Geradores de Números Pseudo-Aleatórios de Alta Taxa Baseados no Mapa de Arnold Discreto

Carlos E. C. Souza, Davi Moreno, Daniel P. B. Chaves e Cecilio Pimentel

***Resumo*— Neste trabalho, são propostos dois geradores de números pseudo-aleatórios (PRNG, *pseudo-random number generator*) baseados no mapa de Arnold discreto sobre o anel de inteiros $\mathbb{Z}_{2^m}$. O primeiro PRNG usa operações de ou-exclusivo (XOR) e permutações entre bits, sem requerer multiplicações. O segundo apresenta uma metodologia para aumentar a dimensionalidade das sequências geradas, aumentando a taxa de bits gerados por amostra. As propriedades estatísticas dos PRNGs são analisadas com a suíte estatística NIST. Os PRNGs propostos são implementados em uma FPGA (*field-programmable gate array*) Xilinx Virtex-5. Observa-se que o máximo *throughput* obtido por um dos PRNGs propostos atinge 17 Gb/s, que é 2,4$\times$ maior do que o obtido com esquemas estado da arte.**

***Palavras-Chave*— Geradores de números pseudo-aleatórios, mapas caóticos, caos discreto, FPGA.**

***Abstract*— In this work we propose two pseudo-random number generators (PRNG) based on the discrete Arnold map over the integer ring $\mathbb{Z}_{2^m}$. The first PRNG employ or-exclusive (XOR) and permutation between bits, discarding multiplication operations. The second PRNG proposes a methodology to increase the dimension of the generated sequences, increasing the rate bits/sample. The statistical properties of the PRNGs are analyzed using the statistical suite NIST. The proposed PRNGs are implemented in the FPGA (field-programmable gate array) Xilinx Virtex-5 and their hardware consumption is analyzed. We show that the maximum throughput obtained by one of the proposed PRNGs is 17 Gb/s, which is 2,4$\times$ the throughput obtained by state of the art schemes.**

***Keywords*— Pseudo-random number generator, chaotic maps, discrete chaos, FPGA.**

I. Introdução

Mapas caóticos discretos são mapas iterativos definidos sobre estruturas algébricas finitas tais como aneis de inteiros ou corpos finitos. Inicialmente formalizado em [1], o conceito de caos discreto é definido por um processo de discretização de mapas caóticos reais com uma subsequente periodização das sequências discretas geradas. Outras propostas geram o caos discreto definindo mapas caóticos tradicionais diretamente sobre aneis de inteiros ou corpos finitos [2]–[7]. Neste caso, todas as operações são realizadas em aritmética inteira que replicam exatamente a dinâmica caótica, sem a degradação desta decorrente de operações de arredondamento e truncamento encontradas em mapas reais que podem comprometer a estatística das sequências geradas.

Os autores são do Departamento de Eletrônica e Sistemas da Universidade Federal de Pernambuco, Recife-PE, e-mails: {carlos.ecsouza, davi.moreno, daniel.chaves, cecilio.pimentel}@ufpe.br.

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Código de Financiamento 001.

Alguns projetos de geradores de números pseudo-aleatórios (PRNG, *pseudo-random number generator*) utilizam mapas caótios discretos para geração de sequências binárias pois estas, assim como no caso de mapas reais, possuem tendência a descorrelação após poucas iterações. Em [5] é apresentada uma família PRNGs baseados no mapa de Arnold discreto sobre o anel de inteiros $\mathbb{Z}_{3^m}$. No referido trabalho, os PRNGs propostos utilizam sequências unidimensionais denominadas sequências-z, sendo estas definidas empregando uma multiplicação entre as variáveis x e y do mapa de Arnold discreto.

Um método usual para analisar o consumo de hardware de PRNGs é implementá-los em FPGA (*field-programmable gate array*) para estimar o número de unidades lógicas requeridas para executar todas as operações bem como o *throughput* alcançado pela implementação. Desta forma, em [7] é proposto um PRNG baseado em sequências-z sobre $\mathbb{Z}_{2^m}$ e é feita a implementação em FPGA para os aneis $\mathbb{Z}_{2^m}$ e $\mathbb{Z}_{3^m}$ e mostra-se que o PRNG sobre $\mathbb{Z}_{2^m}$ apresenta um menor consumo de hardware e *throughput* superior. Além disso, é mostrado que o PRNG proposto em [7] possui vantagens em relação a outras arquiteturas de PRNGs propostos na literatura, como por exemplo PRNGs baseados em mapas caóticos reais [8]–[10] e em geradores congruentes lineares [11].

