Chapter 9

Cellular Automata and Randomization: A Structural Overview

Monica Dascălu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79812

Abstract

The chapter overviews the methods, algorithms, and architectures for random number generators based on cellular automata, as presented in the scientific literature. The variations in linear and two-dimensional cellular automata model and their features are discussed in relation to their applications as randomizers. Additional memory layers, functional nonuniformity in space or time, and global feedback are examples of such variations. Successful applications of cellular automata random number/signal generators (both software and hardware) reported in the scientific literature are also reviewed. The chapter includes an introductory presentation of the mathematical (ideal) model of cellular automata and its implementation as a computing model, emphasizing some important theoretical debates regarding the complexity and universality of cellular automata.

Keywords: random number generator, complexity, universal computing model, cellular automata, self-programmable cellular automata, self-organization, synthesis of cellular automata, BIST

1. Introduction

Cellular automata (CA), as a massive parallel computing architecture of highest granularity, consist of a network of finite-state machines with only local interactions. The evolution of the system is based on the evolution of all its components. Starting from simple configurations and applying local (simple) rules, some cellular automata display a complex behavior.

The cellular automata model is connected to significant landmarks in the artificial intelligence domain, including the origins of the artificial life concept [1]. It is considered to be one of the main representatives of the so-called self-organizing artificial systems, together with neural

networks and genetic algorithms. These three models have also in common the natural inspiration, as each of them replicates some features or constructive principles of natural systems [2]. For the genetic algorithms, the natural analogy is the evolutionist idea of combining individuals with certain qualities to obtain a better individual, together with the survival of the fittest concept. For neural networks, it is the resemblance with the natural neural systems and the way biologic neurons transmit information combining the input stimuli. For cellular automata, it is the structural analogy—large systems consisting of simple elements having only local interactions, like a volume of gas particles.

A different manifestation of self-organization is associated with the three models [3]. For genetic algorithms, the self-organization is the process of solving an optimization problem, combining parts of solutions as in the analogy with the biological evolution. In the case of neural networks, the self-organization is associated with the learning algorithms (which may imply millions of coefficients for deep learning, convolutional neural networks). For cellular automata, the self-organization refers to the coherent global evolution, which sometimes displays patterns or regularities, in a system that does not have a central, global control.

One of the main features of cellular automata is considered to be their simplicity. However, the very first cellular automata, proposed by von Neumann and Ulam in the early 1950s had 29 states per cell. It was a mathematical model of an artificial system capable of self-reproduction [1] (able to construct a similar object). This concept of artificial self-reproduction opens the new horizon of the so-called artificial life, since reproduction is considered the main feature of the living beings.

Probably the most famous example of cellular automata is the Game of Life [4], invented by John Conway in 1970. This example of two-dimensional binary cellular automata is the best illustration for the main feature of cellular automata: a simple, regular structure displays a vast phenomenology, which may manifest a certain order starting from random states. In the Game of Life, the cells of two-dimensional cellular automata may have only two states (0 and 1, meaning “dead” or “alive”). In the next time step, the cells will live, die, or be born depending on the number of living neighbors in the present moment. The significance that Conway associated to the Game of Life evolution is only symbolic but became iconic for artificial life.

Like the whole domain of artificial intelligence, cellular automata have passed through periods of development and stagnation. Successful applications in modeling and simulation of complex processes, cryptography, image processing, etc., were developed and reported in the scientific literature (for a review of applications, see [5, 6]). Random number generation is one of the typical applications for cellular automata, with applications like built-in self-test (BIST) of integrated circuits and cryptography.

The rest of the chapter is organized as follows: Section 2 presents the mathematical (ideal) model of the cellular automata, the structure and phenomenology of the corresponding computing (real-life) model, and its applications. In order to study and use cellular automata, one should understand both the potential and limitations of the model—Section 3 discusses some of its fundamental issues (complexity vs. simplicity, universality vs. difficulty of synthesis, massive parallelism vs. sequential simulation).

The following sections are dedicated to applications of cellular automata for random number generators and reviews the results reported in scientific literature. Section 4 gives some examples of cellular automata randomizers. Two trends have been identified: the use of the classical, regular model (often related to a theoretical approach in search of better statistical properties), and the alteration of the model, in order to obtain a more unpredictable evolution. Hence, an important topic is the modification of the model. Section 5 discusses the actual implementations of the cellular automata randomizers that were proposed by different authors and their performances. The chapter ends with a section of concluding remarks.

2. Cellular automata: the model

The presentation in this section is only a basic overview of concepts and features of cellular automata and is based on classical books on the topic like [2, 7, 8].

2.1. The model—ideal and real

The cellular automata model is defined as a combination of structure and functionality. The structure consists of finite automata and a network of interconnections between them. The functionality is completely defined by local rules that depend on local conditions, related to the type of interconnection network. Specifically, this means that there is no global control. The basic components (finite-state machines) are named cells. The next state of each cell is computed based on the present state of the cell and its neighbors (in the network). All cells are updated synchronously. **Figure 1** suggests how the next state is computed in linear cellular automata, for one cell. All cells' states are computed in a similar manner.

There are two assumptions for the ideal model: infinity and regularity. In reality, the architecture is finite and boundary conditions (functional and topologic) should be defined. Depending on these conditions, cellular automata may be also irregular, at least at the edges.

