Gerador de Números Verdadeiramente Aleatórios Implementado em FPGA

Estevan Rodrigues de Lima[1], Sibilla Batista da Luz França[1],
[1] Departamento de Engenharia Elétrica, Curitiba, Brasil
estevanlima16@gmail.com

Resumo*—Um gerador de números verdadeiramente aleatórios (True Random Number Generator – TRNG) é um componente essencial em sistemas criptográficos. As sequências randômicas são amplamente usadas como chaves confidenciais, vetores de inicialização e valores de preenchimento. Embora os Geradores de Números Pseudoaleatórios (Pseudo Random Number Generators – PRNGs) sejam suficientes para grande quantidade de aplicações, eles são um alvo fácil de potenciais ataques à segurança criptográfica. Os TRNGs aparecem como uma solução à questão de segurança criptográfica. Neste artigo são apresentadas implementações de TRNGS em Field Programmable Gate Array (FPGA), utilizando-se a abordagem de anéis osciladores, os quais consistem em um número ímpar de portas NOT. Diferentes arquiteturas são apresentadas, através das quais obtém-se maiores velocidades utilizando poucos recursos de hardware quando comparadas a outras implementações descritas na literatura. Uma dessas implementações consiste em 20 anéis osciladores, cada um com 3 portas NOT, operando a uma frequência máxima de 550 MHz. Os resultados passaram em ambos os testes de aleatoriedade NIST (National Institute of Standards and Technology) Test Suite e DIEHARDER Test Suite.

I. INTRODUÇÃO

A segurança da informação tem influenciado substancialmente a comunicação e os sistemas de computação. Números aleatórios exercem uma função importante em criptografia e são usados em praticamente todos os protocolos e algoritmos criptográficos [1]. A demanda crescente por algoritmos e protocolos criptográficos traz junto a necessidade de geradores de números aleatórios de alta velocidade utilizando poucos recursos. Embora os PRNGs satisfaçam uma grande quantidade de aplicações, eles são extremamente vulneráveis a possíveis ataques. Os TRNGs aparecem como uma solução definitiva a essa questão [2]. Neste trabalho são estudados TRNGs baseados em anéis osciladores, apresentando estruturas modificadas, que consistem em um aprimoramento do TRNG baseado em anéis osciladores com árvores binárias de portas XOR, proposto por Xu et al [3]. Neste projeto foi obtido um gerador com frequências maiores que as obtidas por [3]. Um dos fatores responsáveis pela alta frequência obtida na implementação de Xu et al. é a sincronia de processamento entre as portas XOR e o *clock*, evitando problemas de *overload*, isto é, sobrecarga nas portas XOR. Isto exime a necessidade de um estágio de pós-processamento, o qual foi utilizado em implementações mais antigas para conseguir a aprovação nos testes de aleatoriedade, como em [4]. O artigo é apresentado como segue: Na seção 2 é apresentado o TRNG baseado em anéis osciladores, apontando trabalhos relacionados presentes na literatura. Na seção 3 novas estruturas são apresentadas. Na seção 4 são mostrados os resultados obtidos para as abordagens propostas. E finalmente, na seção 5 é apresentada a conclusão.

II. TRNG BASEADO EM ANÉIS OSCILADORES

Há diversas fontes de aleatoriedade de um TRNG, entre elas: anéis osciladores, *Delay-Locked Loop* (DLL) e *Phase-Locked Loop* (PLL). O princípio básico envolvido nestas fontes de aleatoriedade é o *jitter* (atraso). Como não se baseiam em algoritmos determinísticos, são muito mais seguros quando comparados aos PRNGs.

A fonte de aleatoriedade do TRNG é o *jitter* ou atraso temporal criado por cada anel oscilador. Esse atraso temporal pode ser definido como o desvio do sinal periódico em relação ao sinal de *clock* como referência. Normalmente o *jitter* é uma propriedade indesejada em um sistema, mas seu comportamento é útil para a geração de sinais randômicos em um TRNG. O *jitter* tem uma distribuição Gaussiana ao redor de cada transição de *clock* entre estado lógico baixo e estado lógico alto [1]. Um anel oscilador consiste em um número ímpar de portas lógicas NOT, conectados em anel, como apresentado na Figura 1.

