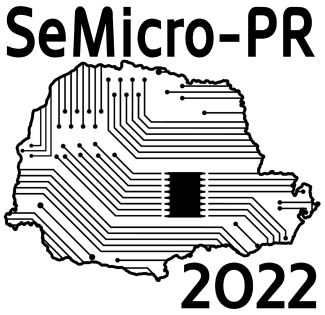

Design de um TRNG em FPGA Baseado em Anéis Osciladores

Luca Werlich Vieira, Leonardo Pelanda Jarek, Sibilla Batista da Luz França
Departamento de Engenharia Elétrica , Curitiba, Brasil
Universidade Federal do Paraná (UFPR)
lucawerlichv@gmail.com, leopelanda4@gmail.com, sibillabl@gmail.com

Resumo – Geradores de Números Aleatórios (Random Number Generators – RNGs) têm aplicações em vários campos. Embora os Geradores de Números Pseudoaleatórios (Pseudo Random Number Generators – PRNGs) sejam suficientes para diversas aplicações, eles são um alvo fácil de potenciais ataques à segurança criptográfica. Os Geradores de Números Verdadeiramente Aleatórios (True Random Number Generators – TRNGs) aparecem como uma possível solução para estes casos. A proposta deste trabalho é implementar um TRNG baseado em anéis osciladores em um Field Programmable Gate Array (FPGA), buscando um gerador de alta velocidade e qualidade, utilizando poucos recursos de hardware. Para isso, diferentes arquiteturas foram estudadas e possíveis modificações apresentadas. Neste trabalho são analisados os impactos no desempenho do TRNG diante de modificações no número de anéis osciladores, número de inversores no anel, bem como utilização de flip-flops para realizar a amostragem em determinados pontos da arquitetura. Diferentes arquiteturas propostas na literatura foram implementadas e comparadas, verificando-se que as arquiteturas de melhor desempenho são aquelas que utilizam um menor número de inversores por anel, assim como, um maior número de anéis osciladores. A validação do gerador, isto é, se a sequência gerada é realmente aleatória, é realizada por meio de testes estatísticos do NIST Randomness Test Suite (National Institute of Standards and Technology).

I. Introdução

A segurança da informação é um requisito fundamental para o funcionamento de diversas atividades. Geradores de números aleatórios exercem uma função importante dentro da criptografia e são usados em diversos protocolos e algoritmos criptográficos [1].

Embora muitas aplicações tanto dentro da criptografia, quanto em outras áreas como jogos eletrônicos, estatística e simulação, tenham suas exigências obedecidas por Geradores de Números Pseudoaleatórios (PRNGs), outras aplicações requerem o nível de segurança que só os Geradores de Números Verdadeiramente Aleatórios (TRNGs) podem conferir. TRNGs são sistemas não-determinísticos que inevitavelmente precisam extrair a aleatoriedade, também chamada de entropia, de algum processo físico.

Dentre as fontes de entropia estudadas pela literatura está o jitter presente em anéis osciladores. Por conta disso, este trabalho tem por objetivo geral implementar um TRNG baseado em anéis osciladores em um FPGA, buscando um gerador de alta velocidade e qualidade, utilizando poucos recursos de hardware.

II. Revisão da Literatura

A. Geradores de Números aleatórios

Os geradores são divididos em dois tipos: geradores de números pseudoaleatórios (PNRGs), e geradores de números verdadeiramente aleatórios (TRNGs). Um PRNG pode ser programado em um software ou implementado em hardware, e possui a característica de sempre gerar a mesma saída para determinada entrada de dados. TRNGs são sistemas não-determinísticos que inevitavelmente precisam obter a aleatoriedade de algum processo físico. É impossível elaborar um programa de computador que se comporte como um TRNG [2]. Por conta disso, de acordo com [4], um TRNG é composto por uma fonte de entropia, um mecanismo de coleta, e, se necessário, um estágio de pós-processamento. Exemplos de fontes de entropia incluem o ruído térmico de transístores, o ruído de diodos zener em polarização reversa e o jitter em circuitos digitais.

