doi:10.21311/001.39.1.01

Global Feedback Self-Programmable Cellular Automaton Random Number Generator

Dan Mocanu[1,2], Alexandru Gheolbanoiu[2],Radu Hobincu[2],Lucian Petrica[2,*]
[1]*Ixia LLC, IxNovation Research Laboratory, Bucharest, Romania*
[2]*Politehnica University of Bucharest, Romania*

Abstract: We present a cellular automaton (CA) based pseudo-random number generator (PRNG) which utilizes its own output to modify its internal rule set in order to produce a high-quality, cryptographically secure random number sequence. Previous work on self-programmable cellular automatons (SPCAs), i.e. CAs which can modify their own rule set,has utilized local feedback, whereby the state of each cell is utilized to select the next rule to be applied to the CA cell. We utilize instead a global feedback which selects a new rule, and a cell to which the new rule is to be applied, based on the entire CA state. We evaluate the quality of global feedback SPCA (GF-SPCA) utilizing the ENT and NIST statistical test suites. We also implement the GF-SPCA in field programmable gate array (FPGA) technology and evaluate its resource utilization. Our analysis demonstrates that the GF-SPCA is the first cryptographically strong SPCA, and is resource-efficient when implemented in FPGA, requiring under 300 look-up tables.

Key Words: Cellular Automaton, Random Number Generator, FPGA

1. INTRODUCTION

Random number generators (RNGs) are essential for the generation of cryptographic keys for secure online communication, and have become a necessary part of any digital system. Other applications of RNGs are statistical simulation algorithms based on the Monte-Carlo method and random event triggering for video games. Since all of these applications can be executed on e.g. a desktop computer, the processing system needs to include a RNG of sufficiently good quality. A true RNG is desirable for cryptographic applications because one cannot predict its output under any circumstances, but is difficult to implement in digital logic, which is deterministic. Additionally, a true RNG cannot reproduce a given sequence of generated numbers, a feature which is important in the simulation of physical and mechanical processes.

Pseudo-random number generators (PRNGs) are deterministic circuits or software which create the appearance of randomness. PRNGs are characterized by an internal initial state, called the seed, which is transformed by a mathematical function or algorithm in discrete steps. At each step, a random number is generated. By recreating the seed, the entire sequence of generated random numbers may be repeated. Linear Feedback Shift Registers (LFSR) are the most popular class of PRNG due to their efficient hardware implementation (Golomb 1982).

Several researchers have attempted to harness the properties of cellular automata for random number generation. A cellular automaton (CA) is a network of cells in a finite dimension space, whereby each cell has a set number of possible states which are updated periodically based on a rule which takes into account the previous state of the cell and the states of other cells in a neighborhood. A CA may be uniform, i.e., all cells have the same rule, or non-uniform, with different cells having different rules. The one-dimensional (1D) and two dimensional (2D) CAs have seen more research interest. A random number is formed by concatenating the 1-bit states of all CA cells. CAs are efficiently implemented in digital logic, including in Field Programmable Gate Arrays (FPGA).

In this work, we present a non-uniform 1D CA PRNG which achieves good randomness properties by implementing a feedback mechanism which periodically modifies the rule of individual cells in the underlying CA. At each step, the particular cell which is to be modified and the next rule to be applied are selected randomly based on the PRNG outputs. In this way, the feedback mechanism operates as a "randomness amplifier" for the CA, resulting in increased randomness while maintaining the simplicity and efficiency of the CA cells, which facilitates hardware implementation. The effectiveness of our approach is demonstrated with industry-standard randomness benchmarks. We also evaluate the resource utilization of the proposed PRNG implemented in FPGA.

