# Método Eficiente Para o Cálculo de uma Função Exponencial em FPGA

José M. L. Filho, Karlo G. Lenzi, Erick R. Sousa, Lésnir F. Porto, Luís G. P. Meloni

***Resumo*— O uso de *chips* de FPGA vem crescendo nas últimas duas décadas devido a sua capacidade de reconfiguração pelo usuário e os avanços de ferramentas de prototipagem rápida. Sendo assim, este artigo apresenta o projeto de uma função exponencial de alto-desempenho para um *chip* FPGA (*Field Programmable Gate Array*) da *Xilinx*, utilizando a ferramenta de prototipagem rápida *Xilinx System Generator for DSP*$^{TM}$ e uma placa de desenvolvimento da *Nallatech XtremeDSP Kit-IV* com uma Virtex-4SX. Este trabalho mostra uma forma eficiente de implementação em hardaware da função exponencial visando o uso em sistemas de comunicações.**

***Palavras-Chave*— FPGA, Resolução de Função, Exponencial, System Generator, Prototipagem rápida**

***Abstract*— In the last two decades, FPGA chips have been rising, mostly because of its capacity of reconfiguration and a methodology of rapid prototyping tools. This paper presents the design of a high-performance exponential function in a FPGA *chip* using the System Generator for DSP$^{TM}$ development tool and a Nallatech XtremeDSP Development Kit-IV with a Virtex-4. In this context, the objective is to show an efficient of implementation in hardware for a communication system projects. This paper shows an efficient way of hardaware implementation of exponential function used in communications systems.**

***Keywords*— FPGA, Function Evaluation, Exponential, System Generator, Rapid Prototyping**

## I. Introdução

O uso de *chips* FPGA para projetos de *software-defined radio* tem crescido muito nos últimos anos [1][2]. A capacidade de reconfiguração e de paralelismo destes dispositivos, junto com ferramentas de desenvolvimento de alto-nível para modelagem e prototipagem rápida de sistemas [3][4], faz de sua escolha uma opção interessante comparada a outros dispositivos DSP (*Digital Signal Processing*) baseados em arquiteturas fixas [5].

Dentre os circuitos empregados no projeto destes sistemas de comunicação digital, está o uso de funções matemáticas transcendentais, tais como seno/cosseno, logaritmo, exponencial, arco-tangente, etc. Estas funções desempenham papéis importantes em todo o processo de comunicação como, por exemplo, na geração de sinais de modulação e portadora, na estimação de potência, na compressão e modulação/demodulação de sinais [6][7][8]. Outras áreas de conhecimento que também possuem matemática intensiva podem se beneficiar do uso de FPGAs para aumentar o desempenho de seus sistemas, uma vez que sistemas baseados em microprocessadores dificilmente implementam tais funções em hardware, sendo obrigados a emularem tais funções via software [9][10].

Fatores relevantes no projeto destes circuitos incluem aspectos de: desempenho, consumo de energia e área de silício [10][11][12], onde muitas vezes a melhoria de um destes parâmetros prejudica os demais. Para o projeto de funções matemáticas outro fator também importante que deve ser considerado é a precisão dos resultados. A proposta deste artigo é apresentar um método eficiente de se implementar uma função exponencial de alto-desempenho com precisão de alguns dígitos decimais em uma arquitetura baseada em FPGA através do uso de ferramentas de prototipagem rápida.

A organização deste artigo é a seguinte: na Seção II apresenta-se um breve comentário sobre aproximações de funções matemáticas, descrevendo os métodos existentes e a análise de erro de aproximação; a seção III discute a implementação de uma função exponencial através de aproximação polinomial; a seção IV aborda a metodologia de implementação desta função em hardware; a seção V ilustra o diagrama de blocos da função exponencial implementada; a seção VI apresenta os resultados da implementação, e finalmente, na seção V a conclusão.

