Implementação de um Equalizador Adaptativo em FPGA Utilizando o Algoritmo CORDIC

Bruno Sens Chang, Francisco J. A. de Aquino, Carlos Aurélio F. da Rocha

***Resumo*— Neste artigo apresentamos uma implementação em uma FPGA de um equalizador adaptativo que utiliza o algoritmo LMS trigonométrico. As funções trigonométricas necessárias às operações de filtragem e adaptação dos coeficientes são calculadas com rotações CORDIC. Este algoritmo impõe a necessidade da definição de um hipercubo, no qual deve estar contido o ponto de mínimo da função objetivo. Este artigo propõe a adoção do procedimento multisplit para facilitar a definição desse hipercubo. Simulações realizadas indicam que a utilização deste procedimento permite a convergência do algoritmo sem a necessidade de uma busca exaustiva das coordenadas do hipercubo.**

***Palavras-Chave*— CORDIC, equalização adaptativa, FPGA, procedimento multisplit.**

***Abstract*— In this paper we present a FPGA implementation of an adaptive equalizer using the trigonometric LMS algorithm. The trigonometric functions needed for filtering and coefficient adaptation are evaluated with CORDIC rotations. This algorithm needs a hypercube definition, in which the minima of the objective function must be contained. This paper proposes the adoption of the multisplit procedure to ease the hypercube definition. Simulations show that, when using this procedure, the algorithm converges without the need for an exaustive search for the hypercube coordinates.**

***Keywords*— CORDIC, adaptive equalization, FPGA, multisplit procedure.**

I. INTRODUÇÃO

O principal problema em um sistema de comunicação é o surgimento das interferências entre símbolos (IES) devido à transmissão do sinal por um canal de comunicação não ideal. Tais interferências corrompem a mensagem transmitida, sendo necessária a utilização de um dispositivo de compensação no receptor. Filtros equalizadores são comumente utilizados para compensar a IES.

Desde seu surgimento, em 1958, o algoritmo CORDIC (*Coordinate Rotation Digital Computer*) é utilizado em várias aplicações de comunicações e processamento de sinais, devido a sua eficiência na solução de problemas trigonométricos [1]. Entretanto, para filtros adaptativos, a utilização do algoritmo CORDIC era limitada aos filtros em treliça, já que os cálculos, a cada estágio desse tipo de filtro, podem ser diretamente relacionados a rotações angulares [2], [3].

Para filtros adaptativos transversais, o algoritmo LMS trigonométrico foi introduzido por Chakraborty *et al.* em 2005 [4]. Nesse algoritmo, os cálculos necessários para as operações de filtragem e atualização dos coeficientes podem ser realizados simultaneamente por processadores CORDIC. No entanto, o artigo citado não é claro quanto ao método de definição das coordenadas do hipercubo que contém o ponto de mínimo da função objetivo.

Neste artigo é apresentada uma implementação em uma FPGA (Field Programmable Gate Array) de um equalizador adaptativo que utiliza o algoritmo LMS trigonométrico com o procedimento multisplit. Este procedimento aumenta a potência do sinal transformado, diminuindo assim os valores dos coeficientes do equalizador e tornando possível a convergência do algoritmo LMS trigonométrico com um hipercubo fixo.

As FPGAs têm sido freqüentemente utilizadas em sistemas de comunicações devido à sua grande flexibilidade, eficiência e possibilidade de reprogramação dinâmica. Na implementação do equalizador utilizou-se uma FPGA Virtex-4 produzida pela Xilinx.

O restante do artigo é organizado como segue. A Seção II contém uma breve introdução ao algoritmo CORDIC. Já a Seção III apresenta o equalizador adaptativo baseado em CORDIC com a utilização do procedimento multisplit. A Seção IV apresenta o procedimento multisplit. As principais características de uma FPGA da série Virtex 4 da Xilinx e do seu ambiente de desenvolvimento, o System Generator, são apresentadas na Seção V. A Seção VI contém os resultados de simulação do equalizador implementado na FPGA. Finalmente, a seção VII conclui o trabalho com algumas observações e propostas de continuidade para este trabalho.

II. CORDIC

O algoritmo CORDIC oferece uma formulação que, com apenas pequenas modificações, pode calcular de maneira eficiente funções trigonométricas, hiperbólicas, entre outras. O CORDIC foi desenvolvido por Jack E. Volder, em 1956, para substituir os computadores analógicos do sistema de navegação do bombardeiro B-58 por computadores digitais, já que os analógicos eram pouco precisos. Walther [5] foi o primeiro a estender o trabalho de Volder, a fim de resolver outros problemas, tais como relações hiperbólicas e sistemas lineares. Outros pesquisadores seguiram seu caminho, e muitas propostas de utilização dos modos de operação do CORDIC foram apresentadas.