Em [12] é proposto um mapa caótico sobre os reais com dimensão cinco e é apresentado um PRNG com *throughput* de aproximadamente 6,8 Gb/s. Em [13] é proposta uma família de PRNGs baseados em vários atratores caóticos que não requerem operação de multiplicação e atingem uma taxa de transferência de aproximadamente 7,2 Gb/s. Além, disso, devido à ausência de multiplicações, o consumo de hardware é inferior a outras propostas [8]–[12].

Neste trabalho são apresentados dois PRNGs (denominados de PRNG 1 e PRNG 2) baseados no mapa de Arnold discreto sobre o anel de inteiros $\mathbb{Z}_{2^m}$. O primeiro PRNG utiliza uma operação de pós-processamento na representação binária das amostras caóticas geradas pelo mapa baseada nas operações de ou-exclusivo (XOR) e permutações entre bits, sem empregar multiplicações. O segundo PRNG apresenta uma metodologia para aumentar a dimensionalidade das sequências geradas pelo mapa de Arnold discreto definindo um particionamento na representação binária das amostras caóticas com subsequentes multiplicações entre sub-blocos da partição. As propriedades estatísticas das sequências binárias geradas pelos dois PRNGs propostos são analisadas com a suíte estatística NIST versão SP 800-22 [14]. Os PRNGs são implementados em FPGA e o consumo de hardware e *throughput* de ambos são analisados e comparados com a literatura. Por exemplo, o PRNG 1 atinge um *throughput* de 10 Gb/s com complexidade menor que a obtida em [12] e [13]. Um *throughput* superior a 17 Gb/s é

obtido com o PRNG 2 ao custo de maior complexidade em relação ao PRNG 1.

O restante deste trabalho está divivido da seguinte forma. Na Seção II é feita uma breve revisão do mapa de Arnold discreto. O PRNG 1 e suas propriedades estatísticas obtidas com a suíte NIST são detalhados na Seção III. Na Seção IV são detalhados o projeto e os resultados do NIST para o PRNG 2. A implementação em FPGA e comparação com alguns trabalhos recentes da literatura são apresentadas na Seção V. Por fim, na Seção VI são discutidas as considerações finais.

II. Preliminares

O mapa de Arnold discreto $\Gamma : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q$ é definido por

$$\Gamma(x, y) = (2x + y, x + y) \pmod q \tag{1}$$

em que $q = p^m$, $m \in \mathbb{N}$, $m \geq 2$, é uma potência do número primo p e $\mathbb{Z}_q$ é o conjunto das classes de equivalência dos números inteiros módulo q. A aplicação iterativa de Γ sobre uma condição inicial $(x_0, y_0) \in \mathbb{Z}_q \times \mathbb{Z}_q$ gera sequências bidimensionais $s = \{(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots\}$, em que o n-ésimo par $(x_n, y_n) \in \mathbb{Z}_q \times \mathbb{Z}_q$ é dado por

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_{n-1} \\ y_{n-1} \end{pmatrix} \pmod q. \tag{2}$$

A dinâmica iterativa definida por Γ gera sequências necessariamente periódicas, pois a cardinalidade do conjunto $\mathbb{Z}_q \times \mathbb{Z}_q$ é finita. Em particular, Γ pode ser escrito em função dos elementos da sequência de Fibonacci $F = \{0, 1, 1, 2, 3, 5, 8, \ldots\}$, tal que (x_n, y_n) é dado por [15]

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} F_{2n+1} & F_{2n} \\ F_{2n} & F_{2n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \pmod q \tag{3}$$

em que F_n é o n-ésimo termo de F. Portanto, o período do mapa de Arnold discreto pode ser calculado em função do período da sequência de Fibonacci módulo q, conhecido como período de Pisano [16]. No caso do anel $\mathbb{Z}_{2^m}$, o período de Pisano é $T = 3 \times 2^{m-1}$, e o período do mapa de Arnold discreto é $T_A = T/2 = 3 \times 2^{m-2}$ [15], [17].