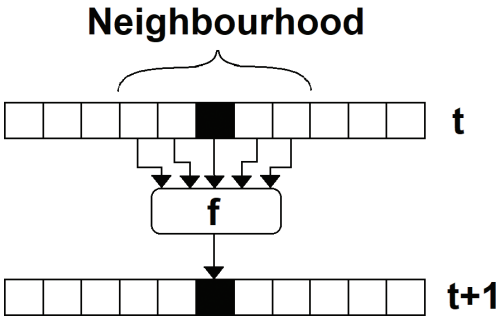

Figure 1. The computation of the next state for one cell, in linear cellular automata.

The most important features of the generic cellular automata computing model are: it is discrete in space and in time, it is finite and regular (except for the limit conditions), and it is parallel. The model can be regarded, in terms of computation theory, as an elementary single instruction multiple data (SIMD) architecture, since all cells perform identical operations [2]. The system is synchronous; therefore, in simulations, the algorithm that computes the next configuration can update cells' states in any order. The local rules, or functions, or laws, are deterministic, implying a global deterministic evolution (which is, however, often hard to predict).

The complexity of the resulting structure, as a digital system, is depending on the number of cells' states, the number of cells (the dimension), and the dimension of the neighborhood, related to the type of interconnection network [9].

2.2. Definitions of specific terms

Before going further, here is a short list of specific terms related to cellular automata that will be used in this chapter:

- cell—the elementary computing element, which is a finite-state machine (**Figure 2**),
- binary cellular automata—each cell has only two states, encoded on 1 bit,
- dimension of cellular automata—total number of cells,

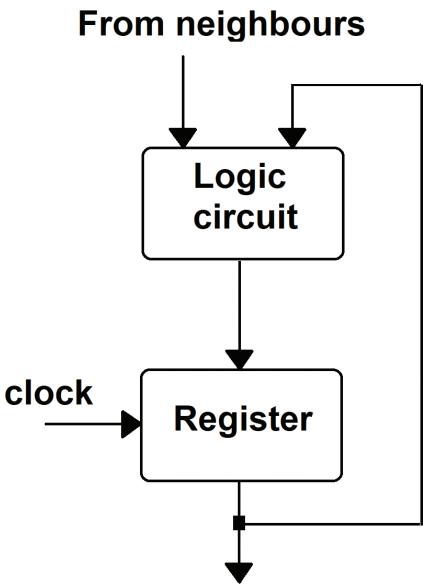

Figure 2. The cell as a finite-state machine connected to the cells in the neighborhood.

- neighborhood—the set of cells, which are involved in the computation of the next state (see **Figures 1** and **2**),
- neighborhood dimension—the number of cells involved locally in the computation of the next state value,
- von Neumann and Moore neighborhood—in two-dimensional rectangular topologies, these types of neighborhoods contain the closest four neighbors (van Neumann), or eight neighbors (Moore), plus the central cell
- limit conditions (or boundary conditions) refer to the interconnection of the cells at the edges
- cyclic boundary conditions—the opposite edges are adjacent,
- rule—the function performed by each cell,
- rule 30—one of the most studied evolution rules, introduced by Wolfram in [10],
- configuration—vector containing all cells' states,
- seed—the initial configuration or initial state,
- trajectory—evolution in the configurations' space,
- basin of attraction—graph representation that illustrates the cyclic trajectory in the configurations' space,
- programmable cellular automata—in the context of an implementation, it allows the modification of the rule or/and topology.

A frequently used denomination of local rules was introduced by Wolfram in [8]. In the context of linear binary cellular automata with neighborhood dimension of three, each rule (which is a Boolean function with three variables) is designated by a decimal number, equal to the binary number obtained from the look-up table of the function. See Appendix 1 for most frequently used rules.

In applications, the synthesis of cellular automata implies to define the particular topology (number of states per cell, number of cells, type of interconnection network, dimension of neighborhood, border conditions), the local rules, timing conditions, and the seed, in order to obtain the desired functionality. For instance, in image processing applications, the initial configuration is the image. The local rules are established in order to obtain the desired function (for instance, edge detection). The automata will run for a certain number of states or until it reaches a stable configuration.

In the previous example, the input data are the initial configuration, the output data are the final configuration, and the computation related to the image processing task is done by the evolution of the global state, through local changes. Computation with cellular automata may also be considered in terms of propagation and combination of patterns, in an analogy with propagation of signals and logical combination of inputs.

2.3. Taxonomy of cellular automata

The classification of cellular automata is based either on structure (topology) or behavior (function and/or phenomenology).

The topology refers mainly to the type of network and local connections (neighborhoods and boundary conditions). In linear cellular automata, the cells are connected in a row (vector) to their nearest neighbors. Further subdivision of linear cellular automata is based on the neighborhood dimension, which is one of the main factors that affect the complexity of the cell. In two-dimensional cellular automata models, the interconnection network is two dimensional, typically rectangular, but also hexagonal networks have been explored for specific applications. However, topologically any two-dimensional network can be transformed in a rectangular one, by choosing an appropriate neighborhood [5]. In typical rectangular connections, there are two most used neighborhoods: von Neumann neighborhood which contains the four adjacent cells on the vertical and horizontal lines, and the central cell itself; and Moore neighborhood that contains the lateral neighbors, the central cell, and the cells adjacent at corners (**Figure 3**).