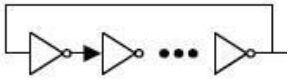

Figura 1. Anel oscilador [5]

Como descrito por Xiufeng Xu et al. [3], há dois possíveis estados para a saída do gerador: um é a saída sem instabilidade 0 ou 1 e o outro é a saída instável quando os

inversores estão trabalhando. Quando ocorre a borda de subida do clock acompanhada do estado instável dos inversores na entrada de dados do flip-flop, a saída será randômica.

Em [4], é proposto um TRNG com 114 anéis osciladores, com cada anel possuindo 13 inversores, como mostra a Figura 2. A frequência de operação do circuito é de 40 MHz.

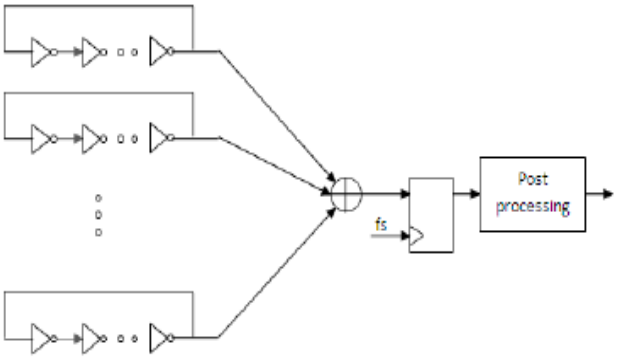

Figura 2. TRNG proposto por [4]

Nesta implementação um estágio de pós-processamento é requerido, pois todas as saídas dos anéis osciladores são conectadas em uma única porta XOR, consequentemente, todos os sinais são assíncronos, forçando a porta XOR a trabalhar em estado sobrecarregado. Em [5], propõe-se um aprimoramento da estrutura apresentada em [4], alcançando uma frequência de operação de 100 MHz. Nesta abordagem, são colocados flip-flops para a amostragem após cada saída de anel oscilador, tornando-a síncrona, não sendo necessário um estágio de pós-processamento. Tal estrutura é apresentada na Figura 3.

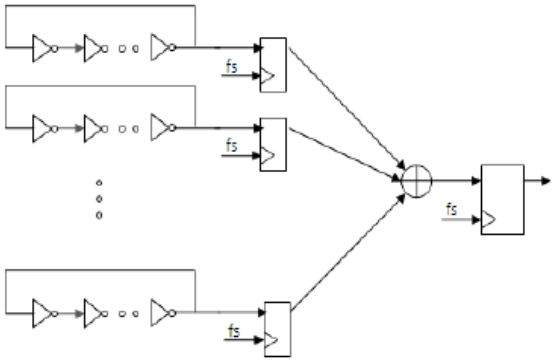

Figura 3. Estrutura aprimorada apresentada por [5]

Um TRNG de maior frequência que a do gerador proposto por [5] é apresentado por Xu et al. [3], o qual alcança uma frequência de operação de 300 MHz utilizando 16 anéis osciladores. A estrutura é descrita na Figura 4.

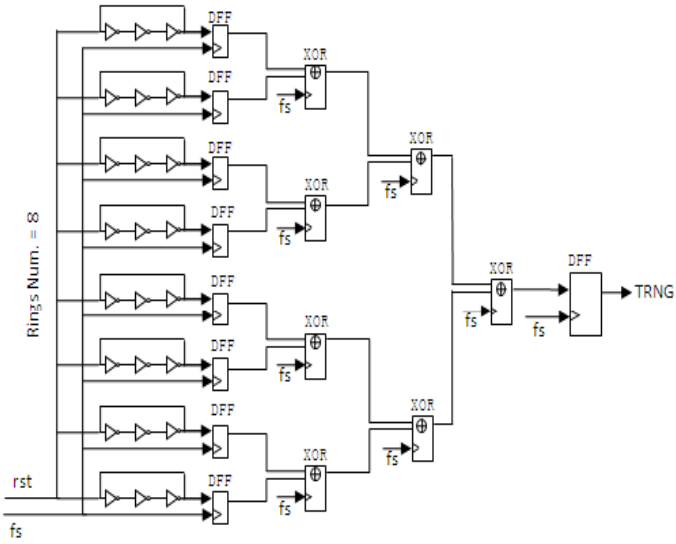

Figura 4. TRNG proposto por [3]

A diferença desta abordagem em relação às anteriores é o fato que cada porta XOR processa apenas dois sinais de entradas em sincronia com o clock, evitando problemas de *overload* e a necessidade de um estágio de pós-processamento.