## II. Aproximação de Funções Matemáticas

Para facilitar o projeto e implementação de circuitos digitais para resolução de funções matemáticas, busca-se registringir as operações deste circuito a operações de soma, subtração, multiplicação e divisão. Deste modo, é preciso usar métodos de aproximação de funções matemáticas complexas capazes de satisfazerem tais requisitos. Tais métodos para aproximar funções transcendentais através de operações elementares e a forma de se avaliar a precisão destas aproximações são descritos a seguir.

### A. Métodos de Aproximação

Existem quatro métodos de aproximação comumente empregados:

- Método Iterativo: baseia-se em algoritmos de busca, onde o processo de cálculo é repetido até que uma condição de precisão seja satisfeita, sendo, portanto uma abordagem mais complexa de implementação. Por ser um algoritmo de busca, ele não apresenta um desempenho fixo, uma vez que o tempo de execução depende do algoritmo empregado e mesmo das condições de iniciação.
- Método de Tabela (*Look-up Table*): são mais simples e normalmente empregados em aplicações onde a precisão não é um fator limitante. Este método utiliza uma tabela

Laboratório RT-DSP, Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas, Campinas, Brasil, E-mails: matias-filho@gmail.com; lenzi@decom.fee.unicamp.br; erochasousa@yahoo.com.br; porto@decom.fee.unicamp.br; meloni@decom.fee.unicamp.br

de valores previamente calculados da função. Seu objetivo é apenas buscar, de acordo com o valor de entrada, o resultado mais próximo. Sua desvantagem está no fato de que quanto maior a precisão exigida, maior será a tabela de valores, implicando em uma maior quantidade de memória alocada, o que pode ser um fator proibitivo para aplicações embarcadas.

- Método de Aproximação Polinomial: Através de coeficientes, pode-se emular a resolução de funções matemáticas complexas apenas com operações elementares de multiplicação, soma e divisão (para o caso de polinômios racionais). Quanto maior a ordem do polinômio, maior será a sua precisão, entretanto, também maior será o número de operações a serem realizadas.
- Método Híbrido: Combinação dos métodos anteriores.

Dentre os métodos existentes, a aproximação polinomial é a mais utilizada [13]. Também é comum que algumas implementações façam uso de técnicas hibridas, onde normalmente a aproximação polinômial é empregada em conjunto com método de tabelas.

### *B. Erro de Aproximação*

Se $f(x)_{aprox}$ for uma aproximação de $f(x)_{real}$, então o erro relativo é defidido pela equação:

$$E_{relativo} = \left[1 - \frac{f(x)_{aprox}}{f(x)_{real}}\right], f(x)_{real} \neq 0 \qquad (1)$$

Se $f(x)_{aprox}$ aproxima-se de $f(x)_{real}$ até a (p + 1)-ésima casa decimal, dizemos que $f(x)_{aprox}$ é exata até a (p + 1)-ésima casa decimal, ou que a aproximação $f(x)_{aprox}$ tem (p + 1)-ésima casas decimais de precisão [14].

## III. Função Exponencial

Segundo [15] pode-se calcular o valor da exponencial através da relação matemárica:

$$e^x = e^{n\frac{\ln(2)}{2^k}+r} = e^{n\frac{\ln(2)}{2^k}} e^r = 2^{\frac{n}{2^k}} e^r \qquad (2)$$

onde $x$ é o valor de entrada da função, $r$ é o valor residual, $n$ é um valor inteiro obtido pela divisão do valor de entrada $x$ pela expressão $\frac{\ln(2)}{2^k}$ e $k$ é o fator de redução da função. O valor $k$ determinará o grau da redução de domínio, pois limitará o valor de $r$ ao intervalo [0, $\frac{\ln(2)}{2^k}$]. Como $n$ é um inteiro, haverão $k$ diferentes restos para esta divisão. Assim, é comum que a implementação desta função utilize a técnica de tabela para resolver divisões não exatas.

