebe-se que o algoritmo **GD** diverge para valores de μ acima de 0.5. Então, a partir desta análise define-se que o algoritmo será executado para μ dentro faixa $0.01 < \mu < 0.5$. Logo, obtém-se o gráfico de convergência do algoritmo como mostra a Figura (6).

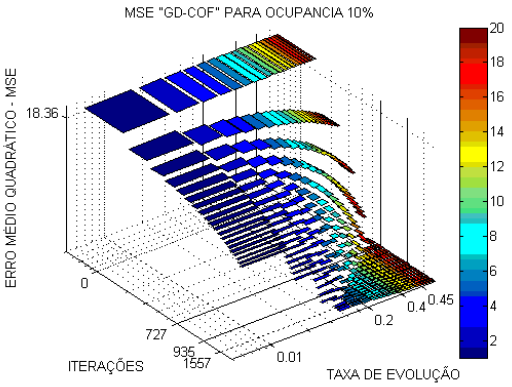

Figura 6: Convergência do algoritmo **GD**.

Note que quando o valor do número de interações é igual a zero os valores são constantes independentemente da variação de μ. Uma vez que o **GD** não executa nenhuma interação e tendo em vista que o mesmo inicializa com valores calculados pelo **DM**, então seu **MSE** será idêntico ao **MSE** calculado entre o **DM** e o **COF**. Percebe-se que a medida em que o número de iterações aumenta com o aumento da taxa de evolução o **MSE** entre o **GD** e o **COF** diminui.

Determinado a faixa de valores para μ o passo seguinte consiste em encontrar o número mínimo de interações em que o desvio entre o **COF** e o **GD** seja inferior 1%. Diversas simulações foram realizadas para encontrar o mínimo de interações considerando a condição estabelecida. O resultado é apresentado na Figura (7), onde é possível concluir que 20 iterações é o suficiente para se obter desvio inferior a 1%.

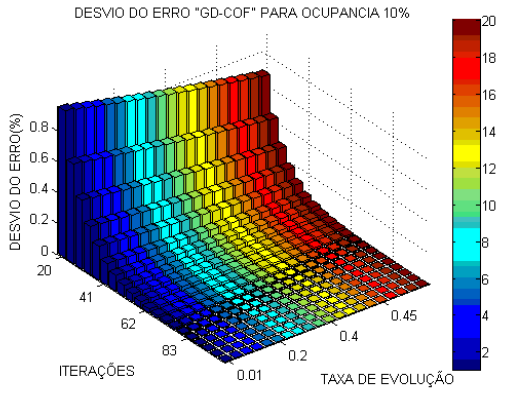

Figura 7: Identificação do valor mínimo de iterações para desvio de 1%

Definido os valores da taxa de convergência e número de interações, a seguir é apresentado o algoritmo proposto.

Data:

- $\mathbf{r}$: Sinal de entrada digitalizado
- $\mathbf{G}$: Sinais de referência
- $\mathbf{a}[\mathbf{0}] = \mathbf{G^{-1}}\ \mathbf{r}$: Amp. Estimada DM

Result:

- $\hat{\mathbf{a}}(n+1)$: Vetor de Amp. Estimadas

$\mu \leftarrow 0.49$, taxa de evolução;
$it \leftarrow 20$, máxima iteração;
$n \leftarrow 0$, valor inicial para convergência;
for $n \leftarrow 1$ ***to*** it **do**
  $\hat{\mathbf{a}}[n+1] \leftarrow \hat{\mathbf{a}}[n] - \mu\mathbf{G}(\mathbf{G}\hat{\mathbf{a}}[n] - \mathbf{r})$;
end

Na simulação foi considerado a existência de *pile-up* no BC central com influência dos BCs a_{-2} e a_{+3}. Desta forma, para um vetor de amplitudes $\mathbf{a} = [a_{-3}\ a_{-2}\ a_{-1}\ a_0\ a_{+1}\ a_{+2}\ a_{+3}]$ foram atribuídos os valores [0 29 0 25 0 0 27]. Após a adição de ruído, os valores de simulação para as amostras de entrada $\mathbf{r}$ são obtidos, [14.23 29 28.69 30.02 17.37 16.89 25.38].