Este trabalho foi parcialmente financiado pela Universidade Federal de Santa Catarina (UFSC), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), através do projeto CAPES-COFECUB número 544/07. Bruno Sens Chang é aluno de doutorado na UFSC no Grupo de Pesquisa em Comunicações (GPqCom), bolsista CNPq, e-mail: brunosens-chang@gmail.com. F. J. A. de Aquino é professor no CEFET-Ce e aluno de doutorado na UFSC (GPqCom), e-mail: fcoalves_aq@cefet-ce.br. C. A. Rocha é professor no Departamento de Engenharia Elétrica da UFSC, laboratório GPqCom. E-mail: aurelio@eel.ufsc.br.

O CORDIC é um algoritmo que utiliza um método iterativo de rotações vetoriais, utilizando apenas deslocamentos e adições [6]. Tal característica o torna muito interessante pela fácil implementação em *hardware*, especialmente naqueles sem multiplicadores.

O algoritmo é derivado das equações gerais de rotação de um vetor com coordenadas (x,y) por um ângulo ϕ qualquer, como é representado na Figura 1:

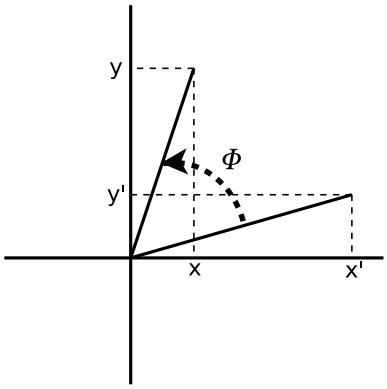

Fig. 1. Rotação Vetorial

Esta rotação vetorial pode ser expressa por:

$$x^{'} = x\cos\phi - y\sin\phi, \tag{1}$$

$$y^{'} = y\cos\phi + x\sin\phi, \tag{2}$$

onde *x'* e *y'* são as novas coordenadas.

Tais equações podem ser modificadas para atingirem a seguinte forma:

$$x^{'} = \cos\phi\left[x - y\tan\phi\right], \tag{3}$$

$$y^{'} = \cos\phi\left[y + x\tan\phi\right]. \tag{4}$$

Limitando tan ϕ_i a $\pm 2^{-i}$, é possível resolver estas equações utilizando apenas operações de soma e deslocamento. Ângulos quaisquer, dentro de uma certa precisão, podem ser formados por sucessivas rotações cada vez menores.

Para que não haja rotações desnecessárias, é necessário somar os ângulos das iterações já realizadas. Este processo é feito com outro somador, que adiciona os ângulos correspondentes às rotações calculadas.

As equações básicas do algoritmo são:

$$x^{'}_{i+1} = k_i\left[x^{'}_i - y^{'}_i \cdot d_i \cdot 2^{-i}\right], \tag{5}$$

$$y^{'}_{i+1} = k_i\left[y^{'}_i + x^{'}_i \cdot d_i \cdot 2^{-i}\right], \tag{6}$$

onde d_i = ± 1 e k_i é um ganho. O sinal + ou - de d_i vai depender do sinal de z_i, com

$$z_{i+1} = z_i - d_i \cdot \tan^{-1} 2^{-i}, \tag{7}$$

$$d_i = 1 \text{ se } z_i > 0, \tag{8}$$

$$d_i = -1 \text{ se } z_i < 0, \tag{9}$$

onde z_0 é o ângulo de rotação final.

O ganho k_i, introduzido pelas rotações inerentes ao algoritmo, depende do número de iterações; quando $n \to \infty$, seu valor é 0,607253. Este ganho pode ser compensado antes ou depois das iterações.

III. Equalização Adaptativa Trigonométrica

Para analisar a formulação matemática do algoritmo LMS trigonométrico, considera-se o procedimento de "*steepest descent*" de filtragem FIR ótima. Assim, é necessário definir:

- a seqüência de entrada:

$$\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^t, \tag{10}$$

- N números positivos A_k, $k = 0, 1, \ldots, N-1$,
- o vetor de coeficientes do filtro:

$$\mathbf{w}(n) = [w(0), w(1), \ldots, w(N-1)]^t, \tag{11}$$

- e a resposta desejada $d(n)$.

Primeiramente, são escolhidos N números positivos A_k, $k = 0, 1, \ldots, N-1$, para a definição do hipercubo que contém os mínimos da superfície de desempenho de erro. Tal hipercubo terá os vértices $[\pm A_0, \pm A_1, \ldots \pm A_{N-1}]$. Desta maneira, pode-se expressar os coeficientes w_k do filtro como:

$$w_k = A_k \sin\theta_k, k = 0, 1, \ldots, N-1, \tag{12}$$

com $-\pi/2 < \theta < \pi/2$.