Através de análises no ambiente matemático *Maple Inc.*, Lenzi propôs utilizar $k$ = 6 [16], o que implicará em um polinômio de segunda ordem, descrito pela equação 3.

$$e^r = 1 + (r) + 0.501617376(r)^2 \qquad (3)$$

A partir do esquema de Horner [16] pode-se reescrever a equação 3.

$$e^r = 1 + (1 + 0.501617376r)r \qquad (4)$$

O erro relativo de aproximação deste polinômio é de 2,23x$10^{-8}$ dentro de um intervalo reduzido de [0 ; 0,01] [17].

Sendo $k$ = 6, então temos $2^k$ = 64 entradas, para calcular as divisões não exatas. O resto dessa divisão (de $n$ pela constante $2^k$) representa o índice da tabela que contém os valores fracionários dessa operação. A tabela I apresenta tais valores.

TABELA I
TABELA PARA $k$ = 6.

| ind | tabela(ind) | ind | tabela(ind) |
|---|---|---|---|
| 0 | 1 | 32 | 1,41421356 |
| 1 | 1,01088927 | 33 | 1,42961334 |
| 2 | 1,02189715 | 34 | 1,44518081 |
| 3 | 1,03302488 | 35 | 1,46091779 |
| 4 | 1,04427378 | 36 | 1,47682615 |
| 5 | 1,05564518 | 37 | 1,49290773 |
| 6 | 1,06714040 | 38 | 1,50916443 |
| 7 | 1,07876080 | 39 | 1,52559815 |
| 8 | 1,09050773 | 40 | 1,54221083 |
| 9 | 1,10238258 | 41 | 1,55900440 |
| 10 | 1,11438674 | 42 | 1,57598085 |
| 11 | 1,12652162 | 43 | 1,59314215 |
| 12 | 1,13878863 | 44 | 1,61049033 |
| 13 | 1,15118923 | 45 | 1,62802742 |
| 14 | 1,16372486 | 46 | 1,64575548 |
| 15 | 1,17639699 | 47 | 1,66367658 |
| 16 | 1,18920712 | 48 | 1,68179283 |
| 17 | 1,20215673 | 49 | 1,70010635 |
| 18 | 1,21524736 | 50 | 1,71861930 |
| 19 | 1,22848054 | 51 | 1,73733384 |
| 20 | 1,24185781 | 52 | 1,75625216 |
| 21 | 1,25538076 | 53 | 1,77537649 |
| 22 | 1,26905096 | 54 | 1,79470908 |
| 23 | 1,28287001 | 55 | 1,81425218 |
| 24 | 1,29683955 | 56 | 1,83400809 |
| 25 | 1,31096121 | 57 | 1,85397913 |
| 26 | 1,32523664 | 58 | 1,87416763 |
| 27 | 1,33966752 | 59 | 1,89457598 |
| 28 | 1,35425555 | 60 | 1,91520656 |
| 29 | 1,36900242 | 61 | 1,93606179 |
| 30 | 1,38390988 | 62 | 1,95714412 |
| 31 | 1,39897967 | 63 | 1,97845603 |

## IV. Metologia de Implementação

Neste trabalho utilizou-se a ferramenta *System Generator for DSP*$^{TM}$ (SysGen) da *Xilinx Inc.* para implementar uma função exponencial (equação 2) segundo o método descrito na seção anterior.

A ferramenta de modelagem e simulação visual SysGen é executada juntamente com o *Mathworks Simulink*$^{TM}$ (Simulink) do *Matlab Inc.*, que, juntos, permitem ao projetista de hardware, a partir do esquema modelado do seu projeto, simular um fluxo de dados e, ainda, sintetizar um circuito digital em *chip* FPGA da *Xilinx Inc.* [18]. Este *chip* inclui alguns recursos tais como: SRL (*Shift Register Logic*), MUX (Multiplexadores), LUT (*Lookup Tables*), Memórias, Registradores *Latch*, Lógica Aritmética, IOBs (*Input Output Blocks*) e PIP (*Programmable Interconnect Point*), permitindo, assim, diversas maneiras de roteamento [19]. É importante ressaltar que através dos *chips* FPGAs o projetista de hardware pode, facilmente, reconfigurar a lógica do sistema, diminuindo o tempo de desenvolvimento, sobretudo, nas etapas de teste e depuração.

