Estudo comparativo de sequências binárias pseudoaleatórias geradas por autômatos celulares

Sílvia Regina Leite Magossi e Marco Aurélio Amaral Henriques

***Resumo*— Autômato Celular (AC) é um sistema dinâmico que, a partir de iterações locais simples, pode exibir comportamentos complexos. Por este motivo ele é um bom candidato a gerador de números pseudoaleatórios. Este trabalho avalia a geração de números pseudoaleatórios por ACs unidimensionais em diferentes configurações, buscando indícios sobre as que seriam mais adequadas. As comparações são baseadas em testes estatísticos padronizados do NIST e os resultados mostram que os AC unidimensionais de raio dois geram números pseudoaleatórios com melhores características de aleatoriedade que os ACs de raio um, uniformes ou híbridos, sendo portanto mais recomendados para aplicações que exigem sequências binárias aleatórias com baixo custo.**

Palavras-chave – Autômato Celular, PRNG, números randômicos, números pseudoaleatórios

Abstract*— *Cellular automata (CA) are dynamic systems that exhibit complex behavior from simple local iterations. For this reason CAs are good candidates for pseudorandom number generators. This work analyzes the pseudorandom number generation by one-dimensional CAs working in different configurations, looking for those with the best characteristics. The comparisons are based on standard statistical tests proposed by NIST and the results show that radius two CAs generate pseudorandom numbers with better characteristics than radius one, uniform or hybrid, which makes them recommended for applications that require random binary sequences at a low cost.

Keywords—Cellular Automata, CAs, PRNG, random numbers, pseudorandom numbers

I. Introdução

Na década de 50 o matemático John von Neumann propôs um modelo de auto-reprodução biológica [1] a partir de um espaço celular, utilizando autômato celular (AC). A partir disso o interesse científico por autômatos celulares aumentou e outras aplicações surgiram em diversas áreas tais como: biologia [2], processamento de imagens [3] e processamento paralelo [4]. Stephen Wolfram, na década de 80, iniciou uma nova vertente de aplicações de ACs, utilizando-os como modelos matemáticos para sistemas estatísticos de auto-organização [5]. Dentre as características de auto-organização mencionadas por Wolfram, uma identificava regras aleatórias em sua evolução [6]. Esse tipo de AC pode ser utilizado em diversas aplicações que exijam o uso de números aleatórios, como sistemas de segurança por exemplo, por possibilitar a criação de geradores de números pseudoaleatórios (PRNGs) a um baixo custo. Este trabalho realiza uma comparação entre as sequências pseudoaleatórias geradas por diferentes tipos e tamanhos de ACs, permitindo que tenhamos maior clareza quanto ao desempenho de um PRNG baseado nestas estruturas.

II. Características de um Autômato Celular

A estrutura investigada por Wolfram pode ser vista como um arranjo discreto de células, onde cada célula assume os valores 0 ou 1. A cada intervalo de tempo as células se alteram e essas alterações dependem do próprio estado da célula e dos estados das células vizinhas, à esquerda e à direita. As células evoluem em tempo discreto a partir de alguma regra determinística.

Os ACs mais simples são os Autômatos Celulares Elementares (AC_{Elem}) unidimensionais, consistindo de um arranjo em que cada célula pode assumir valores 0 ou 1 (branco ou preto) como representado na Fig. 1.

Fig. 1. Autômato celular binário com 17 células

Normalmente são utilizados na caracterização de um AC os parâmetros k, quantidade de estados por célula, e r (raio), distância da célula central em relação à sua vizinha mais distante.

Os ACs com configuração k=2 e r=1 usam 2r+1 = 3 células binárias em suas regras de evolução e são chamados de ACs de vizinhança três e têm $k^{2r+1} = 8$ possíveis padrões, como apresentado na Fig. 2. A figura também mostra o estado futuro da célula central para cada padrão.

A partir destas 8 configurações possíveis, tem-se um total de $2^8 = 256$ distintos mapeamentos referentes a todas as configurações de vizinhança para o próximo estado.

O próximo estado de cada uma das células pode ser considerado um dos bits de um padrão de oito bits e o equivalente número decimal é chamado de número da regra.

Fig. 2 – Os 8 padrões de evolução de um AC elementar

Vamos utilizar a seguinte notação para caracterizar um AC:

i: a posição de uma célula individual em um arranjo unidimensional de células;

t : tempo (discreto);

$\mathbf{a}_i^{(t)}$: estado de saída da *i*-th célula no *t*-th passo de tempo.

