Um Critério de Otimização para Implementação de um CORDIC Paralelo e sua Aplicação

Tiago D. Perez, Eduardo R. de Lima e Luís G. P. Meloni

***Resumo*— Neste artigo, propomos uma arquitetura e uma abordagem de otimização para a implementação do algoritmo CORDIC (COrdinate Rotation Digital Computer) em paralelo, com o objetivo de se reduzir a área e a latência de computação, mantendo a degradação da Taxa de Erro de Bits (BER) em valores aceitáveis. Otimizações similares são encontradas na literatura, porém utilizam como métrica o MSE (Mean Square Error). Demonstramos através de exemplos (Demappers, 8-PSK/16-APSK/32-APSK) que esta não é a melhor abordagem, pois não apresenta uma relação entre o MSE e a métrica de avaliação do sistema, que no caso deste trabalho é a BER.**

***Palavras-Chave*— CORDIC paralelo, Hard-Demapper, VHDL, ASIC, FPGA**

***Abstract*— In this work, we propose an architecture and an optimization approach for the CORDIC (COrdinate Rotation Digital Computer) implementation in parallel aim at reducing the area and the computation latency, while preserving the Bit Error Rate (BER) degradation at acceptable levels. Similar optimizations are found in the literature; nevertheless, they are based on MSE (Mean Square Error). We demonstrate, using examples (Demappers, 8-PSK/16-APSK/32-APSK) that this is not the best approach, because it does not present the relation between the MSE and the system evaluation metric, which in our case is the BER.**

***Keywords*— parallel CORDIC, Hard-Demapper, VHDL, ASIC, FPGA**

I. Introdução

Em sistemas de comunicação a informação transmitida é degradada conforme as características do meio de transmissão. Alterações introduzidas pelo canal de transmissão, e/ou pelo *hardware* do receptor, como por exemplo interferência inter-simbólica e defasagem de frequência podem ser mitigados por equalizadores e sincronizadores, respectivamente, na recepção da informação.

Em comunicações digitais, utiliza-se decisores do tipo *hard-demapper* em processos de equalização[8] e sincronismo[9]. Apesar do uso tradicional de *sof-demappers* para a geração de *soft-bits*, pode-se também utilizar os *hard-demappers* como um decisor para a geração dos bits de dados. No entanto, isso leva a degradação do desempenho BERxSNR. A decisão do *hard-demapper* é feita utilizando-se de regiões delimitadas na plano complexo. Decisores que suportam modulações do tipo PSK (do inglês: *Phase-Shift-Keying*), realizam o cálculo do módulo e argumento dos símbolos recebidos. Em aplicações voltadas ao Processamento Digital de Sinais em VLSI, cálculos como estes são recorrentes e apresentam uma significante complexidade computacional. Para viabilizar a realização destes cálculos, utiliza-se algoritmos especializados.

O algoritmo básico CORDIC (COrdinate Rotation DIgital Computer)[1][2] é uma técnica iterativa eficiente para computar rotações vetoriais, e funções elementares (soma, multiplicação, raiz quadrada, funções trigonométricas e logarítmos). Os cálculos deste algoritmo podem ser realizados por uma sequência de somas e deslocamentos de bit, portanto é bastante adequado para implementações em VLSI.

Usualmente o CORDIC utiliza valores de tangente tabelados para o cálculo do ângulo residual, consumindo uma quantidade significativa de memória ROM. Neste artigo, é proposta uma aproximação para os valores de tangente, para que a tabela de valores completa não seja necessária.

A organização deste artigo é descrita a seguir. Na seção II, é descrito o algoritmo básico CORDIC e são dados exemplos de sua utilização. Na seção III, é derivada a aproximação proposta para os valores da tangente e a arquitetura do CORDIC otimizado é detalhada. Na seção IV, é discutida a implementação do CORDIC otimizado no decisor hard-demapper. Na seção V, é descrita a avaliação de desempenho do hard-demapper com o CORDIC otimizado. Os resultados e comparações são discutidos na seção VI. As conclusões são apresentadas na seção VII.

II. Algoritmo CORDIC básico

O algoritmo CORDIC pode realizar operações vetoriais e rotações de vetores em três sistemas de coordenadas: lineares, circulares e hiperbólicas. O algoritmo realiza uma sequência de micro-rotações utilizando apenas operações de somas e deslocamentos de bit, combinadas com operações de pesquisa em tabela. A arquitetura do CORDIC básico é constituída por M células compostas por somadores e deslocadores de bit (*Shift*), conforme apresentado na Fig. 1.

