Journal of Cellular Automata, Vol. 11, pp. 63–89
Reprints available directly from the publisher
Photocopying permitted by license only

©2015 Old City Publishing, Inc.
Published by license under the OCP Science imprint,
a member of the Old City Publishing Group.

Parallel Application of Hybrid DNA Cellular Automata for Pseudorandom Number Generation

GEORGIOS CH. SIRAKOULIS

Department of Electrical & Computer Engineering,
Democritus University of Thrace, Xanthi, GR67100, GREECE
E-mail: gsirak@ee.duth.gr

Received: May 29, 2014. Accepted: May 4, 2015.

With the advent of massively parallel scientific computation, the parallel generation of pseudorandom numbers has become essential. During the last decades several researchers have successfully implemented Cellular Automata (CA) as Pseudorandom Number Generators (PRNGs). On the other hand, recently Autonomous DNA Turing Machines and DNA Cellular Automata were proposed as cellular computing devices that can serve as reusable, compact computing devices to perform (universal) computation. In this paper, we introduce a methodology for the design of one-dimensional (1-d) Hybrid Autonomous DNA Cellular Automata (HADCA), able to run in parallel, different CA rules with certain modifications on their molecular implementation and information flow compared to their origins. In this aspect, an easy to use HADCA simulator was developed to encourage the possible use of the biological inspired computation tool. As a result, the proposed 1-d HADCAs are used to generate high-quality random numbers which can pass the statistical tests of DIEHARD, one of the most well known general test suites for randomness, proving their suitability as PRNGs.

Keywords: Cellular automata; hybrid autonomous cellular automata; DNA molecular reactions; pseudorandom number generation

1 INTRODUCTION

Since real random numbers can be obtained from some physical phenomena only, they are difficult to be employed in real applications. Hence, pseudorandom numbers generated by artificial designed patterns have to be used

instead. Pseudorandom number sequences are widely used in digital computing and communication applications, such as cryptography, VLSI testing, coding, D-A conversion, etc. [1, 2]. One commonly used type of pseudorandom number generators (PRNGs) is based on Cellular Automata (CA). More specifically, it was back in 1986 when Wolfram [3] first applied CA in pseudorandom number generation. The intensive interest in this field can be attributed to the phenomenal growth of the VLSI technology that permits cost-effective realization of the simple structure of local-neighborhood CA. Wolfram's work in [4] proved that one randomness of the patterns generated by maximum-length CA is significantly better than other widely used methods, such as linear feedback shift registers (LFSRs). As a result, the last years, several one dimensional (1-d) and two dimensional (2-d) CA PRNGs have been proposed [1, 2, 5–12]. The proposed 1-d CAs PRNGs are advantageous in terms of easy implementation, modularity and required silicon area but, in some cases, present reduced randomness, as they fail to pass certain test suites [13–15]. On the other hand, the statistical random tests of the proposed 2-d CAs PRNGs result in that their randomness is much better than that of 1-d CAs PRNGs. However, taking into account design complexity, ease of large scale implementation, maximum switching frequency, time delay and computation efficiency it is difficult to state which one is better.

On the other hand, during the last decades, certain research emphasis has been placed on building synthetic molecular machinery from DNA. In specific, biological systems in which individual molecules act, singly and in concert, as specialized machines result are called DNA machines. It was back in the late 80s when the researchers first started to think of using DNA molecules not only as cornerstone for their biological research but also as possible virtual machines able to perform tasks that are currently beyond our reach. More specifically, DNA nanomachines were made by self-assembly, using techniques that rely on the sequence-specific interactions that bind complementary oligonucleotides together in a double helix [16]. DNA machines can be logically designed since DNA assembly of the double helix is based on strict rules of base pairing that allow portions of the strand to be predictably connected based on their sequence [17]. Consequently, several tasks can be accomplished since devices that change state in response to an external trigger might be used for molecular sensing, intelligent drug delivery or programmable chemical synthesis.

In the latter days, certain advances have been reported, like the DNA machine reported by Bernard Yurke and co-workers at Lucent Technologies in the year 2000, who constructed molecular tweezers out of DNA [17]. Furthermore, Wang *et al.* have built a molecular machine out of DNA that could act as a logic device for chemical sensing and medicine delivery [18]. In details, they built three DNA tweezers that are activated by the inputs H^+/OH^-; $Hg2^+$/cysteine; nucleic acid linker/ complementary antilinker to

yield a 16-states finite-state automaton. The outputs of the automata are the configuration of the respective tweezers (opened or closed) determined by observing fluorescence from a fluorophore/quencher pair at the end of the arms of the tweezers [18].

In correspondence to the aforementioned works, Yin *et al.* successfully managed to use DNA molecules as micromachines capable of universal computing [19]. As also mentioned in [18], it is of significant importance, compared to earlier DNA machines, that unlike its ancestors DNA machines, the proposed nano devices have a degree of memory, making them potentially programmable. The system exhibits a memory because each current state and output depends not only on the source configuration but also on past states and inputs. Moreover, the reported design of an autonomous unidirectional DNA mechanical device capable of universal sequential computation, termed as Autonomous DNA Turing Machine is also capable of complex translational motion which authors defined as universal translational motion [19]. In [20], the same authors extend their previous work and obtain the design of an Autonomous DNA Cellular Automaton (ADCA). By mimicking a 1-d universal CA, the ADCA can perform parallel universal computation, and in the process, demonstrate well coordinated parallel motion.

In this paper we introduce the design and implementation of simultaneous evolution of more than one 1-d CA rules in parallel based on certain modifications of molecular implementations and information flow compared to their origins [20]. More specifically, we have introduced a methodology depicturing the possible needed modifications so as to implement different CA rules resulting into 1-d Hybrid ADCA (HADCA) and in this direction we have developed a simulator to extend the computational usage of the given DNA based CA structures. Taking into account design complexity, maximum switching frequency, timing delay and computation efficiency of CAs we investigated the resulted HADCA as efficient PRNGs. For doing so, we have implemented several CA rules, like rules 110, 30 and 90 and, moreover, we have applied their parallel output as possible PRNGs. Eventually, the produced 1-d HADCA results generate high-quality random numbers which can pass the statistical tests of DIEHARD suite of Marsaglia [13], which seem to be one of the most well known general test suites for measuring the quality of a PRNG. As a result, the presented simulation results can be considered as a promising application of DNA computing.

2 AUTONOMOUS DNA (ADCA) CELLULAR AUTOMATA: DESIGN AND IMPLEMENTATION

2.1 ADCA Preliminaries

In this Section a brief outlook of the original proposed Autonomous DNA Cellular Automaton (ADCA) is provided [20]. Although most of the details

can be found in [20], for readability and comprehension reasons, the key points of the proposed method are also described here to help the reader to appreciate the forecoming modifications as found in the next Sections. More specifically, ADCA operates in a solution system and it is composed of four parts, namely a rigid *symbol track*, a linear array of *dangling DNA molecules* tethered to the symbol track, a set of *floating DNA molecules*, and a group of floating *protein enzymes*.

More specifically, the *symbol track* provides a rigid structural platform on which the dangling-molecules are tethered. It can be implemented, for example, as a rigid addressable DNA lattice, such as the barcode DNA lattice reported in [21]. The array of dangling-molecules, also called symbol-molecules, tethered to the symbol track represent the array of cells (symbols) in the CA (and hence the name symbol-molecule). A *dangling-molecule* is a duplex DNA fragment, with one end tethered to the symbol track via a flexible single strand DNA fragment and the other end possessing a single strand DNA extension (the sticky end). Due to the flexibility of the single strand DNA linkage, a dangling-molecule moves rather freely around its joint on the symbol track. In the presented approach the only possible interactions between two dangling-molecules are those between two immediate neighbors. This requirement can be ensured by properly spacing the dangling-molecules along the rigid track. Furthermore, in addition to the array of dangling-molecules, the system contains floating-molecules. A *floating-molecule* is a free floating (unattached to the symbol track) duplex DNA segment with a single strand overhang at one end (sticky end). A floating-molecule floats freely in the solution and thus can interact with another floating-molecule or a dangling-molecule provided that they possess complementary sticky ends. There are two kinds of floating-molecules: the rule-molecules and the assisting-molecules. The rule-molecules collectively specify the computational rules and are the programmable part of the ADCA, while the assisting-molecules assist in carrying out the operations of the ADCA [20]. Finally, the system also contains floating DNA ligase and three types of DNA endonucleases. The *enzymes* perform ligations and cleavages on the DNA molecules to effect the designed structural changes and hence the information processing.