B. TRNGs baseados em anéis osciladores

A entropia dos TRNGs baseados em anéis osciladores é proveniente do jitter do seu sinal de onda. O jitter é o desvio de periodicidade de um sinal (que deveria ser

periódico) devido ao ruído térmico ou eletrônico. Anéis osciladores são circuitos digitais compostos por um número ímpar de portas inversoras ligadas em série em forma de anel. De acordo com [4], devido ao feedback, após um estímulo lógico inicial na entrada do primeiro inversor, a saída de qualquer um dos inversores oscilará indeterminadamente entre 0 e 1. Portanto, pode-se obter uma onda quadrada quando se amostra qualquer ponto do anel.

Como os anéis osciladores são implementados fisicamente utilizando componentes de eletrônica digital em FPGA, existe jitter na sua forma de onda. Para extrair o jitter utilizam-se diversos anéis osciladores e aplica-se a função XOR à saída dos anéis. A estrutura básica de um TRNG baseado em anéis osciladores, proposta por [4], pode ser observada na Figura 1. Nessa arquitetura, o jitter presente no sinal dos anéis é a fonte de entropia e a porta XOR, o mecanismo de coleta. Essa arquitetura também necessita de um estágio de pós-processamento e há um flip-flop que amostra o sinal de saída da XOR.

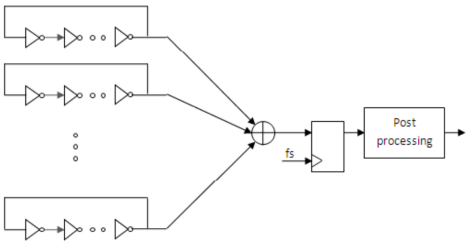

Figura 1 - Arquitetura proposta por [4].

Para o desenvolvimento desse trabalho, foram analisados alguns trabalhos na literatura que estudaram diferentes arquiteturas para TRNGs baseados em anéis osciladores, analisando o impacto de diferentes arranjos, necessidade de flip-flops para a realização da amostragem em diferentes pontos, estágio de pós-processamento, entre outras configurações. Entre os artigos analisados encontra-se a seguinte arquitetura, proposta por [5], conforme a Figura 2. Propõe-se um flip-flop para amostrar a saída de cada um dos anéis osciladores, bem como é empregada uma árvore binária de portas XOR, que evita problemas de sobrecarregamento nas portas XOR. Essas melhorias tornam desnecessária a presença de um estágio de pós-processamento.

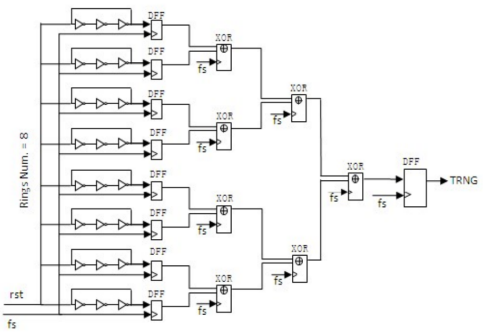

Figura 2 - Arquitetura proposta por [5].

C. *Testes estatísticos para avaliação da aleatoriedade produzida pelos geradores de números aleatórios.*

No âmbito da geração de números aleatórios é necessário realizar uma bateria de testes estatísticos que permitam aceitar ou refutar a aleatoriedade dos dados. Há testes implementados para esse propósito, que funcionam baseados em testes de hipótese. Neste trabalho, a verificação da aleatoriedade dos números gerados pelo TRNG foi realizada com o NIST Test Suite [8], conforme a Tabela 1:

TABELA 1. Testes realizados pelo pacote NIST. Fonte: [8]