Silvia Regina Leite Magossi e Prof. Dr. Marco Aurélio Amaral Henriques, Faculdade de Engenharia Elétrica e Computação – FEEC - Universidade Estadual de Campinas (UNICAMP), Campinas – SP – Brasil, E-mails: srlm@dca.fee.unicamp.br, marco@dca.fee.unicamp.br .

A função de transição para um AC de raio um é apresentada na Eq. (1).

$$a_i^{(t+1)} = f[a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}] \quad (1)$$

onde f representa a função de transição local que evolui como uma lógica combinacional.

A regra local para um autômato celular pode ser considerada uma função Booleana de uma célula com suas vizinhas. Uma possível função equivalente módulo dois é apresentada na Eq. (2).

$$a_i^{(t)} = f[\sum_{j=-r}^{j=r} a_{i+j}^{(t-1)}] \quad (2)$$

Como exemplo, apresentamos na Tabela 1 os padrões de formação associados às regras 30 e 150. A denominação da regra de um AC_{Elem} advém da conversão para base 10 do padrão binário do novo estado da célula central.

Tabela 1 – Padrões de formação para um AC pelas regras 30 e 150

Autômato celular elementar									
Estado de vizinhança	111	110	101	100	011	010	001	000	Regra
Novo estado da célula central	0	0	0	1	1	1	1	0	30
	1	0	0	1	0	1	1	0	150

As correspondentes expressões lógicas para as regras 30 e 150 são apresentadas nas Eqs. (3) e (4) abaixo.

$$\text{Regra 30}: \; a_i^{(t+1)} = a_{i-1}^{(t)} \oplus (a_i^{(t)} \bigwedge a_{i+1}^{(t)}) \quad (3)$$

$$\text{Regra 150}: \; a_i^{(t+1)} = a_{i-1}^{(t)} \oplus a_i^{(t)} \oplus a_{i+1}^{(t)} \quad (4)$$

Os operadores $\oplus$ e $\bigwedge$ denotam as operações lógicas "ou exclusivo" e "ou inclusivo", respectivamente. De um modo geral, as células evoluem de acordo com as iterações da regra escolhida, como mostra a Eq. (5).

$$a_i^{(t)} = F[a_{i-r}^{(t-1)}, a_{i-r+1}^{(t-1)}, \ldots, a_i^{(t-1)}, \ldots, a_{i+r-1}^{(t-1)}, a_{i+r}^{(t-1)}] \quad (5)$$

Na evolução de ACs, temos três possíveis condições limites de fronteira: fronteira nula, fronteira periódica e fronteira intermediária. A condição de fronteira nula de um AC ocorre quando o vizinho esquerdo da célula mais à esquerda e o vizinho direito da célula mais à direita são considerados nulos (zero). Um AC é dito ter fronteira periódica quando as células mais à esquerda e mais à direita são consideradas vizinhas. Um AC tem condições de fronteira intermediária quando o próximo estado das células das extremidades do AC depende apenas delas próprias e de suas vizinhas que fazem parte do AC. A Fig. 3 representa um AC com fronteira periódica, configuração que foi a utilizada neste trabalho.

Se todas as células do AC obedecem a uma mesma regra dizemos que é um AC uniforme, caso contrário dizemos que é um AC híbrido. A Fig. 4 mostra a evolução de um AC uniforme pela regra 30, partindo de um estado inicial com apenas uma célula igual a um.

Fig. 3 - AC com fronteira periódica

Fig. 4 - Diagrama espaço-tempo da evolução de um AC de 31 células pela regra 30

III. Utilização de autômatos celulares como geradores de números pseudoaleatórios

Um AC binário unidimensional elementar com um número razoável de células (64 ou mais), uma regra adequada (como a regra 30 por exemplo) operando com um grande número de iterações pode ser usado como um gerador de sequências binárias pseudoaleatórias tomadas linha a linha a cada iteração [6]. Uma outra forma de obter os bits de uma sequência aleatória poderia ser coletando-os na vertical, isto é, coletando os bits de uma mesma célula à medida que o AC vai evoluindo.

Em ambos os casos é preciso avaliar a qualidade (aleatoriedade) da sequência de bits obtida. Uma forma de se fazer isso é por meio de testes estatísticos de aleatoriedade, como aquele proposto pelo NIST [7].