The theoretical analysis of two-dimensional cellular automata is an open field of research, and most often, the results are extensions of the better-known case of linear automata. Two-dimensional cellular automata are very important in applications, as for instance in image processing, where the image corresponds directly to the configuration of the system. Most of modeling applications also involve two-dimensional extensions, therefore, use two-dimensional cellular automata [11].

2.4. Phenomenology of cellular automata—Wolfram's taxonomy

Probably the most important contribution in understanding cellular automata capabilities as a computer model is Wolfram's experimental approach [8]. Wolfram explored the behavior of linear cellular automata for all rules (with a neighborhood of dimension 3), starting from different "seeds" or initial states. Wolfram established a classification based on the statistical

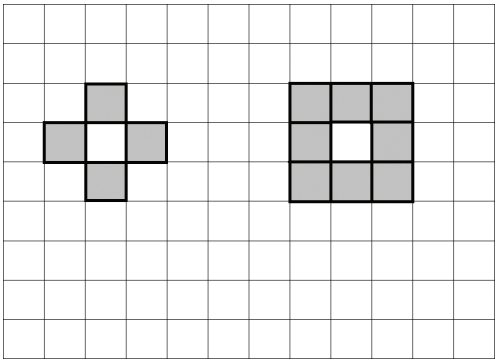

Figure 3. Neighborhoods in two-dimensional cellular automata: von Neumann (left) and Moore (right) neighborhoods.

properties of the linear cellular automata evolution, which is particularly important from the perspective of applications of cellular automata in generation of pseudorandom sequences.

Class 1. Cellular automata that evolve to a homogenous finite state. Automata belonging to this first class evolve from almost all initial states, after a finite number of steps, to a unique homogenous configuration, in which all cells have the same state or value. The evolution of class 1 cellular automata completely "destroys" any information on the initial state.

Class 2. Cellular automata that have a periodical behavior. The configurations are divided in basins of attraction, the periodical evolution of the global configuration depending on the initial state. The states' space being classified in attractor cycles; such cellular automata can serve as "filters."

Class 3. Cellular automata exhibiting chaotic or pseudorandom behavior. This class is particularly significant for ideal (infinite) cellular automata. Cellular automata belonging to the third class evolve, from almost all possible initial states, not periodically, leading to "chaotic" patterns. After an enough long evolution (number of steps), the statistical properties of these patterns are typically the same for all initial configurations, according to Wolfram. Although in practice cellular automata will always be limited to a finite number of cells, this class can be extrapolated maintaining the local rules. Such automata can be used for pseudorandom pattern generation [8, 10].

Class 4. Cellular automata having complicated localized and propagating structures. They can be viewed as computing systems in which data represented by the initial configuration are processed by time evolution, the result being contained in the final configuration of attractor cycle of configurations.

In conclusion, cellular automata are systems consisting of a regular network of identical simple cells that evolve synchronously according to local rules that depend on local conditions. Although the structure is very simple and regular, it produces a vast phenomenology, and therefore, cellular automata are often described as an "artificial universe" that has its own specific local laws [1].

3. Theoretical and practical issues

This section discusses specific issues of cellular automata that one should be aware before deciding to use or investigate this model.

3.1. Parallelism: hardware vs. simulation

Massive parallelism is one of the definitory features of cellular automata. However, simulations are often used instead of actual implementations, and for very good reasons: there are no commercially available hardware versions, hardware accelerators, or co-processors for the cellular automata computational model. The full performance of the massive parallel architecture will never be reached by simulation; in fact, the high granularity will make the simulations slow, even for small dimensions.

There are several hardware implementations for cellular automata reported in the scientific literature (for a recent review, see [12]). Most of them are particular implementations for specific applications. The so-called "cellular automata machines," including CAM (Cellular Automata Machine, project started in the 1980s at MIT [13]) and CEPRA (Cellular Processing Architectures, first prototype in the 1990s at the Darmstadt University, in Germany [14]) never reached industry. Both projects combine serial processing and pipeline techniques to emulate the parallelism of the cellular automata architecture.

Hence, the first paradox of cellular automata: a model of massive parallelism, reduced to sequential computation.

3.2. The problem of synthesis

The synthesis of cellular automata implies to establish the structure and functionality (including the states per cell, the dimension, the topology, and the local rule) that may solve a certain problem or perform a specific computation. This problem of synthesis is the root problem of massive parallel computing the decomposition of a computing problem in elementary tasks to be performed repeatedly by a large number of cells for a certain number of cycles. For cellular automata, the problem of synthesis was approached, for instance, from the perspective of evolutionary computing (similar with genetic algorithms) [15], but still remains the biggest challenge of cellular automata applications.

In terms of computation theory, Wolfram considers that the model of cellular automata is universal, which means that it can solve any computable problem (claiming the equivalence with the Turing machine [16]). There is no general acceptance of the computing universality of the model, but particular rules were proved to be computational universal [16–18], meaning that they can perform any function (which, again, does not mean that they can solve any computing problem). In spite of such theoretical results, a huge challenge still remains: there is no algorithm or method for synthesis of particular cellular automata.

This is the second paradox of cellular automata: its computing capability is not reflected in applications, as there is no synthesis method. As massive parallel computing systems, cellular automata seem ideal for hardware implementation. Theoretically, they can compute any function; however, the difficulty of synthesis was a major drawback in the development of applications.