2.2 Structural Changes

Figure 1 top row depicts an example abstract CA in its top panel, and a corresponding ADCA in its bottom panel [19]. For simplicity and clarity, the floating enzymes and the floating DNA molecules in the ADCA are omitted; the symbol track, as well as the duplex and sticky end portions of a dangling-molecule, is depicted as a thick line segment; the flexible hinge of a dangling-molecule as a thin curve. The leftmost symbol-molecule is a special *initiator*

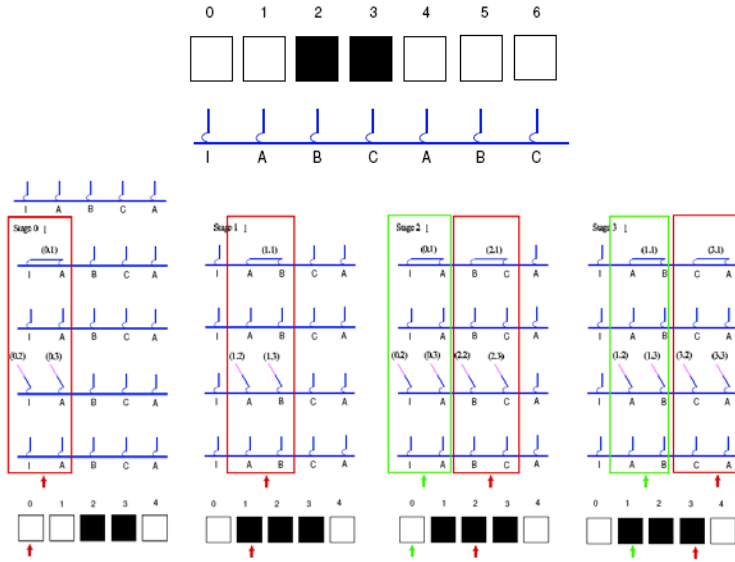

FIGURE 1
Structural changes during the operation of an ADCA. In lower line red (dark) and green (grey) boxes indicate two pipelined [19].

dangling molecule, *I*, representing the cell 0 in the abstract CA. To the right of *I*, three types of dangling-molecules, *A*, *B*, and *C*, are positioned evenly along the track in a periodic order such that cells $3i$+1, $3i$+2, and $3i$+3, where i is a non-negative integer, in the abstract CA are represented in the ADCA by symbol-molecules *A*, *B*, and *C*, respectively. The symbol-molecules differ in their *default sticky ends*, i.e. the sticky ends they possess in their respective initial configurations before the reaction starts. The color of each cell in the abstract CA is encoded in a corresponding symbol-molecule in the ADCA.

Figure 1, bottom line, illustrates structural changes [19]. During the operation of the ADCA, the initiator molecule *I* sends out a "reaction wave" that travels down the track from left to right. A critical novel property of the ADCA is that multiple reaction waves can travel down the track in a "pipelined" fashion. However, a "synchronization" mechanism has been carefully engineered so that a reaction wave that starts at a later stage can never overtake one that starts earlier. This ensures the synchronization of the state changes of the ADCA, and hence its correct operation. In the same Figure 1 bottom line two consecutive reaction waves are shown, respectively indicated with red (dark) and green (grey) boxes and arrows [19]. Next, the first reaction wave (indicated by dark boxes and arrows) is presented, and the structural changes of ADCA are described.

In Stage 0, the reaction wave starts at the initiator I at position 0, then travels sequentially to A in Stage 1, B in Stage 2, and C in Stage 3. The reaction wave finishes one full cycle in Stages 1, 2, and 3, and thus goes on inductionally down the track. In Stage i, where $i = 0, 1, 2, 3$, three types of reactions occur, namely reactions $i.0$, $i.1$. and $i.2$.

In specific, stage 0, I has a complementary sticky end to its right neighbor A and is thus ligated to A, and the ligation product is subsequently cleaved by an endonuclease (Reaction 0.1). Next, I is "modified" by an assisting-molecule, depicted as a pink (grey) line segment, and restored to its default configuration (Reaction 0.2). The "modification" will be implemented as ligation and cleavage events and will be described in detail in Section 3. In a parallel reaction 0.3, A is also modified by another assisting-molecule such that A will possess a complementary sticky end to B, and thus the reaction wave is ready to enter Stage 1 (Reaction 0.3).

In Stage 1, similar structural changes occur as in Stage 0. However, after reaction 1.1, A will possess a sticky end that encodes the state, i.e. color, information of itself, its left neighbor I, and its right neighbor B. In the ensuing reaction 1.2, a rule-molecule corresponding to a transition rule 1-d CA rule 110 recognizes A's sticky end and effects a state transition of molecule A. A will then be modified by an assisting-molecule and restored to its default configuration, encoding its new state. In the example shown in Figure 1, bottom line, a rule-molecule corresponding to rule 110 changes the color encoded in A from WHITE to BLACK. In the parallel reaction 1.3, B will be modified to posses a complementary sticky end to C.

In Stages 2 and 3, reactions of the same nature as in Stage 1 will occur. Details are omitted for brevity.

2.3 Information Flow

Here a short description of the information flow during the operation of the ADCA is given [19]. For ease of exposition, the information encoding DNA molecule is denoted as $X^a[y]^b$, where X is its duplex portion, $[y]$ is its sticky end portion, and a and b respectively represent the state information encoded in X and $[y]$. There are two ways to encode information a in the duplex X. In specific a is encoded as a unique DNA sequence GTA * ; and a is encoded as the number of base pairs (L bp in the Figure 1) between an endonuclease recognition site and the sticky end of DNA molecule. The sequence of the sticky end $[y]$, in this case CGC, encodes the state information b. Furthermore, $[\overline{y}]$ is used to denote the complementary sticky end of $[y]$. Finally, ligation and cleavage events are represented as follows: the ligation of two

* The primary nucleobases are cytosine (DNA and RNA), guanine (DNA and RNA), adenine (DNA and RNA), thymine (DNA) and uracil (RNA), abbreviated as C, G, A, T, and U, respectively.

molecules $X^a[y]^b$ and $[\overline{y}]^c Z^d$ is described by the equation:

$$X^a[y]^b + [\overline{y}]^c Z^d \rightarrow XY \tag{1}$$

Suppose XY incorporates an endonuclease recognition site and is cut into $X^{a'}[u]^{b'}$ and $[\overline{u}]^{c'} Z^{d'}$. This is represented as:

$$XZ \rightarrow X^{a'}[u]^{b'} + [\overline{u}]^{c'} Z^{d'} \tag{2}$$

Now, from the above two equations results,

$$X^a[y]^b + [\overline{y}]^c Z^d \rightarrow X^{a'}[u]^{b'} + [\overline{u}]^{c'} Z^{d'} \tag{3}$$

To demonstrate the practicality of the proposed design, a detailed description of the molecular implementation of the ADCA is given in [19]. More specifically, in [19], all the detailed info regarding the molecular implementation step-by-step as well as the complete set of DNA molecules that constitute ADCA evolution are found in the main text as well as in the appendix.