Como cada $w_k \in (-A_k, +A_k)$ e os valores A_k são fixos, os coeficientes w_k dependem unicamente dos ângulos θ_k. Desta maneira, a variável a ser efetivamente adaptada é θ_k.

Define-se o filtro ótimo de Wiener $\hat{\mathbf{w}} = [\hat{w}(0), \hat{w}(1), \ldots, \hat{w}(N-1)]^t$ pela minimização da função de erro $J(n) = E[e(n)e^*(n)]$, onde $e(n) = d(n) - \mathbf{w}^t\mathbf{x}(n)$. O erro médio quadrático é função dos coeficientes do filtro **w**, e define a superfície de desempenho de erro de dimensão $N+1$.

Uma busca utilizando o algoritmo "*steepest descent*" é feita dentro do hipercubo no espaço θ para alcançar o $\widehat{\theta}$ ótimo. Utilizando-se procedimentos matemáticos relativamente simples, o vetor gradiente

$$\nabla_\theta J(n) = [\partial J(n)/\partial\theta_0, \partial J(n)/\partial\theta_1, \ldots, \partial J(n)/\partial\theta_{N-1}]^t \tag{13}$$

pode ser escrito como:

$$\nabla_\theta J(n) = -2\Delta(\mathbf{p} - \mathbf{R}\mathbf{w}), \tag{14}$$

onde:

$$\mathbf{w} = [A_0\sin\theta_0, A_1\sin\theta_1, \ldots\ A_{N-1}\sin\theta_{N-1}]^t, \tag{15}$$

$$\mathbf{p} = E[\mathbf{x}(n)d(n)], \tag{16}$$

$$\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^t(n)], \tag{17}$$

e Δ é uma matriz diagonal com o j-ésimo elemento dado por:

$$\delta_{j,j} = A_j\cos\theta_j, j = 0, 1, \ldots, N-1. \tag{18}$$

Assim, a iteração $\theta(i)$ é calculada como:

$$\theta(i+1) = \theta(i) - (\mu/2)\nabla_\theta J(n)|_{\theta=\theta(i)}, \tag{19}$$

onde μ é o passo de adaptação do algoritmo.

Para passar do modo "*steepest descent*" para a forma LMS, é necessário substituir as esperanças na Equação 14 por seus valores instantâneos, a fim de obter uma estimativa do gradiente no instante *n*. Desta forma, chega-se ao algoritmo LMS trigonométrico, cujas equações são dadas a seguir:

$$e(n) = d(n) - \sum_{k=0}^{N-1} A_k \sin\theta_k(n) x(n-k), \qquad (20)$$

$$\theta(n+1) = \theta(n) + \mu\Delta(n)\mathbf{x}(n)e(n). \qquad (21)$$

O algoritmo LMS trigonométrico é especialmente apropriado para a realização por processadores CORDIC.

Para uma realização *pipelined* do LMS trigonométrico, é mais apropriado considerar o LMS atrasado (DLMS) [7], [8]. Neste, os coeficientes no instante *n* são atualizados utilizando uma estimativa anterior do gradiente (por exemplo, um instante $(n-D)$, onde D é um inteiro). Assim, a equação de atualização dos coeficientes do LMS trigonométrico atrasado é dada por:

$$\theta(n+1) = \theta(n) + \mu\Delta(n-D)\mathbf{x}(n-D)e(n-D). \qquad (22)$$

Esta variação do LMS necessita de um valor de passo menor para convergência, mas possibilita a adoção de uma taxa de amostragem muito maior [9].

IV. Filtragem Multisplit

O procedimento multisplit parte do pressuposto de que qualquer seqüência finita pode ser expressa pela soma de suas partes simétricas e anti-simétricas. Portanto, qualquer filtro FIR pode ser implementado conforme mostrado na Figura 2, onde $\mathbf{w}_s$ é a parte simétrica do filtro, enquanto $\mathbf{w}_a$ é a parte anti-simétrica [10].

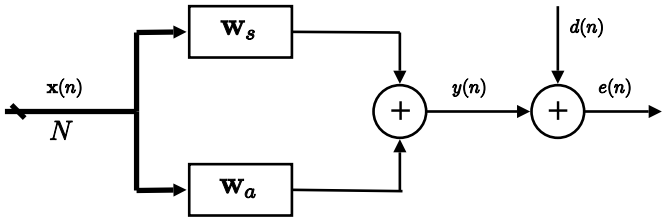

Fig. 2. Filtragem split

Com o vetor de coeficientes do filtro FIR dado por

$$\mathbf{w} = [w_0, w_1, \ldots, w_{N-1}]^t, \qquad (23)$$

pode-se mostrar que [10]:

$$\mathbf{w}_s = (\mathbf{w} + \mathbf{J}_N\mathbf{w})/2 \qquad (24)$$

e

$$\mathbf{w}_a = (\mathbf{w} - \mathbf{J}_N\mathbf{w})/2, \qquad (25)$$

onde $\mathbf{w}_s$ é a parte simétrica da seqüência, $\mathbf{w}_a$ a antisimétrica e $\mathbf{J}_N$ é a matrix de reflexão NxN.