No processo de identificação do *pile-up* o **DM** é calculado, resultando em um vetor com valores [0 23.78 0 29.18 0 0 22.75]. Para um limiar de 5 ADC *counts*, o **DM** identifica *pile-up* no BC central e nos BCs a_{-2} e a_{+3}, como era esperado.

A Figura (5) apresenta a curva de convergência do presente método, para este exemplo.

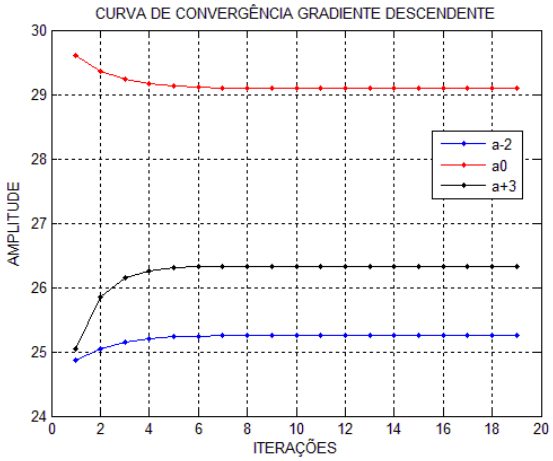

Figura 8: Curva de convergência para estimação da amplitude utilizando algoritmo GD

Para a comparação do algoritmo proposto com o método **COF**, a seguir apresenta-se uma tabela comparativa com os resultados.

	a_-2	a_0	a_+3
REAL	22.00	25.00	27.00
COF	25.25	29.09	26.32
GD	25.00	29.01	26.00

Tabela 1: Comparativo entre COF e GD para os sinais observados.

4 Implementação

Diante das características da arquitetura de um dispositivo FPGA (Meyer-Baese, 2007), a seguir descreve-se o *Top level* do núcleo do processador para execução seqüencial do algoritmo **GD**. Posteriormente são apresentados os detalhes da simulação do processador em FPGA

4.1 Arquitetura do Processador em FPGA

A Figura (9) mostra o *Top level* e suas características. O bloco PC consiste basicamente de um contador que incrementa a cada instrução realizada, indicando a posição do programa ou aplicação que o *hardware* está executando no momento. O mesmo possui também sinais de controle para carregamento externo, no caso de uma instrução de mudança de controle de fluxo ou subrotina.

Na arquitetura *pipeline* desenvolvida, enquanto a instrução atual é executada pela ULA, o bloco Prefetch faz a busca da próxima instrução na memória de programa, separando suas informações de opcode e operando e gerando os sinais de controle para mudança de fluxo de programa, caso a próxima instrução a ser executada seja uma instrução de salto ou chamada para subrotinas. Já o Instrdecoder recebe os campos separados pelo Prefetch e decodifica o que a instrução determina, comandando os blocos para realizar a operação desejada.

O Stack Pointer tem a função de colocar e retirar os dados do topo da memória, simulando uma pilha. Este recurso é essencial para o desenvolvimento do compilador C. Para suportar sub-rotinas no *hardware*, o bloco Stack armazena em qual posição de memória o programa estava anteriormente, para quando esta rotina terminar, voltar ao local correto onde a aplicação estava. O bloco Int2float converte para tipo float os dados recebidos do tipo inteiro para se adequar ao dado guardado na memória e realizar as contas corretamente e com precisão.

Finalmente o bloco RegFile implementa um ponteiro relativo para endereçamento da memória de dados. Este recurso permite a implementação, em software, de ponteiros e vetores.

A Figura (10) mostra o núcleo do processador, chamado de ULA (Unidade Lógica Aritmética), que realiza as operações de soma, multiplicação e divisão de números em ponto flutuante.

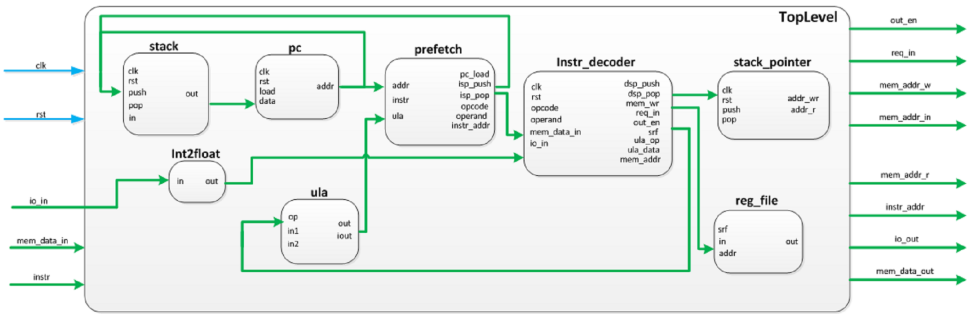

Figura 9: Diagrama de blocos da Unidade Lógica Aritmética.