3 HYBRID AUTONOMOUS DNA CELLULAR AUTOMATA (HADCA)

3.1 HADCA Basics and the Corresponding Simulator

In this subsection the ADCA evolution based on rule 110 as well as on different rules than the one found in the original paper [20] is described. More specifically, rules 90 and 30 are also examined, selected mainly because of their previous application on pseudorandom number generation as found in literature [10]. However, the most interesting part refers to the simultaneous application in parallel of all the aforementioned rules resulting in Hybrid DNA Cellular Automata (HADCA) concept. It should be made clear that every other 1-d CA rule could be also selected without any loss of generality for the proposed HADCA design method. The key point of generating successful HADCA is located on the requested modifications of the molecular reactions for the development of the presented CAs as described in the following subsection. However, for implementation reasons, a simulator has been developed in Matlab that enables the HADCA generation for the researcher without any prior knowledge of the molecular implementation of the provided hybrid CA DNA model.

As mentioned before, each ADCA corresponds to the following initial molecular structural configuration: $IABCABCABC$ where A, B and C can be found in either white (W) or black (B) state corresponding to logic 0 or 1,

respectively, while the I state corresponds by default to white (W) or in other words to logic 0 for our simulator. It should be also mentioned, once again, that I, A, B and C are DNA molecules that consisted of the four well known DNA bases adenine (A), thymine (T), cytokine (C) and guanine (G) which are represented by numbers of the quaternary number system, namely 0, 1, 2 and 3 for our simulation needs.

The presented ADCA evolves based on some simple rules. For $t = 0$, the initial configuration of ADCA is one-dimensional array $[1, n]$ where n stands for the number of the ACDA cells. The cell state of an arbitrary cell, i.e. $c(t, i)$, where t the evolution time and i positive integer, depends on the state of the left neighboring cell of $c(t, i)$ found on two (2) previous time steps, i.e. $l(t-2, i-1)$, the state of the $c(t, i)$ itself during previous time step, i.e. $c(t-1, i)$ and the state of the right neighboring cell in just last time step, i.e. $r(t-1, i+1)$. The boundary conditions of the presented CA remain fixed, in other words the virtual cell that has been added as left neighboring cell of $c(t, 1)$ cell is always found on 0. Cell state changes only when triangle inequality $i \leq t$ applies.

In view of the foregoing, some simulation results for each of the tested rules will be provided. More specifically, evolution results after 60 time steps for rule 110 arrived from our simulator are provided in Figure 2 while the corresponding ADCA evolution is depictured in Figure 3, respectively. Beyond the fact that the aforementioned rule applied also in [20], rule 110 has been proven to be a Turing machine able for universal computing.

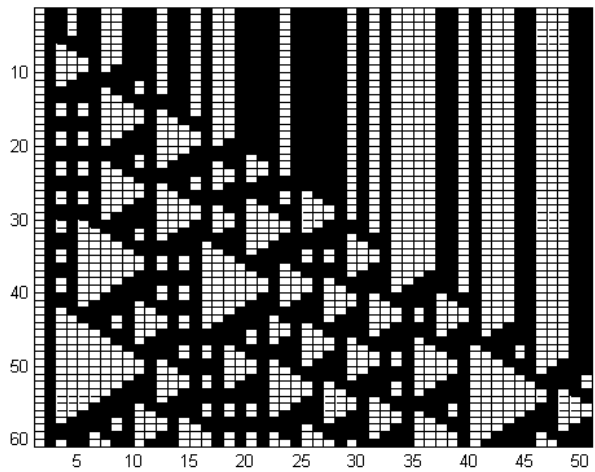

FIGURE 2
Simulation of ADCA rule 110 evolution

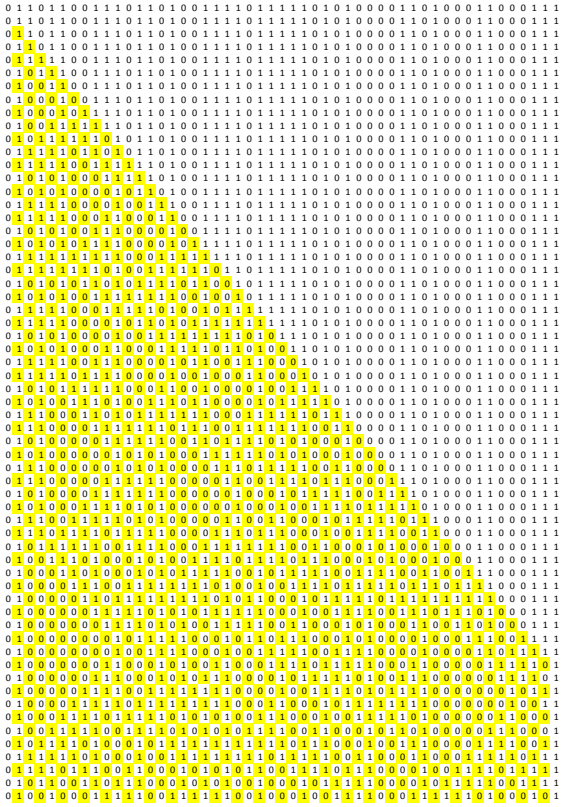

FIGURE 3
ADCA evolution for rule 110 (CA cells subject to change are depictured with yellow colour)

The application of another CA rule, namely rule 90 than the one proposed in [20] resulted in somehow different behavior for ADCA as found in Figures 4 and 5. It is clear that three (3) initial 3-tuples out of 2^3 3-tuples are different in case of rule 90 when compared with rule 110 evolution. We will use the aforementioned B and W letter convention for describing these 3-tuples [8], i.e. f(WBW), f(BWW), f(BWB), where function f corresponds to any transition rule, resulting B, W and B, in case of rule 110 and W, B and W in case of rule 90, respectively. How these different results for the exact initial 3-tuples affect the information flow of the ADCA? In case of rule 110, WBW and BWW result in the same state as the previous ones meaning that the molecular reactions taken into account are $i.1$, $i.2$ and $i.3$ [20], while in case BWB the new state is different than the one found before and eventually,

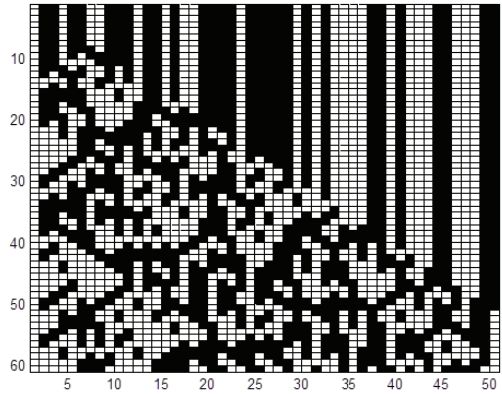

FIGURE 4
Simulation of ADCA rule 90 evolution.

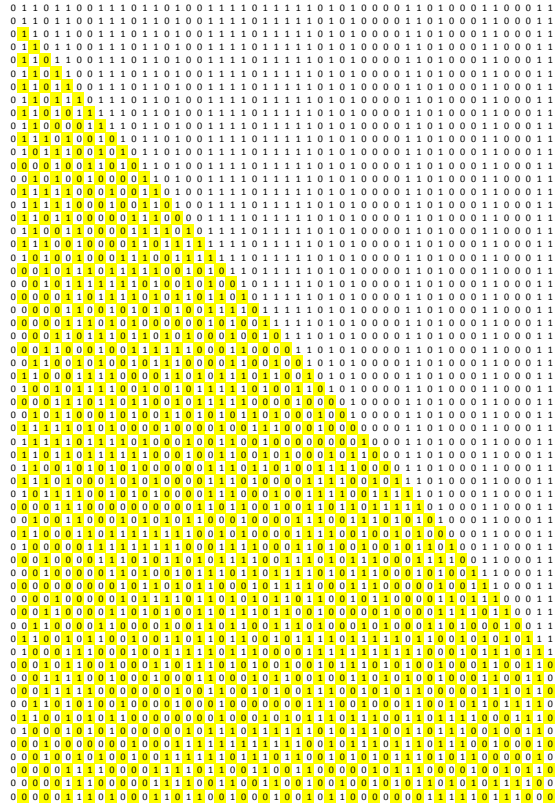

FIGURE 5
ADCA evolution for rule 90 (CA cells subject to change are depictured with yellow colour).