Na seção seguinte, é apresentado um projeto de PRNG baseado em sequências binárias geradas pela dinâmica iterativa do mapa de Arnold discreto sobre o anel $\mathbb{Z}_{2^m}$.

III. PRNG 1

Inicialmente, é gerada uma sequência bidimensional $s = \{(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots\}$ pela aplicação iterativa de (2) sobre uma condição inicial $(x_0, y_0) \in \mathbb{Z}_{2^m} \times \mathbb{Z}_{2^m}$. Em particular, consideramos $m = 32$ e $m = 64$ como em [5], [7]. Cada elemento do n-ésimo par (x_n, y_n) é representado por um bloco binário $B_n = b_{m-1}b_{m-2}\cdots b_1 b_0$ de comprimento m, de forma que a cada iteração de (2) são gerados dois blocos binários, associados às variáveis x e y, os quais denotamos por $B_n^{(x)}$ e $B_n^{(y)}$, respectivamente.

Uma análise estatística em cada bloco $B_n^{(x)}$ e $B_n^{(y)}$ indica que estes possuem bits fortemente correlacionados entre si e, além disso, a correlação é mais expressiva entre os bits menos significativos. Uma abordagem usualmente empregada para gerar sequências binárias com boas propriedades de aleatoriedade (ser aprovada em alguma suite de testes estatísticos) é definir operações de pós-processamento destes blocos. Seja a taxa r (em bits/amostra) o número de bits gerados após o pós-processamento para cada par de blocos gerados $B_n^{(x)}$ e $B_n^{(y)}$. Para se obter um PRNG com baixa complexidade, propomos uma operação XOR entre os sub-blocos mais significativos de $B_n^{(x)}$ e os menos significativos de $B_n^{(y)}$, conforme descrito a seguir.

TABELA I

Proporções obtidas com a suíte NIST para o PRNG 1 sobre os aneis $\mathbb{Z}_{2^{32}}$ e $\mathbb{Z}_{2^{64}}$. Nos testes múltiplos (indicados por *) é mostrada apenas a proporção mínima.

Teste estatístico	$\mathbb{Z}_{2^{32}}$	$\mathbb{Z}_{2^{64}}$
Frequency	1	0,99
Block Frequency	0,99	1
Cumulative Sums*	0,99	0,98
Runs	0,98	0,99
Longest Run	0,99	0,99
Rank	1	0,99
FFT	0,97	0,98
Non-Ovla. Temp.*	0,96	0,96
Ovla. Temp.	0,99	0,99
Universal	1	0,97
Approximate Entropy	1	1
Ran. Exc.*	0,97	0,97
Ran. Ex. Var.*	0,97	0,97
Serial*	0,99	0,99
Linear Complexity	1	1

Seja (x_n, y_n) o n-ésimo par. Particionamos $B_n^{(x)}$ e $B_n^{(y)}$ em m/ℓ sub-blocos de comprimento ℓ, com $\ell = 8$ para $m = 32$ e $\ell = 16$ para $m = 64$. Nos dois casos, o número de sub-blocos em cada bloco é igual a 4. Vamos denominar estes sub-blocos de $B_n^{(x_i)}$ e $B_n^{(y_i)}$, $i = 1, 2, 3, 4$. Por convenção, a sub-blocos de menor índice são atribuídos bits mais significativos de $B_n^{(x)}$ ($B_n^{(y)}$); assim, o sub-bloco $B_n^{(x_1)}$ ($B_n^{(y_1)}$) contêm os bits mais significativos, enquanto o sub-bloco $B_n^{(x_4)}$ ($B_n^{(y_4)}$) contêm os bits menos significativos. A operação de pós-processamento consiste na concatenação dos sub-blocos $B_n^{(x_i)} \oplus B_n^{(y_{5-i})}$, em que $\oplus$ denota a operação XOR, para $i = 1, 2, 3, 4$. Neste caso, a taxa de bits por amostra do PRNG 1 é $r = m$.

A Tabela I detalha os resultados obtidos para o PRNG 1 usando a suíte estatística NIST versão SP 800-22 [14]. As simulações consideram um conjunto de 100 sequências de comprimento 10^6 cada. O nível de confiança empregado é $\alpha = 0{,}01$, conforme recomendado em [14]. As sequências binárias geradas apresentam sucesso em todos os testes para os dois aneis considerados, com proporções equivalentes. Isto indica que o PRNG 1 é escalonável e pode ser implementado com valores maiores de m, de acordo com a necessidade da aplicação.

