Cellular Automaton pRNG with a Global Loop for Non-Uniform Rule Control

Alexandru Gheolbanoiu, Dan Mocanu, Radu Hobincu, and Lucian Petrica
Politehnica University of Bucharest
alexandru.gheolbanoiu@arh.pub.ro

Abstract—Pseudo-random number generation is an important ingredient of many cryptography applications, as well as scientific applications based on statistical sampling, e.g., Monte Carlo methods. Several methods have been proposed for generating pseudo-random numbers, but these are generally either (i) based on cryptographic cypher algorithms and expensive to implement in hardware (e.g., large silicon area, low energy efficiency) or (ii) based on linear-feedback shift registers, which can be efficiently implemented in hardware but are not sufficiently random. This paper presents a pseudo-random number generator which utilzes a configurable cellular automaton network which generates the output stream of numbers, and a feedback loop which monitors the randomness properties of the output stream and adjusts the parameters of the network in order to optimize its cryptographic performance. We demonstrate that introducing this additional feedback loop increases the overall entropy of the system, improving the quality of the pseudo-random sequence over other cellular implementations or LFSRs. We also analyze the effect of multiple configurations of the proposed generator architecture. We evaluate the generator against several standard benchmarks to illustrate its performance and we also provide an evaluation of its hardware implementation which demonstrates comparable implementation efficiency to LFSRs.

Keywords—cellular automaton, random number generator, LFSR, feedback, FPGA.

I. 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 are statistical simulation algorithms based on the Monte-Carlo method and even 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 because one cannot predict its output under any circumstances, and once a sequence of such numbers has been generated, someone cannot predict if and when it will be generated again [1]. True RNGs are difficult to implement in a digital system because such a system is inherently deterministic, but physical processes like the initialization of random access memories may be utilized for RNG purposes [cite something here].

The easier approach is to combine different algorithms and mathematical functions for the purpose of generating numbers that create the appearance of randomness. These generators are called Pseudo Random Number Generators (PRNG). The most popular such PRNGs are the Linear Feedback Shift Registers (LFSR) due to their efficient hardware implementation [2]. However, these generators present an unwanted property: periodicity. Thus, after a sequence of N numbers, where N is the repetition period of the generator, the exact same sequence will start being generated again. Attempts to increase the period N are made through the use of Non-Linear Feedback Shift Register (NLFSR) [3] [4] [5]. The research on these generators is still ongoing and the construction of large NLFSRs with guaranteed long periods remains an open problem.

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 and the states of other cells in a neighbourhood. A CA may be uniform, meaning all cells have the same rule, or non-uniform, with different cells having different rules. The one-dimensional (1D) and two-dimensional (2D) automata have seen more research interest, with the most well-known cellular automaton being Conway's Game of Life [6], a 2D automaton which has been proven to be Turing complete [7]. Much of the research on CA-based RNG has focused on the identification of CA rules which lead to good randomness properties. Wolfram in particular has performed extensive analysis on 1D cellular automata rules, and for the remainder of this work we will utilize the rule naming conventions defined in [8].

In this work, we attempt to create a one-dimensional CA RNG with good randomness properties by analyzing the cell network in its entirety. Instead of integrating different algorithms and hardwired mechanisms within the cells, i.e., searching for the perfect (and most likely complex) CA rule, we maintain the simplicity and flexibility of the cell in order to facilitate hardware implementation. With the addition of a feedback mechanism, which we call the global loop and which is able to reconfigure the cell rules, RNG properties are improved, as demonstrated with industry-standard randomness benchmarks.

This paper is structured as follows. Section II will present existing work on PRNGs, with focus on methods based on cellular automata. Section III introduces the proposed system architecture and Section IV describes its implementation. Section V presents the evaluation methodology and results, while in Section VI we make concluding remarks and outline areas for future research.

ISBN: 978-1-61804-237-8

II. Cellular Automaton RNGs

In 1986, Stephen Wolfram first tried to construct RNGs through the use of Cellular Automata [9]. His main focus was to demonstrate that one-dimensional uniform CA networks are able to generate random numbers of higher quality than most LFSR generators [10]. Since then, multiple works have attempted to improve on the idea of 1D CA RNGs [11] [12] [13]. Some attempts have even been made on the construction of 2D CA RNGs [14]. Most of these studies focused more on the CA cell itself, the rules to be used and the evolution of set cells, resulting in complex circuits which, in most cases, are impractical.