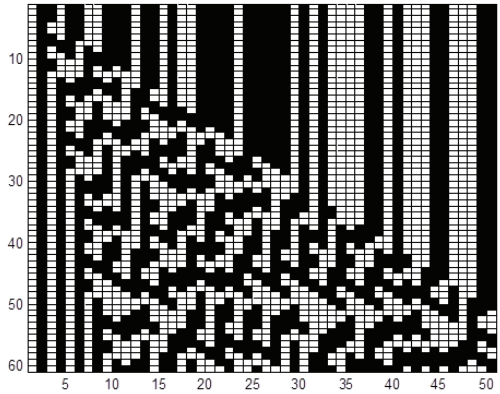

FIGURE 6
Simulation of ADCA rule 30 evolution.

molecular reactions $i.1$, $i.2.1$, $i.2.2$ and $i.3$, occur where $i = 1, 2, 3, 4$. In case of rule 90, WBW and BWW configurations result in news states compared with the previous one meaning that the molecular reactions taken into account are $i.1$, $i.2.1$, $i.2.2$ and $i.3$, while in case BWB the new state is the same as before eventually, molecular reactions $i.1$, $i.2$ and $i.3$, take place where $i = 1, 2, 3, 4$. For the rest of the 3-tuples rules 110 and 90 result in identical molecular reactions. More specifically, when WWW, WBB and BBW occur new state remains the same as before and eventually, molecular reactions $i.1$, $i.2$ and $i.3$, take place, while in WWB the new state is different than the one found before and eventually, molecular reactions $i.1$, $i.2.1$, $i.2.2$ and $i.3$, occur where $i = 1, 2, 3, 4$.

The application of another CA rule, namely rule 30, has been also examined as depicted in Figures 6 and 7. Using Wolfram's classification scheme, Rule 30 is a Class III rule, displaying aperiodic, chaotic behaviour. This rule is of particular interest because it produces complex, seemingly random patterns from simple, well-defined rules. Because of this, rule 30 has also been used as a random number generator in Mathematica, [3] and has also been proposed as a possible stream cipher for use in cryptography. As in case of rules 110 and 90, certain similarities and differences arise in case of rule 30 ADCA evolution, when molecular reactions occur based mostly on the 3-tuples that result in different states when applied to each of the aforementioned rules. For example, in case of rule 30 initial configurations BWW, BWB and BBW result in different states when applied to rules 110 and 90. However, in two of these cases (BWW and BWB) rule 30 results the same as rule 90 and the only case that rule 30 differs from both aforementioned rules is BBW. As before, while for rule 110, the BBW configuration results

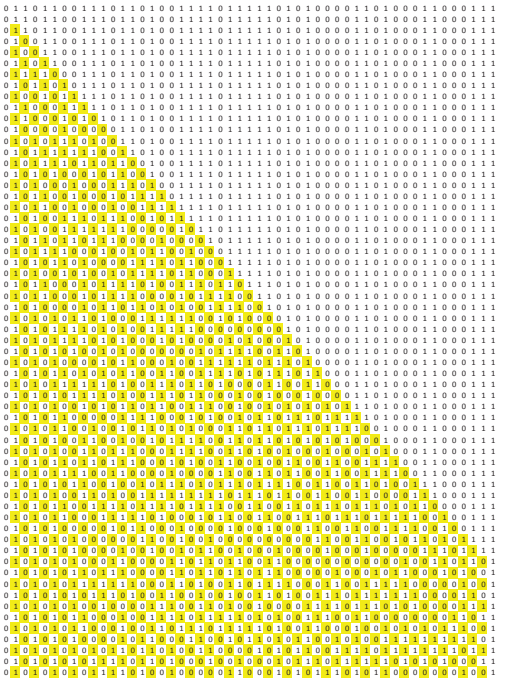

FIGURE 7
ADCA evolution for rule 30 (CA cells subject to change are depictured with yellow colour).

in a new state the same as before, and the molecular reactions $i.1$, $i.2$ $i.3$, take place, for rule 30, this case, namely BBW, results in a different new state compared to the one before and therefore the molecular reactions $i.1$, $i.2.1$, $i.2.2$ and $i.3$, will happen where $i = 1, 2, 3, 4$. For the rest of the cases the exact same reactions occur for any of the above CA rules.

It is now clear that the provided simulator can evolve any of the 1-d CA rules, taking into account Wolfram's decimal numbering convention for describing these rules and the aforementioned molecular reactions modifications. The user does not need to know the biological "background" of the under study rules and the simulations are in every case straightforward, since the developed code detects and provides automatically the requested changes to information flow of molecular reactions.

Moreover, in the context of the aforementioned simulator simultaneous application in parallel of all the aforementioned rules resulting in Hybrid DNA Cellular Automata (HADCA) concept is also described. In specific, for most of the CA cells, (for visualization reasons we have chosen the number of CA cells to be equal to 50) rules 110 applies. For $15 \leq i < 39$, where $i \in$

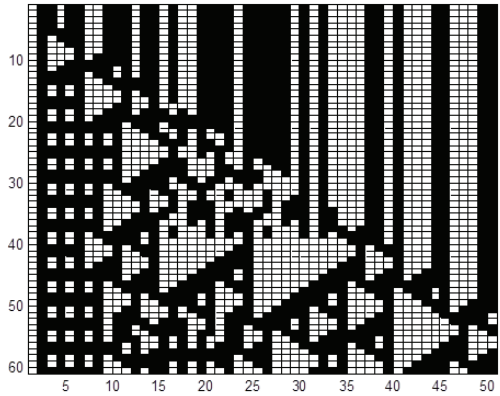

FIGURE 8
Simulation of HADCA rules 110, 90 and 30 evolution in parallel.

[1, 50] rule 90 applies for time t, $15 \leq t < 39$. Furthermore, rule 90 applies also for the cells where $9 \leq i < 24$ but this time for time t, $51 \leq t < 55$ in order to confirm that cells change their state when triangle inequality $i \leq t$ applies. Finally, for CA cells where $24 \leq i < 33$ rule 30 applies for time steps $24 \leq t < 33$. As mentioned before for the rest of CA cells same rule, namely 110 applies when inequality $i \leq t$ applies. The HADCA results are depictured in Figures 8 and 9 for 60 time steps colored by the application of each rule, respectively.

Finally, we have tested our simulator and the corresponding HADCA for different initial sequences when applying all the aforementioned rules in parallel. In such a way we are trying to investigate the applicability of the proposed scheme under random initial conditions. It is clear as found in Figures 10 and 11 after the rules parallel application for 60 time steps as described above, where results are colored by the application of each rule, namely once again 30, 90 and 110, respectively, that the proposed scheme succeeds to reproduce the requested results without problems.

It should be noticed that the above choices regarding the length and evolution time of each of the aforementioned rules were arbitrary and can be easily modified by the simulator user. However, to apply these rules several modifications on the original information flow of the ADCA model should be made.