Padrões complexos podem ser gerados também por ACs híbridos, nos quais cada célula obedece uma regra distinta durante toda a sua evolução. Um AC híbrido bem conhecido é aquele que usa as regras 90 e 150 em suas células de maneira a reproduzir o comportamento de uma estrutura conhecida como *Linear Feedback Shift Register*, a qual pode gerar padrões aleatórios com ciclo de repetição máximo (2^N-1, onde N é o número de células do AC), se determinadas condições forem satisfeitas [8].

Podemos usar também ACs de raio dois, os quais têm 2^5=32 padrões elementares e 2^{32} regras distintas, muitas delas com comportamento caótico. Nestes ACs cada célula depende de outras quatro para definir seu estado futuro, o que pode tornar bem menos previsível seu comportamento, melhorando sua característica como PRNG. Até onde conhecemos, não existem estudos na literatura comparando a qualidade das sequências aleatórias geradas por ACs de raio 1 e 2, sendo este o foco principal deste trabalho.

IV. Classes de Autômatos Celulares

Baseado em propriedades estatísticas dos ACs de vizinhança três, Wolfram [5] classificou os autômatos em quatro grandes classes. Medidas estatísticas de ordem e caos nos padrões gerados pela evolução dos ACs foram usadas para distinguir as 4 classes de comportamento.

Classe 1: a partir de um estado inicial aleatório, o AC evolui após um número finito de passos para um único estado homogêneo, em que todas as células ficam com o mesmo valor. São exemplos da Classe 1 as regras 0 e 255.

Classe 2: a partir de um estado inicial aleatório, são geradas estruturas simples e estáveis. Estas exibem estados persistentes. Sua aparência é semelhante a listras com ciclos de períodos curtos, contendo a mesma estrutura persistente. São exemplos da Classe 2 as regras 2 e 240.

Classe 3: a evolução de AC da Classe 3, a partir de um estado inicial aleatório, conduz a padrões caóticos, totalmente irregulares em sua evolução, tornando os ACs desta classe bons candidatos para a geração de sequências pseudoaleatórias. O aumento da entropia e a natureza irreversível manifestam-se com a evolução do autômato. São exemplos da Classe 3 as regras 45, 90, 150 e 30.

Classe 4: os ACs desta classe, evoluem com um comportamento complexo. Em muitos casos é possível prever uma evolução para um comportamento estável, após um número finito de passos. Em alguns casos são encontradas estruturas estáveis ou periódicas persistentes, como mostram as regras 137 e 169.

V. Aplicação do conjunto de testes do NIST

Para avaliar Geradores de Números Aleatórios (RNGs) e Pseudoaleatórios (PRNGs), o Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST) [7] desenvolveu um conjunto de 15 testes: *Frequency (Monobit), Frequency within a Block, Runs, Longest-Run-of-Ones in a Block, Binary Matrix Rank, Discrete Fourier Transform (Spectral), Non-overlapping Template Matching, Overlapping Template Matching, Maure's "Universal Statistical", Linear Complexity, Serial, Approximate Entropy, Cumulative Sums, Random Excursions* e *Random Excursions Variant.* Estes testes estão documentados e disponibilizados como um conjunto de programas.

A fim de se obter resultados mais confiáveis, é recomendada a utilização de alguns parâmetros básicos, tais como o fornecimento de pelo menos 55 sequências de um milhão ou mais bits para os testes. Decidimos trabalhar com 100 sequências de um milhão de bits para ter uma boa margem de segurança e facilitar a interpretação dos resultados, o que exige a produção de 10^8 bits para testes de cada AC avaliado.

Para produzir sequências de bits aleatórios, programas utilizando a estrutura dos ACs como PRNGs foram desenvolvidos em Python, gerando 100 sequências binárias com 10^6 bits cada a partir da regra 30 com raio um, regras 90/150 (AC híbrido) com raio um e regra 1436194405 com raio dois. Os programas foram configurados para processar a evolução dos ACs a partir de estados iniciais randômicos produzidos pela função randint() do Python. Os tamanhos adotados para os ACs foram de 32, 64 e 128 células. Os bits gerados pela evolução dos ACs foram coletados por linha e por coluna, tendo sido usada uma célula central na coleta por coluna. A Tabela 2 apresenta os parâmetros adotados para evolução dos ACs escolhidos.

Tabela 2 – Parâmetros para geração das sequências

Regra	Raio	Tamanho	Iterações (linha)	Iterações (coluna)
30	1	32	31250	10^6
	1	64	15625	10^6
	1	128	7813	10^6
90/150 (AC híbrido)	1	32	31250	10^6
	1	64	15625	10^6
	1	128	7813	10^6
1436194405	2	32	31250	10^6
	2	64	15625	10^6
	2	128	7813	10^6

As sequências binárias geradas pelos ACs investigados, foram submetidas aos 15 testes do NIST, os quais exigiram alguns ajustes de parâmetros internos específicos conforme é mostrado na Tabela 3.