2. CELLULAR AUTOMATON PRNGs

Cellular automata consist of a lattice of identical interconnected cells, whereby each cell is a finite state automaton. The values of the state of the cells evolve in discrete time steps according to deterministic rules that specify the next state of each cell as a function of the current state and the states of neighboring cells. An elementary CA consists of a sequence of cells arranged on a line. Each cell has two possible states, and two neighbors, to the left and right in the 1D arrangement. A cell and its two neighbors form a neighborhood of 3

* Corresponding author.
E-mail: lucian.petrica@upb.ro (Lucian Petrica)

cells, whose state is a collection of the states of the cells within the neighborhood. There are 2^3 = 8possible neighborhood states. A CA rule is a function which, given the neighborhood state as input, produces the next state of the center cell as output. There are then 2^8 = 256 possible rules, which each rule identified by the 8-bit number formed by concatenating the values of the next state of the center cell, for each of the 8 possible neighborhood states. Rule 30 is illustrated in **Error! Reference source not found.**, along with the states of a 31-cell CA in 16 consecutive time steps. Black squares indicate cells of state '1', and white squares indicate cells of state '0'.

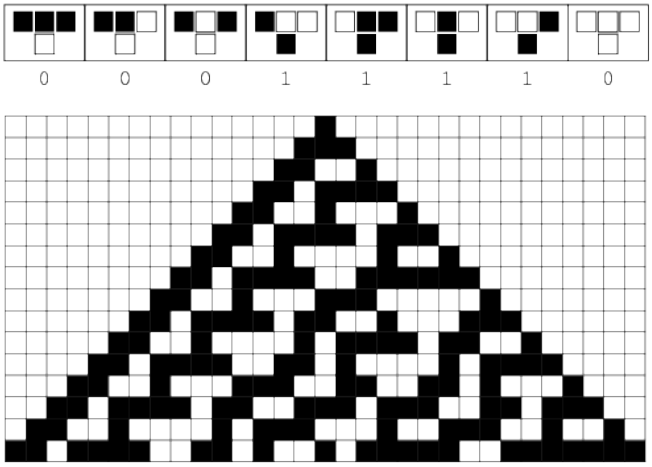

Figure 1 Output of 31-cell rule 30 CA

2.1 CA AS Source of randomness

In 1986, Stephen Wolfram first constructed PRNGs with cellular automata (Wolfram, Cryptography with cellular automata 1986) in order to demonstrate that 1D uniform CAs are able to generate random numbers of higher quality than most LFSR generators(Wolfram, Random sequence generation by cellular automata 1986). Multiple works have attempted to improve on the idea of 1D CA RNGs (De la Guia-Martinez 1997)(Kokolakis 1997)(Matsumoto 1998). Some researchers have experimented with 2D CA PRNGs(M. M. Tomassini 2000). Most of these studies focused on the CA cell itself and the CA rules.

Wolfram evaluated each of the 256 possible CA rules utilizing a suite of randomness tests and measured the potential of each rule to be used in a PRNG(Wolfram, Random sequence generation by cellular automata 1986). His study concluded with a taxonomy of CA rules, consisting of three classes defined by their potential for randomness, class I being the most predictable and class III being the most chaotic, out of which rule 30 best creates the appearance of chaotic behavior. One problem with rule 30 CA-based PRNGs is that two output streams generated with different initial conditions (seeds) present a strong correlation with each-other. Another problem is the short cycle period of the output, i.e., the output sequence is repeated after a small number of iterations.

In 1989, Hortensius et al. proposed the first non-uniform CA network for random number generation (Hortensius 1989). Instead applying the same rule to the entire CA, two or more rules would be utilized by different cells in the network. The authors evaluated a combination of cells with rule 90 and cells with rule 150 within the same CA. This configuration reduced the correlation between two output streams and the periodicity of the generated numbers.