3.2 Modifications to Molecular Reactions Based on Simultaneous Application of Different CA rules in Parallel

In this subsection the molecular reactions taking place in every stage as well as the resulting modifications caused by different CA rule application are

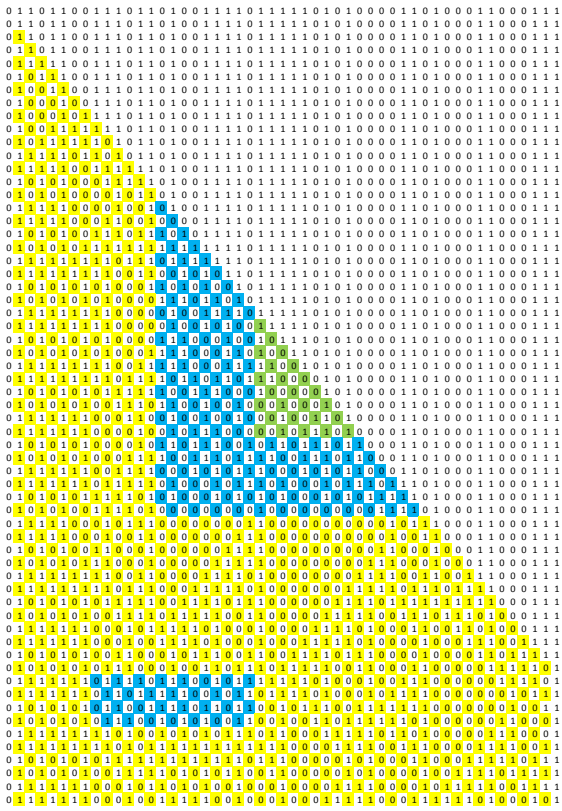

FIGURE 9
HADCA evolution for simultaneous application of rules 110, 90 and 30 in parallel. CA cells subject to change are depictured with colour. Rule 110 applies to yellow cells, while rules 90 and 30 apply to blue and green ones, respectively.

presented. It should be mentioned that the below modifications can be applied to any other 1-d CA rules without loss of generality. In such a way the embedded changes of the HADCA as depicted in the aforementioned simulator allow the user to develop any rules combination of the hybrid DNA CA with minimum effort.

In ***stage 0***, for both possible cases (WWW and WWB) since $a \in$W,B, all three aforementioned rules 110, 90 and 30 will be identical. In case WWW, the new state will be W, while in case WWB the resulting state will be B. Therefore, the same molecules will be used for all the molecular reactions in every case (WWW WWB) the used molecules will be the same no matter which rule is taken into account. The only difference between WWW and WWB is found on the application of different molecules R^{ia} and T^{ia},

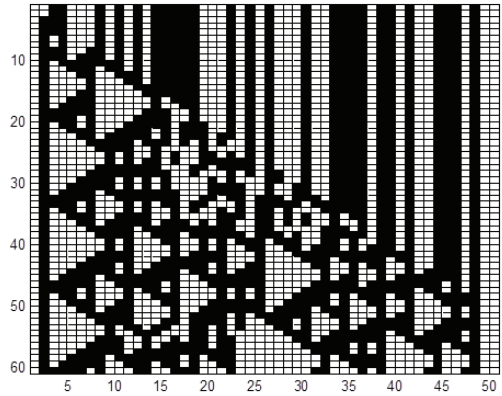

FIGURE 10
Simulation of HADCA rules 110, 90 and 30 evolution in parallel with random initial conditions.

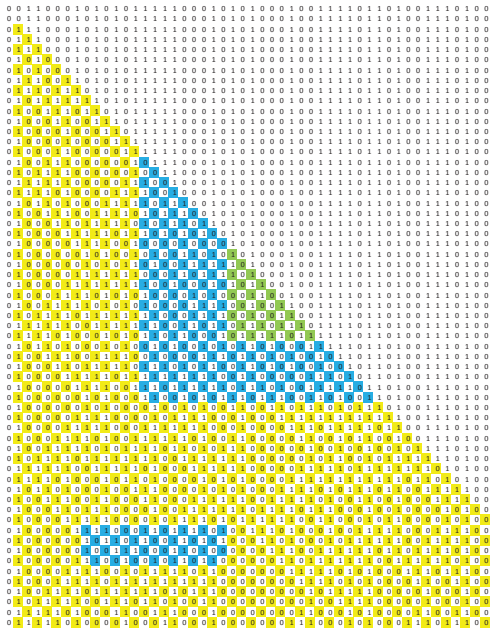

FIGURE 11
HADCA evolution for simultaneous application of rules 110, 90 and 30 in parallel. As before the same rules for the exact specified conditions have been considered. The only difference compared to Figure 9 is the application of the whole scheme to different initial conditions. Once again, CA cells subject to change are depictured with colour. More specifically, rule 110 applies to yellow cells, while rules 90 and 30 apply to blue and green ones, respectively.

respectively. Consequently, the following reactions apply:

$$\text{Reaction 0.1}: \quad I^{i}[u] + [\overline{u}]A^{a} \rightarrow I[t]^{ia} + [\overline{t}]^{ia}A \tag{4}$$

$$\text{Reaction 0.2}: \quad I[t]^{ia} + [\overline{t}]^{ia}R \rightarrow I^{i}[u] + [\overline{u}]R \tag{5}$$

$$\text{Reaction 0.3}: \quad T^{ia}[t]^{ia} + [\overline{t}]^{ia}A \rightarrow T[\overline{v}] + [v]A^{ia} \tag{6}$$

resulting to the following sequence of reactions:

$$S_{0,1} = (0.1, 0.2, 0.3). \tag{7}$$

In ***stage 1*** we have four different cases, due to the fact i=W.

1st case when iab=WWW.

In this case all three aforementioned rules 110, 90 and 30 will be identical. The new state will remain W and the same molecules will be used for the resulting reactions no matter which rule is taken into account.

$$\text{Reaction 1.1}: \quad A^{ia}[v] + [\overline{v}]B^{b} \rightarrow A[t]^{iab} + [\overline{t}]^{iab}B \tag{8}$$

$$\text{Reaction 1.2}: \quad A^{q}[t]^{iab} + [\overline{t}]^{iab}R^{a'} \rightarrow A^{a'}[\overline{u}] + [u]R \tag{9}$$

$$\text{Reaction 1.3}: \quad T^{ab}[t]^{iab} + [\overline{t}]^{iab}B \rightarrow T[\overline{w}] + [w]B^{ab} \tag{10}$$

thus, resulting to the following sequence of reactions:

$$S_{1,1} = (1.1, 1.2, 1.3) \tag{11}$$

2nd case when iab=WWB.

As before all three aforementioned rules 110, 90 and 30 will be identical. The new state will change to B and for the resulting reactions, the same molecules will be used no matter which rule is taken into consideration. The following additional reactions take place as follows:

$$\text{Reaction 1.1}: \quad A^{ia}[v] + [\overline{v}]B^{b} \rightarrow A[t]^{iab} + [\overline{t}]^{iab}B \tag{12}$$

$$\text{Reaction 1.2.1}: \quad A^{q}[t]^{iab} + [\overline{t}]^{iab}R^{a'} \rightarrow A[e]^{qa'} + [\overline{e}]^{qa'}R \tag{13}$$

$$\text{Reaction 1.2.2}: \quad A[e]^{qa'} + [\overline{e}]^{qa'}E \rightarrow A^{a'}[\overline{u}] + [u]E \tag{14}$$

$$\text{Reaction 1.3}: \quad T^{ab}[t]^{iab} + [\overline{t}]^{iab}B \rightarrow T[\overline{w}] + [w]B^{ab} \tag{15}$$

Stage 1 : iab		Rule 110	Rule 90	Rule 30
1	WWW	$S_{1,1}$	$S_{1,1}$	$S_{1,1}$
2	WWB	$S_{1,2}$	$S_{1,2}$	$S_{1,2}$
3	WBW	$S_{1,1}$	$S_{1,2}$	$S_{1,1}$
4	WBB	$S_{1,1}$	$S_{1,1}$	$S_{1,1}$

TABLE 1
Four (4) cases of ***Stage 1*** (iab) and the required reactions emanate from the application of CA rules 110, 90 and 30, respectively.

and the corresponding sequence of reactions will be:

$$S_{1,2} = (1.1, 1.2.1, 1.2.2, 1.3) \tag{16}$$

3rd case when iab=WBW.