Frequency Monobit Test	Teste da proporção de zeros e uns para uma sequência inteira. Quanto mais próxima de 50% a proporção for, mais aleatória a sequência.
Frequency Test within a Block	Teste da proporção de zeros e uns para uma sequência de M bits.
Cumulative Sums	Transforma os bits de valor 0 em -1, e em seguida compara o valor da soma acumulada das sequências parciais com o valor esperado para sequências aleatórias. Engloba dois testes distintos, um percorrendo a sequência do começo ao fim (Forwards), e outro do fim até o começo (Backwards).
Runs Test	Determina se a transição entre zeros e uns é muito rápida ou muito lenta. Isto está associado ao tamanho das sequências ininterruptas de bits idênticos entre cada transição.
Test for the Longest Run of Ones in a Block	Avalia a maior sequência ininterrupta de bits idênticos e se este tamanho é consistente com o esperado para uma sequência aleatória.

III. MATERIAIS E MÉTODOS

A ferramenta de síntese Quartus II foi utilizada para implementar as arquiteturas de TRNG, já que a FPGA utilizada no trabalho foi a Intel Stratix II, de forma a permitir a comparação do desempenho dos geradores com aqueles implementados nos artigos. A linguagem de descrição de hardware utilizada foi o VHDL na qual o atributo keep foi utilizado de modo que a ferramenta de síntese não simplifique a implementação dos vários inversores ligados em série que compõem os anéis.

Foi feita a implementação de 3 arquiteturas propostas por [9]. A Figura 3 apresenta a Implementação 1, escolhida para servir de base para a testagem neste trabalho. A disposição dessa arquitetura é uma variação daquela proposta em [5], cada uma dessas estruturas horizontais é denominada “Nível”, sendo que a

arquitetura apresentada na Figura 3 possui 4 níveis. Essa arquitetura foi escolhida por ter um formato mais simples, já que o foco da pesquisa é estudar os efeitos dos números de inversores e anéis.

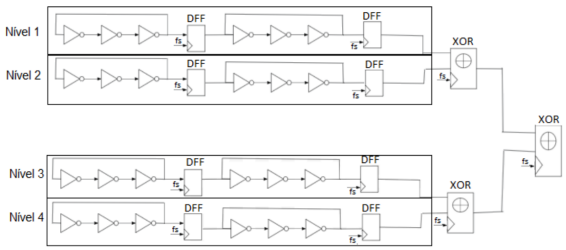

Figura 3 - Implementação 1 com 4 níveis.

IV. Resultados e Discussão

Foram implementadas no software Quartus II diferentes configurações (número de níveis, número de inversores) da Implementação 1. Foram avaliadas configurações do TRNG com 4, 8, 16 ou 32 níveis. Cada nível possui dois anéis osciladores com 3, 5 ou 13 inversores, e dois flip-flops. Cada configuração apresenta uma frequência de clock máxima (Fmax) informada pela ferramenta de síntese.

Para cada configuração foram geradas sequências de bits empregando três frequências de clock distintas (F1, F2, F3) com base na frequência de clock máxima Fmax correspondente. F1, F2 e F3 são menores que Fmax e têm 50 MHz de diferença entre si, exceto para os valores de 333 MHz e 143 MHz, que foram escolhidos pois possuem valores de período inteiros (3 ns e 7 ns, respectivamente). Períodos inteiros facilitam a visualização das formas de onda.

A Tabela 2 sintetiza as combinações de configurações e frequências de clock testadas.

Foram avaliadas todas as 36 combinações relacionadas na Tabela 2 com tamanho de sequência gerada pelo Quartus II igual a 16.000 bits. Neste conjunto foram executados 6 testes de aleatoriedade (T1: Frequency; T2: Block Frequency; T3: Cumulative Sums Forwards; T4: Cumulative Sums Backwards; T5: Runs; T6: Longest Run) realizados pela bateria de testes do NIST. A Tabela 2 relaciona também as combinações que foram aprovadas nesses testes.