IV. PRNG 2

Similarmente ao caso anterior, cada bloco $B_n^{(x)}$ e $B_n^{(y)}$ é particionado em m/ℓ sub-blocos de comprimento ℓ cada. Em particular, consideramos $\ell = 4$ e $\ell = 8$ para $m = 32$, $\ell = 8$ e $\ell = 16$ no caso $m = 64$. Denominamos os sub-blocos da partição como $B_n^{(x_i)}$ e $B_n^{(y_i)}$, $i = 1, 2, \ldots, m/\ell$. Convencionamos, assim como no PRNG 1, que os blocos com maior índice contêm os bits mais significativos.

Definimos o sub-bloco $B_n^{(z_i)}$ pela multiplicação dos sub-blocos da partição das variáveis x e y, ou seja, $B_n^{(z_i)} = B_n^{(x_i)} \times B_n^{(y_i)}, i = 1, 2, \ldots, m/\ell$. Devido à operação de multiplicação, são necessários 2ℓ bits para representar cada sub-bloco $B_n^{(z_i)}$. Esta etapa pode ser interpretada como um aumento na dimensão das sequências bidimesionais geradas pelo mapa de Arnold discreto, pois a cada par (x_n, y_n) associamos uma m/ℓ-upla $(B_n^{(z_1)}, B_n^{(z_2)}, \ldots, B_n^{(z_{\ell/m})})$ de comprimento $2m$ bits. Na próxima etapa, são construídos blocos de comprimento 2ℓ definidos pela concatenação dos sub-blocos $B_n^{(x_i)}$ e $B_n^{(y_i)}$. Definimos estes blocos como $B_n^{(w_i)} = B_n^{(x_i)} + B_n^{(y_i)}$, $i = 1, 2, \ldots, m/\ell$, em que $+$ indica a operação de concatenação entre blocos binários.

A etapa seguinte consiste em eliminar a correlação entre os bits menos significativos da sequência binária original gerada pelo mapa de Arnold discreto. Para isto, executamos a operação XOR entre os blocos $B_n^{(z_i)}$ e $B_n^{(w_i)}$ gerados na etapa anterior, de forma a operar entre bits mais e menos significativos. No caso $m = 32$ e $\ell = 8$, definimos

$$\begin{aligned}
\beta_n^1 &= B_n^{(z_1)} \oplus B_n^{(w_4)} \oplus B_n^{(w_3)} \\
\beta_n^2 &= B_n^{(z_2)} \oplus B_n^{(w_3)} \oplus B_n^{(w_2)} \\
\beta_n^3 &= B_n^{(z_3)} \oplus B_n^{(w_2)} \oplus B_n^{(w_1)} \\
\beta_n^4 &= B_n^{(z_4)} \oplus B_n^{(w_1)} \oplus B_n^{(w_4)}
\end{aligned}$$

e para $m = 32$ e $\ell = 4$ definimos

$$\begin{aligned}
\beta_n^1 &= B_n^{(z_1)} \oplus B_n^{(w_8)} \oplus B_n^{(w_5)} \\
\beta_n^2 &= B_n^{(z_2)} \oplus B_n^{(w_7)} \oplus B_n^{(w_4)} \\
\beta_n^3 &= B_n^{(z_3)} \oplus B_n^{(w_6)} \oplus B_n^{(w_3)} \\
\beta_n^4 &= B_n^{(z_4)} \oplus B_n^{(w_5)} \oplus B_n^{(w_2)} \\
\beta_n^5 &= B_n^{(z_5)} \oplus B_n^{(w_4)} \oplus B_n^{(w_1)} \\
\beta_n^6 &= B_n^{(z_6)} \oplus B_n^{(w_3)} \oplus B_n^{(w_8)} \\
\beta_n^7 &= B_n^{(z_7)} \oplus B_n^{(w_2)} \oplus B_n^{(w_7)} \\
\beta_n^8 &= B_n^{(z_8)} \oplus B_n^{(w_1)} \oplus B_n^{(w_6)}.
\end{aligned}$$

Os casos $m = 64$ com $\ell = 8$ e $\ell = 16$ são definidos da mesma forma que os casos $\ell = 4$ e $\ell = 8$ para $m = 32$.

