Geração de Ruídos com Padrão Gaussiano Utilizando Sistemas Embarcados

N. Maciel, E. Marques e R. Coelho

***Resumo*— A avaliação da influência de ruídos em sistemas de comunicação é primordial para a obtenção de estimativas da medida BER (*Bit Error Ratio*). Este trabalho apresenta um gerador de ruídos com padrão de amostras Gaussianas baseado no método Box-Muller. A principal contribuição deste trabalho está na regra de escolha das retas para a aproximação das funções do Box-Muller. A implementação do gerador foi realizada utilizando-se a tecnologia FPGA (*Field Programmable Gate Array*) Stratix EP1S25F1020C5 da Altera. A solução proposta acarretou em reduzida utilização de elementos lógicos (5,1%) e de memória RAM (0,1%) dos recursos do FPGA.**

***Palavras-Chave*— Gerador de Ruídos com Padrão Gaussiano, Método Box-Muller, Algoritmos implementados em *Hardware*, FPGA, Aproximação de funções.**

***Abstract*— The evaluation of the influence of noise in communication systems is vital to obtain estimates of the BER (*Bit Error Ratio*) measure. This paper presents a Gaussian noise generator based on the Box-Muller method. The main contribution of this work is the method for choosing lines to approximate the Box-Muller functions. The implementation of the generator was performed using FPGA(*Field Programmable Gate Array*) Altera Stratix board EP1S25F1020C5. The implementation of the proposed solution resulted in a reduced usage of logics elements (5.1 %) and RAM (0.1 %) of FPGA features.**

***Keywords*— Gaussian Noise Generator, Box-Muller Method, Algorithms implemented in Hardware, FPGA, Function approximation.**

I. INTRODUÇÃO

Nos últimos anos, o desenvolvimento de sistemas de comunicação digital com altas taxas de transmissão se tornou uma realidade. Neste cenário, as estimativas de BER devem prover maior acurácia. Assim, geradores de ruídos mais precisos, eficazes e rápidos, são fundamentais para aprimorar a avaliação de desempenho dos sistemas de comunicações.

Recentemente, os avanços em campos programáveis, baseados em FPGA, ganharam muita atenção. Um fator importante é a redução dos custos destas tecnologias embarcadas que vem popularizando o seu emprego. Além disto, estudos comprovam que a geração de amostras em *hardware* pode ser até 50 vezes mais rápida que em *software* [3].

A maioria dos métodos para a geração de amostras Gaussianas envolve transformações em distribuições uniformes para a obtenção do padrão. As funções transformações propostas por Box e Muller [1] definem o principal método, proposto na literatura, para a geração de amostras Gaussianas. Boutillon *et al* [2] propuseram uma arquitetura em *hardware* para esta geração. Esta arquitetura foi desenvolvida pela Xilinx e por outros autores [3].

Este trabalho apresenta um gerador de ruídos Gaussianos baseado no método Box-Muller. O objetivo é a proposta de um método para a escolha das retas de aproximação das funções transformações, o que permite uma maior precisão na geração. Para a geração das amostras uniformes utilizou-se o método Tkacik [4]. O gerador foi implementado numa placa FPGA Stratix EP1S25F1020C5. A avaliação do desempenho do gerador de ruídos considerou amostras de 8, 16 e 32 bits. O gerador implementado alcança uma representatividade de até $6{,}7\sigma$ para amostras de 32 bits.

N. Maciel e E. Marques são bolsistas do Programa PIBITI/CNPq. Os autores são do Instituto Militar de Engenharia (IME), Departamento de Engenharia Elétrica, Laboratório de Comunicações e Sistemas Ópticos (LaRSO), Emails: {nilsonmpj,elainecmarques,coelho}@ime.eb.br.