3.3. Complexity and self-organization

In the introductory section, we have mentioned two features of the cellular automata model: the self-organization and the complexity. Depending on the context, different authors looked at cellular automata from both perspectives: as complex systems that, starting from a random initial state, manifest the self-organization property, or as simple, regular systems that exhibit a very complex, random behavior [1, 4, 7, 8].

In [19], a possible reconciliation of the two perspectives includes them both in the concept of "apparent complexity," understood as a "complex phenomenal appearance backed by a structural simple generative rule" (in order to be efficiently used, the term "apparent complexity"

should be further, more clearly, defined). The same evolution or pattern may appear complex or simple, depending on the perspective of the analysis. In the particular case of linear cellular automata, of one does not know the rule and the initial state, it is difficult to infer them by analyzing the behavior. We recognize here again the problem of synthesis, as this issue is directly related to the previous one.

This dichotomy complexity—self-organization is the third paradox of cellular automata.

3.4. Variations of the model

The last important issue is related to the fact that the actual field of cellular automata research, particularly in application development, has adopted a lot of variations of the ideal model (for which the theory was developed). We will mention the most important ones, those who are significant for the topic of this chapter. For a detailed overview, see [3].

Hybrid or inhomogeneous cellular automata: either the cellular space is inhomogeneous (different structures of neighborhoods and cell properties, topological variations), or local rules vary in the cellular space. Local rules may also be modified after a number of time steps, in order to obtain a specific processing of the global configuration. Block hybrid cellular automata are a particular case of inhomogeneous cellular automata. The cellular structure is here divided in homogenous subdomains or blocks.

Automata with structured states' space: the cell's state is considered to be the combination of some significant parameters or state values. As in the case of finite-state machine design, such structuration leads to a global simplification, mainly regarding the dimension of the local rules space.

Multilevel cellular automata: a more complex model, built as a hierarchical structure of interconnected cellular automata (the model is not simply a multidimensional structure).

Self-programmable cellular automata: the local rules change in time, depending on the evolution, and may be implemented as multilevel cellular automata.

Asynchronous automata: the cells' states are updated in a certain order, taking into account the new values obtained for the neighboring cells already updated.

Cellular automata with supplementary memory layer: the computation of the following states takes into account the "history" of the system, or a number of previous states of each cell. These previous states are loaded in the memory layer. This model is practically a network of elementary processors.

Nondeterministic and probabilistic cellular automata: the next state is established in a nondeterministic manner or according to a certain distribution of probability. Due to its versatility, this model can be successfully used in complicated modeling applications.

In composite systems, the basic cellular automata model is considered as a space that interferes with autonomous mobile structures or agents that evolve in the cellular space. This model simplifies some modeling effort for example in the case of particles diffusion.

The list above is not exhaustive but gives an image of the versatility of cellular automata developments, starting from the very restrictive, regular, infinite ideal model. On the other extreme, there are random Boolean networks, often called n-k networks that allow any inhomogeneous set of rules and also random connectivity.

A global theory of all models derived from cellular automata does not exist by now and probably will never exist. Because most of the applications in this field are derived empirically and also empirically tested, this is not truly a weak point, as it can be compensated by appropriate experiments.

4. Random number generation with cellular automata

Randomization is essential for several artificial intelligence applications: cryptography, data mining, games, integrated circuits verification, etc. Cellular automata have been proposed and used as pattern generators such as: universal pattern generators, pseudorandom signal generators, generation of numerical sequences with predefined statistic properties, BIST generators for integrated circuit's verification [6].

The block scheme of a pseudorandom signal generator is represented in **Figure 4**. The output is considered to be pseudorandom since it is obviously produced by an algorithm (transformation), starting from the initial state (seed). The output sequence appears to be random for someone who does not know the generation algorithm, meaning that the previous number in the sequence cannot be computed, nor the next numbers predicted, based on the present output.

The apparent complexity of cellular automata phenomenology makes them useful tools for pseudorandom number/signal generators (frequently, the word "pseudo" is skipped). In this section, we will present such some examples of cellular automata random number generators reported in scientific literature, while next section will discuss their properties, implementation, and applications.

4.1. Examples of cellular automata pseudorandom number generators

Based on the phenomenology of cellular automata introduced by Wolfram in [8], the class 3 linear, binary automata were proposed as pseudorandom generators [10]. There are basically

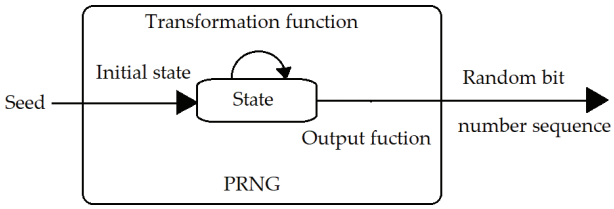

Figure 4. Block scheme of a pseudorandom signal/number generator (PRNG).

two ways to consider the output of such a generator, either as a one-bit digital signal (given by the evolution of the state of one particular cell), or an n-bit output (a subset of the global configuration). The statistical properties depend both on the rule and on the initial state, and the choice of the n-bit vector (because of intrinsic correlation of neighboring cells).

Note that all cellular automata will eventually enter into an attraction cycle—the length of the trajectory is related not only to the rule, the initial state but also to the dimension of the automata. This is why the "infinity" assumption often is mentioned in theoretical demonstrations regarding statistic properties of cellular automata randomizers. For better properties, larger automata will be used.