As condições de simetria e anti-simetria das seqüências $\mathbf{w}_s$ e $\mathbf{w}_a$ são expressas, respectivamente, como:

$$\mathbf{w}_s = \mathbf{J}_N\mathbf{w}_s \qquad (26)$$

e

$$\mathbf{w}_a = -\mathbf{J}_N\mathbf{w}_a. \qquad (27)$$

Estas equações podem ser facilmente obtidas a partir das equações 24 e 25.

As condições de simetria e anti-simetria dos filtros $\mathbf{w}_s$ e $\mathbf{w}_a$ podem ser impostas via restrições lineares. Tais restrições podem ser expressas como:

$$\mathbf{C}_s^t\mathbf{w}_s = \mathbf{f}_s \qquad (28)$$

e

$$\mathbf{C}_a^t\mathbf{w}_a = \mathbf{f}_a, \qquad (29)$$

onde $\mathbf{C}_s$ é a matriz de restrição de simetria de dimensão $Nx(N-K)$, definida por:

$$\mathbf{C}_s = \begin{bmatrix} \mathbf{I}_K \\ -\mathbf{J}_K \end{bmatrix}, \qquad (30)$$

onde $\mathbf{I}_k$ é a matriz identidade de tamanho K e $\mathbf{C}_a$ é a matriz de restrição de anti-simetria de dimensão Nx$(N-K)$, que é definida por:

$$\mathbf{C}_a = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{J}_K \end{bmatrix}. \qquad (31)$$

Os vetores de resposta $\mathbf{f}_s$ e $\mathbf{f}_a$, que devem ser nulos a fim de satisfazerem as restrições, têm dimensão Kx1.

Aplicando o GSC (*Generalized Sidelobe Canceller*) [11], é possível transformar este problema de otimização restrito em um irrestrito. Assim, pode-se representar o esquema de filtragem *split* com o esquema da Figura 3.

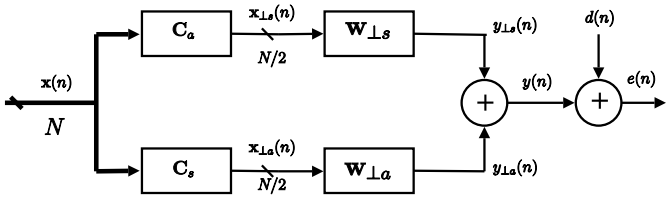

Fig. 3. Filtragem split usando o GSC

Pode-se repetir o processo de *splitting* dos filtros $\mathbf{w}_{\perp s}$ e $\mathbf{w}_{\perp a}$ em suas partes simétricas e anti-simétricas, seguindo os mesmos conceitos anteriores. Depois de L passos, que consistem em 2^{l-1} $(l = 1, 2, \ldots, L)$ operações de *splitting* cada, chega-se a um conjunto de 2^L filtros de ordem zero.

Desta maneira, o equalizador adaptativo trigonométrico também pode ser implementado através de um conjunto de 2^{l-1} filtros paralelos de um único coeficiente, onde a seqüência de entrada de cada filtro é dada por:

$$\mathbf{x}_\perp(n) = \mathbf{T}^t\mathbf{x}(n), \qquad (32)$$

onde

$$\mathbf{T} = \begin{bmatrix} \mathbf{C}_{aL}^t & C_{aL-1}^t & \cdots & \mathbf{C}_{a1}^t \\ \mathbf{C}_{sL}^t & \mathbf{C}_{sL-1}^t & \cdots & \mathbf{C}_{s1}^t \\ \vdots & & & \\ \mathbf{C}_{sL}^t & \mathbf{C}_{sL-1}^t & \cdots & \mathbf{C}_{s1}^t \end{bmatrix}^t_{NxN} \qquad (33)$$

e

$$\mathbf{x}_{\perp}(n) = [x_{\perp 0}(n), x_{\perp 1}(n), \ldots, x_{\perp N-1}(n)]^t. \quad (34)$$

Para $N = 2^L$, $\mathbf{T}$ é uma matriz composta por +1s e -1s, em que o produto interno entre quaisquer duas colunas é zero, ou seja, as colunas de $\mathbf{T}$ são mutuamente ortogonais. Por este motivo, as colunas de $\mathbf{T}$ podem ser permutadas entre si sem afetar as características da matriz. Uma dessas permutações transforma a matriz $\mathbf{T}$ na matriz de Hadamard de ordem N, possibilitando a representação do equalizador adaptativo trigonométrico multisplit, como mostrado na Figura 4.