Das 36 combinações avaliadas, 9 obtiveram aprovação em todos os 6 testes realizados pelo pacote NIST. Este resultado é demonstrado na Figura 4. Destas 36 combinações, apenas 8 não foram aprovadas simultaneamente nos testes T1 a T4. Este resultado é demonstrado através da Figura 5.

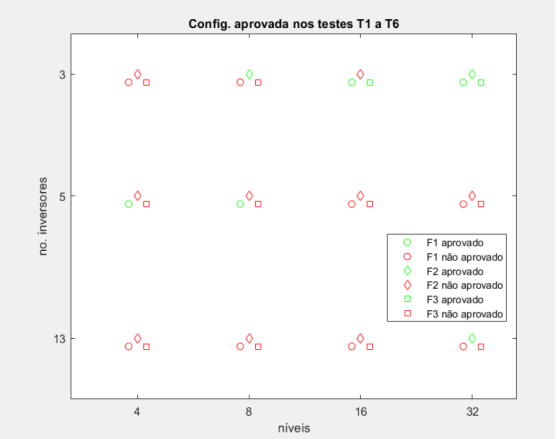

Figura 4 - Configurações aprovadas nos testes T1 a T6

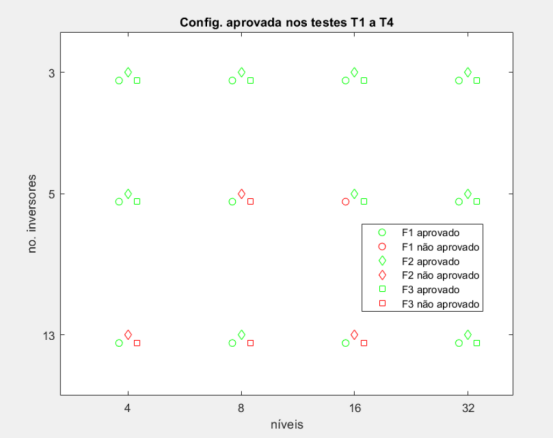

Figura 5 - Configurações aprovadas nos testes T1 a T4

Considerando as 3 frequências em que cada configuração foi avaliada, e os 6 testes realizados, então cada configuração foi submetida a 18 testes. A configuração que obteve aprovação em todos os testes, independente da frequência de clock foi 32 níveis, 3 inversores. Por outro lado, considerando todas as 3 frequências de clock avaliadas, a configuração que obteve aprovação no menor número de testes (8 dentre os 18) foi 4 níveis, 13 inversores. Este resultado vai de acordo com as informações presentes na revisão da literatura de que quanto maior o número de níveis e menor o número de inversores melhor a performance do TRNG.

Cada configuração foi avaliada em 3 frequências de clock distintas. Foi contabilizado o número de aprovações em testes de aleatoriedade que cada configuração obteve em cada uma das frequências. Para cada configuração foi realizada a classificação da frequência de clock que resultou no maior número de aprovações de testes de aleatoriedade. Com isto a frequência F2 foi classificada em 1º lugar, e as frequências F1 e F3 ficaram empatadas em 2º lugar. A configuração 8 níveis, 13 anéis osciladores, obteve aprovação em 4, 4, e 3 testes nas frequências F1, F2, F3, respectivamente. Com isto as frequências F1 e F2 empataram em 1º lugar e a frequência F3 ficou em 2º lugar.

A Tabela 3 sintetiza a quantidade de vezes em que cada frequência foi classificada em 1º, 2º ou 3º lugar quanto à aprovação nos testes de aleatoriedade. Observa-se que, devido à possibilidade de empate na classificação, o somatório de 1º nem de 2º nem de 3º lugar totaliza 12. Pode ser observado que as frequências F1 e F2 obtiveram um desempenho muito equivalente ao passo que a frequência F3 obteve uma performance nitidamente inferior às demais. Isto sugere que, quanto maior a frequência de clock para uma dada configuração melhor a performance quanto à aleatoriedade da sequência gerada. Na literatura avaliada não havia sido encontrada nenhuma relação entre a frequência de clock e desempenho quanto à aleatoriedade da sequência gerada.