III. ARQUITETURA PROPOSTA

Buscou-se, então, novos arranjos na arquitetura do TRNG, variando a posição dos anéis osciladores, flip-flops e portas XOR. Foram implementadas sete diferentes arquiteturas, das quais apenas as três melhores serão apresentadas neste artigo, ou seja, aquelas que alcançaram frequências mais elevadas com poucos recursos lógicos.

Novos arranjos nas configurações dos anéis inversores foram analisados, obtendo-se melhores resultados dos que os alcançados por [3]. Um novo arranjo testado é apresentado na Figura 5, ao qual deu-se o nome de Implementação 1. Nele há 12 anéis osciladores, 16 flip-flops D e 7 portas XOR sendo usados. A diferença em relação à abordagem de [3] está na inserção de 4 anéis no estágio 2, conforme apresentado na Figura 5. Há 8 anéis osciladores no primeiro estágio e 4 anéis osciladores no segundo, sendo estes intercalados. A frequência máxima do gerador é de 200 MHz, acima deste valor, a sequência randômica não passa em todos os testes NIST e DIEHARDER. No estágio 1 cada anel oscilador é inicializado com um sinal de reset. Após a inicialização, os anéis inversores trabalham e, portanto, geram uma sequência aleatória. No estágio 2 o sinal de saída do flip-flop do estágio 1 é enviado à entrada do anel oscilador a cada dois ciclos de clock. No ciclo de clock no qual este sinal não é utilizado, a realimentação é o que está em operação, fazendo os anéis do estágio 2 atuarem.

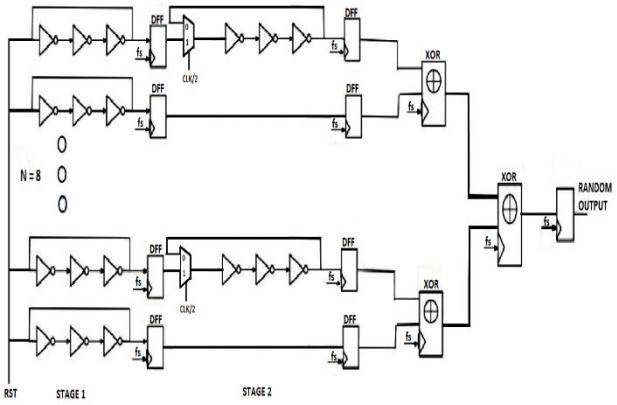

Figura 5. Implementação 1

Em seguida foi testada uma nova implementação, à qual deu-se o nome de Implementação 2, que utiliza menos recursos que a imediatamente anterior, operando a uma frequência máxima de 200 MHz. Esta implementação é apresentada na figura 6. Nela, há uma separação dos anéis osciladores em dois estágios: cada um composto por 4 anéis osciladores. Há a utilização de 8 anéis osciladores, 8 flip-flops D e 3 portas XOR. O funcionamento dos anéis no estágio 2 ocorre do mesmo modo que na Implementação 1. Há redução no número de elementos lógicos em relação à implementação anterior.