A etapa final consiste em concatenar os blocos β_n^i. Esta operação define o n-ésimo bloco binário de saída do PRNG 2, o qual denotamos por β_n e é definido por

$$\beta_n = \beta_n^1 + \beta_n^2 + \cdots + \beta_n^{m/\ell}. \tag{4}$$

Como cada bloco β_n^i tem comprimento 2ℓ, o comprimento de cada bloco β_n é dado por $m/\ell \times 2\ell = 2m$. Logo, para um PRNG 2 sobre $\mathbb{Z}_{2^m}$ a taxa de bits por amostra é dada por $r = 2m$, ou seja, o PRNG 2 tem o dobro da taxa de bits por amostra em relação ao PRNG 1.

TABELA II

PROPORÇÕES OBTIDAS COM A SUÍTE ESTATÍSTICA NIST PARA O PRNG 2 SOBRE OS ANEIS $\mathbb{Z}_{2^{32}}$ E $\mathbb{Z}_{2^{64}}$. NOS TESTES MÚLTIPLOS (INDICADOS POR *) É MOSTRADA APENAS A PROPORÇÃO MÍNIMA.

-	$\mathbb{Z}_{2^{32}}$		$\mathbb{Z}_{2^{64}}$	
Teste Estatístico	$\ell = 4$	$\ell = 8$	$\ell = 8$	$\ell = 16$
Frequency	0,99	1	1	1
Block Frequency	1	1	1	1
Cumulative Sums*	0,98	1	1	1
Runs	0,99	1	0,99	1
Longest Run	0,99	1	0,99	0,98
Rank	0,97	0,98	1	0,99
FFT	0,96	0,98	0,98	0,98
Non-Ovla. Temp.*	0,96	0,96	0,97	0,96
Ovla. Temp.	0,99	0,98	0,99	0,99
Universal	0,99	0,99	0,99	1
Approximate Entropy	0,98	0,98	0,98	0,99
Ran. Exc.*	0,96	0,98	0,98	0,96
Ran. Ex. Var.*	0,98	0,98	0,96	0.97
Serial*	0,99	0,99	0,97	0,99
Linear Complexity	0,97	0,98	0,97	0,97

A Tabela II detalha os resultados obtidos com a suíte NIST para o PRNG 2. As sequências binárias geradas pelo PRNG 2 são aprovadas em todos os testes estatísticos. Também pode-se observar que para $\mathbb{Z}_{2^{32}}$ as proporções são ligeiramente melhores quando $\ell = 8$. De forma análoga, para $\mathbb{Z}_{2^{64}}$ as proporções são melhores quando $\ell = 16$. Este comportameto indica que proporções melhores são obtidas quando se particiona os blocos binários com comprimentos maiores. Entretando, acarreta uma piora no consumo de hardware e no *throughput*, conforme detalhado na seção a seguir. Outra característica do PRNG 2 é a escalonabilidade, como também ocorre no PRNG 1.

V. IMPLEMENTAÇÃO EM FPGA

Para analisar o consumo de recursos do PRNG 1 e do PRNG 2, estes foram implementados na FPGA Xilinx Virtex-5 (V5) XC5VLX50T-1FFG1136 (ML505). Consideramos $q = 2^{32}$ e $q = 2^{64}$ no caso do PRNG 1 e $q = 2^{32}$ no caso do PRNG 2. Observe que a taxa de bits por amostra é $2m$ para o PRNG 2, portanto, quando $q = 2^{64}$ a taxa é 128 bits/amostra, sendo proibitivo para a implementação na Virtex-5.

Na Tabela III são detalhados os resultados obtidos pela FPGA para o PRNG 1 e para o PRNG 2, bem como para os PRNGs propostos em [12], [13]. O PRNG proposto em [13] emprega uma representação de 32 bits e utiliza 24 bits por amostra. O PRNG 1 sobre $\mathbb{Z}_{2^{32}}$ também utiliza uma representação de 32 bits mas com uma taxa de bits por amostra superior. Além disso, consome 71% de LUTs e 97% de FFs, e *throughput* 140% maior em comparação com [13]. No caso do PRNG 1 sobre $\mathbb{Z}_{2^{64}}$, observa-se que o consumo de LUTs e FFs é basicamente o dobro em relação ao caso $\mathbb{Z}_{2^{32}}$, porém o *throughput* não é linear com m, sendo de aproximadamente 158% da taxa do PRNG 1.