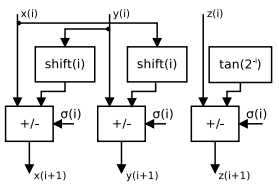

Fig. 1. Célula Básica do Algoritmo CORDIC.

As equações básicas do CORDIC podem ser obtidas a partir de uma rotação vetorial. Dado o vetor $v(x, y)$, sua rotação por um ângulo θ é $v(x', y')$, onde

Tiago D. Perez e Eduardo R. de Lima. Instituto de Pesquisas Eldorado, Campinas-SP, Brasil, Luís G. P. Meloni. Universidade Estadual de Campinas, Campinas-SP, Brasil, E-mails: eduardo.lima@eldorado.org.br, tiago.perez@eldorado.org.br, meloni@decom.fee.unicamp.br.

$$\begin{cases} x' = xcos(\theta) + ysin(\theta) & (1) \\ y' = ycos(\theta) - xsin(\theta) & (2) \end{cases}$$

As equações acima podem ser simplificadas, de tal forma que independam do cálculo de funções trigonométricas. Então, o fator cosseno é omitido e o ângulo θ é otimizado por $\theta \approx \sum_{i=0}^{M} \sigma_i \theta_i$, onde $\sigma_i = \{-1, 1\}$ e $\theta_i = tan^{-1}(2^{-i})$. Portanto, em forma de matriz

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \approx K \prod_{i=0}^{M-1} \begin{bmatrix} 1 & \sigma_i 2^{-i} \\ -\sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3)$$

Onde

$$K = \prod_{i=0}^{M-1} cos(\theta_i) = \prod_{i=0}^{M-1} \frac{1}{\sqrt{1 + 2^{-2i}}} = \prod_{i=0}^{M-1} K_i \quad (4)$$

O produtório da equação (3) pode ser interpretado como $M-1$ micro-rotações, com suas direções decididas por σ_i. A decisão de $\sigma_i \in \{-1, 1\}$ é feita para que o angulo residual z_{i+1} convirja para zero [3]:

$$z_{i+1} \equiv z_i - \sigma_i \theta_i \equiv \theta - \sum_{m=0}^{i} \sigma_m \theta_m \quad (5)$$

Onde $\sigma_i = sinal(z_i)$. A computação final de x' e y' consiste em duas partes. A primeira é a iteração de $M-1$ somas e deslocamentos de bit,

$$\begin{cases} x_{i+1} = x_i - \sigma_i 2^{-i} y_i & (6) \\ y_{i+1} = y_i + \sigma_i 2^{-i} x_i & (7) \\ z_{i+1} = z_i - \sigma_i tan^{-1}(2^{-i}) & (8) \end{cases}$$

Seguido por um fator de compensação de escala:

$$x'_M = Kx_M, \quad y'_M = Ky_M \quad (9)$$

Portanto, o resultado (x'_M, y'_M) são as coordenadas da rotação do vetor $v(x, y)$ por um ângulo θ, após $M-1$ micro-rotações efetuadas iterativamente e compensado por um fator de escala, como é mostrado na Fig. 2. O número de iterações $M-1$ é relacionado com a acurácia dos bits fracionários.

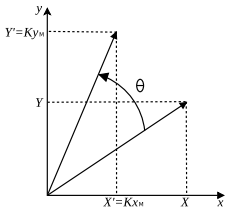

Fig. 2. Rotação Vetorial Por Um Ângulo θ.

As equações básicas do CORDIC podem ser reformuladas de maneira generalizada e unificada, adequando-as para o cálculo de rotações nos sistemas de coordenada circular, hiperbólico e linear. É introduzida uma nova variável m, cujo valor define o sistema de coordenada utilizado. O CORDIC generalizado é formulado como a seguir:

$$\begin{cases} x_{i+1} = x_i - m\sigma_i 2^{-i} y_i & (10) \\ y_{i+1} = y + i + \sigma_i 2^{-i} x_i & (11) \\ z_{i+1} = z_i - \sigma_i \alpha_i & (12) \end{cases}$$

A utilização dos possíveis modos e sistemas de coordenadas é descrita na Tabela I. Os resultados obtidos para cada modo de operação e para cada sistema de coordenada, são resumidos na Tabela II. Para o modo hiperbólico é necessário executar as iterações $i = 4, 13, 40, \ldots, k, k+1, \ldots$ duas vezes para garantir convergência, pois $\sum_{j=i+1}^{\infty} tanh^{-1}(2^{-j}) < tanh^{-1}(2^{-i})$. Em consequência, o fator de escala converge para $K_h = 0.8281$ [4].