Wolframs work focused on analyzing the potential of different CA rules for random number generation with the use of a one-dimensional uniform CA network. In order to do this, he applied, in turn, each of the 256 possible rules to all the cells forming the one-dimensional network and, using a suite of randomness tests, measured the potential of each rule to be used in a RNG [10]. 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. The first problem that remained was that two output streams generated through the use of the same CA rule, but with different initial seeds, presented a strong correlation with each-other in both time and space. The second problem was the periodicity that the generated numbers presented.

In 1989, Hortensius et al. proposed the first non-uniform CA network for random number generation [15]. Instead of having the entire network of cells apply the same rule, two or more rules would be utilized by different cells in the network. In his research, he evaluated a combination of cells with rule 90 and cells with rule 150 within the same network. 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, but this time, through the use of the DIEHARD evaluation suite [16]. He also 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 RNG, with site spacing, not all bits generated at one moment by the network are utilized for the output number. And, with time spacing, only bits generated at a certain moment of time are used for the output number. 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 can improve their randomness quality through genetic algorithms. Each cell would be able to change its rule at the end of a generation cycle depending on the entropy it and its neighbors presented [16]. 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 [17] [18].

Other attempts have been made at improving the CA RNGs through the use of traditional genetic evolution algorithms where the network is analyzed in its entirety and modifications are applied to all the cells depending on the results [19] [20]. This practically steps away from the CA network itself, ignores the local loops made between the cells, but applies a global loop, whereby the output of the network is analyzed and, based on different genetic algorithms, modifications are made to the entire network. This type of configuration holds promise for improved randomness properties, but has not been formally evaluated in the existing literature, until now. Our work aims to construct and evaluate such a CA configuration, with focus on both its RNG properties and the efficiency of its hardware implementation.

III. Global Feedback CA RNG

Taking guidance from previous work, we propose to construct a minimalistic CA RNG which is able to satisfy most of the quality requirements that are now placed on RNGs for cryptographic use. Our goal is to design and formally evaluate a system which is capable of outputting a high entropy sequence of numbers whilst maintaining the hardware resource usage to a minimum.

The principal challenge towards the intended goal is how to prevent the CA RNG from remaining stuck in a steady state or in short cycles. This requirement may be achieved in one of two ways, either by implementing a generic algorithm in each cell for rule updates, or by implementing a global feedback loop which is be able to control the entire network by updating rules in individual cells. Because we desire to obtain a small structure, the area and complexity of individual cells must be minimized. Therefore, we choose to implement a global feedback loop to control the rules within the CA network based on certain criteria.

Another design choice is whether he global loop can appoint only one rule to the whole network, resulting in a uniform network, or appoint several different rules to different parts of the network, resulting in a non-uniform network. In order to fully take advantage of a global loop, and in following with the findings of previous work on non-uniform CA, we elect to implement a feedback loop which is able to control the rule for each individual cell within the network. Therefore, the CA network will be generally non-uniform and the cells will be capable of retaining rules applied to them by the external loop.

As previously stated, the global feedback loop will collect the output of the network, analyze its properties and apply the required modifications, i.e., change the rules of different cells individually. This process is illustrated in Figure 1. An important design consideration is the nature of the feedback. Evaluations were carried out on negative feed-back mechanisms, e.g., an external system would calculate the entropy of the network and, in case is decreasing, modify the cells rules in an attempt to revitalize the RNG, but these proved incapable of ensuring good randomness. Similar results were obtained

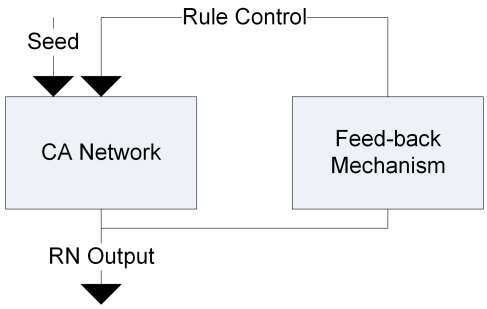

Fig. 1: CA with Global Feedback Loop

utilizing a toggle checker system to count the transitions of each bit of the CA network output within a time frame and, should a bit get stale, call for a rule change. Therefore, negative feedback was discounted and the focus was switched to positive feed-back mechanism, whereby the mechanism collects the generated output of the CA network and, at fixed intervals of time and through the use of the entropy collected, applies a new rule to a randomly selected CA cell.