Tabela 3 – Tamanho dos blocos adotados nos testes do NIST

[1]	Block Frequency	10000
[2]	Non Overlapping Template	10
[3]	Overlapping Template	10
[4]	Approximate Entropy	10
[5]	Serial	16
[6]	Linear Complexity	5000

VI. Estudo comparativo

A Regra 30 foi exaustivamente estudada e avaliada por Wolfram[6], demonstrando habilidade em produzir sequências de bits randômicas.

O AC híbrido 90/150 discutido em [8] também apresenta uma evolução randômica. Seu desempenho em termos de PRNGs é considerado bom na literatura, com o benefício adicional de que pode gerar sequências de ciclo máximo.

A Regra 1436194405 foi apresentada na ref. [9], que utilizou um algoritmo genético para encontrar regras randômicas para ACs unidimensionais de raio dois.

Para efeito de avaliação visual, apresentamos na Fig. 5 os diagramas espaço-tempo da evolução (500 iterações) de ACs de 128 células pelas regras 30, 90/150 e 1436194405. Visualmente, ACs de raio dois parecem ter características mais aleatórias que os de raio unitário.

VII. Análise dos resultados

Todas as sequências geradas pelos ACs descritos foram submetidas aos testes do NIST que emitiu uma série de relatórios detalhando o desempenho de cada sequência.

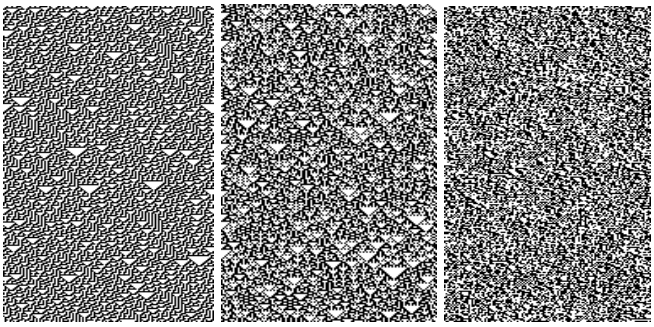

Fig. 5 - Evolução por 500 iterações das regras 30, 90/150 e 1436194405 em AC de 128 células

Apresentamos na Tabela 4 a consolidação dos relatórios referentes à regra 30 onde os bits foram coletados por linha. Nesta tabela observamos que ocorre uma evolução gradativa nos testes aprovados com o aumento do tamanho do AC. O caso com 32 células teve aprovação em somente sete dos 15 testes, mas este resultado melhorou quando o tamanho do AC foi aumentado para 64 (nove aprovações) e 128 (dez aprovações). A coluna de percentual mínimo mostra a fração mínima necessária de 100 sequências de 10^6 bits que precisam ser aprovadas para que os bits sejam considerados aleatórios pelo teste em questão (é um parâmetro calculado pelos programas do NIST e tem escala diferenciada nos testes 12 e 13). No AC considerado, nota-se também um aumento do percentual de sequências de 10^6 bits aprovadas à medida em que o tamanho do AC aumenta. De 72,1% de sequências consideradas aleatórias no AC de 32 células, este índice passou para 78,5% no AC de 64 células e para 84,3% no AC de 128 células, mostrando claramente que AC maiores permitem obter sequências consideradas aleatórias com mais facilidade. Provavelmente a razão para tal resultado se deve ao número muito maior de estados pelos quais o AC pode passar, diminuindo significativamente as chances de ficar preso em um ciclo de forma perceptível pelos testes.

Consolidamos na Tabela 5 estes e todos os demais testes feitos com os outros ACs nos mesmos três tamanhos já citados. Com base nestes resultados, podemos constatar que há significativa melhoria na qualidade das sequências de bits gerados, do ponto de vista da aleatoriedade, quando tais sequências são obtidas a partir de uma única coluna do AC. Praticamente em todos os casos, mesmo havendo uma ligeira diminuição no percentual médio de sequências aprovadas quando a coleta se deu por coluna, houve um aumento no número de testes aprovados. Isto pode ser explicado pela menor relação entre os bits de uma dada coluna, visto que são fortemente influenciados pelos seus vizinhos e pelos vizinhos dos vizinhos à medida que as iterações avançam. No caso de coleta de bits por linha, pode-se constatar uma relação de dependência relativamente clara entre os bits de uma linha e os da linha seguinte. Neste caso o comportamento específico da regra está impresso na sucessão de linhas, já que todas as células do AC estão representadas.