TABELA I
MODOS DE OPERAÇÃO DO CORDIC

Modo	**Valor de σ_i**	**Coordenada**	α_i	**m**
Vetorial	sinal(y_i)	Circular	$tan(2^{-i})$	1
Rotação	sinal(z_i)	Hiperbólica	$tanh(2^{-i})$	-1
-	-	linear	2^{-i}	0

Exemplos de algumas das possíveis funções, que podem ser computadas utilizando as equações generalizadas do CORDIC, são mostradas na Tabela III. As três últimas funções são obtidas de modo indireto.

TABELA II
RESULTADOS DAS EQUAÇÕES GENERALIZADAS DO CORDIC

m	**Modo de Rotação**	**Modo Vetorial**
1	$x_M = K[x_0 cos(z_0) - y_0 sin(z_0)]$ $y_M = K[y_0 cos(z_0) + y_0 sin(z_0)]$ $z_M = 0$	$x_M = k\sqrt{x_0^2 + y_0^2}$ $y_M = 0$ $z_M = z_0 + tan^{-1}(y_0/x_0)$
0	$x_M = x_0$ $y_M = y_0 + x_0 z_0$ $z_0 = 0$	$x_M = x_0$ $y_M = 0$ $z_M = z_0 + (y_0/x_0)$
-1	$x_M = K_h[x_0 cosh(z_0) - y_0 sinh(z_0)]$ $y_M = K_h[y_0 cosh(z_o) + y_0 sinh(z_0)]$ $z_M = 0$	$x_M = K_h\sqrt{x_0^2 - y_0^2}$ $y_M = 0$ $z_M = z_0 + tanh^{-1}(y_0/x_0)$

TABELA III
ALGUMAS FUNÇÕES COMPUTADAS ATRAVÉS DO CORDIC

Modo	**m**	$\mathbf{x_0}$	$\mathbf{y_0}$	$\mathbf{z_0}$	$\mathbf{x_M}$	$\mathbf{y_M}$ ou $\mathbf{z_M}$
Rotação	1	K	0	θ	$cos(\theta)$	$y_M = sin(\theta)$
Vetorial	1	1	a	0	$K\sqrt{a^2+1}$	$y_M = tan^{-1}(a)$
Rotação	-1	K_h	0	θ	$cosh(\theta)$	$y_M = sinh(\theta)$
Rotação	-1	a	a	θ	Kae^{θ}	$y_M = Kae^{\theta}$
Vetorial	-1	a	1	0	$K\sqrt{a^2-1}$	$z_M = cot^{-1}(a)$
Vetorial	-1	$a+1$	$a-1$	0	$2K_h\sqrt{a}$	$z_M = 0,5ln(a)$
Vetorial	-1	$a+b$	$a-b$	0	$2K_h\sqrt{ab}$	$z_M = 0,5ln(a/b)$
Vetorial	0	$sen(\theta)$	$cos(\theta)$	0	$sen(\theta)$	$z_M = tan(\theta)$
Vetorial	0	$senh(\theta)$	$cosh(\theta)$	0	$senh(\theta)$	$z_M = tanh(\theta)$
Vetorial	0	$ln(b)$	$ln(a)$	0	$ln(b)$	$z_M = log_b a$

III. CORDIC OTIMIZADO

Operações que possuem o CORDIC em seu ramo de realimentação, são bastantes dependentes do atraso inicial, portanto, necessitam utilizar arquiteturas em paralelo para a implementação do CORDIC. Arquiteturas em paralelo consomem mais área, porém o atraso inicial é reduzido em relação a arquiteturas pipeline [6].

Para compensar o aumento da área da arquitetura em paralelo do CORDIC, pode-se reduzir o consumo de memória ROM utilizada para armazenar os valores de $tan^{-1}(2^{-i})$, utilizados para o cálculo do ângulo residual.