Foi utilizada a interface PCI para comunicar o usuário em um PC com o kit de desenvolvimento da *Nallatech XtremeDSP Kit-IV*. Além da interface PCI, neste kit da *Nallatech* temos um USB e um JTAG para rotear os recursos de uma Virtex-4 SX. E ainda, o kit apresenta DAC, ADC, clock externo, clock interno, LEDs para o usuário e Memória ZBT SRAM.

## V. Implementação

Através da interligação e configuração de alguns blocos do SysGen e do Simulink, implementou-se um sistema calcula a função exponencial (Figuras 1, 2 e 3). Deve-se lembrar que apenas os blocos do SysGen são sintetizados em *chip* FPGA. A funcionalidade de cada bloco desta implementação é explicado a seguir:

- *System Generator*: este bloco determina características do projeto como o modelo do *chip* utilizado, freqüência de operação do circuito, ferramenta utilizada para síntese, o tipo de linguagem de descrição de hardware, etc.

- *Resource Estimator*: faz uma estimação dos recursos que o sistema vai utilizar de um *chip* FPGA;

- *Gateway In* e os *Gateway Out*: representam, respectivamente, a entrada e a saída do sistema sintetizado em *chip*, realizam também a convergência de ponto flutuante para fixo e vice-versa;

- *AddSub*: fixa uma operação de soma ou subtração dos seus dois dados de entrada;

- *CMult*: implementa uma operação de ganho referente ao seu valor no dado de entrada;

- *Mult*: faz a multiplicação de suas duas entradas;

- *Constant*: gera uma constante para o sistema;

- *Convert*: converte uma amostra de entrada para um outra representação aritmética, com o intuito de diminuir a utilização de recursos em FPGA;

- *Counter*: implementa uma contagem limitada ou ilimitada, podendo ser: crescente, decrescente ou crescente/decrescente;

- *ROM*: implementa uma memória de apenas leitura;

- *Delay*: gera um atraso nos dados (chamado também de *shift register*), sendo este utilizado para adicionar uma latência no projeto, e assim, equiparando os dados no mesmo período de tempo. Uma unidade de tempo será chamado de um *delay*;

- Os blocos *To Workspace* nomeado de ”saída r”, ”saída er”, ”saída table” e ”saída exp sysgem” pertencem ao *Simulink*. Eles geram um vetor de dados referente ao tempo de simulação do sistema que são analisados em ambiente *Matlab*.

A implementação em Sysgen foi dividida em 3 subsistemas, representada pelas seguintes figuras:

Na Figura 1 calcula-se utilizando o método de aproximação polinomial. Para cada dado de entrada no sistema, multiplica-se pela constante $\frac{log(2)}{(2^6)}$, retira-se a aparte fracionária dessa multiplicação e, em seguinda, multiplica-se por outra constante $\frac{(2^6)}{log(2)}$. E ainda, subtrai-se o valor do dado de entrada pelo valor resultante das multiplicações consecutivas. Por fim, passa pelo restante dos blocos que são representados pela equação 4. Este subsistema tem 5 delays.