Another important design consideration is the selection of the new rule to apply to a given cell. Randomly generating a new rule between 0 and 255 was not expected to yield good results, since most CA rules do not exhibit good randomness properties. Additionally, this architecture requires that each cell contain the circuits required to implement all 256 rules, leading to large area footprint and circuit complexity. Therefore, our proposed mechanism chooses between a fixed set of rules. Based on the research done by Tomassini [16], we selected rules 105, 165, 90 and 150 to be the only candidates for a new rule. We decided for all cells to follow rule 105 initially, not only because Tomassinis work proved rule 105 to provide the best randomness, but also because it allows CA oscillation even when the initial state of cell, i.e., the seed, is all zeroes or all ones. For example, if the seed is 0 the CA will oscillate between 0x0000...0000 and 0xFFFFFFFF until the feedback mechanism applies the first rule change, and will subsequently exhibit random output.

Finally, we add to the design a site and time spacing mechanism, as illustrated in Figure 2 in order to avoid the output correlation problems which usually occur with CA RNGs. From the output of the entire CA network, denoted N, of length L_N, the site spacing mechanism selects and passes on only the outputs of cells located at set spatial intervals. The length of the interval is denoted S_s. Consequently, for a site spacing value of 2, only the outputs of cells 0, 3, 6, 9, etc. are utilized for generating an output number. We denote S the output after site spacing, of length L_S. Conversely, if the system is required to generate random numbers of a certain length L_S, with a set site spacing, the required number of CA cells is given by Equation 1. The higher number of cells required increases the CA RNG circuit size, and it is desired to have a site spacing value as small as possible.

$$L_N = (S_s + 1) * L_S \tag{1}$$

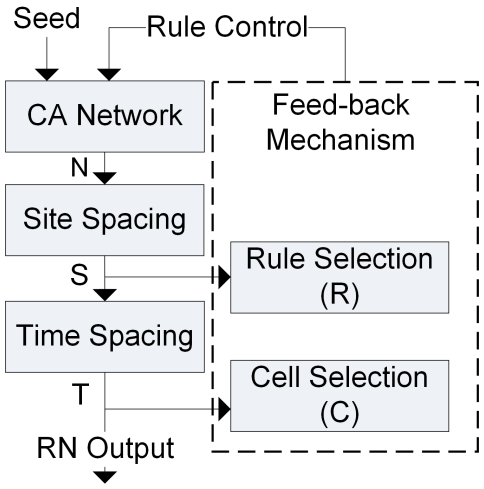

Fig. 2: CA with Site and Time Spacing

The time spacing mechanism forwards, at regular time intervals, the output values of the site spacing mechanism to the output of the CA RNG. For example, if the numbers S_0, S_1, S_2, S_3, S_4, S_5, etc. are output by the site spacing mechanism, with a time spacing value of 1, only S_0, S_2, S_4, etc. are selected. This mechanism decreases the correlation of consecutive output words but reduces the RNG throughput. The output of the time spacing mechanism is denoted T. The time spacing value T_s is also desired to be as small as possible, in order to reduce the number of circuit operations per output word, and therefore increase the circuit energy efficiency.

In order to select one of the four available CA cell rules, the rule selection mechanism collects the output of the site spacing and determines the selection bits R_0 and R_1 according to Equations 2 and 3. These two bits will be generated using the collected randomness of the site spacing output.

$$R_0 = (\sum_{i=0}^{L_{RN}/2-1} S_i + R_0) \mod 2 \tag{2}$$

$$R_1 = (\sum_{i=L_{RN}/2}^{L_{RN}-1} S_i + R_1) \mod 2 \tag{3}$$

At each CA network generation cycle, a new value is output by the site spacing circuit and a new rule is selected. The cell selection block determines when and which cell is to be modified. There are three possibile strategies for the timing of rule changes, (i) at each generation cycle, (ii) at a fixed interval or (iii) at a random interval. Strategy (i) is expected to be inffective, since the cell selection mechanism does not have enough time to gather entropy information, which leads to a poor rule selection randomization. Strategy (iii) is discarded because the system has a single entropy source and both the change interval and cell selection would be calculated based on it, resulting in a strong correlation between the two. Hence, we opted for strategy (ii), updating the cell rules ar fixed intervals of time. When the time to change a rule is reached, the block