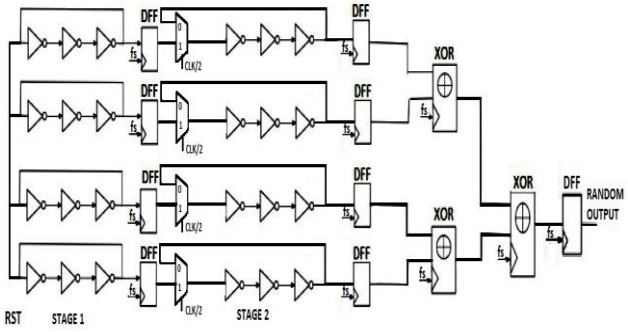

Figura 6. Implementação 2

Por fim, é apresentada na Figura 7 a Implementação 3, a qual opera na frequência mais elevada, 550 MHz, quase duas vezes a frequência apresentada em [3]. Esse valor é limitado pela frequência restrita máxima da FPGA Stratix II.

Nesta abordagem, são inseridos anéis osciladores após a segunda camada de portas XOR. Os anéis osciladores do primeiro estágio são inicializados com um sinal de reset, e os anéis posteriores trabalham como descrito para as implementações 1 e 2. Totalizam-se 20 anéis osciladores nesta abordagem, 20 flip-flops D e 16 portas XOR. São apenas 4 anéis osciladores acrescentados após uma amostragem de flip-flops e duas operações XOR.

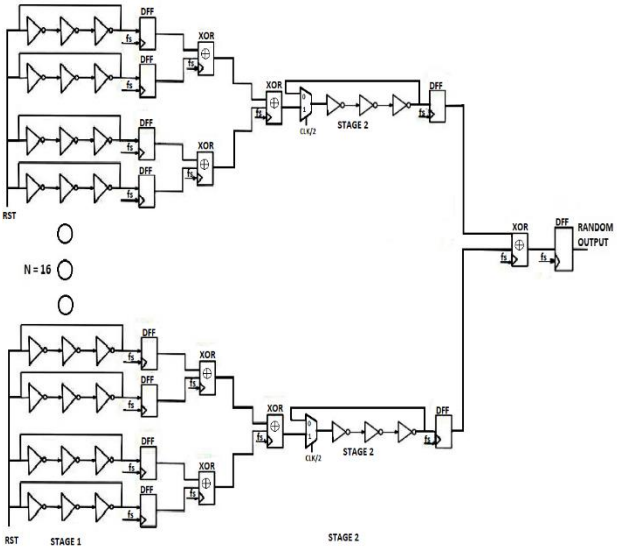

Figura 7. Implementação 3

IV. RESULTADOS

A implementação foi realizada para uma FPGA Altera Stratix II, utilizando a ferramenta Quartus II. Todas as implementações foram validadas em ambos os testes NIST Test Suite e DIEHARDER Test Suite para as frequências máximas de 200 MHz (Implementação 1 e 2) e 550 MHz (Implementação 3). Uma comparação da relação de frequência e recursos das abordagens apresentadas no presente artigo com as implementações de outras publicações é apresentada na Tabela 1. As informações nos espaços demarcados por traço não são fornecidas nas publicações em questão.

Os parâmetros números de anéis osciladores, registradores e LUTs (*Look-up Table)* representam consumo de *hardware* na FPGA.

TABELA 1. COMPARAÇÕES ENTRE IMPLEMENTAÇÕES

Arquitetura	Freq. (Hz)	Número de Anéis Osciladores	Reg.	LUTs
Implementação 1	200 M	12	24	19
Implementação 2	200 M	8	10	9
Implementação 3	550 M	20	34	35
[3]	300 M	16	-	-
[5]	100 M	25	26	57
[6]	160 M	-	-	25
[7]	4 M	10	19	160
[8]	250 M	32	-	-
[9]	100 k	8	-	-

Observa-se que a Implementação 3 atinge a maior frequência de operação, o que o faz o gerador de maior velocidade dentre todos os apresentados. Também se nota que as implementações 1 e 2 utilizam poucos recursos lógicos e apresentam uma alta velocidade comparadas às outras implementações apresentadas na literatura.

Nas Figuras 8, 9 e 10 são apresentados os resultados da simulação na ferramenta Quartus II para as implementações 1, 2 e 3, respectivamente. Nelas são apresentados os sinais de *clock, reset* e a saída randômica.