Na Figura 2 calcula-se utilizando o método tabela. Para cada dado de entrada no sistema existe uma posição de memória relacionada. Insere-se 5 delays nos dados obtidos da memória

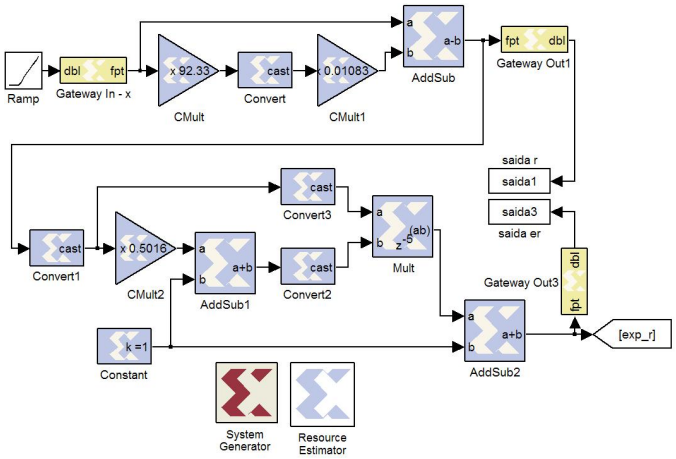


Fig. 1. Subsistema do SysGen utilizando o método polinomial.

para equiparar com o fluxo de dados obtidos do subsistema representado pela figura 2.

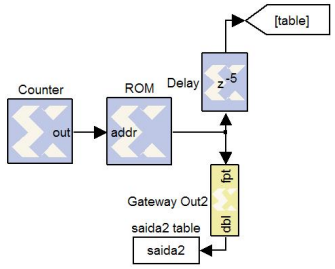


Fig. 2. Subsistema do SysGen utilizando o método tabela.

Na Figura 3, a partir do bloco *Mult*, ocorre a junção dos subsistemas anteriormente citados, acumulando um total de 10 delays. Os dados do ociloscópio, ilustrado pelo bloco *Scope* do Simulink, são analisados na seção VI.

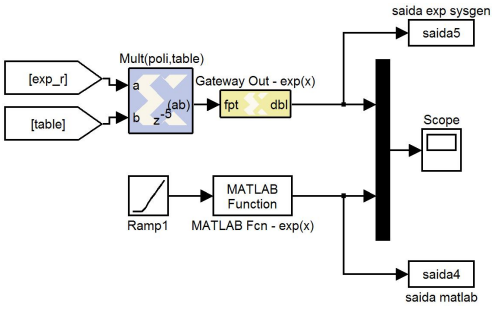


Fig. 3. Subsistema do SysGen que unifica os métodos anteriores.

## VI. Resultados Obtidos

Na Figura 4, temos um gráfico obtido do ociloscópio da Figura 3. Pode-se ver que o sistema tem um tempo de resposta de 10 delays. A curva é o função exponencial gerada pelo bloco *MATLAB Fcn - exp(x)*, e os degraus são gerados pelos blocos do SysGen. À medida que os valores de entrada do

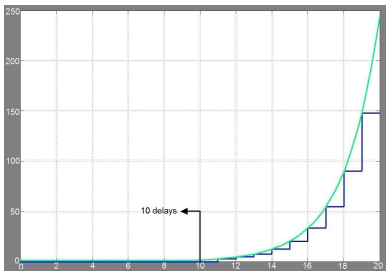


Fig. 4. Sinal gerado da Simulaçao do SysGen.

sistema aumentam, os degraus acompanham a curva exponencial.

Calcula-se o erro relativo da função exponecial do sistema SysGen. Os dados obtidos está no gráfico na Figura 5.

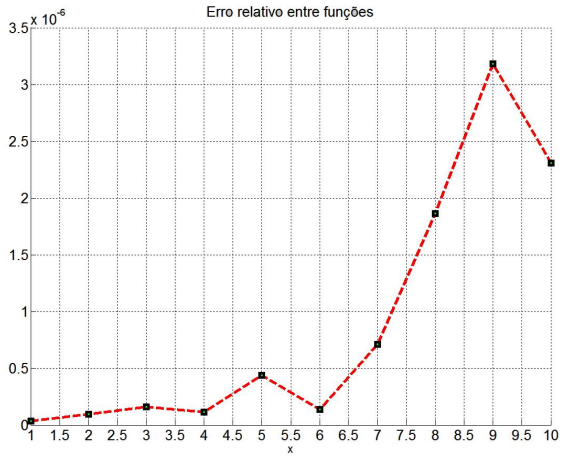


Fig. 5. Erro relativo da Simulação do MatLab.