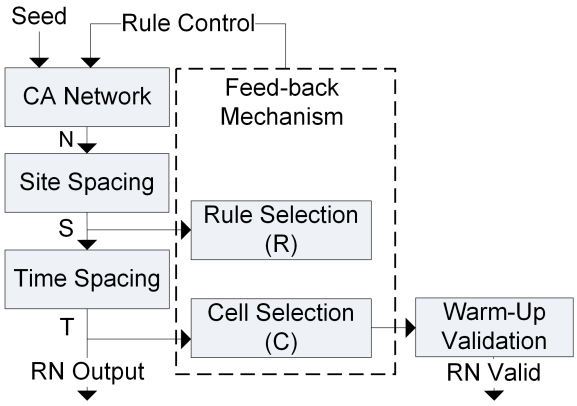

Fig. 3: Final CA RNG Structure

reads the current time spacing output and, based on its value, selects a cell C to have its rule changed with the one currently selected by the rule selection block. This strategy creates a new condition: the rule change interval needs to be at least greater than the time spacing interval, else two consecutive changes may be applied to the same cell only because time spacing has not yet generated a new number.

$$C = T_s \mod L_N \tag{4}$$

The two mechanisms described above ensure a good randomization of the cell rules based only on the output of the network and with minimal circuit complexity and size. We provision an additional external connection to the rule selection block so that the user may input extra random values in order to increase the gathered entropy. The CA network is initially uniform, with rule 105 controlling all the cells, but becomes non-uniform after the first interval of the rule change mechanism in the feedback loop. Consequently, the quality of the first generated numbers will strongly depend on the initial seed. To remove this weakeness in the RNG design, all output numbers of the RNG are discarded until a certain number of rule changes has have occurred. This is called the warm-up period and in order to control it, we introduce an additional block into the design. The warm-up period value represents the number of rule changes that must occur before the output numbers are considered valid. Notably, another reason for not using a random interval for rule changing is that the length of the warm-up period could not be predicted and may in some cases become very large.

Most parameters described above as being part of the different mechanisms (time spacing, rule change interval and warm-up period) are controllable by the user depending on the required performance. The generated number length and site spacing values are constant because they directly impact the number of cells used and the system structure. Tomassini recommends in his work a site spacing of 1 or 2 and a time spacing between 1 and 4. The optimal values for the rest of the parameters are determined after implementation and analysis of the RNG.

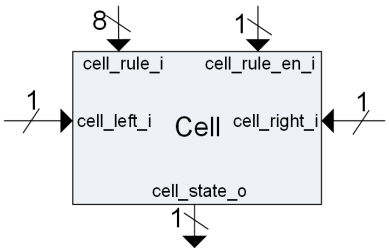

Fig. 4: CA Cell

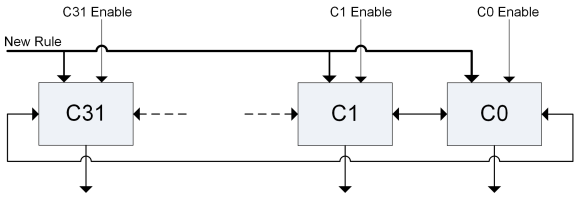

Fig. 5: CA with null site spacing

IV. Implementation

In order to enable the evaluation of the proposed CA RNG hardware structure, we implemented it in VHDL. In our proposed CA RNG architecture, in order for the cells to be able to retain the assigned rule and the current state, each requires 9 flip-flops, 1 for the current state and 8 for the rule storage. Because each cell may have only one out of four rules, two flip-flops for rule storage would be enough to retain a corresponding encoded value. However, we desired to implement a more flexible structure which allows us to experiment with multiple feed-back rule control mechanisms. Hence, each cell requires an additional 8 bit port for the new rule input and an update enable 1 bit port. A diagram of the cell, with all inputs and outputs, is presented in Figure 4.