In this case, rules 110 and 30 result in the same manner while rule 90 differs. The new state for rules 110 and 30 is B, and for the resulting reactions, the same molecules will be used for rules 110 and 30, respectively. The reactions take place exactly like in the aforementioned 1st case, since a does not change its state. Consequently, the sequence of reactions found in equation 11, i.e. $S_{1,1} = (1.1, 1.2, 1.3)$ will apply. The new state of rule 90 will change to W and the molecules that will be used for the molecular reactions will differ from the ones used for rules 110 and 30. More specifically, all the reactions look alike the 2nd case reactions since a changes its state, i.e. sequence $S_{1,2} = (1.1, 1.2.1, 1.2.2, 1.3)$ of equation (16) follows.

4th case when iab=WBB. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new case will be still and for the resulting reactions, the same molecules will be used no matter which rules apply to. All the corresponding reactions are exactly the same with the 1st case due to the fact that a does not change and sequence $S_{1,1} = (1.1, 1.2, 1.3)$ of equation (11) is applied to.

Table 1 summarizes all four (4) cases of ***Stage 1*** according to the applied rules and the corresponding reactions:

Stage 2 corresponds to eight different cases.

1st case when abc=WWW. In this case all three aforementioned rules are identical. The new state will remain W and the same molecules will be used for the resulting reactions no matter which rule is taken into account. The following reactions take place:

$$\text{Reaction 2.1}: \quad B^{ab}[w] + [\overline{w}]C^{c} \rightarrow B[t]^{abc} + [\overline{t}]^{abc}C \tag{17}$$

$$\text{Reaction 2.2}: \quad B^{q}[t]^{abc} + [\overline{t}]^{abc}R^{b'} \rightarrow B^{b'}[\overline{v}] + [v]R \tag{18}$$

$$\text{Reaction 2.3}: \quad T^{bc}[t]^{abc} + [\overline{t}]^{abc}C \rightarrow T[\overline{u}] + [u]C^{bc}, \tag{19}$$

and the resulting sequence of reactions is as follows:

$$S_{2,1} = (2.1, 2.2, 2.3) \tag{20}$$

2nd case when abc=WWB. In this case all three aforementioned rules are identical. The new state will change to B and for the resulting reactions, the same molecules will be used no matter which rule is taken into consideration. The following reactions take place:

$$\text{Reaction 2.1}: \quad B^{ab}[w] + [\overline{w}]C^{c} \rightarrow B[t]^{abc} + [\overline{t}]^{abc}C \tag{21}$$

$$\text{Reaction 2.2.1}: \quad B^{q}[t]^{abc} + [\overline{t}]^{abc}R^{b'} \rightarrow B[e]^{qb'} + [\overline{e}]^{qb'}R \tag{22}$$

$$\text{Reaction 2.2.2}: \quad B[e]^{qb'} + [\overline{e}]^{qb'}E \rightarrow B^{b'}[\overline{v}] + [v]E \tag{23}$$

$$\text{Reaction 2.3}: \quad T^{bc}[t]^{abc} + [\overline{t}]^{abc}C \rightarrow T[\overline{u}] + [u]C^{bc} \tag{24}$$

and the resulting sequence of reactions is as follows:

$$S_{2,2} = (2.1, 2.1.2, 2.2.2, 2.3) \tag{25}$$

3rd case when abc=WBW.

In this case, rules 110 and 30 result in the same manner while rule 90 differs. The new state for rules 110 and 30 is B, and for the resulting reactions, the same molecules will be used for rules 110 and 30, respectively. The reactions take place exactly like in the aforementioned 1st case, since b does not change its state. Consequently, sequence $S_{2,1} = (2.1, 2.2, 2.3)$ of equation (20) is applied to. The new state of rule 90 will change to W and the molecules that will be used for the molecular reactions will differ from the ones used for rules 110 and 30. More specifically, all the reactions look alike the 2nd case reactions since b changes its state and the sequence of reactions $S_{2,2} = (2.1, 2.1.2, 2.2.2, 2.3)$ of equation (25) takes place.

4th case when abc=WBB.

In this case all three aforementioned rules 110, 90 and 30 will be identical, the new state will be still and for the resulting reactions, the same molecules will be used no matter which rules apply to, and, as a result, sequence $S_{2,1} = (2.1, 2.2, 2.3)$ takes place.

5th case when abc=BWW. In this case, rules 90 and 30 result in the same manner while rule 110 differs. The new state for rule 110 remains W. b does not change its state and the reactions happen alike $S_{2,1} = (2.1, 2.2, 2.3)$. The new state of rules 90 and 30 will change to B and the molecules that will be

used for the molecular reactions will differ from the ones used for rule 110. Since b changes its state, $S_{2,2} = (2.1, 2.1.2, 2.2.2, 2.3)$ reactions take place.

6th case case when abc=BWB. In this case, rules 90 and 30 result in the same manner while rule 110 differs. The new state of rule 110 changes to B. Since b changes its state, $S_{2,2} = (2.1, 2.1.2, 2.2.2, 2.3)$ reactions take place. The new state of rule 90 and 30 will remain to W and the molecules that will be used for the molecular reactions will differ from the ones used for rules 110. Since b does not change its state, $S_{2,1} = (2.1, 2.2, 2.3)$ reactions take place.

7th case case when abc=BBW. In this case, rules 110 and 90 result in the same manner while rule 30 differs. The new state for rules 110 and 90 remains B, and for the resulting reactions, the same molecules will be used for rules 110 and 90, respectively. Once again, b does not change its state, and the reactions of $S_{2,1} = (2.1, 2.2, 2.3)$ are used in the same way as before. The new state of rule 30 will change to W and the molecules that will be used for the molecular reactions will differ from the ones used for rules 110 and 90. More specifically, in this case b changes its state and all the reactions follow, i.e. $S_{2,2} = (2.1, 2.1.2, 2.2.2, 2.3)$.

8th case case when abc=BBB. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new state will change to B and for the resulting reactions, the same molecules will be used no matter which rules apply to. Consequently, all the corresponding reactions apply one after the other like in $S_{2,2} = (2.1, 2.1.2, 2.2.2, 2.3)$.

Once again Table 2 summarizes all eight (8) cases of ***Stage 2*** according to the applied rules and the corresponding reactions.

As found with the previous stages, ***Stage 3*** also corresponds to eight different cases as follows.

1st case when bca=WWW. In this case all three aforementioned rules are identical. The new state will remain W and the same molecules will be used

Stage 2 : *abc*		**Rule 110**	**Rule 90**	**Rule 30**
1	WWW	$S_{2,1}$	$S_{2,1}$	$S_{2,1}$
2	WWB	$S_{2,2}$	$S_{2,2}$	$S_{2,2}$
3	WBW	$S_{2,1}$	$S_{2,2}$	$S_{2,1}$
4	WBB	$S_{2,1}$	$S_{2,1}$	$S_{2,1}$
5	BWW	$S_{2,1}$	$S_{2,2}$	$S_{2,2}$
6	BWB	$S_{2,2}$	$S_{2,1}$	$S_{2,1}$
7	BBW	$S_{2,1}$	$S_{2,1}$	$S_{2,2}$
8	BBB	$S_{2,2}$	$S_{2,2}$	$S_{2,2}$

TABLE 2
Eight (8) cases of ***Stage 2*** (abc) and the required reactions emanate from the application of CA rules 110, 90 and 30, respectively.