Quanto maior o valor de $x$, maior será o erro relativo. Em representação de números em ponto flutuante, ao representar um número grande, com uma certa quantidade bits no expoente, o número de bits para representar a mantissa diminui, sendo ela responsável pela precisão de um valor.

Na Figura 6, tem-se a estimação da ocupação dos recursos uma *chip* FPGA da família Virtex-4 da *Xilinx* que tem na placa de desenvolvimento da *Nallatech XtremeDSP Kit-IV*. A porcentagem é a relação do tamanho ocupado pelo tamanho total de recursos, e assim, representando a quantidade total de recursos utilizados no momento do mapeamento. A partir da porcentagem, percebe-se então que foi ocupado poucos recursos em FPGA com esta implementação.

Pode-se diminuir os recursos em *chip* FPGA, em contra partida, perde-se precisão no cálculo da função exponencial em hardware.

## VII. Conclusões

A metodologia proposta neste artigo representa uma maneira eficiente de sintetizar e implementar em FPGA um algoritmo, que utiliza métodos de aproximação de uma função aritmética, afim de serem utilizadas em sistemas de telecomunicações. Demonstrou-se uma implementação em

```
Release 7.1i Map H.38
Xilinx Mapping Report File for Design 'simulacao'

Design Information
------------------
Command Line   : map -o simulacao_map.ncd -intstyle xflow simulacao.ngd
simulacao.pcf
Target Device  : xc4vsx35
Target Package : ff668
Target Speed   : -10
Mapper Version : virtex4 -- $Revision: 1.26.6.3 $
Mapped Date    : Thu Jul 06 01:00:13 2007

Design Summary
--------------
Number of errors:      0
Number of warnings:   26
Logic Utilization:
  Number of Slice Flip Flops:        1,527 out of  30,720    4%
  Number of 4 input LUTs:            1,665 out of  30,720    5%
Logic Distribution:
  Number of occupied Slices:                        1,015 out of  15,360    6%
    Number of Slices containing only related logic:   1,015 out of   1,015  100%
    Number of Slices containing unrelated logic:          0 out of   1,015    0%
      *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:           1,741 out of  30,720    5%
  Number used as logic:              1,665
  Number used as a route-thru:          56
  Number used as Shift registers:       20
  Number of bonded IOBs:               171 out of     448   38%
  Number of BUFG/BUFGCTRLs:              1 out of      32    3%
    Number used as BUFGs:                1
    Number used as BUFGCTRLs:            0
  Number of FIFO16/RAMB16s:              1 out of     192    1%
    Number used as FIFO16s:              0
    Number used as RAMB16s:              1

  Number of RPM macros:            4
Total equivalent gate count for design:  34,454
Additional JTAG gate count for IOBs:  8,208
Peak Memory Usage:  224 MB
```

Fig. 6. Recursos utilizados em FPGA.

hardware da função exponencial através da ferramenta de prototipagem rápida para projetos de DSP *Xilinx System Generator for DSP*$^{\mathrm{TM}}$ em um kit de desenvolvimento da *Nallatech*. Conforme as exigênia de um projeto, o usuário pode adaptar o seu sitema, mesmo em campo, para atender as suas necessidades. Um exemplo de adaptaçao é fazer ajustes nos blocos da implementação do *SysGen* para aumentar ou diminuir a precisão do resultado. Portanto, a utilização dos métodos de aproximação em hardware e as ferramentas de prototipagem rápida são eficientes na construção de sistemas para o processo comunicações.

## Agradecimentos

Agradecemos a FINEP pelo suporte recebido no convênio nº 0.1.06.1179.00 e ao Instituto Eldorado pelo apoio ao Laboratório de Processamento Digital de Sinal de Multimídia em Tempo Real da Unicamp (RT$^{\mathrm{M}}$DSP).