The CA cells are arranged as a 1D network with its extremity cells sharing a connection, as illustrated in Figure 5. As mentioned in Section III, the number of required cells is given by the site spacing value and the generated number length. For our analysis, we select a length of 32 bits and, with a site spacing of 0 or 1, we require 32 or 64 cells within the network. In order to reduce wiring fan-out issues, new rules are transmitted via a common 8 bit bus to all the cells, while each cell has an individual enable signal.

The site spacing mechanism is implemented by connecting only the appropriate state outputs of the network to the output. The time spacing mechanism contains a counter-based timer which, upon reaching the selected time spacing interval, signals a buffer to store the output of the site spacing mechanism. The output of the buffer is connected directly to the output of the RNG. The cell selection block and warm-up validation block also consist of counter-based timers which signal when the rule change takes place and when the warm-up is done. The rule selection mechanism is illustrated in Figure 6 and consists of a memory for the two rule selection bits, i.e., 2 flip-flops, which are updated according to Equations 2 and 3 at each generation cycle.

ISBN: 978-1-61804-237-8

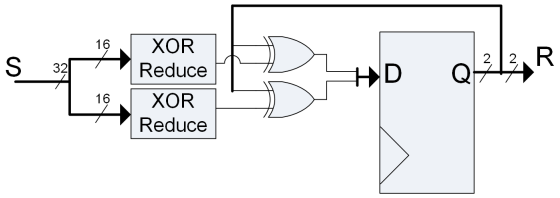

Fig. 6: Feed-Back Rule Selection

The implementation is parametric, with 4 architectural parameters: site spacing, time spacing, warm-up period and rule change interval. We encoded these configurations with a unique name containing all the parameters: S[site spacing]T[time spacing]R[rule change interval]W[warm-up period]. For example S1T3R5W50 is the configuration with site spacing 1, time spacing 3, rule change interval 5, and the warm-up period is 50 rule change intervals.

V. Evaluation

This section presents an evaluation of the proposed CA RNG structure with regard to randomness properties and the efficiency of its hardware implementation on FPGA, i.e., area and maximum frequency.

A. Methodology

We targeted the Xilinx Virtex-6 FPGA architecture for the evaluation of the implementation efficiency, and utilize LUT count and maximum frequency as metrics. To determine whether the system exhibits good randomness properties, we utilize three popular RNG evaluation suites, namely the ENT [21], DIEHARD [22] and NIST [23] suites. Our intention is to verify that the RNG performs well on all the selected suites, of which ENT is the least demanding and the NIST, issued by the foremost authority on public information security in the United States, is the most difficult.

Simulations are performed with site spacing 0 and 1, time spacing 0 to 5, warm up period of 50 and rule change interval of 1 to 5. Utilizing these parameter values and the associated simulation environment we obtained a set of 30 sequences of random numbers, each generated by the CA RNG with a specific configuration. Site spacing beyond 1 is not evaluated because the hardware implementation is expected to become unfeasably large. Of these 30 configurations, we keep only those that do not exhibit short cycles, i.e., an output value is not observed more than once in every seven consecutive output words.

The first evaluation is performed on ENT, which runs 7 tests to help discern the quality of the random sequence. These tests are entropy, optimum compression, chi square distribution, arithmetic mean, Monte Carlo value for Pi, and serial correlation coefficient. ENT is the only benchmark which outputs a set of absolute results for the 7 tests it runs. Although it is not generally regarded as the most relevant benchmark for random number generators, the fact that it outputs absolute values allows us to utilize the results for selecting a number of configurations to go forward. We select the best performing configurations for each test, which continue to the DIEHARD and NIST statistical benchmarks.

DIEHARD contains 12 statistical tests that output a p-value, which should be uniform on [0,1) if the input file contains truly independent random bits. A p-value of 1 or 0 means the input sequence has failed the test. After validating the remaining CA RNG configuraton with DIEHARD we continue by running the NIST Suite. This evaluation suite has been developed by the Random Number Generation Technical Working Group (RNG-TWG) between 1997 and 2010 as a benchmark for RNGs and PRNGs used in cryptographic applications. NIST contains 15 tests and, similar to DIEHARD, outputs a p-value that determines if the input sequence has passed or failed the test.