Sabe-se que $tan^{-1}(2^{-i})$ é praticamente linear para $i \geq 3$, e pode ser expressado como [5]:

$$\lim_{i\to\infty} \frac{tan(2^{-i})}{2^{-i}} = 1 \tag{13}$$

Para valores grandes de i:

$$tan^{-1}(2^{-i}) \approx 2^{-i} \tag{14}$$

Pode-se relacionar a aproximação (14) com a acurácia de bits fracionários, para determinar o melhor índice i, tal que o erro de aproximação seja menor que a acurácia. Dada a acurácia de bits fracionários B, tem-se:

$$2^{-m} - tan^{-1}(2^{-m}) < 2^{-B} \tag{15}$$

É mostrado em [5] que $m = (B - log_2 3)/3$ é o menor valor que satisfaz a equação (15).

Dado o íncice m, utiliza-se $t = m - 1$ células básicas e $B - t$ células otimizadas. A arquitetura do CORDIC otimizado é mostrada na Fig. 3.

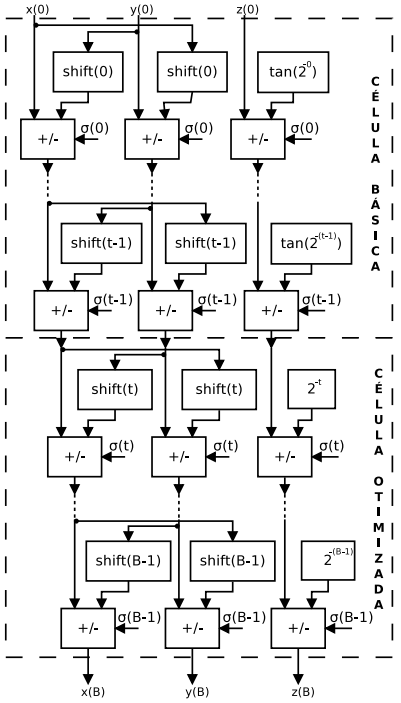

Fig. 3. Arquitetura do CORDIC em Paralelo Otimizado.

IV. Aplicação do CORDIC Otimizado: Hard-Demapper

O CORDIC otimizado é aplicado em um decisor hard-demapper, que comporta as modulações: 8-PSK (do inglês: *Phase Shift Keying*), 16-APSK (do inglês: *Amplitude Phase Shift Keying*) e 32-APSK. As decisões são feitas conforme regiões delimitadas no plano complexo. Para cada modulação, delimita-se uma região de decisão por símbolo. Para as modulações 32-APSK e 16-APSK, as regiões são delimitadas por semicírculos de raios e ângulos diferentes. Para a modulação 8-PSK os semicírculos diferem apenas no ângulo. As regiões de decisão, para o primeiro quadrante, da modulação 16-APSK são mostradas na Fig. 4. As decisões são baseadas no módulo e no argumento dos símbolos de entrada.

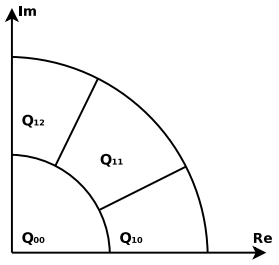

Fig. 4. Regiões de Decisão: 16-APSK.

Utiliza-se o CORDIC para o cálculo do módulo e do ângulo dos símbolos de entrada do *hard-demapper*. Seguindo a Tabela II, utiliza-se o modo vetorial e sistema de coordenada circular ($m = 0$). Dado o símbolo de entrada $s = s_i + js_q$, o CORDIC é alimentado como a seguir:

$$\begin{cases} x_0 = s_i & (16) \\ y_0 = s_q & (17) \\ z_0 = 0 & (18) \end{cases}$$

O resultado obtido é: $|s| = Kx_B$ e $\angle s = z_B$. A arquitetura do *hard-demapper* é mostrada na Fig. 5.

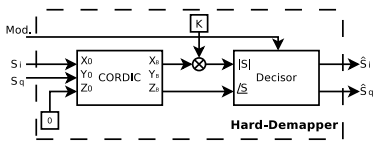

Fig. 5. Arquitetura do Hard-Demapper.