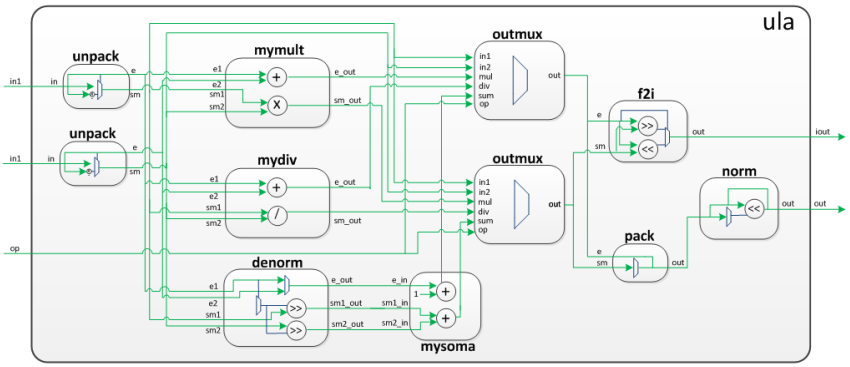

Figura 10: Diagrama de blocos da Unidade Lógica Aritmética.

A ULA constitui-se dos blocos Unpack, Mymult, Mydiv, Denorm, Mysoma, Outmux, F2i, Pack e Norm. O bloco Unpack é responsável por separar o número recebido em expoente e mantissa do ponto flutuante, verificando se o mesmo é negativo. Em caso afirmativo, o mesmo já realiza o complemento de 2. Já o bloco Mymult realiza a multiplicação dos números da entrada da ULA, multiplicando os valores das mantissas e somando os expoentes. O bloco exporta o resultado dessas duas operações em saídas distintas.

O Mydiv tem como função realizar a divisão dos números da entrada da ULA, dividindo os valores das mantissas e subtraindo os expoentes. Ele exporta o resultado dessas duas operações em saídas distintas porém, este bloco não é utilizado no algoritmo Gradiente Descendente implementado. O Denorm iguala os expoentes, realizando o deslocamento de bits da mantissa, preparando os números para serem subtraídos ou adicionados.

O bloco Mysoma, após preparar as mantissas, realiza a soma das mesmas e ajusta o expoente. O bloco outmux verifica qual foi a operação indicada pela instrução e seleciona o resultado do expoente e a mantissa deste. O bloco F2i recebe o resultado no formato de ponto flutuante e transforma em inteiro através do deslocamento adequado de bits. Já o bloco Pack tem como função realizar o empacotamento do expoente e mantissa informando se o número é positivo ou negativo. Por fim, o bloco Norm verifica o número formado pelo empacotamento e normaliza o mesmo para deixá-lo no padrão de ponto flutuante adotado para o processador.

A utilização da ULA como núcleo do processador otimiza os recursos em FPGA para a reconstrução de múltiplos sinais quando comparado aos recursos utilizados pela implementação do **COF** por meio de bancos de filtros. A saber, se considerarmos a implementação do **COF** por meio de filtros FIR para a estimação do pulso central apenas, seriam necessários sete multiplicadores e seis somas. Já com a utilização da ULA para execução do algoritmo **GD**, apenas um multiplicador e uma soma são utilizados para todas as possíveis combinações de *pile-up*.

4.2 Resultados do Processador em FPGA

Através do compilador projetado, desenvolveu-se as subrotinas em linguagem C para o algoritmo **GD**. O compilador foi desenvolvido com o parse generator GNU bison e um gerador de análise lexical (FLEX) (Donnelly and Stallman, 2008; Paxson, 1995) capaz de reconhecer padrões de texto e codificar no formato da instrução do processador. O compilador gera as intruções em assembler para o processador através das subrotinas em C. Estas instruções são então executadas pelo *hardware* digital para o co-processamento do algoritmo. O Hardware digital foi desenvolvido em linguagem descritiva

HDL, utilizando o sintetizador da Altera Quartus II.