Finally, we evaluate the remaining configurations on TestU01 [24], a benchmark consisting of four sub-tests. We remove from our initial set of CA RNG configuration those that have failed one of the randomness benchmarks, and evaluate the remaining configurations for FPGA implementation efficiency. As target FPGA architectures, we select Xilinx Spartan-3 and Virtex-6. Spartan-3 is selected for direct comparison to previous work on FPGA random number generation in [25], while Virtex-6 is a more modern architecture.

TABLE I: Inter-stream correlation

Configuration	Seed 1	Seed 2	Correlation
S1T2RxW50	Binary '0'	Binary '1'	-0.0038
S1T2RxW50	Pattern 0xA	Pattern 0x5	0.001747
S1T3RxW50	Binary '0'	Binary '1'	0.000995
S1T3RxW50	Pattern 0xA	Pattern 0x5	-0.005248

B. Results

The initial short cycle evaluation results in the elimination of all configurations with null site spacing, therefore leaving only 15 configurations for further analysis. The ENT evaluation does not further eliminate any of the remaining configurations, as all exhibit good performance on the ENT banchmarks. The NIST benchmark passes on all remaining configurations. DIEHARD fails on all configurations with time spacing smaller than 2, therefore only 10 out of the initial 30 configurations are selected for evaluation with TestU01. Of these, all except S1T3R4W50 pass the randomness test.

We also analyzed the inter-stream correlation of the winning configurations, which is the correlation between streams generated with the same configuration but with different seeds. Ideally, output streams from different seeds are completely uncorrelated. The correlation evaluations were performed between two pairs of seeds, consisting of the binary representation of decimal values 0 and 1, and the repeating patterns of 0xA and 0x5 respectively. The calculated correlation is a number between -1 and 1, ideally 0. The correlation between the streams generated by the seeds are presented in Table I. All configurations with the same rule change interval performed identically and were therefore compressed in the same table entry.

ISBN: 978-1-61804-237-8

TABLE II: FPGA Implementation

Configuration	Spartan-3 LUT/FF	Spartan-3 F_{max}	Virtex-6 LUT/FF	Virtex-6 F_{max}
S1T2R2W50	358/380	177	237/227	641
S1T2R5W50	358/380	177	241/231	641
[25]	307/202	181	-	-

Finally, we synthesized the circuit for the target FPGA architecture of Xilinx Spartan-3 and Virtex-6. Table II gives an overview of the best and worst implementation results of the evaluated configurations, set against implementation results from previous work in Thomas et al. [25]. From previous work we selected the smallest implementation which passed the DIEHARD and Crush benchmarks, since Crush is a part of TestU01. For all remaining CA configurations, the theoretical maximum frequency is calculated at 600 MHz, and estimated area is similar. It must be noted that, while Spartan-3 results are comparable, our work is optimized for Virtex-6 and therefore Spartan-3 performance may suffer.

VI. Conclusion and Future Work

We have presented a pseudo-random number generator consisting of a one-dimensional cellular automaton and a feedback loop which monitors the CA outputs and modifies the CA rules at set time intervals in order to improve the randomness properties of the RNG. The generator was designed with hardware efficiency in mind, and the resulting structure is capable of passing all the selected randomness benchmarks, while also occupying very little area when implemented in a modern FPGA and is capable of operating at a high frequency of over 600 MHz.

Future work will focus on the continued analysis of the randomness properties of the proposed CA RNG architecture, and on comparisons to other methods of random number generation, with regard to both quality of output stream and hardware implementation efficiency. In this work we have explored a small number of the possible configurations of the CA RNG architecture, and future work will also concentrate on expanding the analysis to a larger number of configurations.

Acknowledgment

Part of this work was carried out with funding and support from POSDRU/159/1.5/S/132397 ExcelDoc postdoctoral program.

References

[1] S. Srinivasan, S. Mathew, R. Ramanarayanan, F. Sheikh, M. Anders, H. Kaul, V. Erraguntla, R. Krishnamurthy, and G. Taylor, "2.4 ghz 7mw all-digital pvt-variation tolerant true random number generator in 45nm cmos," in *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*. IEEE, 2010, pp. 203–204.

