SeMicro-PR 2023

Análise e projeto de diferentes arquiteturas de TRNGs baseados em anéis osciladores implementados em FPGA

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

Resumo*—Com o avanço da computação, a segurança em sistemas de informação é crucial. A criptografia depende de números aleatórios, geralmente gerados por True Random Number Generators (TRNGs). Estes exploram fenômenos físicos imprevisíveis como o jitter, para criar sequências aleatórias. Neste estudo, buscamos melhorar a eficiência de um gerador que utiliza anéis osciladores em FPGA para gerar sequências binárias verdadeiramente aleatórias. A arquitetura do gerador proposto utiliza, além da aleatoriedade produzida pelo jitter nos anéis osciladores, a instabilidade gerada ao ligar e desligar a oscilação dos anéis (wake-up e shutdown). Este gerador, implementado em uma FPGA Stratix II, utiliza 319 ALUTs e 32 DFFs, operando a uma frequência máxima de 530 MHz. Todas as sequências geradas foram validadas pelo software NIST.

I. Introdução

Nos sistemas de criptografia, números de natureza aleatória desempenham um papel crucial, atuando como geradores de chaves confidenciais, simétricas e públicas, além de servirem como fontes para senhas. Em diversos algoritmos e protocolos, a aleatoriedade é intrínseca à computação. Em todas essas aplicações, a segurança é vinculada à aleatoriedade da fonte [1]. Sendo assim, a concepção de geradores de números aleatórios é um tópico relevante.

Os geradores são construídos com diversos anéis osciladores controlados de forma a aprimorar a aleatoriedade dos números produzidos. Os anéis osciladores são formados por um número ímpar de portas lógicas inversoras conectadas em sequência, sendo a saída da última inversora conectada à entrada da primeira. O sinal gerado apresenta uma frequência de oscilação aleatória devido ao *jitter* presente nos elementos lógicos. Para esta implementação são utilizados anéis com *wake-up* e *shutdown*, modos de ligar e desligar a oscilação do anel de forma a criar mais instabilidade, incrementando o nível de aleatoriedade da sequência binária gerada.

Os geradores desenvolvidos neste trabalho de pesquisa são descritos utilizando a linguagem de descrição de *hardware* VHDL, implementados em FPGA (*Field Programmable Gate Array*), e sintetizados no *software Quartus II* . E por fim as sequências binárias resultantes são verificadas através do software estatístico NIST, que informa se os números gerados são aleatórios, qualificando o gerador como um TRNG.

II. Geradores de números Verdadeiramente aleatórios

Os Geradores de Números Verdadeiramente Aleatórios (TRNGs) são dependentes de aspectos físicos para extrair a fonte de aleatoriedade. Os fenômenos físicos que trazem esse comportamento são o ruído térmico, ruído de diodos, turbulência em discos rígidos, flutuação em conversores A/D e *jitter* [1]. O *jitter* é a característica física que gera a aleatoriedade nos anéis osciladores. Esse fenômeno pode ser descrito como a variação no tempo de transição dos níveis de tensão em uma porta lógica [2].

Um TRNG baseado em anéis osciladores explora o efeito de variação de tempo (*jitter*) nas portas inversoras para gerar uma oscilação interna em uma frequência imprevisível. Uma FPGA reproduz esse efeito internamente [3]. Na figura 1, é apresentado um modelo simplificado de um anel oscilador.

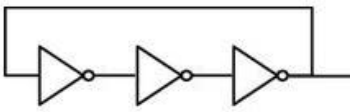

Figura 1. Estrutura básica de um anel oscilador [1].

O modelo proposto por [2] contém múltiplos anéis em paralelo de forma que a oscilação é combinada por portas XOR, e assim a sequência de bits é gerada e registrada.

Em trabalhos posteriores feitos por [3] adiciona um flip-flop do tipo D na saída de cada anel oscilador, desse modo obtém um melhor resultado na saída do gerador, tornando desnecessária a etapa de pós-processamento. Outra vantagem dessa arquitetura é que a sincronização fornecida pelo *clock* nos registradores permite que o gerador trabalhe em uma frequência mais elevada.

A arquitetura de TRNG proposta por [4] utiliza registradores na árvore de XOR, figura 2. Desta maneira, foi possível aumentar a performance do gerador, alcançando a frequência de 100 MHz, sem distorção na qualidade dos números gerados. Nesta arquitetura foram utilizadas três portas inversoras em cada anel.