II. ARQUITETURA DA GERAÇÃO DE RUÍDOS COM PADRÃO GAUSSIANO

O método Box-Muller define que a partir de duas variáveis aleatórias uniformes independentes, u_0 e u_1, no intervalo [0,1) são geradas duas amostras, x_0 e x_1, com distribuição Gaussiana N(0,1), através das funções f, g_0 e g_1; onde:

$$f(u_0) = \sqrt{-2ln(u_0)}$$
$$g_0(u_1) = sen(2\pi u_1) \quad g_1(u_1) = cos(2\pi u_1)$$
$$x_0 = f(u_0).g_0(u_1) \quad x_1 = f(u_0).g_1(u_1)$$

A arquitetura do bloco de geração de ruídos com padrão Gaussiano é apresentada na Fig. 1. A geração foi implementada em 3 etapas. A primeira etapa envolve a geração das amostras uniformes u_0 e u_1 utilizando o método Tkacik [4]. A segunda abrange a implementação eficiente das funções. A terceira etapa utiliza o TLC (Teorema do Limite Central).

A. Método Tkacik

O método Tkacik consiste em dois osciladores independentes com uma geração LFSR (*Linear Feedback Shift Register*) e outra CASR (*Cellular Automata Shift Register*). A saída é composta por uma combinação destas duas gerações. Estes osciladores dão uma maior aleatoriedade e imprevisibilidade. Atualmente, este método é utilizado pela Motorola [4] para a geração de números aleatórios.

B. Proposta para escolha das retas das aproximações das funções transformações

A utilização de métodos computacionais das funções demandaria muito tempo. Já uma LUT (*look-up table*) [3] demandaria requisitos de memória. Portanto, optou-se por um processo baseado numa aproximação linear não uniforme das funções transformações. O processo faz aproximações das funções por retas [2]. O intervalo de aproximação é dividido em um conjunto de subintervalos. Uma LUT é utilizada para

armazenar apenas os coeficientes de cada subintervalo. Com isso, é possível implementar as funções através somente de multiplicadores, somadores e lógicas elementares.

A função f possui grandes curvaturas próximas de 0 e de 1. Caso segmentos uniformes fossem usados, teria-se grandes erros nestas regiões. Assim, um grande número de pequenos segmentos são necessários para aproximar com melhor precisão essas regiões não-lineares. Contudo, a parte central da função é relativamente linear. Isto permite, para uma mesma precisão de aproximação, uma quantidade menor de segmentos. Visando uma maior simplicidade do circuito, optou-se por usar segmentos que cresçam com um fator de dois, de 0 a 0.5, de forma que os segmentos vão se tornando cada vez menores a medida que se aproxima do 0. E segmentos que diminuam com um fator de dois de 0.5 a 1, que faz a melhor aproximação semelhante a anterior para a região próxima de 1. Este último diferencia da proposta em [2]. A diferença do proposto em [3] se encontra no números de retas necessários para passar no testes estatísticos.

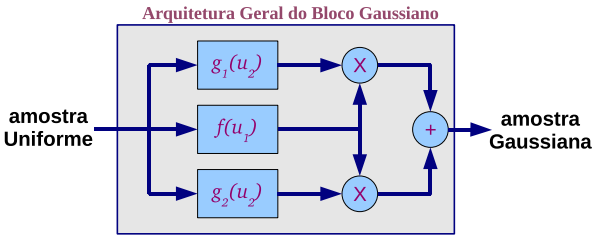

Fig. 1. Arquitetura do método Box-Muller.

Para aproximar as funções g_0 e g_1, aproveitou-se a simetria das mesmas. O eixo x pode ser considerado como quatro regiões relacionadas por simetria. Portanto, a aproximação destas funções é feita no intervalo de entrada [0, 1/4) da função g_1. O domínio é segmentado uniformente e se utiliza regressão linear para a escolha das melhores retas.

III. Implementação do Gerador de Ruídos

A implementação da geração de ruídos Gaussianos foi realizada para valores de u_0 de 8, 16 e 32 bits, sendo denominadas Gauss-08, Gauss-16 e Gauss-32. Enquanto que a quantidade de bits utilizadas para u_1 foi 8, 16 e 16 bits, respectivamente.