for the resulting reactions no matter which rule is taken into account. Namely, the following reactions take place:

$$\text{Reaction 3.1}: \quad C^{bc}[u] + [\overline{u}]A^{a} \rightarrow C[t]^{bca} + [\overline{t}]^{bca}A \tag{26}$$

$$\text{Reaction 3.2}: \quad C^{q}[t]^{bca} + [\overline{t}]^{bca}R^{c'} \rightarrow C^{c'}[\overline{w}] + [w]R \tag{27}$$

$$\text{Reaction 3.3}: \quad T^{ca}[t]^{bca} + [\overline{t}]^{bca}A \rightarrow T[\overline{v}] + [v]A^{ca} \tag{28}$$

resulting to the following sequence of reactions:

$$S_{3,1} = (3.1, 3.2, 3.3) \tag{29}$$

2nd case when *bca*=WWB. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new state will change to B and for the resulting reactions, the same molecules will be used no matter which rules apply to. In this case, reactions as found in the following equations occur:

$$\text{Reaction 3.1}: \quad C^{bc}[u] + [\overline{u}]A^{a} \rightarrow C[t]^{bca} + [\overline{t}]^{bca}A \tag{30}$$

$$\text{Reaction 3.2.1}: \quad C^{q}[t]^{bca} + [\overline{t}]^{bca}R^{c'} \rightarrow C[e]^{qc'} + [\overline{e}]^{qc'}R \tag{31}$$

$$\text{Reaction 3.2.2}: \quad C[e]^{qc'} + [\overline{e}]^{qc'}E \rightarrow C^{c'}[\overline{w}] + [w]E \tag{32}$$

$$\text{Reaction 3.3}: \quad T^{ca}[t]^{bca} + [\overline{t}]^{bca}A \rightarrow T[\overline{v}] + [v]A^{ca} \tag{33}$$

resulting to the following sequence of reactions:

$$S_{3,2} = (3.1, 3.2.1, 3.2.2, 3.3) \tag{34}$$

3rd case when *bca*=WBW. In this case, rules 110 and 30 result in the same manner while rule 90 differs. The new state for rules 110 and 30 is B, and for the resulting reactions, the same molecules will be used for rules 110 and 30, respectively. The reactions take place exactly like in the aforementioned 1st case, i.e. sequence of reactions $S_{3,1} = (3.1, 3.2, 3.3)$, since *c* does not change its state. The new state of rule 90 will change to W and the molecules that will be used for the molecular reactions will differ from the ones used for rules 110 and 30. More specifically, all the reactions look alike the 2nd case reactions, i.e. $S_{3,2} = (3.1, 3.2.1, 3.2.2, 3.3)$ since *c* changes its state.

4th case when *bca*=WBB. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new state will be still, *c* does not change and for the resulting reactions, i.e. $S_{3,1} = (3.1, 3.2, 3.3)$ the same molecules will be used no matter which rules apply to.

5th case when *bca*=BWW. In this case, rules 90 and 30 result in the same manner while rule 110 differs. The new state for rule 110 remains W and since *b* does not change its state, reactions in $S_{3,1} = (3.1, 3.2, 3.3)$ take place. The new state of rules 90 and 30 will change to B and the molecules that will be used for the molecular reactions in this case, i.e. $S_{3,2} = (3.1, 3.2.1, 3.2.2, 3.3)$, will differ from the ones used for rule 110, since *c* changes its state.

6th case case when *bca*=BWB. In this case, rules 90 and 30 result in the same manner while rule 110 differs. The new state of rule 110 changes to B and since *c* changes its state, the reactions in $S_{3,2} = (3.1, 3.2.1, 3.2.2, 3.3)$ take place. The new state of rule 90 and 30 will remain to W and the molecules that will be used for the molecular reactions in this case, i.e. $S_{3,1} = (3.1, 3.2, 3.3)$, will differ from the ones used for rule 110, since *c* does not change its state.

7th case when *bca*=BBW. In this case, rules 110 and 90 result in the same manner while rule 30 differs. The new state for rules 110 and 90 remains B, and since *c* does not change its state, reactions in $S_{3,1} = (3.1, 3.2, 3.3)$ take place. The new state of rule 30 will change to W and the molecules that will be used for the molecular reactions in this case, i.e. $S_{3,2} = (3.1, 3.2.1, 3.2.2, 3.3)$, will differ from the ones used for rules 110 and 90, since *c* changes its state.

8th case when *bca*=BBB. In this case all three aforementioned rules 110, 90 and 30 will be identical, the new state will change to W since *c* changes its state and for the resulting reactions in $S_{3,2} = (3.1, 3.2.1, 3.2.2, 3.3)$, the same molecules will be used no matter which rules apply to.

As before Table 3 summarizes all eight (8) cases of ***Stage 3*** according to the applied rules and the corresponding reactions.

Finally, ***Stage 4*** corresponds to the eight different following cases.

Stage 3 : *bca*		**Rule 110**	**Rule 90**	**Rule 30**
1	WWW	$S_{3,1}$	$S_{3,1}$	$S_{3,1}$
2	WWB	$S_{3,2}$	$S_{3,2}$	$S_{3,2}$
3	WBW	$S_{3,1}$	$S_{3,2}$	$S_{3,1}$
4	WBB	$S_{3,1}$	$S_{3,1}$	$S_{3,1}$
5	BWW	$S_{3,1}$	$S_{3,2}$	$S_{3,2}$
6	BWB	$S_{3,2}$	$S_{3,1}$	$S_{3,1}$
7	BBW	$S_{3,1}$	$S_{3,1}$	$S_{3,2}$
8	BBB	$S_{3,2}$	$S_{3,2}$	$S_{3,2}$

TABLE 3
Eight (8) cases of ***Stage 3*** (*bca*) and the required reactions emanate from the application of CA rules 110, 90 and 30, respectively.

1st case when cab=WWW. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new state will remain W and the same molecules will be used for the resulting reactions in this case no matter which rule is taken into account. Namely, the following reactions take place:

$$\text{Reaction 4.1}: \quad A^{ca}[v] + [\bar{v}]B^{b} \rightarrow A[t]^{cab} + [\bar{t}]^{cab}B \tag{35}$$

$$\text{Reaction 4.2}: \quad A^{q}[t]^{cab} + [\bar{t}]^{cab}R^{a'} \rightarrow A^{a'}[\bar{u}] + [u]R \tag{36}$$

$$\text{Reaction 4.3}: \quad T^{ab}[t]^{cab} + [\bar{t}]^{cab}B \rightarrow T[\bar{w}] + [w]B^{ab} \tag{37}$$

thus, resulting to the corresponding sequence of reactions given below:

$$S_{4,1} = (4.1, 4.2, 4.3) \tag{38}$$

2nd case when cab=WWB. As before all three aforementioned rules 110, 90 and 30 will be identical. The new state will change to B and for the resulting reactions, the same molecules will be used no matter which rule is taken into consideration. In this case, reactions as found in the following equations occur:

$$\text{Reaction 4.1}: \quad A^{ca}[v] + [\bar{v}]B^{b} \rightarrow A[t]^{cab} + [\bar{t}]^{cab}B \tag{39}$$

$$\text{Reaction 4.2.1}: \quad A^{q}[t]^{cab} + [\bar{t}]^{cab}R^{a'} \rightarrow A[e]^{qa'} + [\bar{e}]^{qa'}R \tag{40}$$

$$\text{Reaction 4.2.2}: \quad A[e]^{qa'} + [\bar{e}]^{qa'}E \rightarrow A^{a'}[\bar{u}] + [u]E \tag{41}$$

$$\text{Reaction 4.3}: \quad T^{ab}[t]^{iab} + [\bar{t}]^{iab}B \rightarrow T[\bar{w}] + [w]B^{ab} \tag{42}$$

resulting to the corresponding sequence of reactions given below:

$$S_{4,2} = (4.1, 4.2.1, 4.2.2, 4.3) \tag{43}$$

3rd case when cab=WBW. In this case, rules 110 and 30 are identical, while rule 90 differs. The new state for rules 110 and 30 is B, and since a does not change its state, for the resulting reactions of sequence $S_{4,1} = (4.1, 4.2, 4.3)$, the same molecules will be used for rules 110 and 30, respectively. The new state of rule 90 will change to W and since a changes its state, the molecules that will be used for the molecular reactions will differ from the ones used for rules 110 and 30. More specifically, all the reactions look alike the 2nd case reactions, i.e. $S_{4,2} = (4.1, 4.2.1, 4.2.2, 4.3)$ since a changes its state.