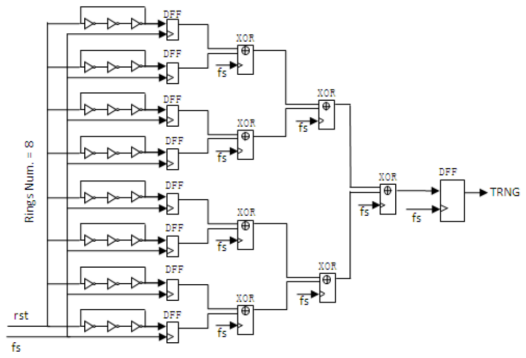

Figura 2. TRNG com árvore de xor com registradores[4].

A arquitetura proposta por [5] traz uma modificação no arranjo dos anéis osciladores de modo a colocá-los em cascata. O autor adiciona um anel oscilador na saída do outro anel, de forma a obter a aleatoriedade de cada anel a cada período do clock.

Em outro estudo, uma modificação na estrutura do anel oscilador é realizada com o objetivo de aumentar a aleatoriedade [6]. O conceito principal apresentado nesta abordagem é ligar e desligar o anel oscilador através da alteração do número de portas inversoras no anel. Quando for um número ímpar o anel oscila e quando for um número par o anel para de oscilar.

O autor [7] utilizou desse princípio para implementar em um gerador de números aleatórios. O autor passa a utilizar a nomenclatura de anéis osciladores com *shutdown* e *wake-up*, para o desligamento e acionamento do anel, respectivamente. Nas figuras 3 e 4 estão as formas de onda que mostram o comportamento do circuito no momento do chaveamento. O sinal em amarelo representa o clock de amostragem, sinal em vermelho a saída do gerador, e sinal em azul a saída de um anel oscilador. Para o *shutdown*, figura 3, o anel registra o valor instável após o desligamento. No caso no *wake-up*, figura 4, o valor registrado é obtido após o anel ser ativado.

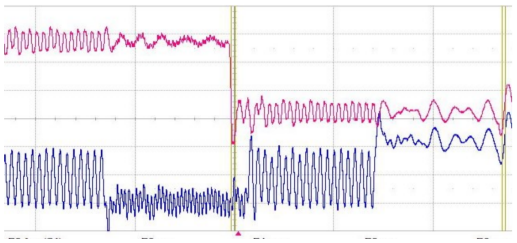

Figura 3. Anéis controlados com *shutdown* [7].

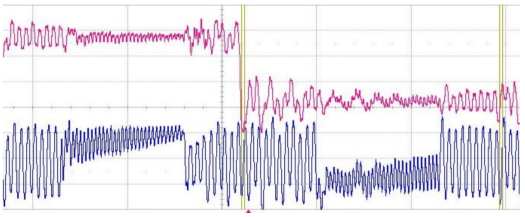

Figura 4. Anéis controlados com *wake-up* [7].

A arquitetura [8] faz uma adaptação dos modelos propostos por [5] e inclui o controle do anel de [7]. O autor introduz anéis osciladores controlados (AOC) no modo de *wake-up*, com 13 unidades inversoras. Esta implementação foi capaz de gerar números aleatórios em 320 MHz.

III. Desenvolvimento e Resultados

As atividades deste trabalho foram desenvolvidas com o objetivo de aplicar os modelos das arquiteturas propostas na literatura, em especial a implementação [7], e a arquitetura simplificada [8]. Após diversas simulações e modificações foi possível, usando o controle de *wake-up* e *shutdown*, obter um gerador capaz de produzir sequências binárias a uma frequência de amostragem de 530 MHz.

A arquitetura utiliza anéis osciladores controlados (AOC) que possuem 17 portas inversoras, o circuito do anel está ilustrado na figura 5. O anel também é construído com um multiplexador para o sinal de *reset*, que é inserido no início da simulação do circuito. Outro multiplexador é adicionado para fazer o controle, quando desativado mantém a oscilação no anel, quando é ativado pelo sinal de controle o circuito passa a utilizar um número par de portas inversoras de modo que a oscilação cessa, nesses momentos de chaveamento é gerada uma instabilidade na oscilação. Ao final do anel é utilizado um flip-flop, que amostra o sinal presente no anel. O *jitter* e a instabilidade na oscilação são as fontes de aleatoriedade do gerador.

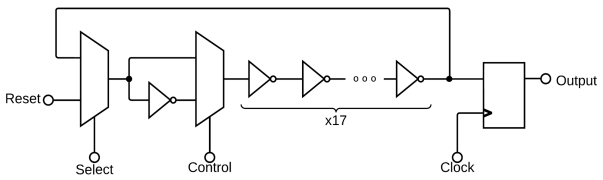

Figura 5. Anel oscilador controlado (AOC) com 17 portas inversoras.