Sabendo que o segundo nível de filtragem possui tempo de processamento máximo de 10 us e visto que 20 iterações são suficientes para um desvio inferior a 1%, então cada loop do algoritmo pode ser repetido a uma taxa de 2 MHz. Porém, para o tempo de processamento de 10 us, 7.5 us são utilizados para outras aplicações. Sendo assim, os algoritmos **DM** e **GD** precisam ser executados em um tempo total de 2.5 us. Se o resultado do **DM** é obtido com 7 ciclos de clock e o GD com 20 ciclos, então a taxa a ser utilizada no método proposto é de 10.8 MHz.

A tabela a seguir mostra a comparação entre os resultados obtidos em ambiente de simulação Matlab e os resultados obtidos após simulação em FPGA.

	a_-2	a_0	a_+3
GD MATLAB	25.00	29.01	26.00
GD FPGA	25.00	29.01	26.00

Tabela 2: Comparativo entre **GD** Matlab e **GD** FPGA.

5 Conclusões e Trabalhos Futuros

Através das formulações apresentadas, comprova-se que o método proposto, baseado no algoritmo do Gradiente Descendente, é capaz de estimar as amplitudes de sinais sobrepostos com eficiência similar à formulação atual do COF com inversão de matrizes. O método apresentado evita esta inversão possibilitando sua implementação em FPGA para o processamento *online*. Através da simulação do processador comprovou-se a viabilidade do algoritmo para reconstrução *online* de sinais em condições de alta luminosidade, uma vez que sua formulação resulta em operações simples de produto e soma e hardware otimizado. Para trabalhos futuros pretende-se implementar em FPGA os algoritmos Gradiente Descendente Adaptativo e Conjugado, o que viabiliza a utilização de ponto fluntuante. Estes algoritmos já foram simulados em ambiente Matlab obtendo excelentes resultados. Para o caso do **GD** Conjugado, os resultados mostram que o número de interações é idêntico a quantidade de *pile-up* observados.

Referências

Adragna, P. e. a. (June 2006). The atlas hadronic tile calorimeter: from construction toward physics, *Nuclear Science* **53**: 1275 – 1281.

ATLAS Group (1996a). Atlas liquid argon calorimeter technical design report, *Technical report*, CERN.

ATLAS Group (1996b). Atlas tile calorimeter technical design report, *Technical report*, CERN/LHCC/96−42.

Bertsekas, D. P. (1996). *Constrained Optimization and Lagrange Multiplier Methods*, 1st edn, Athena Scientific.

Cleland, W. and Stern, E. (1994). Signal processing considerations for liquid ionization calorimeters in a high rate environment, *Nuclear Instruments and Methods in Physics Research* **A338**: 467–497.

Collaboration, T. A. (2008). The atlas experiment at the cern large hadron collider, *Journal of Instrumentation* **3**(08003).

Conde, M. P., Aracena, I., Brelier, B. and Cranmer, K. (2008). Implementation and performance of the atlas second level jet trigger, *Journal of Physics: Conference Series* **119**(2): 022029.

Donnelly, C. and Stallman, R. (2008). *Bison*, Free Software Foundation, Inc.

Evans, L., B. P. (2008). Lhc machine, *Journal of Instrumentation* **JINST 3**(S08001).

Fullana, E. (2006). Optimal filtering in the atlas hadronic tile calorimeter, *IEEE Transactions On Nuclear Science* **53**(4).

Haykin, S. (1996). *Adaptative Filter Theory*, 3rd edn, Prentice Hall.

Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing − Estimation Theory*, 1st edn, Prentice Hall.

L. Filho, A. Cerqueira, D. D. J. S. (2012). Calorimeter signal response deconvolution for online energy estimation in presence of pile-up, *Atl-tilecal-int-2012-008*, CERN Document Serve.

Meyer-Baese, U. (2007). *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd edn, Springer.

Palestini, S. (2003). The muon spectrometer of the atlas experiment, *Nuclear Physics B* **125**: 337–345.

Paxson, V. (1995). *FLEX*, University of California.

Ros, E. (2003). Atlas inner detector, *Nuclear Physics B - Proceedings Supplements* **120**: 235–238.

Wigmans, R. (2000). *Calorimetry: Energy Measurement in Particle Physics*, Oxford University Press.