TABELA III
RESULTADOS OBTIDOS PELA IMPLEMENTAÇÃO EM FPGA DO PRNG 1 SOBRE $\mathbb{Z}_{2^{32}}$ E $\mathbb{Z}_{2^{64}}$ E DO PRNG 2 SOBRE $\mathbb{Z}_{2^{32}}$ COM $\ell = 4$ E $\ell = 8$. TAMBÉM SÃO MOSTRADOS ALGUNS RESULTADOS DA LITERATURA.

-	-	-	PRNG 1		PRNG 2	
Método	[13]	[12]	$\mathbb{Z}_{2^{32}}$	$\mathbb{Z}_{2^{64}}$	$\ell = 8$	$\ell = 4$
FPGA	V5	Z7	V5	V5	V5	V5
LUTs	178	2017	127	255	448	278
FFs	65	3458	63	128	136	144
DSPs	0	8	0	0	0	0
Máx. Freq. (MHz)	298,597	113	314,37	248,69	190,33	278,86
Bits/Ciclo	24	60	32	64	64	64
Throughput (Mbps)	7166,328	6780	10059,73	15916,44	12181,20	17847,18
Potência à Máx. Freq. (mW)	500	73	772	959	909	1105
Potência/Throughput (mW/Mbps)	0,070	0,011	0,077	0,060	0,075	0,062

Comparando o PRNG 2 com $\ell = 8$ e $\ell = 4$, pode-se ver que o caso $\ell = 4$ apresenta melhor performance, com 62% do consumo de LUTs, 105% do consumo de FFs e 146% do *throughput*, em comparação com o caso $\ell = 8$. Portanto, conclui-se que particionamentos em blocos de comprimento menor são mais vantajosos em termos de recursos de hardware e *throughput*. No caso de [12] o número de bits por amostra é 64, sendo o mesmo do PRNG 2 sobre $\mathbb{Z}_{2^{32}}$. Observa-se que nos dois casos, $\ell = 8$ e $\ell = 4$ o PRNG 2 supera [12] em LUTs, FFs e *throughput*. É importante observar que a FPGA em [12] é a Xilinx Zynq-7000 (Z7).

Para estimar a potência necessária para a implementação dos PRNGs propostos, empregamos o *Xilinx XPower Analyzer*. Observa-se que os PRNGs propostos consomem maior potência em relação a [12], [13]. Entretanto, quando se considera a potência normalizada pelo *throughput*, os PRNGs propostos apresentam consumo de potência equivalente a [13] para o PRNG 1 sobre $\mathbb{Z}_{2^{32}}$ e para o PRNG 2 com $\ell = 8$. Nos casos do PRNG 1 sobre $\mathbb{Z}_{2^{64}}$ e do PRNG 2 com $\ell = 4$ a potência normalizada é menor que em [13]. O PRNG proposto em [12] consome tanto a menor potência quanto a menor potência normalizada, ao custo de um maior consumo de recursos em comparação aos outros PRNGs.

VI. CONCLUSÕES

Neste trabalho, foram propostos dois PRNGs de alta taxa baseados no mapa de Arnold discreto sobre os aneis de inteiros $\mathbb{Z}_{2^{32}}$ e $\mathbb{Z}_{2^{64}}$. As propriedades estatísticas dos PRNGs propostos foram avaliadas com a suíte estatística NIST, apresentando aprovação em todos os testes. Ambos os geradores foram implementados em FPGA para analisar a complexidade de implementação e consumo de recursos. Comparações com propostas recentes da literatura mostram que os PRNGs propostos apresentam *throughput* superior e, no caso do PRNG 1, consumindo um menor número de LUTs e FFs. Já no caso do PRNG 2, apesar de consumir mais recursos que o PRNG 1, apresenta *throughput* superior às demais propostas da literatura. Além disso, quando se considera a potência normalizada pelo *throughput* os PRNGs propostos apresentam eficiência energética competitiva.