O método Tkacik gera amostras uniformes com 32 bits. Para a implementação da Gauss-32, foram utilizados dois blocos Tkacik. As aproximações das funções foram realizadas com as quantidades de retas apresentadas na Tabela I. Foi obtido um erro máximo de 0,0105 para a função f, e 0,0003 para as funções g_0 e g_1. Após a geração, somou-se duas amostras, melhorando o padrão segundo os testes estatísticos, concordando assim, com o TLC.

TABELA I

Quantidade de retas

Implementação	Função f	Função g_0	Função g_1
Gauss-08	14	16	16
Gauss-16	30	32	32
Gauss-32	47	32	32

IV. Resultados

A geração de 1 milhão de amostras foi realizada para cada implementação, cujos histogramas são apresentados na Fig. 2.

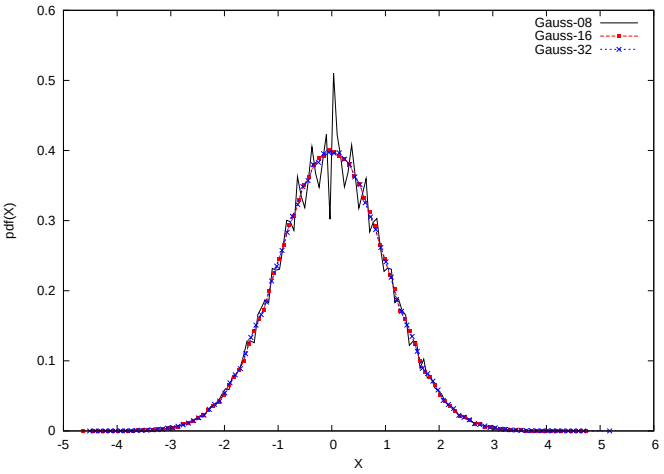

Fig. 2. Histogramas das implementações de 8, 16 e 32 bits para 1 milhão de amostras.

A geração é realizada na taxa de 100 milhões de amostras por segundo, com uma amostra por *clock*. Ainda foram utilizados 16 blocos DSP (*Digital Signal Processing*) de 9 bits para multiplicadores e somadores. As quantidades de elementos lógicos, a máxima representatividade em termos de desvio-padrão alcançado (σ_{max}) e as quantidades de bits de RAM utilizados são mostrados na Tabela II. Os percentuais apresentados são relativos à capacidade em elementos lógicos e bits de RAM do FPGA.

TABELA II

Recursos utilizados do FPGA

Implementação	Elem. Lógicos	σ_{max}	bits de RAM
Gauss-08	306 (1,2%)	$3{,}3\sigma$	448 (0,02%)
Gauss-16	1146 (4,6%)	$4{,}6\sigma$	1664 (0,08%)
Gauss-32	1296 (5,1%)	$6{,}7\sigma$	2024 (0,10%)

V. Conclusões

Neste trabalho foi apresentado um gerador de ruídos com padrão Gaussiano para 3 implementações diferentes, utilizando o método Box-Muller. A arquitetura envolve aproximações por retas das funções deste método. O gerador de ruídos pode alcançar uma representatividade máxima de até $6{,}7\sigma$. Através deste gerador, pode-se obter 100 milhões de amostras por segundo. Geradores de ruídos $1/f$ e fractais podem ser implementados em *hardware* em trabalhos futuros.

Referências

[1] Box G., Muller M., "A Note on the Generation of Random Normal Deviates", *Annals Math. Statistics*, Vol. 29, pp. 610-611, 1958.

[2] Boutillon E., Danger J. L., Ghazel A., "Design of High Speed AWGN Communication Channel Emulator", Vol.55, No.6, *Analog Integrated Circuits and Signal Processin, 34*, 2003.

[3] Lee D., "A Gaussian Noise Generator for Hardware-Based Simulations", Vol.53, No.12, *IEEE Transactions on Computers*, 2004.

[4] Tkacik T. E.,"A Hardware Random Number Generator", *Springer-Verlag Berlin Heidelberg*, 2003.