Wolfram analyzed the properties of several rules and proposed a generator based on the so-called "rule 30," for a one-bit signal generated by the central cell [10]. **Figure 5** presents the results of the simulation of a 64-cells linear cellular automata with rule 30 (Verilog HDL simulation), for 25 time steps.

Further research was oriented in discovering better rules and initial configurations (seeds) for improving the statistical properties of the sequence. A simple solution in this direction implies the use of two different rules, thus obtaining heterogenous cellular automata. Hortensius et al. proved in [20] that the results obtained with heterogenous rules (two different local rules) are better than in homogenous automata using either of those rules. Commonly used rules are listed in **Appendix 1**.

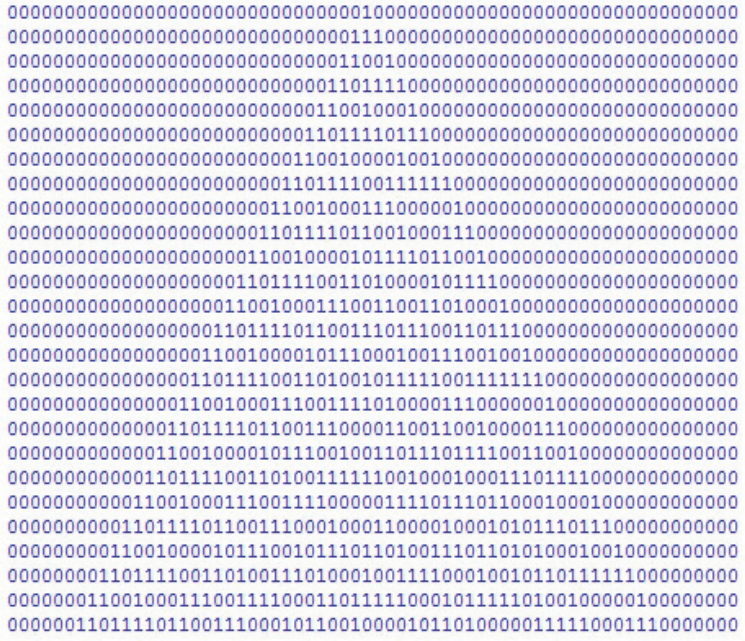

Figure 5. The output of a 64-cells linear cellular automata with rule 30 (Verilog HDL simulation).

Sipper and Tomassini [21] included two innovative improvements: the selection of rules based on an evolutionary algorithm (four rules were selected, namely 90, 105, 150, and 165) and the so-called "time spacing" of the sequence, meaning that the output sequence skips some time steps. This technique improves the quality of the sequence, according to the authors, as compared to classical random number generators.

In a later work, Tomassini and Sipper also proposed two-dimensional cellular automata [22] in order to improve the complexity of the evolution of the generator. Another two-dimensional cellular automata for pseudorandom sequence generation was presented in [23] and was applied by the authors in BIST and cryptography hardware implementations [6]. The difference is mainly quantitative in a two-dimensional topology, but also the neighborhood dimension is higher (and the rules more complex).

One-dimensional topology is simpler from the perspective of hardware implementation. Still theoretical research is done on linear cellular automata in order to discover binary functions or class of functions that may be used in random sequence generation. In [24], the authors prove that uniform linear cellular automata with nearest neighbor's connections can meet cryptographic requirements for random sequences (The selection of rules and seeds was done empirically.)

The examples given so far illustrate the theoretical and experimental effort to prove the properties of cellular automata and to discover significant combinations of rules and initial states that lead to "good" pseudorandom sequences. All these examples are based on the basic cellular automata model. The only modification of the model is, for some of these examples, the heterogeneity, but this is restricted to the implementation of two/four rules in a static manner to subsets of cells.

4.2. Examples based on variations of the cellular automata model

Another important trend of research is to improve the properties of the pseudorandom sequence introducing more substantial modifications to the cellular automata model, in order to obtain a more complex structure and expected behavior. A first example, presented in [9], uses a homogenous programmable cellular automata of 256 cells, with a global feedback loop. Each cell supports any of the 256 possible rules, but the actual rule is selected by the feedback (depending on the global state, with a simple transformation like module 2 sum). Such an approach is now called self-programmable, since the rule is not changed externally, but here the global reaction acts like a central control. According to the author, the initial state influences the properties of the generator.

Another structural modification of the cellular automata model is presented in [25]. Here, an additional memory layer was used to improve the complexity of the evolution, and the next state is computed based on the present state of the neighborhood and the previous state of the cell (or the entire neighborhood). The local rules are 4-variable functions, in the simplest form.

In physical implementations, the actual dimension of the cellular automata is reduced. Several authors proposed randomization schemes that improve the quality of the generators (to compensate the limited dimension). In [26], the output of the hybrid cellular automata is combined

with the output of a linear feedback shift register (LSFR) to produce a 1-bit random output using 32-bit cellular automata. A combination of rules 90 and 150 was applied in the cellular automata.

The technique of self-programming, meaning that the local rules are changed during the evolution, was presented in [27]. The local rules are changed based on local conditions (in [9], the global configuration determines the rule). The result is a hierarchical structure of cellular automata. Each cell can switch between two rules (90 and 165, 150, and 105 were used), depending on a "super rule" based on the state of a larger neighborhood (up to 7 cells). The idea of the "super rule" is to interrupt static behavior (the patterns that appear in low-quality pseudorandom signals).