## Referências

[1] Chris Dick, *The Platform FPGA: Enabling the Sofrware Radio*, Xilinx Inc., San Jose, CA, 2002.

[2] Peter B. Kenington, *Software Defined Radio*, ed. 1, 2005.

[3] Miroslav Licko, Jan Schier, Milan Tichy, Markus Kuhl, *MATLAB/Simulink Based Methodology for Rapid-FPGA-Prototyping*, FPL 2003, LNCS 2778, pp. 984987, 2003.

[4] Ana Toledo, Cristina Vicente-Chicote, Juan Suardíaz, Sergio Cuenca, *Xilinx System Generator Based HW Components for Rapid Prototyping of Computer Vision SW/HW Systems*, LNCS 3522, pp. 667674, 2005.

[5] Patrick Lysaght, *Future Design Tools for Platform FPGAs*, Xilinx Research Labs, 2100 Logic Drive, San Jose, Ca., USA, 2005.

[6] Zdenek Pohl, Jan Schier, Miroslav Licko, Antonin Hermanek, Milan Tichy, Rudolf Matousek, Jiri Kadlec, *Logarithmic Arithmetic for Real Data Types and Support for Matlab/Simulink Based Rapid-FPGA-Prototyping*, 2003, IEEE.

[7] Erick R. Sousa, José M. L. Filho, Karlo G. Lenzi, Lésnir Porto, Luís G. Meloni, *Compressão e Expansão de Amplitude de Sinais em Sistemas OFDM*, XXV Simpósio Brasileiro de Telecomunicação- SBrT 2007, Recife, PE.

[8] Chris Dick, Fred Harris, Michael Rice, *FPGA Implementation of Carrier Synchronization for QAM Receivers*, Journal of VLSI Signal Processing 36, 5771, 2004.

[9] Nicolas Brisebarre, Jean-Michel Muller, Arnaud Tisserand, *Computing Machine-Efficient Polynomial Approximations*, ACM Transactions on Mathematical Software, Vol. 32, No. 2, June 2006, Pages 236256.

[10] Dong-U Lee, Altaf Abdul Gaffar, Oskar Mencer, Wayne Luk, *Optimizing Hardware Function Evaluation*, IEEE Trasactions on Computers, Vol. 54, NO. 12, December, 2005.

[11] W. Rhett Davis, *Getting High-Performance Silicon from System-Level Design*, Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI03), 2003.

[12] Juinn-Dar Huang, Jing-Yang, Wen-Zen Shen, *ALTO: An iteractive Area/Performance Tradeoff Algorithm for LUT-Based FPGA Technology Mapping*,IEEE Transactions on Very Large Scale Integration (VLSI) System, VOL. 8, NO. 4, August, 2000.

[13] Jean-Michel Muller, *Elementary Functions, Algorithms and Implementation*, 2nd ed., 2006, XXII, 266 p. 36 illus., Hardcover, ISBN: 0-8176-4372-9.

[14] James Stewart, *Cálculo*, ed. 4, vol. II, 2005.

[15] Ping Tak Peter Tang, *Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic*, Argonne National Laboratory, ACM Transactions on Mathematical Software, Vol. 15, No. 2, June 1989.

[16] C. Sidney Burrus, James W. Fox, Gary A. Sitton, and Sven Treitel, *Horners Method for Evaluating and Deflating Polynomials*, Rice University, November 26, 2003.

[17] Lenzi, K.; Saotome, O. *Optimized Math Functions for a Fixed-Point DSP Architecture*, Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07), IEEE Computer Society, October, 2007, No prelo.

[18] Xilinx Inc., System Generator for DSP$^{\mathrm{TM}}$, *http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=system_generato* [on line], acessado em 02/05/2007.

[19] Uwe Meyer-Baese, *Signal Processing with Field Programmable Gate Arrays: Signals and Communication Technology*, ed. 2, p. 1-10, 2004.