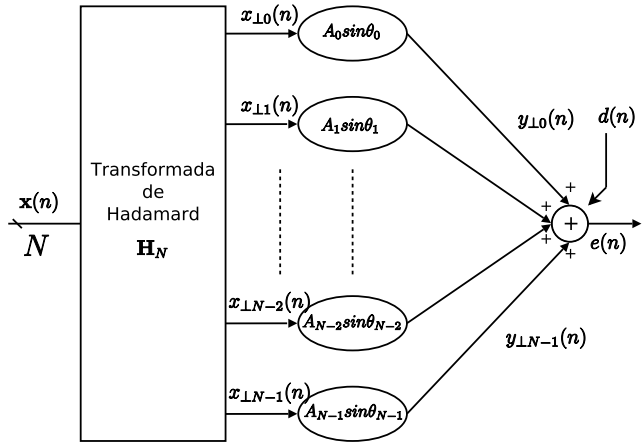

Fig. 4. Equalizador Adaptativo Trigonométrico Multisplit

A transformada de Hadamard é muito utilizada em sistemas de processamento digital de sinais devido à sua fácil implementação e baixo custo computacional. A matriz utilizada pela transformada de Hadamard é composta por ± 1, e tem dimensão de N x N, sendo $N = 2^L$. Uma matriz de Hadamard de ordem N pode ser formada a partir de uma de ordem $N/2$, seguindo o seguinte esquema:

$$\mathbf{H}_{2^L} = \begin{bmatrix} \mathbf{H}_{2^{L-1}} & \mathbf{H}_{2^{L-1}} \\ \mathbf{H}_{2^{L-1}} & -\mathbf{H}_{2^{L-1}} \end{bmatrix}, L = 2, \ldots, \text{ sendo } \mathbf{H}_1 = [1]. \quad (35)$$

A transformada rápida de Hadamard é uma maneira computacionalmente mais eficiente de realizar os cálculos da transformada de Hadamard. Enquanto a transformada normal necessita de N^2 operações de adição e subtração para fornecer o resultado, a transformada rápida requer apenas *LN* adições e subtrações, $N = 2^L$, como pode ser visto na Figura 5. Como menos operações são necessárias, o espaço necessário para implementação da transformada de Hadamard em uma FPGA também é menor.

Um dos efeitos do procedimento multisplit utilizando a transformada de Hadamard é o aumento da potência do sinal transformado. Em várias aplicações, tais como [10] e [12], uma normalização da potência deste sinal é utilizada. Mas para o algoritmo LMS trigonométrico, é possível explorar esse aumento da potência para simplificar o procedimento de definição do hipercubo.

O aumento da potência do sinal de entrada causa a diminuição do valor dos coeficientes do filtro adaptativo. E

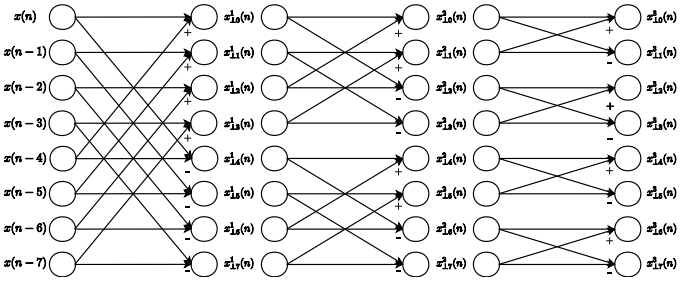

Fig. 5. Transformada Rápida de Hadamard

com o aumento da potência, pode-se considerar apenas o valor $\sin(\theta_k)$ para os coeficientes, fixando assim as coordenadas do hipercubo como unitárias e descartando a necessidade da definição de um hipercubo através de outros valores A_k.

Assim, as equações de filtragem e adaptação dos coeficientes no algoritmo LMS trigonométrico utilizando o procedimento multisplit podem ser reescritas como:

$$\theta(n+1) = \theta(n) + \mu\Delta(n)\mathbf{x}(n)e(n), \quad (36)$$

$$e(n) = d(n) - \sum_{k=0}^{N-1} \sin\theta_k(n)x(n-k), \quad (37)$$

e Δ torna-se uma matriz diagonal com o j-ésimo elemento dado por:

$$\delta_{j,j} = \cos\theta_j, j = 0, 1, \ldots, N-1. \quad (38)$$

V. FPGA VIRTEX-4 E SYSTEM GENERATOR

Nos últimos anos, a tecnologia de dispositivos FPGAs têm evoluído significativamente, alcançando elevados níveis de densidade, altos índices de desempenho e menores custos de fabricação. Esta evolução tem tornado cada vez menor a distância entre FPGAs e CIs (Circuitos Integrados) de uso específico. Além dos avanços em capacidade, desempenho e redução de custos, os fabricantes de FPGAs têm introduzido, no decorrer dos anos, cada vez mais recursos de reconfigurabilidade.