In 1999, Tomassini et al. reanalyzed the potential for chaos of the CA rules in uniform networks (M. M. Tomassini 1999) and pointed out the importance of site spacing and time spacing in the attempt to reduce the correlation of two random bit streams generated with the same CA rule, but with different initial seeds. Unlike the traditional PRNG, with site spacing, the output random number is constructed by concatenating only some (notall) CA state bits. And, with time spacing, state bits are samples at certain moment of time, instead of being sampled in every CA time step. His evaluation concluded that, utilizing site and time spacing, rule 105 presents the most chaotic behavior, followed by 165, 90 and 150. He also introduced the idea that individual cells would be able to change the r rule at the end of a time step depending on its and its neighbors state (M. M. Tomassini 1999). This concept was termed cellular programming, and focused on the idea of self-evolving non-uniform CA networks, but on a local scale. From this idea, there have been many published researches that focus on the local evolution and control of a CA cell(Guan 2004)(Hoe 2012).Other attempts have been made at improving the CA RNGs through the use of rule evolution algorithms where the CA output is analyzed in its entirety and modifications areapplied to the rules of all or some of the cells (Stefan 1998)(Sipper 1997). This type of configuration holds promise for improved randomness properties, but has not been formally evaluated in the existing literature, until now.

Up to date, no cryptographically strong SPCA has been proposed or evaluated in previous work. Our work aims to construct and evaluate such a CA configuration, with focus on both its PRNG properties and the efficiency of its hardware implementation.

3. GLOBAL FEEDBACK SPCA

In this section we introduce the Global Feedback Self-Programmable CA (GF-SPCA), a minimalistic CA PRNG that is able to satisfy the quality requirements of modern statistical test suites. Our goal is to design and optimize the GF-SPCA with regard to quality but also FPGA resource utilization and energy efficiency.

3.1. Entropy feedback

The defining characteristic of GF-SPCA is a global feedback of information from the output of the PRNG into the CA rule selection decision. The feedback mechanism collects the generated output of the GF-SPCA PRNG at fixed intervals. This random data is utilized to select one or more of the CA cells as targets for rule update, and to select a new rule to apply to the target cells. Figure 2 presents the entropy feedback concept.

Global entropy feedback allows great flexibility with regard to the actual utilization of the output random data to modify the CA rules. However, it is not obvious whether assigning rules randomly to cells is beneficial

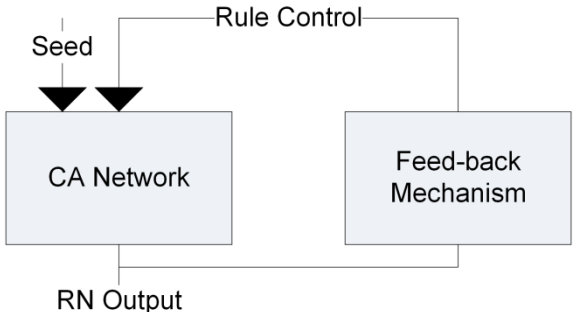

Figure 2 Globalentropy feedback SPCA

to the overall quality of a CA PRNG. Based on the studies conducted by Tomassini and Wolfram, most rules do not generate pseudo-random CA behaviour, therefore the possible set of new rules generated by the rule control mechanism must be carefully constrained.

Based on Tomassini's research(M. M. Tomassini 1999), rules 105, 165, 90 and 150are the only cryptographically strong CA rules. We therefore select these as the only candidates for a new rule. Consequently, the GF-SPCA feedback mechanism collects the output of the network during a time frame and based on it, selects which cells to have their rule changed and which of the four additive rules to replace the previous CA cell rule. Rule 105 is selected as the initial rule of the CA cells, because Tomassini's work proves this to be the strongest cryptographically.

Another important design parameter regards the selection of cells for rule update. Previous work on SPCA has utilized a rule update mechanism which modifies the rules of all cells at each update interval. While this approach has yielded quality CA-based PRNGs, the aggressive update schedule potentially reduces the period of the resulting PRNG. Work in (Hoe 2012) on hybrid SPCA-LFSR PRNGs indicate that the period of a PRNG may be extended by combining two cyclical processes, namely a SPCA and a LFSR. Through careful parameter selection, the period length of the resulting PRNG can be extended to the least common multiple of the period lengths of the SPCA and LFSR respectively. Based on this result, we choose to update a single CA cell rule in each GF-SPCA update interval, thereby extending as much as possible the cycle length of the rule update process, and therefore the cycle length of the GF-SPCA itself. The slower update process is not detrimental to output randomness, since a CA output is sufficiently random over short intervals of time.