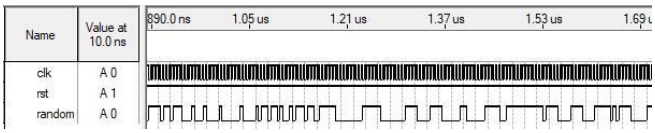

Figura 8. Resultados para a Implementação 1

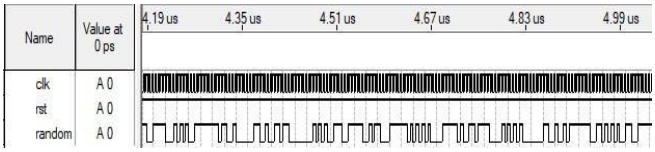

Figura 9. Resultados para a Implementação 2

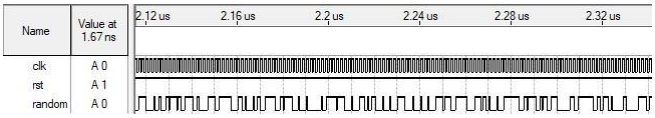

Figura 10. Resultados para a Implementação 3

Pelos resultados gráficos, percebem-se sequências aparentemente aleatórias. Para comprovação da real aleatoriedade, a sequência de bits de saída de cada gerador implementado foi armazenada e validada pelos testes NIST e DIEHARDER.

Ambos os testes NIST e DIEHARDER consistem em uma bateria de testes estatísticos distintos entre si. Cada um destes avalia a aleatoriedade de uma forma particular. Por exemplo, no teste *Frequency Monobit Test*, proposto pelo NIST, é avaliada a frequência de zeros e uns em uma sequência inteira, indicando quão próxima a frequência de uns está de 0,5. Já o teste *Runs Test* indica o número total de "*runs*" na sequência de *bits* analisada, onde cada *run* é uma sequência ininterrupta de bits idênticos. O teste determina se a transição entre zeros e uns é muito rápida ou muito lenta.

Os limitadores para as frequências de operação do TRNG são frequência restrita máxima da FPGA na qual o gerador é implementado e a aprovação nos testes de aleatoriedade.

V. CONCLUSÃO

Foram estudadas várias abordagens de geradores de números aleatórios baseados em anéis osciladores, comparando os resultados experimentais obtidos. Em seguida, estruturas aprimoradas foram propostas, Implementação 1, 2 e 3. Foi demonstrado que é possível a obtenção de um melhor desempenho com poucas modificações no arranjo da estrutura do TRNG, como demonstrado nas Implementação 1 e 2. A Implementação 3 opera em uma frequência duas vezes maior que a abordagem proposta por [3], utilizando apenas 4 anéis osciladores a mais. As arquiteturas propostas alcançam frequências elevadas de operação com pouco consumo de *hardware*. A real aleatoriedade dos TRNGs apresentados é confirmada com a aprovação nos testes de aleatoriedade NIST *Test Suite* e DIEHARDER *Teste Suite.*

REFERÊNCIAS

[1] Sammy H. M. Kwok and Edmund Y. Lam. FPGA-based High-speed True Random Number Generator for Cryptographic Applications. IEEE, 2006.

[2] J.-L. Danger, S. Guilley, P. Hoogvorst. High speed true random number generator based on open loop structures in FPGAs. Microelectronics Journal 40, 2009

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

[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] 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.

[6] Faqiang Mei, Lei Zhang, Chongyan Gu, Yuan Cao, Chenghua Wang and Weiqiang Liu.A Highly Flexible Lightweight and High Speed True Random Number Generator on FPGA. 2018.

[7] Mehmet Alp Şarkışla and Salih Ergün. An Area Efficient True Random Number Generator Based on Modified Ring Oscillators. IEEE, 2018.

[8] Andrei Marghescu, Paul Svasta and Emil Simion. Optimising Ring Oscillator-based True Random Number Generators Concepto on FPGA. 39th Spring Seminar on Electronics Technology (ISSE), 2016.

[9] Dongsheng Liu, Zilong Liu, Lun Li and Xuecheng Zou. A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards. IEEE Transactions on Circuits and Systems – II: Express Briefs, Vol.63, No. 6, June 2016.