Os recursos de reconfigurabilidade recentemente implantados pelos fabricantes de FPGAs têm possibilitado o projeto de sistemas dinamicamente reconfiguráveis. O termo "dinamicamente reconfigurável" indica a possibilidade de se alterar parcialmente a funcionalidade de um dispositivo enquanto ativo, sem prejudicar o funcionamento de sua lógica restante, que pode estar em operação [13].

A FPGA Virtex-4 produzida pela XILINX, compreende o seguinte conjunto de famílias de FPGAs:

- Virtex-4 LX: aplicações de alto desempenho;
- Virtex-4 SX: aplicações DSP (Digital Signal Processing) de alto desempenho;
- Virtex-4 FX: solução completa e de alto desempenho para aplicações de plataformas embutidas.

Os componentes da Virtex-4 são uma evolução dos componentes já existentes em outras famílias (Virtex, Virtex-E, Virtex-2, Virtex-2 Pro e Virtex-2 Pro X) [14].

A. *System Generator*

O System Generator é uma ferramenta de projeto integrado, em nível de sistema, para FPGAs, que utiliza o Simulink, como suporte de desenvolvimento e é apresentado em forma de uma biblioteca (blockset) [15]. Como ilustrado na Figura 6, o System Generator através de co-simulação gera um arquivo de configuração (*.bit) necessário para a programação da FPGA [15]. O projeto utilizando o System Generator facilita a implementação dos equalizadores em FPGA, pois todos os detalhes de implementação são realizados nesta ferramenta e sem a necessidade de se estar conectado com a placa.

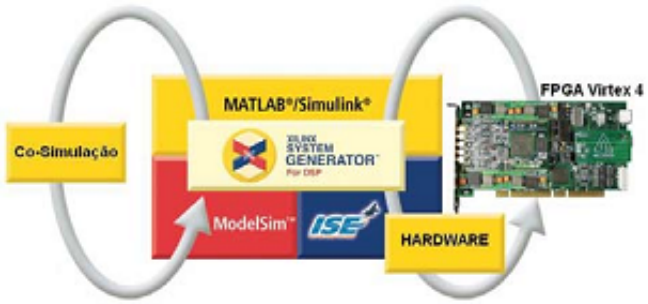

Fig. 6. Fluxo de projeto no System Generator [15]

VI. Resultados

Para as seguintes simulações no System Generator, consideramos que um sinal 4-PAM está sendo transmitido por um canal real. As relações sinal-ruído utilizadas são 20 e 30 dB. Três algoritmos serão comparados em suas formas atrasadas: o LMS convencional, o LMS trigonométrico e o LMS trigonométrico utilizando o procedimento multisplit. Para o algoritmo LMS trigonométrico sem o procedimento multisplit, a busca de valores para o parâmetro A_k foi limitada a valores que são potência de 2. Assim, sua implementação em *hardware* pode ser realizada através de um simples deslocamento, que não consome recursos extras. Os processadores CORDIC utilizados foram implementados na forma paralela. Todos os equalizadores utilizados nas simulações têm oito coeficientes.

Utilizando a ferramenta de estimação de recursos do System Generator, foi realizada uma estimação de área em FPGA para os equalizadores trigonométricos. Os resultados são apresentados na Tabela I.

TABELA I
FPGA Virtex 4 - Estimação de Recursos

Estrutura	Sem Multisplit	Com Multisplit
Slices	10670	12032
Flip-Flops	11630	11686
BRAMs	0	0
LUTs	19091	22427
IOBs	52	60
Bem. Mults.	0	0
TBUFs.	0	0

Como pode-se observar, a adoção do procedimento multisplit não aumenta muito os recursos necessários para o equalizador.

O canal de fase mínima $\mathbf{h}_1 = [0,8316;\ 0,5325;\ 0,4578]$ foi utilizado para a obtenção dos resultados da primeira simulação. O valor de A_k utilizado para o algoritmo LMS trigonométrico atrasado é 1. Os passos de adaptação utilizados para a Simulação 2 são: $\mu_{LMS} = 0,004$, $\mu_{TLMS} = 0,003$ e $\mu_{TLMS-MS} = 0,0005$. 250 curvas de aprendizado foram mediadas para a construção das figuras 7 e 8, que mostram os resultados referentes à Simulação 1.