Há casos em que uma melhor qualidade na coleta por coluna não se manifesta claramente, mas geralmente a diferença é pequena. Entretanto, no AC de raio dois com 32 células há uma queda acentuada quando se troca a coleta de linha para coluna, contrariando as expectativas. Como os resultados desta configuração são ruins (treze das quinze sequências reprovadas em ambos os casos), entendemos que tal queda não possa ser considerada confiável. Acreditamos que a alta taxa de reprovação se deva ao fato de que o tamanho do AC está muito pequeno para lidar com raio dois, já que em 32 células só há, inicialmente, 6 grupos de cinco células totalmente independentes. Devido ao uso de fronteiras periódicas, a influência das células sobre si mesmas e suas vizinhas através das fronteiras acaba ocorrendo com maior facilidade. Por este motivo, neste caso as sequências binárias tiveram aprovação muito baixa nos testes.

Esta seria também a explicação para as melhorias observadas em praticamente todos os resultados quando aumentamos o tamanho dos autômatos de 32 para 64 células. Tais melhorias estão fortemente manifestadas tanto no número de testes aprovados como no percentual médio de sequências de 10^6 bits aprovadas.

Entretanto, chama a atenção o fato de que as melhorias de desempenho não são predominantes quando aumentamos os ACs de 64 para 128 células. Tanto no percentual médio de sequências aprovadas, como no número de testes aprovados temos praticamente o mesmo número de aumentos e de diminuições ao trocarmos o tamanho do AC. Acreditamos que já estamos bem acima do tamanho mínimo de um AC que garanta bons resultados (os ACs de tamanho 64 tiveram excelente desempenho) e, provavelmente, estejamos presenciando oscilações naturais na característica aleatória das sequências de bits geradas, as quais podem alterar se realizarmos outra bateria de testes com novas sequências geradas pelos mesmos ACs de 128 células. Este é um ponto a ser melhor avaliado futuramente.

Outro ponto importante a ser notado nos dados coletados por coluna (mais úteis do ponto de vista de PRNGs) é que o AC de raio dois proporciona resultados melhores que os ACs de raio um híbridos, considerados bons geradores de números pseudoaleatórios, por serem capazes de gerar sequências de ciclo máximo. Isto nos leva a crer que a influência de cinco células na definição do estado de uma delas traz um comportamento mais caótico que aquele em que apenas três células definem o estado de uma. Este é um resultado que nos motiva a ir além e investigar a qualidade da aleatoriedade das sequências de bits geradas por ACs bidimensionais, visto que cada célula tem seu estado futuro definido por cinco ou por nove células. Isto será considerado em um trabalho futuro.

VIII. Conclusões e trabalhos futuros

Neste trabalho foi possível fazer uma comparação detalhada, sob o ponto de vista de testes estatísticos padronizados, da aleatoriedade das sequências de bits geradas por três tipos de autômatos celulares: dois de raio um, sendo um deles híbrido, e um de raio dois. A comparação se baseou

nos dados fornecidos pelo pacote de testes disponibilizado pelo NIST, o qual avalia as sequências binárias sob uma lupa de 15 testes estatísticos diferentes.

Até onde sabemos, este é o primeiro trabalho que faz uma comparação direta entre ACs de raio um e dois e que constata um desempenho superior do AC de raio dois, o que motiva uma investigação mais apurada sobre a qualidade de ACs bidimensionais de raio um como PRNGs.

Com trabalhos futuros pretendemos comprovar com outros testes as características de ACs de 128 células em relação aos de 64 células e avaliar em detalhes o potencial dos ACs bidimensionais de raio um na geração de números pseudoaleatórios. Avaliações de custos de implementação de cada AC também estão entre os planos de investigação futura.

Agradecimentos: agradecemos ao acadêmico Yoshitomi Maehara pelo auxílio nos programas em Python que evoluem os ACs e geram as sequências binárias.