The idea of self-programmable cellular automata, with a hierarchical structure, and a behavior based on local conditions, was developed in several ways since its introduction in [27]. For instance, [28] presents a combination of this technique in hybrid cellular automata, with a linear feedback shift register. The global feedback is used in [29] in the generation of the super rule with the additional selection of the cell whose rule is changed. In [30], the self-programmable cellular automata are optimized for field programable gate arrays (FPGA) implementation, using the splittable look-up table concept and embedded carry logic of the FPGA.

Other variation of the model is the asynchronous update of the cells' states. There are several examples in the scientific literature, of which we mention [31] because it is one of the few examples that use two-dimensional cellular automata (for 1-bit output) and also because it is designed as a randomization algorithm, not targeting the hardware implementation.

In conclusion, several variations of the cellular automata model were developed. The validation of the models is often experimental, in absence of theoretical developments.

5. Implementation and applications of cellular automata randomizers

In this section, we will make some qualitative comments regarding the implementation of cellular automata randomizers, their applications, and the evaluation of their performances.

Compared to other applications of cellular automata, random number generation eludes the problem of synthesis as explained in Section 3. Most of the research is done (and confirmed) empirically, including the selection of functions, seed, and other implementation details. In spite of the theoretical efforts to prove the properties of the classical cellular automata and the good results obtained by several researchers [10, 24], most of the reviewed papers include some variations of the basic model in order to obtain better statistic properties. One of the reasons may be the fact that the "infinite dimension" assumption is the guarantee of good random sequence generation with cellular automata. The performances depend both on the local rules and initial state. Particular cases when specific rules produced either excellent or poor, depending on the seed, are reported in the scientific literature.

Both linear and two-dimensional cellular automata were proposed in scientific literature for pseudorandom number generation. Linear topology is preferred for hardware implementation, although the two-dimensional topology would take the most advantage of the parallel computation.

As an alternative to the mathematical, ideal cellular automata model, several authors introduced variations. The most frequently used models include nonhomogenous cellular automata (with maximum four different rules performed by subsets of cells) and self-programmable cellular automata (that can be considered either as a hierarchical structure or as two parallel cellular automata). Other techniques that impact the apparent complexity use larger neighborhoods in the next state computation, or previous states.

The performances of a random number generator are determined by the statistical properties of the output, although there are other important properties like the dimension of sequence, speed, and costs of implementation. Since random number generators produce sequences that are not really random (being based on an algorithm or a process), their "randomness "is established by several statistical criteria that try to evaluate how difficult is to predict the evolution or compute the previous values.

One of the first systematic approaches to randomness evaluation was done by Knuth [32] who suggested a series of statistical tests (for instance, the frequency test, the autocorrelation or serial correlation test, the run test, etc.). Marsaglia [33] added several tests and launched a battery of 15 statistical tests in 1995. The new state of the art, since 2010, is established by the NIST test suite [34], specially designed for cryptographic applications.

There is no comparative study (and this is over the intention of this chapter) of the performances of all these alternatives presented by different authors. As a general remark, their evaluation is often done at the state-of-the-art level of the time when they were developed. Before the introduction of the Diehard test suite [33], the authors performed their own tests (see, for instance, [8, 20, 21]). Since the late 1990s, the Diehard tests were used to assess the performances of cellular automata randomizers (like in Refs [26–28]). Most of the recent work evaluates the performances based on NIST test suite [24, 30, 31].

Performances should be related to the application of the generators. Most common applications reported in scientific literature are integrated circuit verification (as in BIST pattern generators) and cryptography. Therefore, the performances should be evaluated according to the application. For BIST pattern generators, a comparison with LFSR is a common practice and is sufficient [6] (since LFSR is one of the consecrated methods for hardware random number generation). For cryptography, there are more strict requirements, both for hardware and software versions of random number generation. On the other hand, BIST applications are almost intended for hardware implementation [6, 26, 28].

Figure 6 presents a consecrated architecture for cryptography with cellular automata [25]. The initial message is combined with the cellular automata output to obtain the encrypted message (frequently used transformation is the exclusive or, XOR, function). In the decryption stage, the encrypted message is again combined with the output of identical cellular automata. The key is, in this scheme, the initial state (sometimes combined with the local rule, if several rules are available).

Comparing the hardware implementations of cellular automata reported in the scientific literature, one can notice the positive impact of the development of the FPGA technology,