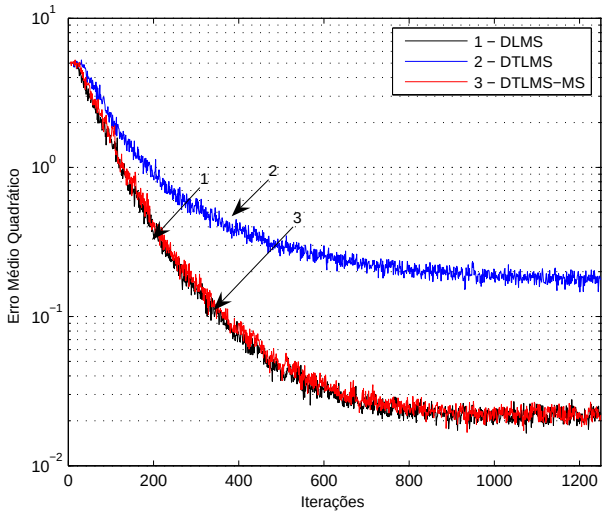

Fig. 7. Curvas de aprendizado da Simulação 1 - SNR 30 dB

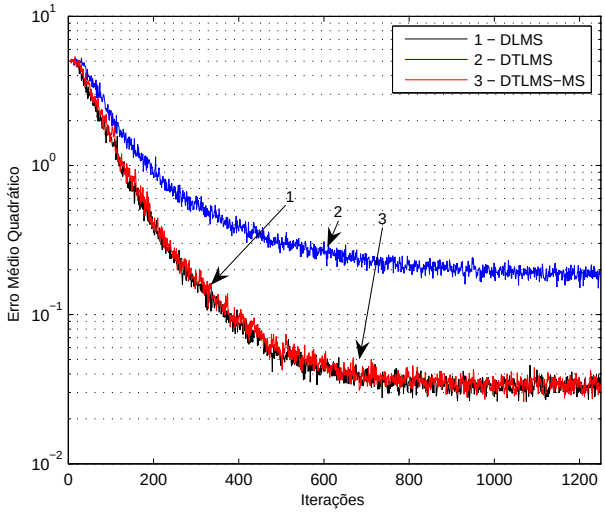

Fig. 8. Curvas de aprendizado da Simulação 1 - SNR 20 dB

Como é possível ver nas figuras 7 e 8, o algoritmo LMS trigonométrico tem o erro médio quadrático na saída muito acima dos outros dois algoritmos, o que indica que o valor adotado para A_k está longe de um valor que permita uma boa convergência. Outros valores de A_k, como 0,5 e 2, fazem com

que a convergência não seja alcançada. Para esta simulação, o desempenho dos outros algoritmos é similar.

Na segunda simulação foi utilizado um canal de fase não-mínima com resposta à amostra unitária dada por $\mathbf{h}_2 = [-0,40;\ 0,85;\ 0,34;\ 0,27]$. Para esta simulação, os passos de adaptação são: $\mu_{LMS} = \mu_{TLMS} = 0,006$ e $\mu_{TLMS-MS} = 0,0008$. O valor de A_k para o algoritmo LMS trigonométrico é 1. As figuras 9 e 10 apresenta as curvas de convergência para a Simulação 2.