3.2.Internal correlation minimization

Selecting the affected cell and the new rule based on the same set of generated random values leads to a strong correlation between the cell and the rule selection. To avoid this potential problem we implement two different processing algorithms for the rule and cell selection in an attempt to reduce the correlation.

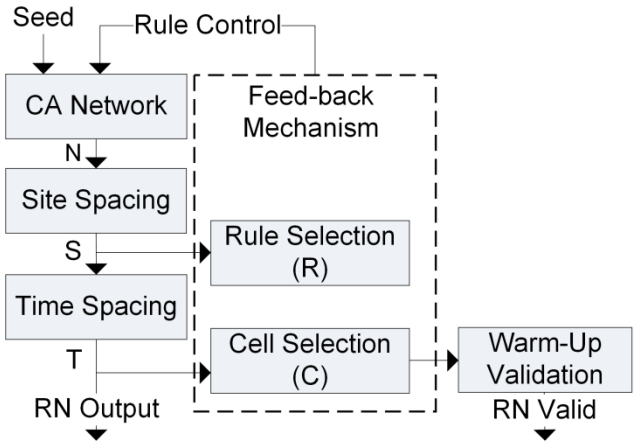

Figure 3 Correlation-minimized GF-SPCA structure

We also utilize both site and time spacing (Marco Tomassini, 1999).A site spacing mechanism is applied to the output of the CA network. This is followed by a time spacing mechanism. As illustrated in Figure 3, the feedback loop takes the site spacing output and uses it as an input for the new rule selection circuitry and the time spacing output and uses it as an input for the cell selection logic.

3.2.1. Site spacing

Figure 3 presents the high level block diagram of the feedback mechanism with site and time spacing application. From the output "N" of the cellular automaton, the site spacing mechanism selects and forwards only the outputs of cells located at set spatial intervals W_S. This means that for a site spacing value of W_S= 2, only the outputs of cells 0, 3, 6, 9, etc. are passed along. In this situation, for a 12-cell CA with 12 bits of state, only 4 random bits are generated. The output of the site spacing logic is noted "S".

$$W_N = (W_S + 1) * W_O \qquad (1)$$

If the system is required to generate a random number of a certain length W_O, with a set site spacing W_S, the CA number of cells W_N is defined in Equation (1). Site spacing increases the number of cells, increasing the CA size. Therefore, it is desirable to have a site spacing value as small as possible, but high enough to remove any unwanted correlation between consecutive bits in the output random number.

3.2.2. Time spacing

The time spacing logic receives as the input the output of the site spacing mechanism and forwards the input value to the output at set time intervals . This means that if the numbers S_0, S_1, S_2, S_3, S_4, S_5, etc. are inputs to the time spacing block, with a time spacing value of W_T = 1,only S0, S2 and S4 are selected.

Time spacing generates an obvious performance issue. Instead producing random numbers at each CA time step, we now produce at W_T+ 1time steps. The output of the time spacing mechanism is noted with "T". The time spacing parameter W_T is also desired to be as small as possible.

3.2.3. Rule selection

The rule selection mechanism collects the output of the site spacing and determines which rule, out of the four available, is to be selected. Hence, for the selection of the rule only two bits are required. These two bits, noted R_0 and R_1, are generated using the collected "randomness" of the site spacing output. At each CA time step, the rule selection logic outputs a new rule to be selected.

$$R_0 = \left(\sum_{i=0}^{\frac{W_O}{2}-1} S_i + R_0 \right) mod\ 2 \qquad (2)$$

$$R_1 = \left(\sum_{i=\frac{W_O}{2}}^{W_O-1} S_i + R_1 \right) mod\ 2 \qquad (3)$$