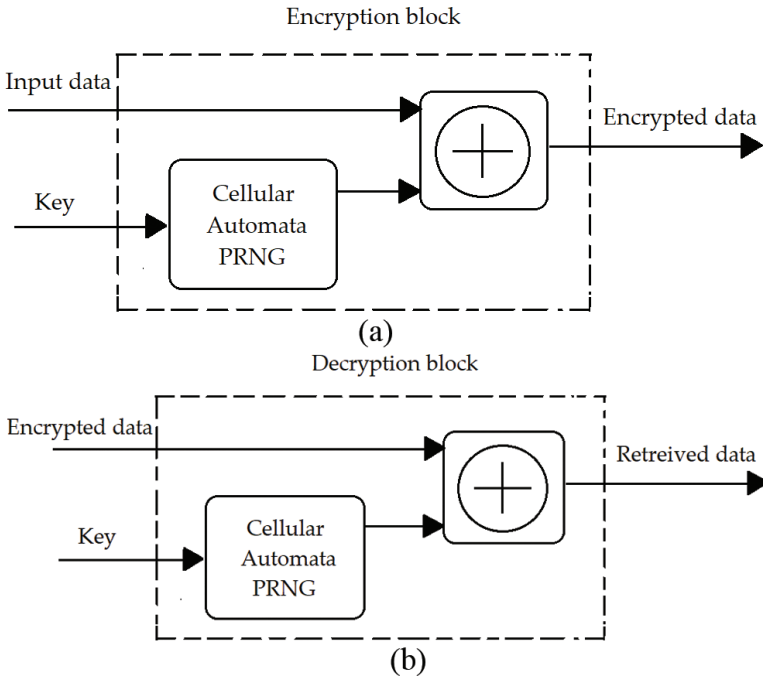

Figure 6. Encryption and decryption schemes in a cellular automata-based cryptographic system.

including for the hardware versions of cellular automata randomizers. Prior the FPGA implementations, the most common used rules were combinations of XOR functions, while the FPGA, look-up table (LUT)-based technology makes possible the implementation of all logic functions with same resources. On the other hand, some applications target application specific integrated circuits (ASIC) circuits (this is the case for BIST generators).

6. Conclusions

Cellular automata are one of the iconic models of massive parallelism, and their development is related to the artificial intelligence and artificial life domains. In absence of an effective hardware implementation, one can say that the model never reached its full potential as a massive parallel machine. Another drawback is the difficulty of synthesis of cellular automata that perform a specific global processing task, based on elementary computations at cell level. However, as dynamic systems with a vast phenomenology, they are an important direction of research in the theory of complexity, efficient modeling, and simulation tools and as the examples in this review have proved, good random number generators. The applications of

cellular automata in random number generation are based on their apparent complex behavior. Although the direct process is very simple, it is not reversible—the complexity of the inverse determination grows exponentially.

Random number generation is one of the successful applications of cellular automata, particularly for cryptography and verification of circuits (BIST generators). This chapter presents a selection of the results presented in the scientific literature. Two main directions of research were identified, based on these examples: the mathematical approach to demonstrate the properties of particular rules and configurations, and the engineering and empiric approach to develop new architectures. Some examples are oriented toward "classic," close to ideal model of cellular automata, while others include consistent modifications at structural and functional levels. Both hardware and software cellular automata randomizers are reported in the scientific literature. The development of the FPGA technology offers efficient solutions for hardware versions, when needed.

Acknowledgements

I want to thank the editor for the suggestions that very much helped to improve this chapter. Also, I would like to thank Eduard Franti for useful remarks and support.

Conflict of interest

The author declares no conflict of interest.

Appendix 1. Frequently used local rules

This appendix presents the nomenclature conventions (**A 1**) for local rules, frequently used local rules (**A 2**) and some rules invoked in the discussion of Turing equivalence of cellular automata.

The denomination of rules introduced by Wolfram [8] refers to 3-bit Boolean functions. Each function is designated by the configuration in the look-up table, read in reverse (hence, in this context, the look-up table is drawn starting with the last value). In **A 1**, the look-up table of the

Present state	111	110	101	100	011	010	001	000
Next state (of the central cell)	0	0	0	1	1	1	1	0

Here, rule 30.

A 1. Each 3-bit rule is designated by the value in the look-up table.

Rule	LUT configuration	Examples
30	00011110	[10]
90	01011010	[20, 21, 26, 27]
105	01101001	[22, 27]
150	10010110	[20, 21, 26, 27]
165	10100101	[22, 27]

A 2. Frequently used local rules for random number generation with linear cellular automata.

rule 30 is given. **Table 2** gives the specific configuration in the look-up table for some of frequently used rules cited in this paper, according to the references list.

In the basic rule in the Game of Life binary, two-dimensional cellular automata, the significance of 0 and 1 is that a cell is dead or alive. With a Moore neighborhood, the rule can be described as follows:

- if a cell is alive, it remains alive in the next step only if it has two of three living neighbors,
- if a cell is dead, it becomes alive only if it has three living neighbors,
- a living cell with less than two neighbors alive will die of loneliness,
- a living cell with more than three neighbors alive will die overcrowded.

In the Game of Life, there are several patterns that persist for ever, either in a dynamic or static manner. For instance, gliders are patterns that are moving. The demonstration of the Turing equivalence for the Game of Life involves flows of gliders that perform logic functions.

Rule 110, in linear cellular automata (binary code 01101110), is a class IV rule in Wolfram's taxonomy and, like Life, is Turing equivalent.