[2] S. W. Golomb, L. R. Welch, R. M. Goldstein, and A. W. Hales, *Shift register sequences*. Aegean Park Press Laguna Hills, CA, 1982, vol. 78.

[3] E. Dubrova, "A list of maximum period nlfsrs." *IACR Cryptology ePrint Archive*, vol. 2012, p. 166, 2012.

[4] R. Gottfert and B. M. Gammel, "On the frame length of achterbahn-128/80," in *Information Theory for Wireless Networks, 2007 IEEE Information Theory Workshop on*. IEEE, 2007, pp. 1–5.

[5] B. Gammel, R. Göttfert, and O. Kniffler, "Achterbahn-128/80: Design and analysis," in *ECRYPT Network of Excellence-SASC Workshop Record*, 2007, pp. 152–165.

[6] J. Conway, "The game of life," *Scientific American*, vol. 223, no. 4, p. 4, 1970.

[7] P. Rendell, "A universal turing machine in conway's game of life," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE, 2011, pp. 764–772.

[8] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of modern physics*, vol. 55, no. 3, p. 601, 1983.

[9] ——, "Cryptography with cellular automata," in *Advances in Cryptology CRYPTO85 Proceedings*. Springer, 1986, pp. 429–432.

[10] ——, "Random sequence generation by cellular automata," *Advances in applied mathematics*, vol. 7, no. 2, pp. 123–169, 1986.

[11] D. De la Guia-Martinez and A. Fuster-Sabater, "Cryptographic design based on cellular automata," in *Information Theory. 1997. Proceedings., 1997 IEEE International Symposium on*. IEEE, 1997, p. 180.

[12] I. Kokolakis, I. Andreadis, and P. Tsalides, "Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators," *Microprocessors and Microsystems*, vol. 20, no. 10, pp. 643–658, 1997.

[13] M. Matsumoto, "Simple cellular automata as pseudorandom m-sequence generators for built-in self-test," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 31–42, 1998.

[14] M. Tomassini, M. Sipper, and M. Perrenoud, "On the generation of high-quality random numbers by two-dimensional cellular automata," *Computers, IEEE Transactions on*, vol. 49, no. 10, pp. 1146–1151, 2000.

[15] P. D. Hortensius, R. D. McLeod, and H. C. Card, "Parallel random number generation for vlsi systems using cellular automata," *Computers, IEEE Transactions on*, vol. 38, no. 10, pp. 1466–1473, 1989.

[16] M. Tomassini, M. Sipper, M. Zolla, and M. Perrenoud, "Generating high-quality random numbers in parallel by cellular automata," *Future Generation Computer Systems*, vol. 16, no. 2, pp. 291–305, 1999.

[17] S.-U. Guan and S. Zhang, "Pseudorandom number generation based on controllable cellular automata," *Future Generation Computer Systems*, vol. 20, no. 4, pp. 627–641, 2004.

[18] D. H. Hoe, J. M. Comer, J. C. Cerda, C. D. Martinez, and M. V. Shirvaikar, "Cellular automata-based parallel random number generators using fpgas," *International Journal of Reconfigurable Computing*, vol. 2012, p. 4, 2012.

[19] G. Stefan, "Looking for the lost noise," in *Semiconductor Conference, 1998. CAS'98 Proceedings. 1998 International*, vol. 2. IEEE, 1998, pp. 579–582.

[20] M. Sipper, *Evolution of parallel cellular machines*. Springer Heidelberg, 1997, vol. 4.

[21] J. Walker. (1998) Ent randomness test. [Online]. Available: http://www.fourmilab.ch/random/

[22] G. Marsaglia and W. W. Tsang, "Some difficult-to-pass tests of randomness," *Journal of Statistical Software*, vol. 7, no. 3, pp. 1–9, 2002.

[23] S. Chari, C. Jutla, J. R. Rao, and P. Rohatgi, "A cautionary note regarding evaluation of aes candidates on smart-cards," in *Second Advanced Encryption Standard Candidate Conference*. Citeseer, 1999, pp. 133–147.

[24] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 4, p. 22, 2007.

[25] D. B. Thomas and W. Luk, "High quality uniform random number generation through lut optimised linear recurrences," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005, pp. 61–68.