3.2.4. Cell selection

The cell selection block determines which cell's rule is to be modified. Changing a rule at each CA time interval is ineffective, since the cell selection mechanism does not have enough time to gather sufficient entropy and leads to a poor rule selection randomization. Changing the rule at random intervals is best, but because the block uses a single entropy source and both the change interval and the cell selection would be computed based on it, a strong correlation appears between the two.

$$C = \mathrm{T}\ mod W_N \qquad (4)$$

A fixed interval solution is a good compromise, with the interval length denoted W_R. When the time to change a rule is reached, the block checks the current time spacing output and, based on it, selects the cell with index "C" to have its rule changed with the one currently selected by the rule selection block. The rule change interval W_R should be greater than the time spacing interval W_T or else two consecutive changes could be applied to the same cell only because time spacing has not yet generated a number.

The time- and site-spacing mechanisms ensure a high enough randomization of the cell rules based only on the output of the network with minimal added complexity. An additional input to the rule selection block could be made so that the user can input extra random values in an attempt to increase the gathered entropy.

3.2.5 Warm-up

The cellular automaton network is initially a uniform one with rule 105 controlling all the cells. This uniformity means that until a certain number of rule changes have occurred, the quality of the numbers will strongly depend on the initial seed. A worst-case scenario is when the seed is 0, and the output numbers would be periodic. To ensure that this does not happen, all output numbers of the RNG are discarded until a certain number of rule changes have occurred. This is called the warm-up period and in order to control it, another block is added to the GF-SPCA structure.

All of the parameters described for the different GF-SPCA mechanisms (W_S, W_T, W_O, W_R) may be modified independently depending on the application requirements. Tomassini recommended in his work a site spacing of 1 or 2 and time spacing between 1 and 4. The optimal values for all the parameters are determined experimentally and presented in the following section.

4. EVALUATION METHODOLOGY

We evaluate the GF-SPCA PRNG family with regard to randomness and FPGA resource utilization, by performing design space exploration on the various configuration parameters of GF-SPCA. In this section we describe the evaluation quality metrics, design space exploration methodology, and software tools utilized.

4.1. PRNG quality

Various statistical tests can be applied to a generated random number sequence to evaluate it in comparison to a truly random sequence. Randomness is a probabilistic property; that is, the properties of a random sequence can be characterized and described in terms of probability. The outcome of statistical tests, when applied to a truly random sequence, are known apriori. A statistical test is formulated to verify the hypothesis that the sequence being tested is random. For each applied test, a decision or conclusion is derived that accepts or rejects the hypothesis, i.e., whether the generator is (or is not) producing random values, based on the sequence that was produced.

4.1.1. ENT Test suite

The ENT[†] test suite applies various tests to sequences of bytes stored in files and reports the results of those tests. The program is useful for evaluating PRNGs for encryption and statistical sampling applications, compression algorithms and other applications where the information density of a file is of interest. ENT is not a particularly stringent randomness test but is capable of delivering a quick estimate of the randomness of a sequence. In our evaluation, we utilize ENT to eliminate GF-SPCA configurations with particularly bad randomness early on in the evaluation. This reduces the time required for design space exploration, because more stringent and time-consuming tests are not performed on bad configurations.

4.1.2. NIST Test suite

The NIST (NIST 2010)statistical test suite evaluates the suitability of PRNGs for cryptographic applications, such as the generation of key material. Generators suitable for use in cryptographic applications

[†]http://www.fourmilab.ch/random

must meet stronger requirements than for other applications. The NIST Test Suite consists of 15 tests which focus on a variety of different non-randomness indicators that may exist in an input sequence.

4.2. FPGA Implementation

Our FPGA implementation of the GF-SPCA targets the Xilinx Zynq-7000 FPGA device and the ZedBoard[‡] development board based on the Zynq device. We selected the Zynq because it combines a FPGA and a dual-core ARM Cortex-A9 processor on a single chip, enabling an easy comparison of software and FPGA based PRNGs on the same processing device.