Como parâmetros de projeto, têm-se os valores de t e de B, que são o número do nível de otimização e a acurácia de bits fracionários, respectivamente.

O valor de B é igual à quantidade de bits utilizados para representar a parte fracionária. A quantização utilizada é $Q2.13$ (1 bit de sinalização, 2 bits para representar parte inteira, e 13 para a parte fracionária), logo $B = 13$.

Para a escolha de t, faz-se um estudo do impacto da otimização utilizada no desempenho do *hard-demapper*, descrito na seção seguinte.

V. Avaliação de Desempenho: Hard-Demapper com CORDIC Otimizado

A métrica utilizada para medir o desempenho do *hard-demapper* é a Taxa de Erro de Bit (BER: do inglês *Bit Error Rate*) versus a razão sinal-ruído (SNR: do inglês *Signal-to-Noise Ratio*). Para a modulação 8-PSK, a equação teórica

da probabilidade de erro de bit, em função da SNR e da defasagem $\Delta\theta$, é dada por [10]:

$$\begin{aligned} P_{BER} =& \frac{1}{3}erfc[\sqrt{\gamma_s}sin(A_1)] + \frac{1}{3}erfc[\sqrt{\gamma_s}sin(A_2)] \\ &+\frac{1}{6}erfc[\sqrt{\gamma_s}cos(A_1)]erfc[\sqrt{\gamma_s}sin(A_3)] \\ &+\frac{1}{6}erfc[\sqrt{\gamma_s}cos(A_2)]erfc[-\sqrt{\gamma_s}sin(A_2)] \end{aligned} \quad (19)$$

Onde, γ_s é a SNR por símbolo, $erfc$ é função de erro complementar dado por

$$erfc(x) = 1 - \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \quad (20)$$

E os coeficientes são dados por

$$\begin{cases} A_1 = \dfrac{\pi}{8} - \Delta\theta & (21) \\ A_2 = \dfrac{\pi}{8} + \Delta\theta & (22) \\ A_3 = \Delta\theta - \dfrac{\pi}{8} & (23) \end{cases}$$

A diferença no cálculo do argumento $\angle s$ entre o CORDIC básico e o CORDIC otimizado, traduz-se como uma desafagem $\Delta\theta$. Portando, são feitas simulações númericas do *hard-demapper* trabalhando com a modulação 8-PSK, para os níveis de otimização iguais a $t = \{1, 2, 3, 4, 5\}$, com os dois CORDIC trabalhando em paralelo para a avaliação do erro quadrático médio (mean square error, MSE), equação (24). Utiliza-se uma quantidade de dados igual à $N_d = 5\ 10^5$. Os valores de MSE obtidos são mostrados na Fig. 6.

$$MSE = \frac{1}{N_d}\sum_{j=1}^{N_d}(z_{B,basico}(j) - z_{B,otimizado}(j))^2 \quad (24)$$

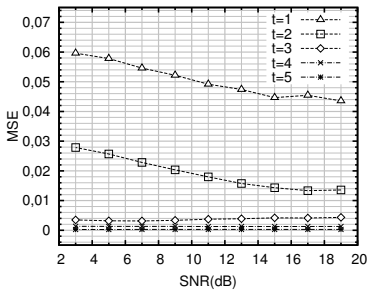

Fig. 6. Erro Quadrático Médio vs SNR, Hard-Demapper: 8-PSK.

De acordo com os resultados obtidos na Fig. 6, é possível traçar as curvas teóricas da probabilidade de erro de bit, equação (19), levando em consideração o MSE de cada nível de otimização t. O resultado é mostrado na Fig. 7.

Note, a otimização só interfere no cálculo do ângulo residual, portanto, o cálculo dos módulos dos símbolos são idênticos para o CORDIC básico e para o CORDIC otimizado. Portanto, o MSE é traduzido em um $\Delta\theta$ no cálculo da BER.

Pela análise da Fig. 7, conclui-se que o nível de otimização que causa menor impacto no desempenho do *hard-demapper*, é $t = 5$.

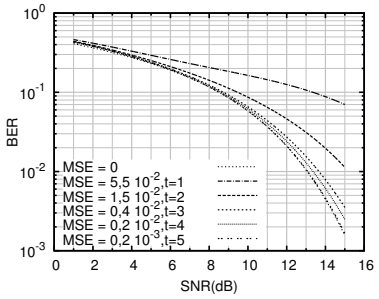

Fig. 7. Curva Téorica da Probabilidade de BER, diferentes $\Delta\theta$: 8-PSK.