Para trabalhos futuros, serão analisadas como as operações de pós-processamento afetam o período das sequências binárias. Apesar das simulações computacionais indicarem que os períodos das sequências geradas pelos PRNGs propostos são iguais ao período do mapa de Arnold, se faz necessária uma demonstração formal desta afirmação.

REFERÊNCIAS

[1] L. Kocarev, J. Szczepanski, J. M. Amigo, and I. Tomovski, "Discrete chaos-I: Theory," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 53, no. 6, pp. 1300–1309, Jun. 2006.

[2] B. Chirikov and F. Vivaldi, "An algorithmic view of pseudochaos," *Physica D: Nonlinear Phenomena*, vol. 129, no. 3, pp. 223 – 235, May 1999.

[3] F. Chen, K. Wong, X. Liao, and T. Xiang, "Period distribution of generalized discrete Arnold cat map for $N = p^e$," *IEEE Trans. Inf. Theory*, vol. 58, no. 1, pp. 445–452, Jan. 2012.

[4] B. Yang and X. Liao, "Some properties of the logistic map over the finite field and its application," *Signal Processing*, vol. 153, pp. 231 – 242, Dec. 2018.

[5] C. E. C. Souza, D. P. B. Chaves, and C. Pimentel, "One-dimensional pseudo-chaotic sequences based on the discrete Arnold's cat map over $\mathbb{Z}_{3^m}$," *IEEE Trans. Circuits and Syst. II: Exp. Briefs*, vol. 68, no. 1, pp. 491–495, Jan. 2021.

[6] C. Wang, Y. Di, J. Tang, J. Shuai, Y. Zhang, and Q. Lu, "The dynamic analysis of a novel reconfigurable cubic chaotic map and its application in finite field," *Symmetry*, vol. 13, no. 8, Aug. 2021.

[7] C. E. C. Souza, D. Moreno, D. P. B. Chaves, and C. Pimentel, "Pseudo-chaotic sequences generated by the discrete Arnold's map over $\mathbb{Z}_{2^m}$: Period analysis and FPGA implementation," *aceito para publicação em IEEE Trans. Instrum. Meas.*, 2022.

[8] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, and S. Celma, "Chaos-based bitwise dynamical pseudorandom number generator on FPGA," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 1, pp. 291–293, Jan. 2019.

[9] I. Koyuncu, M. Tuna, I. Pehlivan, C. Fidan, and M. Alçın, "Design, FPGA implementation and statistical analysis of chaos-ring based dual entropy core true random number generator," *Analog Integrated Circuits and Signal Processing*, vol. 102, pp. 445–456, Feb. 2020.

[10] S. Kalanadhabhatta, D. Kumar, K. K. Anumandla, S. A. Reddy, and A. Acharyya, "PUF-Based secure chaotic random number generator design methodology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 7, pp. 1740–1744, Jul. 2020.

[11] A. Kumar Panda and K. Chandra Ray, "A coupled variable input LCG method and its VLSI architecture for pseudorandom bit generation," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1011–1019, Apr. 2020.

[12] N. T. Nguyen, T. Bui, G. Gagnon, P. Giard, and G. Kaddoum, "Designing a pseudorandom bit generator with a novel five-dimensional-hyperchaotic system," *IEEE Trans. Ind. Electron.*, vol. 69, no. 6, pp. 6101–6110, Jun. 2022.

[13] A. A. Rezk, A. H. Madian, A. G. Radwan, and A. M. Soliman, "Multiplierless chaotic pseudo random number generators," *AEÜ - International Journal of Electronics and Communications*, vol. 113, p. 152947, Jan. 2020.

[14] L. E. B. III *et al.*, *SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Gaithersburg, MD, United States: Nat. Inst. Std. & Technol., 2010.

[15] F. J. Dyson and H. Falk, "Period of a discrete cat mapping," *The American Mathematical Monthly*, vol. 99, no. 7, pp. 603–614, Aug. 1992.

[16] D. D. Wall, "Fibonacci series modulo m," *The American Mathematical Monthly*, vol. 67, no. 6, pp. 525–532, Jun. 1960.

[17] F. Chen, K. Wong, X. Liao, and T. Xiang, "Period distribution of the generalized discrete Arnold cat map for $N = 2^e$," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3249–3255, May 2013.