The GF-SPCA is described in parametric VHDL and synthesized with Xilinx ISE 14.7 for the target FPGA, for each configuration under evaluation. Resource utilization is extracted from the synthesis report in order to evaluate the effects of GF-SPCA parameters on the size of the circuit. We connect the FPGA-implemented GF-SPCA to the operating system and randomness evaluation software through Xillybus[§], a DMA-based data transport core which resides in the FPGA alongside the GF-SPCA. Xillybus enables software executing on the ARM processors to access data in a set of FPGA FIFOs through the Linux file I/O API. In our use-case, the FPGA FIFOs are filled with generated pseudo-random data by the GF-SPCA, and ENT and NIST execute on the ARM processors and read the data to evaluate its randomness.

We utilize PRNGs in previous work as a resource utilization comparison. The MT19937, known as Mersenne Twister(Konuma 2005) is a cryptographically strong LFSR-based PRNG which is freely accessible as VHDL code[**] and has been evaluated for FPGA implementation.MT19937 has a long period and has been extensively utilized in cryptography; it is the default Linux random number generator. Another LFSR-based PRNG, the LFSR-160, is proposed and evaluated in (Thomas, High quality uniform random number generation using LUT optimised state-transition matrices 2007). LUT-SR, a FPGA-optimized LFSR, is presented in (Thomas, The LUT-SR family of uniform random number generators for FPGA architectures 2013). There is no FPGA-based SPCA implementation in the latest FPGA technology; therefore no direct comparison may be made with previous work on SPCA, with regard to resource utilization. All of the above-mentioned PRNGs generate 32 bits per cycle, therefore in our evaluation we restrict W_O to 32.

4.3. Design space exploration

We performed an evaluation of GF-SPCA with the following parameter combinations:

- time spacing W_T varying between 0 and 5
- site spacing W_S of 0 and 1
- warm-upperiod varying between 0 and 50 with an increment of 5
- rule change interval W_R between1 and 5
- bits per cycle W_O always 32.

Each GF-SPCA configuration is synthesized, connected to a Xillybus core, and loaded into the FPGA.We subsequently read through Xillybus10 sequences of 1 million 32-bit GF-SPCA generated random numbers, which are utilized for randomness evaluation with ENT and NIST.ENT is executed first, in order to quickly eliminate the GF-SPCA configuration under analysis if it does not produce quality random numbers. If the configuration passes ENT, it is then subjected to the more stringent NIST tests, which determine if the GF-SPCA configuration is cryptographically strong. The ENT and NIST results are recorded, along with FPGA resource utilization, to enable comparative analysis of GF-SPCA configurations.

5.EVALUATION RESULTS

5.1 ENT and NIST Results

Table 1 presents the configurations which obtained the best results in ENT. The most important ENT test is entropy, which defines the amount of information contained by the bit stream, counted in bits per byte. The ideal value is 8 bits/byte for a completely random source of information. The best GF-SPCA configurations score as high as 7.999988 bits/byte and generate practically incompressible output data. The listed configurations are capable of estimating Pi with the Monte-Carlo method to a high degree of accuracy.

Of note is the fact that some GF-SPCA configurations, e.g. (1, 5, 5), perform rather badly. The (1, 5, 5), not listed in the table, only achieves 4 bits/byte entropy and a compression factor of 46. Any GF-SPCA implementation should only utilize one of the listed best configurations.

[‡] www.zedboard.org
[§] www.xillybus.com
[**] http://www.ht-lab.com/freecores/mt32/mersenne.html

All GF-SPCA configurations which passed the ENT tests were subjected to the NIST tests. Of the 660 configurations under analysis, 593 passed both ENT and NIST, a relatively large number. Of the parameters under analysis, the warm-up period had the least effect on the ENT and NIST results. We could not discern a connection between any of the remaining parameters and the quality of the produced random number sequence.