TABELA IV
RECURSOS UTILIZADOS EM FPGA: CYCLONE V (ALTERA).

ALUT	Registradores
252	66

VI. RESULTADOS E COMPARAÇÕES

As Fig. 8, 9 e 10, mostram o desempenho do *hard-demapper* para as modulação 8-PSK, 16-APKS e 32-APSK, respectivamente, com valores de otimização $t = \{1, 2, 3, 4, 5, 13\}$. Note que $t = 13$ é o equivalente do CORDIC básico e não é utilizada nenhuma célula otimizada.

Como esperado pela análise teórica, o melhor desempenho, utilizando o CORDIC otimizado, para todas modulações foi atingido quando utilizado $t = 5$.

TABELA V
CONTAGEM DE GATES PARA DIFERENTES ARQUITETURAS

Arquitetura	Gates
Prosposto sem Compensação (Fator *K*)	21998
Proposto com Compensação (Fator *K*)	23024
[5]	7280
[7]	6636
[11]	32928
[12]	8337
[13]	9498

TABELA VI
USO DE MEMÓRIA ROM PARA DIFERENTES ARQUITETURAS

Arquitetura	[1][2]	[5]	[7]	Proposto
t	13	2	4	5
ROM (Q2.13)	208	32	64	80

A arquitetura da Fig. 5 foi implementada em VHDL, e sintetizada utilizando a FPGA Cyclone V da Altera. O resultado do desempenho em comparação com o simulado, é mostrado na Fig. 11. Como esperado, o desempenho é praticamente idêntico. Os recursos utilizados são mostrados na Tabela IV.

A Tabebla V, compara a contagem de Gates com outras arquiteturas CORDIC. As arquiteturas, [5], [12] e [13] são implementadas em pipeline, possuem alta latência e baixo consumo de *hardware*. As arquiteturas [11], [7] e a proposta são implementadas em paralelo, possuem baixa latência e alto consumo de *hardware*, com exceção de [7] devido ao alto nível de otimização.

A Tabela VI, apresenta o nível de otimização utilizada em outras arquiteturas CORDIC e o respectivo consumo

de memória ROM para o armazenamento de valores de $tan^{-1}(2^{-i})$.

Para aplicar o CORDIC no *hard-demapper*, utilizado como máquina de decisão no equalizador *Decision-Feedback* [8], é requerido que ele tenha baixa latência, e com a arquitetura proposta, conseguiu-se uma decisão de símbolo por ciclo de relógio, trabalhando a uma frequência de 62.5MHz. Se comparado com a arquitetura [11], a otimização reduziu significativamente o consumo de *hardware* e manteve a latência baixa. Em termos de recursos utilizados e latência, a melhor arquitetura a se utilizar é a [7], porém o nível de otimização utilizado, $t = 2$, não atinge o desempenho esperado para um *hard-demapper*, como mostram as Fig. 9 e 10.

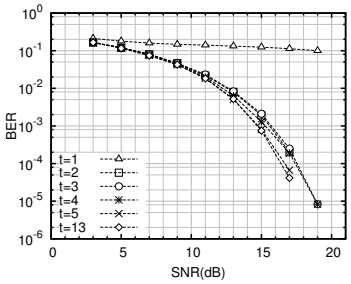

Fig. 8. Comparação de Desempenho do Hard-Demapper: 8-PSK.

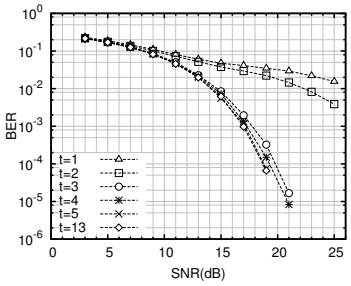

Fig. 9. Comparação de Desempenho do Hard-Demapper: 16-APSK.

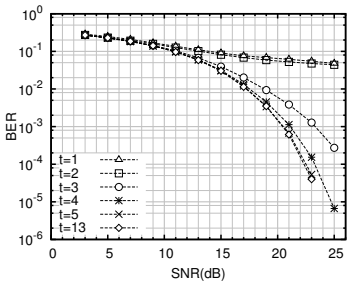

Fig. 10. Comparação de Desempenho do Hard-Demapper: 32-APSK.