Tabela 4 – Detalhamento dos resultados dos testes NIST com ACs baseados na regra 30 (coleta de bits por linha)

Teste Estatístico		AC 32 células			AC 64 células			AC 128 células		
		% mínimo	%	Situação	% mínimo	%	Situação	% mínimo	%	Situação
1	*Frequency*	96,0	98,0	aprovado	96,0	100,0	aprovado	96,0	100,0	aprovado
2	*Block Frequency*	96,0	100,0	aprovado	96,0	100,0	aprovado	96,0	100,0	aprovado
3	*Cumulative Sums*	96,0	98,0	aprovado	96,0	100,0	aprovado	96,0	100,0	aprovado
4	*Runs*	96,0	0,0	reprovado	96,0	0,0	reprovado	96,0	20,0	reprovado
5	*Longest Run*	96,0	97,0	aprovado	96,0	100,0	aprovado	96,0	98,0	aprovado
6	*Rank*	96,0	98,0	aprovado	96,0	99,0	aprovado	96,0	100,0	aprovado
7	*FFT*	96,0	0,00	reprovado	96,0	0,0	reprovado	96,0	0,0	reprovado
8	*Non Overlapping Template*	96,0	85,4	reprovado	96,0	96,8	aprovado	96,0	98,6	aprovado
9	*Overlapping Template*	96,0	94,0	reprovado	96,0	97,0	aprovado	96,0	96,0	aprovado
10	*Universal*	96,0	70,0	reprovado	96,0	94,0	reprovado	96,0	92,0	reprovado
11	*Approximate Entropy*	96,0	0,0	reprovado	96,0	15,0	reprovado	96,0	77,0	reprovado
12	*Random Excursions*	94,7	95,9	aprovado	95,2	98,4	aprovado	95,1	97,9	aprovado
13	*Random Excursions Variant*	94,7	92,5	reprovado	95,2	93,1	reprovado	95,1	96,6	aprovado
14	*Serial*	96,0	55,0	reprovado	96,0	86,5	reprovado	96,0	89,0	reprovado
15	*Linear Complexity*	96,0	98,0	aprovado	96,0	98,0	aprovado	96,0	99,0	aprovado
% médio e número de aprovações			72,1	7 (46,6%)		78,5	9 (60%)		84,3	10 (66,6%)

Tabela 5 – Resultados consolidados de todos os testes com diferentes tipos de ACs

Tamanho do AC			32		64		128	
Regra	Raio	Coleta	Sequências aprovadas (% médio)	Testes aprovados	Sequências aprovadas (% médio)	Testes aprovados	Sequências aprovadas (% médio)	Testes aprovados
30	1	linha	72,1	7 (46,6 %)	78,5	9 (60,0 %)	84,3	10 (66,6 %)
30	1	coluna	84,5	9 (60,0 %)	99,2	15 (100,0 %)	98,5	15 (100,0 %)
90/150	1	linha	84,7	11 (73,3 %)	98,5	11 (73,3 %)	97,6	13 (86,6 %)
90/150	1	coluna	85,8	13 (86,6 %)	92,6	14 (93,3 %)	92,2	14 (93,3 %)
1436194405	2	linha	55,3	2 (13,3 %)	92,1	14 (93,3 %)	90,3	13 (86,6 %)
1436194405	2	coluna	47,1	2 (13,3 %)	99,2	15 (100 %)	98,9	15 (100,0 %)

Referências

[1] J.von Neumann, *The Theory of Self-Reproducing Automata*, A.W. Burks, ed., Univ. of Illinois Press, Urbana and London, 1966.

[2] M.Arbib, *Simple Self-Reproducing Universal Automata*, Information and Control, Vol. 9, 1966, p.177.

[3] K.Preston. at al. *Basics of Cellular Logic with Some Applications in Medical Image Processing*, proc. IEEE, Piscataway, N.J., 1979, p.67.

[4] F.B.Manning, *An Approach to Highly Integrated, Computer-Maintained Cellular Arrays*, IEEE Trans.Computers, Vol. C26,1977.

[5] S. Wolfram. *Statistical mechanics of cellular automata.* Rev.Mod. Phys 55 (1983) 601

[6] S. Wolfram,Random *Sequence Generation by Cellular Automata* , Advances in Applied Mathematics 7, 123-169 (1986)

[7] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray , S. Vo. *NIST-National Institute of Standards and Technology* is an agency of the U.S. Department of Commerce., Revised: April 2010.

[8] Hortensius, Peter D., et al. "*Cellular automata-based pseudorandom number generators for built-in self-test*." IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (1989): 842-859.

[9] P.Bouvry, F.Seredynski, A. Y. Zomaya, *Application of Cellular Automata for Cryptography*, R. Wyrzykowski et al. (Eds.): PPAM 2003, LNCS 3019, pp. 447–454, 2004. Springer-Verlag Berlin Heidelberg 2004.