Author details

Monica Dascălu

Address all correspondence to: monica.dascalu@upb.ro

Research Institute for Artificial Intelligence, Politehnica University of Bucharest, Bucharest, Romania

References

[1] Coveney P, Highfield R. Frontiers of Complexity, the Search for Order in a Chaotic World. London: Faber and Faber; 1995

[2] Garzon M. Models of Massive Parallelism. Berlin: Springer-Verlag; 2012

[3] Dascalu M. Self-Orgnizing Systems. Bucharest: Politehnica Press; 2016. pp. 101-116

[4] Conway J. The Game of Life. Vol. 1970. Scientific American Magazine. pp. 120-123

[5] Dascalu M, Franti E. Automate Celulare – Modelare si Aplicatii. Bucharest: Editura Tehnica; 2009. pp. 119-125

[6] Chaudhuri P, Chowdhury D, Nandi S, Chattopadhyay S. Additive Cellular Automata: Theory and Applications. Vol. 1. Computer Society Press; 1997

[7] Gutowitz H. Cellular Automata and the Sciences of Complexity, I and II. Complexity; 1996;**5**:16-22, **6**:29-35

[8] Wolfram S. Statistical properties of cellular automata. Reviews of Modern Physics. 1983; **58**:601-644

[9] Stefan G. Looking for the lost noise. In: Proceedings of the Semiconductor Conference (CAS'98); 02-04 October 1998. Vol. 2. Sinaia: IEEE; 1998. pp. 579-582

[10] Wolfram S. Random sequence generation by cellular automata. Advances in Applied Mathematics. 1996:123-169

[11] Weimar J. Simulation with Cellular Automata. Berlin: Logos Verlag; 1999

[12] Dascalu M. Cellular automata hardware implementations - an overview. Romanian Journal of Information Science and Technology. 2016;**19**(4):360-368

[13] Toffoli T, Margolus N. Cellular Automata Machines: A New Environment for Modeling. Cambridge, USA: MIT Press; 1987

[14] Halbach M, Hoffmann R. FPGA Implementation of Cellular Automata Compared to Software Implementation. In: Proceedings of the Organic and Pervasive Computing Workshops (ARCS'04); 26 March 2004; Augsburg. Germany

[15] Mitchell M. Computation in cellular automata: A selected review. In: Scuster, Gramms, editors. Nonstandard Computation. Weinheim: VCH Verlagsgessellschaft; September 1996

[16] Wolfram S. Cellular automata as models of complexity. Nature. 1984;**311**:419-424. DOI: 10.1038/311419a0

[17] Sutner K. Universality and cellular automata. In: Margenstern M, editor. Machines, Computations, and Universality. MCU 2004. Lecture Notes in Computer Science. Springer-Verlag; 2004. p. 3354

[18] Lena Di P, Margara L. Computational complexity of dynamical systems: The case of cellular automata. Information and Computation. 2008;**206**(9–10):1104-1116

[19] Malita M, Stefan G. Real complexity vs. apparent complexity in Modeling social processes. Romanian Journal of Information Science and Technology. 2013;**16**(2–3):131-143

[20] Hortensius PD, McLeod RD, Card HC. Parallel random number generator for VLSI systems using cellular automata. IEEE Transactions on Computers. 1989;**38**(10):1466-1473

[21] Sipper M, Tomassini M. Co-evolving parallel random number generator. In: Voigt HM, Embeling W, Rechenberg I, Schwefel HP, editors. Parallel Problem Solving from Nature IV (PPSN IV); 1141. Lecture Notes in Computer Science. Heidelberg: Springer-Verlag; 1996

[22] Tomassini M, Sipper M, Perrenoud M. On the generation of high-quality random numbers by two-dimensional cellular automata. IEEE Transactions on Computers. 2000;**49**:1146-1151

[23] Chowdhury DR, Gupta IS, Chaudhuri PP. A class of two-dimensional cellular automata and applications in random pattern testing. Journal of Electrical Testing: Theory and Applications. 1994;**5**:65-80

[24] Ping P, Xu F, Wang ZJ. Generating high-quality random numbers by next nearest-neighbor cellular automata. In: Proceedings of the 2nd International Conference on Systems Engineering and Modeling (ICSEM-13). Paris, France: Atlantis Press; 2013

[25] Franti E, Dascalu M. More security and autonomy for users: Encryption system with evolutive key. In: Data, Text and Web Mining and their Business Applications. WIT Press; Cambridge Printing; 2007. pp. 335-344

[26] Tkacik TE. A hardware random number generator. Cryptographic Hardware and Embedded Systems. 2003;**2523**:450-453

[27] Guan S, Tan SK. Pseudorandom number generation with self-programmable cellular automata. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2004;**23**(7):1095-1101

[28] Comer JM, Cerda JC, Martinez CD, Hoe DHK. Random number generators using cellular automata implemented on FPGAs. In: Proceedings of the 44th Southeastern Symposium on System Theory (SSST'12); Jacksonville; FL. 2012. pp. 67-72. DOI: 10.1109/SSST.2012.6195137

[29] Mocanu D, Gheolbanoiu A, Hobincu R, Petrica L. Global feedback self-programmable cellular automaton random number generator. Technical Journal of the Faculty of Engineering. 2016;**39**(1):1-9

[30] Petrica L. FPGA optimized cellular automaton random number generator. Journal of Parallel and Distributed Computing. 2018;**111**:251-259

[31] Bilan S, Bilan M, Bilan S. Research of the method of pseudo-random number generation based on asynchronous cellular automata with several active cells. In: Proceedings of the 21st International Conference on Circuits, Systems, Communications and Computers (CSCC 2017); 04 October 2017. 2017. DOI: 10.1051/matecconf/201712502018

[32] Knuth DE. The Art of Computer Programming, Vol. 2: Seminumerical Algorithms. 2nd ed. Boston: Addison-Wesley; 1981

[33] Marsaglia G. Diehard Test Suite. [Internet]. 1998. Available from: https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/ [Accessed: 2018-05-19]

[34] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J, Vo S. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. [Internet]. 2010. Available from: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf [Accessed: 2018-05-19]