VII. Conclusões

A redução de memória ROM foi atingida sem comprometer o desempenho da aplicação, neste caso, o decisor *hard-demapper*. Em comparação com outras arquiteturas, o consumo de ROM poderia ser reduzido ainda mais, porém, o desempenho da aplicação seria comprometido. Portando,

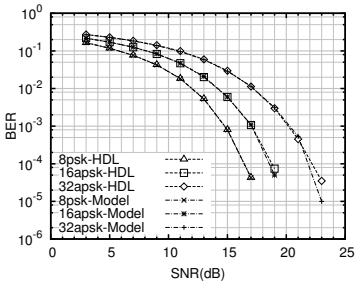

Fig. 11. Comparação de Desempenho do Hard-Demapper: Modelo vs HDL.

para utilizar algoritmos otimizados é necessário o estudo do impacto no desempenho da aplicação, além da análise dos valores de MSE.

Agradecimentos

Os autores agradecem ao Maique Garcia e ao Thiago M. Marson do Instituto de Pesquisas Eldorado pelos resultados de síntese de ASIC.

Referências

[1] J.E. Volder, *"The CORDIC Trigonometric Computing Technique."* IRE Trans. Comput., Vol. EC-8, 1959, pp. 330-334.

[2] J.S. Walther, *A Unified Algorithm for Elementary Functions."* AFIPS Spring Joing Comput. Conf., 1971, pp. 379-385.

[3] J. CHIH, S. CHEN, *Fast CORDIC Algorithm Based on a New Recording Scheme for Rotation Angles and Variable Scale Factors*. Journal of VLSI Signal Processing 33, 19-29, 2003.

[4] P. K. Meher, J. Valls, T. Juang, K. Sridharan, K. Maharatna. *50 Years of CORDIC: Algorithms, Architectures and Applications*. IEEE Transactions on Circuits and Systems-I: Regular Papers.

[5] S. Wang, V. Piuri, and E.E. Swartzlander. *"Hybrid CORDIC algorithms". IEEE Trans. Comput.*, 46(11):1202-1207, 1997.

[6] H. Pahuja, L. Kansal, P. Singh, *CORDIC Algorithm Implementation in FPGA for Computation of Sine & Cosine Signals.* International Journal of Scientific & Engineering Researsh, Vol. 2, Issue 12, December-2011. ISSN 2229-5518

[7] T. Juang, S. Hsiao, M. Tsai. *Para-CORDIC: Parallel CORDIC Rotation Algorithm*. IEEE Transactions on Circuits and Systems-I: Regular Papers, Vol. 51, NO. 8, August 2004.

[8] A. N. Sapper, E. R. de Lima, C. G. Chaves, G. S. da Silva., A. F. R. Queiroz, M. A. C. Fernandes *An Adaptive Equalizer for Reliable Transmissions in DVB-S2 Satellite Communications Under ISI*. IEEE Latin-America Conference On Communications, 2014.

[9] N. Noels, H. Steendam, M. Moeneclaey, H. Bruneel. *Carrier Phase and Frequency Estimation for Pilot-Symbol Assisted Transmission: Bounds and Algorithms.* IEEE Transactions on Signal Processing, Vol. 53, Issue 12, Dec. 2005.

[10] S. Zhang, P. Y. Kam, J. Chen, C. Yu. *Bit-error Rate Performance of Coherent Optical M-ary PSK/QAM using Decision-Aided Maximum Likelihood Phase Estimation.* Optics Express, Vol. 18, No. 12, 24 May 2010.

[11] D. S. Phatak. *Double step branching CORDIC: A new algorithm for fast sine and cosine generation.* IEEE Trans. Computers, vol. 47, pp. 587-607, May 1998.

[12] J. H., J. H. Chois, E. E. Swartzlander, Jr.. *High-speed CORDIC based on an overlapped architecture and a novel σ-prediction method.* J. VLSI Signal Processing, vol. 25, no. 2, pp. 167-178, 2000.

[13] M. Kuhlmann and K. K. Parhi. *P-CORDIC: A precomputation based rotation CORDIC algorithm.* EURASIP J. Appl. Signal Processing, vol. 2002, pp. 936-943, 2002.