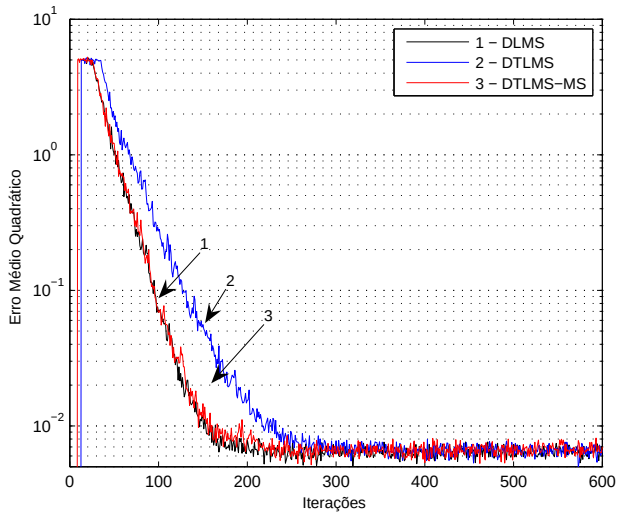

Fig. 9. Curvas de aprendizado da Simulação 2 - SNR 30 dB

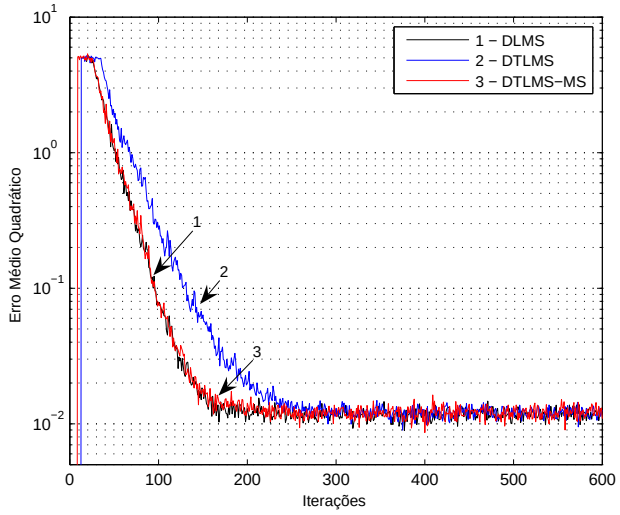

Fig. 10. Curvas de aprendizado da Simulação 2 - SNR 20 dB

Nessa simulação, o valor de A_k adotado permite uma boa convergência, mas os algoritmos LMS e LMS trigonométrico com multisplit convergem mais rapidamente.

VII. Conclusões

Este artigo apresentou um equalizador adaptativo que utiliza o algoritmo LMS trigonométrico para a adaptação dos seus coeficientes. Com os resultados das simulações, indica-se que a adoção do procedimento multisplit faz com que seja desnecessária uma busca exaustiva dos valores das coordenadas do hipercubo, permitindo assim a convergência para vários canais. Para continuar esse trabalho planeja-se a utilização do passo de adaptação variável e a expansão do algoritmo LMS trigonométrico para canais e constelações complexas.

Referências

[1] Jack E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Transactions on Electronic Computers*, v. EC-8, pp. 330–334, Setembro 1959.

[2] Y. H. Hu and H. E. Liao, "CALF: A CORDIC Adaptive Lattice Filter", *IEEE Transactions on Signal Processing*, v. 40-4, pp. 990–993, Abril 1992.

[3] Shin-ichi Shiraishi and Miki Haseyama and Hideo Kitajima, "An Implementation of a Normalized ARMA Lattice Filter with a CORDIC Algorithm", *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, v. 5, pp. 253–256, Maio 1998.

[4] Mrityunjoy Chakraborty and A. S. Dhar anda Moon Ho Lee, "A Trigonometric Formulation of the LMS Algorithm for Realization on Pipelined CORDIC," *IEEE Transactions on Circuits and Systems - II: Express Briefs*, v. 52, pp. 530–534, Setembro 2005.

[5] J. S. Walther, "A Unified Algorithm for Elementary Functions", *Spring Joint Computer Conference*, pp. 379–385, Abril 1971.

[6] Ray Andraka, "A survey of CORDIC algoritms for FPGA based computers", *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, pp. 191–200, Fevereiro 1998.

[7] Guozhu Long and Fuyun Ling and John G. Proakis, "The LMS Algorithm with Delayed Coefficient Adaptation", *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 37-9, pp. 1397–1405, Setembro 1989.

[8] Peter Kabal, "The Stability of Adaptive Minimum Mean Square Error Equalizeres Using Delayed Adjustment", *IEEE Transactions on Communications*, v. 31-3, pp. 430–432, Março 1983.

[9] L. K. Ting and R. Woods and C. F. N. Cowan, "Virtex FPGA Implementation of a Pipelined Adaptive LMS Predictor for Electronic Support Measures Receivers", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 13-1, pp. 86–95, Janeiro 2005.

[10] L. S. Resende and J. M. T. Romano and M. Bellanger, "Split Wiener Filtering With Application in Adaptive Systems", *IEEE Transactions on Signal Processing*, v. 52-3, pp. 636–644, Março 2004.

[11] L.J. Griffiths e C.W. Jim, "An alternative approach to linearly constrained adaptive beamforming", *IEEE Transactions on Antennas and Propagation*, v. AP-30, pp. 27–34, Janeiro 1982.

[12] Francisco J. A. de Aquino, Carlos A. F. da Rocha e Leonardo S. Resende, "Equalização de Canal Usando um Algoritmo LMS Largamente Linear Multi Split", *XXV Simpósio Brasileiro de Telecomunicações - SBrT 2007*, Setembro 2007.

[13] Lysaght, P; Dunlop, J., "Dynamic Reconfiguration of Field Programmable Gate Arrays", *Proceedings of the 3rd International Workshop on Field Programmable Logic and Applications*, pp. 82-94, 1993.

[14] XILINX Inc., "Achieving Breakthrough Performance in Virtex-4 FPGAs", *White Paper: Virtex-4 FPGAs*, Maio 2006.

[15] XILINX Inc., "System Generator for DSP: User Guide - Release 9.2.01", Outubro 2007.