No gerador há 16 anéis, todos têm suas saídas unificadas através de portas XOR síncronas, de modo a reunir os sinais binários de cada anel e obter uma única saída no final, a figura 6 mostra a arquitetura final desse gerador.

Uma modificação proposta neste trabalho, comparando com [8], é no modo de controlar os anéis. Utilizando as duas maneiras *wake-up* e *shutdown*, cada modo de controle é conectado a 8 anéis, metade do gerador para cada modo. O *shutdown* tem o princípio de

desligar a oscilação do anel no momento do evento do *clock*, enquanto o *wake-up* religa no evento do clock.

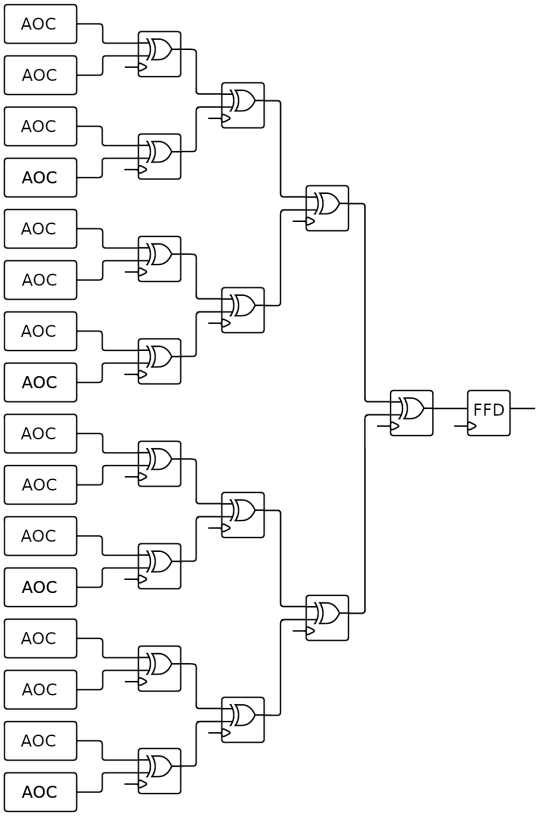

Figura 6. Arquitetura do gerador proposto.

O gerador possui o controle com o modo intercalado, logo o sinal de controle de ambos tem o dobro do período do clock, de modo que em um ciclo seja coletado a instabilidade do *wake-up*, e em outro ciclo de clock é registrado o chaveamento do *shutdown*.

A figura 7 apresenta uma janela de simulação de aproximadamente 230 nanossegundos, sendo possível observar a saída aleatória (sinal *output*).

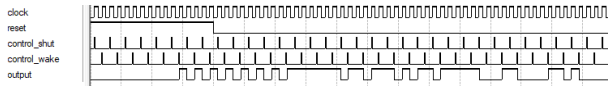

Figura 7. Resultado da simulação do gerador com a saída aleatória.

Para este projeto foram utilizados 319 ALUTs e 32 flip-flops da FPGA Stratix II. O consumo estimado de energia é de 323,63 mW. Comparando com trabalhos anteriores a arquitetura desenvolvida consegue operar em maiores frequências. O gerador de [7] opera na frequência de 4 MHz. A arquitetura mostrada em [8] utiliza 184 ALUTs e é capaz de produzir números aleatórios em 320 MHz. O gerador proposto neste trabalho utiliza 57% mais elementos lógicos que em [8], mesmo assim utiliza apenas 3% do total disponível de ALUTs e menos 1% dos total de flips-flops da FPGA.

Após a simulação, os resultados obtidos, salvos em um arquivo em formato .tbl, são extraídos utilizando um programa em C/C++, com o objetivo de obter apenas os valores da saída aleatória após cada evento do clock. A sequência é analisada pelo NIST SP 800-22 [9], a figura 8 mostra o resultado das análises. Com base no valor P (*p-value*) maior que 0,01 é possível determinar a aleatoriedade da sequência produzida. O arranjo da sequência para ser analisada no NIST foi de 10x10k, respectivamente bitstream e assess, totalizando 100 mil bits. Dado a quantidade de bits utilizada os testes "*universal*", "*random excursion*" e "*random excursion variant*" não são possíveis de serem utilizados, tais testes requerem milhões de bits (informação relatada pelo software NIST), algo que dificulta de produzir, pois aumenta significativamente o tempo de simulação do gerador.