Table 1ENT results of best GF-SPCA configurations

GF-SPCA Configuration (W_S, W_T, W_R)	Entropy [Bits/byte]	Maximum Compression	Arithmetic Mean	Monte-Carlo Pi Deviation	Serial Correlation Coefficient
(1, 3, 5)	7.999988	0	127.4970	0.05	$556*10^{-6}$
(1, 4, 2)	7.999988	0	127.4860	0	$87*10^{-6}$
(0, 1, 1)	7.999983	0	127.4999	0.07	$124*10^{-6}$
(0, 3, 4)	7.999985	0	127.5000	0.01	$-55*10^{-6}$
(0, 4, 5)	7.999985	0	127.5002	0.02	$80*10^{-6}$
(1, 2, 1)	7.999985	0	127.4999	0.01	$-204*10^{-6}$
(1, 2, 2)	7.999986	0	127.4999	0.02	$-113*10^{-6}$
(1, 4, 1)	7.999984	0	127.5002	0.01	$-178*10^{-6}$
(0, 1, 3)	7.999985	0	127.4994	0	$20*10^{-6}$
(0, 4, 2)	7.999986	0	127.5304	0.02	$12*10^{-6}$
(1, 3, 3)	7.999983	0	127.5199	0.03	$12*10^{-6}$
(1, 4, 3)	7.999983	0	127.4725	0.04	$12*10^{-6}$
(1, 5, 2)	7.999984	0	127.5014	0.01	$12*10^{-6}$
(1, 5, 4)	7.999983	0	127.4991	0	$12*10^{-6}$

5.2. Effect of warm-up

In Figure 4 and Figure 5 we present the visual representation of the output of a GF-SPCA configuration with and without the warm-up mechanism respectively. It is evident that without the warm-up mechanism the output presents discernable patterns in the first few generated numbers. Soon afterwards, the rules for different cells are changed and those patterns disappear. The initial patterned output is a potential weakness of the GF-SPCA, which is eliminated by the warm-up mechanism by invalidating the random number output until a certain amount of time passes. In our experiments, 50 clock cycles is sufficient for the initial patterns to dissipate.

Figure 4 Initial GF-SPCA Output (no Warm-Up)

Figure 5 Initial GF-SPCA Output (with Warm-Up)

5.3. FPGA Resource utilization

Table 2 lists the top ten GF-SPCA configurations with regard to FPGA resource utilization, expressed as the number of Look-Up Tables (LUTs) and Flip-Flops. All configurations passed the ENT and NIST tests.

Immediately evident is that all configurations have W_S equal to zero, which is expected. The size of GF-SPCA is proportional to W_S, which determines the size of the underlying CA network. The value of other GF-SPCA parameters does not significantly affect the FPGA resource utilization.

Table 2FPGA resource utilization

GF-SPCA Configuration (W_S, W_T, W_R)	LUTs	Flip-Flops
(0,0,3)	272	807
(0,1,3)	272	808
(0,5,5)	272	809
(0,0,5)	273	808
(0,0,7)	273	808
(0,1,2)	273	809
(0,1,5)	273	809
(0,7,3)	273	813
(0,5,3)	274	813
(0,2,5)	275	813

Table 3 presents a comparison of the GF-SPCA resource utilization against PRNGs in previous work. The GF-SPCA compares favorably to MT19937 and LFSR-160, but is not as resource-efficient as LUT-SR. This may be explained by the fact that LUT-SR is FPGA-optimized, i.e., designed to take advantage of specific architectural features of the target FPGA. Conversely, the presented GF-SPCA implementation is generic, and not optimized for FPGA specifically.

Table 3 Comparison of PRNG resource utilization

PRNG	LUTs	Flip-Flops	RAMs
MT19937 (Konuma 2005)	278	0	2
LFSR-160(Thomas 2007)	448	314	0
LUT-SR (Thomas 2013)	64	64	0
GF-SPCA	272	807	0

6. CONCLUSION

We have designed, implemented, optimized and evaluated the GF-SPCA, a cellular automaton pseudo-random number generator. We have implemented a functional, cryptographic class random number generator, the first to be proposed and implemented based on the self-programmable cellular automaton architecture. The design is comparable in area with the current implementations of other types of hardware random number generators, although FPGA-optimized LFSR-based PRNGs such as the LUT-SR achieve lower resource utilization.