Tabela 2. Síntese de combinações de testes realizados

N	I	F [MHz]	T1	T2	T3	T4	T5	T6
4	3	333	✓	✓	✓	✓		
		300	✓	✓	✓	✓		✓
		250	✓	✓	✓	✓		
	5	300	✓	✓	✓	✓	✓	✓
		250	✓	✓	✓	✓		
		200	✓	✓	✓	✓		
	13	143	✓	✓	✓	✓		
		100	✓					
		50	✓		✓	✓		
8	3	333	✓	✓	✓	✓		✓
		300	✓	✓	✓	✓	✓	✓
		250	✓	✓	✓	✓		✓
	5	300	✓	✓	✓	✓	✓	✓
		250	✓		✓	✓		
		200		✓				
	13	143	✓	✓	✓	✓		
		100	✓	✓	✓	✓		
		50	✓	✓	✓	✓		
16	3	333	✓	✓	✓	✓	✓	✓
		300	✓	✓	✓	✓		✓
		250	✓	✓	✓	✓	✓	✓
	5	300	✓		✓	✓		
		250	✓	✓	✓	✓		✓
		200	✓	✓	✓	✓		✓
	13	143	✓	✓	✓	✓		
		100	✓		✓	✓		
		50	✓		✓	✓		
32	3	333	✓	✓	✓	✓	✓	✓
		300	✓	✓	✓	✓	✓	✓
		250	✓	✓	✓	✓	✓	✓
	5	300	✓	✓	✓	✓		
		250	✓	✓	✓	✓		
		200	✓	✓	✓	✓		
	13	143	✓	✓	✓	✓		
		100	✓	✓	✓	✓	✓	✓
		50	✓	✓	✓	✓		

Tabela 3. Contabilização da classificação quanto à aprovação em testes de aleatoriedade

	1^0 mais aprovado	2^0 mais aprovado	3^0 mais aprovado
F1	8	4	0
F2	7	4	1
F3	4	7	1

V. Conclusão

Com relação às arquiteturas de TRNG implementadas e avaliadas através de testes de aleatoriedade foi possível verificar os comportamentos previstos na literatura que especifica que quanto maior o número de anéis osciladores e menor o número de inversores por anel melhor é o desempenho quanto à aleatoriedade da sequência produzida.

Foi possível descobrir também que quanto maior a frequência de clock de operação do TRNG melhor é o desempenho quanto à aleatoriedade da sequência produzida. Destaca-se que, na literatura consultada, não foi encontrada qualquer referência quanto a esta relação.

Referências

[1] Johnston, David Random Number Generators—Principles and Practices: A Guide for Engineers and Programmers, 2018

[2] Pedroni, Volnei A. Circuit Design with VHDL, 2020

[3] Pedroni, Volnei A. Digital electronics and design with VHDL, 2008

[4] Berk Sunar, William J. Martin, Douglas R. Stinson. A provable secure true random number generator with built-in tolerance to active attacks. IEEE Transaction Computers, 56:109-119, 2007.

[5] Xiufeng Xu, Yuyang Wang. High Speed True Random Number Generator Based on FPGA. International Conference on Information Systems Engineering, 2016.

[6] Knut Wold and Chik How Tan. Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings. International Conference on Reconfigurable Computing and FPGAs, 2008.

[7] Pedro A. Morettin, Wilton O. Bussab. – Estatística Básica, 6. ed. – São Paulo : Saraiva, 2010.

[8] Andrew Rukhin et. al, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, 2010.

[9] Lima, E. , França, S. B. L. . Gerador de Números Verdadeiramente Aleatórios Implementado em FPGA. II Seminários de Microeletrônica do Paraná (SeMicro-PR), 2019.