O gerador apresentou sucesso na maioria dos resultados, o que caracteriza-o como um gerador de números verdadeiramente aleatórios.

```
   P-VALUE    PROPORTION     STATISTICAL TEST
-------------------------------------------------------
   0.534146      10/10       Frequency
   0.534146      10/10       BlockFrequency
   0.739918      10/10       CumulativeSums
   0.213309      10/10       CumulativeSums
   0.739918      10/10       Runs
   0.911413      10/10       LongestRun
   0.350485      10/10       Rank
   0.534146      10/10       FFT
   0.739918      10/10       NonOverlappingTemplate
   0.122325       9/10       OverlappingTemplate
   0.000000 *     0/10    *  Universal
   0.066882       9/10       ApproximateEntropy
      ----      ------       RandomExcursions
      ----      ------       RandomExcursionsVariant
   0.350485      10/10       Serial
   0.066882      10/10       Serial
   0.350485      10/10       LinearComplexity
```

Figura 8. Resultado do NIST do gerador proposto.

Outra maneira de visualizar o resultado do gerador, e de compreender a aleatoriedade dos bits gerados, é tratar os valores binários da saída como números decimais, visualizando a dispersão em gráfico. Se as sequências forem de fato randômicas apresentaram dispersão uniforme, do contrário haverá um padrão. A partir de um programa em C/C++ as sequências binárias são convertidas para números decimais a cada 16 bits, e então normalizado de 0 a 1. A figura 9 mostra o gráfico de dispersão de um gerador que falhou, e a figura 10 mostra os números do gerador proposto, ambos os gráficos foram produzidos com 2000 números.

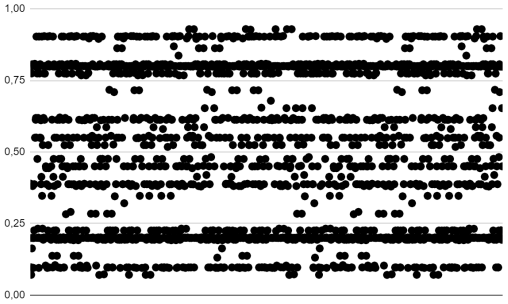

Figura 9. Gráfico de dispersão de um gerador não aleatório.

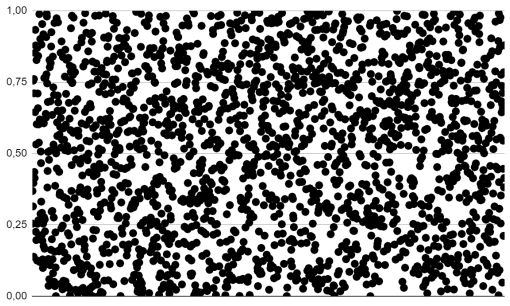

Figura 10. Gráfico de dispersão do gerador aleatório desenvolvido.

IV. Conclusão

O gerador de números verdadeiramente aleatório proposto neste trabalho utiliza o *jitter* e a instabilidade do chaveamento do anel oscilador como fonte de aleatoriedade. O controle do anel é aplicado nos momentos da amostragem de forma a registrar o nível lógico randômico presente no anel oscilador.

A principal modificação apresentada, em relação a [8], é a fusão do método de controle *wake-up* e *shutdown*. A união de tais métodos de controle se mostrou eficiente e possibilitou elevar a frequência de operação. O gerador proposto foi capaz de produzir números aleatórios em um frequência de amostragem de 530 MHz. Comparando com arquiteturas apresentadas na literatura, o gerador obteve um melhor desempenho, operando em frequências mais elevadas.

As sequências binárias produzidas pelo gerador foram validadas pelo *software* estatístico NIST, qualificando o gerador proposto como um TRNG.

Referências

[1] Sammy H. M. Kwok; Edmund Y. Lam. “FPGA-based Highspeed True Random Number Generator for Cryptographic Applications”, 2006.

[2] Xiufeng Xu; Yuyang Wang. “High Speed True Random Number Generator Based on FPGA”, 2016.

[3] Viktor Fischer; Miloˇs Drutarovsky, “True Random Number Generator Embedded in Reconfigurable Hardware”, 2003.

[4] Xiufeng Xu; Yuyang Wang. “High Speed True Random Number Generator Based on FPGA”, 2016.

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

[6] Varchola, M. “New Method of Randomness Extraction Based on a Modified Ring Oscillator for Cryptographic TRNGs Embedded in FPGAs”. 2009.

[7] RM.A. Şarkışla; S. Ergün , “An Area Efficient True Random Number Generator Based on Modified Ring Oscillators” , 2018.

[8] Jarek, L. P.; França, S. B. L. “Estudo e Implementação de um TRNG em Hardware” . V Seminários de Microeletrônica do Paraná (SeMicro-PR). 2022.

[9] NIST Special Publication 800-22, Revision 1 ”A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” 2008.