For future development, a more in-depth theoretical analysis of the effect of all the entropy enhancing mechanisms (site-spacing, time-spacing, warm-up period, and rule-change interval) needs to be conducted in order to provide guidance on selecting the best parameter combinations for maximum output randomness. Further optimization with regard to FPGA resource utilization is also a possibility. Our present implementation of the GF-SPCA is not FPGA-optimized, and we expect that FPGA resource efficiency may be greatly increased by making use of FPGA architectural features.

ACKNOWLEDGMENTS

The work has been funded in part by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132397.

REFERENCES

De la Guia-Martinez, D., and Amparo Fúster-Sabater. "Cryptographic design based on cellular automata." *IEEE International Symposium on Information Theory.* IEEE, 1997. 180.

Golomb, Solomon W. *Shift register sequences.* Aegean Park Press, 1982.

Guan, Sheng-Uei, and Shu Zhang. "Pseudorandom number generation based on controllable cellular automata." *Future Generation Computer Systems* 20, no. 4 (2004): 627-641.

Hoe, David HK, Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and Mukul V. Shirvaikar. "Cellular automata-based parallel random number generators using FPGAs." *International Journal of Reconfigurable Computing*, 2012: 4.

Hortensius, Peter D., Robert D. McLeod, and Howard C. Card. "Parallel random number generation for VLSI systems using cellular automata ." *IEEE Transactions on Computers* 38, no. 10 (1989): 1466-1473.

Kokolakis, I., I. Andreadis, and Ph Tsalides. "Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators." *Microprocessors and Microsystems* 20, no. 10 (1997): 643-658.

Konuma, Shiro, and Shuichi Ichikawa. "Design and evaluation of hardware pseudo-random number generator MT19937." *IEICE transactions on information and systems* 88, no. 12 (2005): 2876-2879.

Matsumoto, Makoto. "Simple cellular automata as pseudorandom m-sequence generators for built-in self-test." *ACM Transactions on Modeling and Computer Simulation* 8, no. 1 (1998): 31-42.

NIST. "Special publication 800-22: A statistical test suite for random and pseudorandom number generators for cryptographic applications." 2010.

Sipper, Moshe. *Evolution of parallel cellular machines.* Heidelberg: Springer, 1997.

Stefan, Gheorghe. "Looking for the lost noise." *IEEE International Conference on Semiconductors.* Sinaia, 1998. 6-10.

Thomas, David B., and Wayne Luk. "High quality uniform random number generation using LUT optimised state-transition matrices." *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* 47, no. 1 (2007): 77-92.

Thomas, David B., and Wayne Luk. "The LUT-SR family of uniform random number generators for FPGA architectures." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, no. 4 (2013): 761-770.

Tomassini, Marco, Moshe Sipper, and Mathieu Perrenoud. "On the generation of high-quality random numbers by two-dimensional cellular automata." *IEEE Transactions on Computers* 49, no. 10 (2000): 1146-1151.

Tomassini, Marco, Moshe Sipper, Mosé Zolla, and Mathieu Perrenoud. "Generating high-quality random numbers in parallel by cellular automata." *Future Generation Computer Systems* 16, no. 2 (1999): 291-305.

Walker, John. *Pseudorandom Number Sequence Test Program.* 2015. http://www.fourmilab.ch/random.

Wolfram, Stephen. "Cryptography with cellular automata." *Advances in Cryptology—CRYPTO'85 Proceedings.* Springer Berlin Heidelberg, 1986. 429-432.

Wolfram, Stephen. "Random sequence generation by cellular automata." *Advances in applied mathematics* 7, no. 2 (1986): 123-169.

Wolfram, Stephen. "Statistical mechanics of cellular automata." *Reviews of modern physics* 55, no. 3 (1983): 601.