4th case when cab=WBB. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new case will be still and since a does

not change for the resulting reactions, i.e. $S_{4,1} = (4.1, 4.2, 4.3)$, the same molecules will be used no matter which rules apply to.

5th case when *cab*=BWW. In this case, rules 90 and 30 result in the same manner while rule 110 differs. The new state for rule 110 remains W and since *a* does not change its state, the reactions in $S_{4,1} = (4.1, 4.2, 4.3)$ take place. The new state of rules 90 and 30 will change to B and the molecules that will be used for the molecular reactions in this case $S_{4,2} = (4.1, 4.2.1, 4.2.2, 4.3)$ will differ from the ones used for rule 110, since *a* changes its state.

6th case case when *cab*=BWB. In this case, rules 90 and 30 result in the same manner while rule 110 differs. The new state of rule 110 changes to B and since *a* changes its state, the following reactions $S_{4,2} = (4.1, 4.2.1, 4.2.2, 4.3)$ take place. The new state of rule 90 and 30 will remain to W and the molecules that will be used for the molecular reactions in this case $S_{4,1} = (4.1, 4.2, 4.3)$ will differ from the ones used for rule 110, since *a* does not change its state.

7th case when *cab*=BBW. In this case, rules 110 and 90 result in the same manner while rule 30 differs. The new state for rules 110 and 90 remains B, and since *a* does not change its state for the resulting reactions, namely $S_{4,1} = (4.1, 4.2, 4.3)$, the same molecules will be used for rules 110 and 90, respectively. The new state of rule 30 will change to W and the molecules that will be used for the molecular reactions in this case $S_{4,2} = (4.1, 4.2.1, 4.2.2, 4.3)$ will differ from the ones used for rules 110 and 90, since *a* changes its state.

8th case when *cab*=BBB. In this case all three aforementioned rules 110, 90 and 30 will be identical. The new state will change to W and since *a* changes its state for the resulting reactions of $S_{4,2} = (4.1, 4.2.1, 4.2.2, 4.3)$, the same molecules will be used no matter which rules apply to.

As before Table 4 summarizes all eight (8) cases of ***Stage 4*** according to the applied rules and the corresponding reactions.

Stage *4* : *cab*		**Rule 110**	**Rule 90**	**Rule 30**
1	WWW	$S_{4,1}$	$S_{4,1}$	$S_{4,1}$
2	WWB	$S_{4,2}$	$S_{4,2}$	$S_{4,2}$
3	WBW	$S_{4,1}$	$S_{4,2}$	$S_{4,1}$
4	WBB	$S_{4,1}$	$S_{4,1}$	$S_{4,1}$
5	BWW	$S_{4,1}$	$S_{4,2}$	$S_{4,2}$
6	BWB	$S_{4,2}$	$S_{4,1}$	$S_{4,1}$
7	BBW	$S_{4,1}$	$S_{4,1}$	$S_{4,2}$
8	BBB	$S_{4,2}$	$S_{4,2}$	$S_{4,2}$

TABLE 4
Eight (8) cases of ***Stage 4*** (*cab*) and the required reactions emanate from the application of CA rules 110, 90 and 30, respectively.

Stage $i = \begin{cases} 2: abc \\ 3: bca \\ 4: cab \end{cases}$		Rule 110	Rule 90	Rule 30
1	WWW	$S_{i,1}$	$S_{i,1}$	$S_{i,1}$
2	WWB	$S_{i,2}$	$S_{i,2}$	$S_{i,2}$
3	WBW	$S_{i,1}$	$S_{i,2}$	$S_{i,1}$
4	WBB	$S_{i,1}$	$S_{i,1}$	$S_{i,1}$
5	BWW	$S_{i,1}$	$S_{i,2}$	$S_{i,2}$
6	BWB	$S_{i,2}$	$S_{i,1}$	$S_{i,1}$
7	BBW	$S_{i,1}$	$S_{i,1}$	$S_{i,2}$
8	BBB	$S_{i,2}$	$S_{i,2}$	$S_{i,2}$

TABLE 5
The general table of the eight (8) cases corresponding to different ***Stages*** i.e. ***2***, ***3*** and ***4*** and the required reactions emanate from the application of CA rules 110, 90 and 30, respectively.

In the next stages of the HADCA methodology stages 2, 3 and 4 are repeated. In general someone could say that the modifications needed for the all cases of these stages are summarised in Table 5. Consequently, with slight modifications the proposed methodology can reproduce any CA rule and most important allow the parallel application of any desired combination of 1-d CA rules.

4 HADCA PRNG

Real random numbers can only be obtained from physical phenomena. The random numbers generated by PRNGs are not truly random, but only pseudorandom. Although this assertion is inevitable, we would still like to obtain sequences that behave as if they were random. Thus, the problem is how to decide whether the sequences are random enough. Statistical (empirical) tests could be a good solution in this respect. If a sequence passed a number of quantitative tests, we could assert that the sequence is random. But we should also note that there are no guarantees, only predictions in numerical practice are possible.

In general, there are four aspects of CA configuration affecting the randomness: transition rule, initial seed, i.e. the initial state configuration in CA, length of the CA, i.e. the total number of cells in a CA, time of execution and boundary conditions. In more details, the randomness of the sequences generated are greatly affected by the rules used, the length of the CA increases the maximum possible cycle length of the pseudorandom sequence, while the effect of initial seed on randomness is trivial. In our case in order to accomplish the aforementioned requirements so as to generate high-quality random numbers, we decided to use the aforementioned three different rules when

applied to CA grid with the help of a real time computer clock sequence taking into account all the resulting numbers of the clock sequence. More specifically, the length and the time interval of each of the above rules result from different products of all the above numbers, namely day, month, year, hour, min and seconds was calculated. The result of this operation produced a binary number which indicated the initial CA configuration and simultaneously the length of the CA. More details regarding the usage of the aforementioned idea can be found on [10]. The random number sequences produced by the HADCA as described above are coded in hexadecimal form.

We used DIEHARD test suites values [13]. Generally, a PRNG which can pass DIEHARD can be considered to present adequate randomness for most applications. The DIEHARD battery consists of 18 different, independent statistical tests. The results of these tests are real valued, between 0 and 1, and are referred to as "P-values". For any given test, a P-value between 0.025 and 0.975 corresponds to a pass at the 0.05 level. A complete description of all the tests in DIEHARD is available in [13]. The proposed HADCA passes successfully the statistical DIEHARD tests. It should also noticed that compared to other PRNGs the proposed HADCA approach is able to generate "true" random numbers, since the described methodology uses, if we can say so, random DNA events without taking into consideration any further random "sources". Finally, some simulation results of the proposed HADCA PRNG are already depictured in Figures 9 and 11.

5 CONCLUSIONS

In this paper 1-d Hybrid Autonomous DNA Cellular Automata (HADCA) were presented. HADCA are able to run simultaneous different CA rules in parallel with certain modifications on their molecular implementation and information flow compared to their origins. A detailed methodology that describes the requested modifications on information flow and molecular reactions caused by different CA rule application was also introduced. In such a way the embedded changes of the HADCA as depicted in the proposed simulator allow the user to develop any rules combination of the hybrid DNA CA with minimum effort. Finally, it was shown that the proposed 1-d HADCA can generate high-quality random numbers which can pass the statistical tests of DIEHARD one of the most well known general test suites for randomness.

As part of future research, taking into account that the batteries of test have been evolved in the past years, the proposed HADCA will be tested with different test suites for randomness like NIST [14] and "TestU01" [15], which can be considered new particularly hard batteries that "stresses" even high-quality PRNGs. Moreover, based on the proposed methodology we will

extent the HADCA usage taking into account all the 1-d CA rules so as to produce a real life RNG. Finally, we do expect that with further enhancements of the proposed methodology to implement 2-d HADCA, several other engineering related problems will be also possible to be solved.

REFERENCES

[1] S. Nandi, B.K. Kar, and P. Pal Chaudhuri, "Theory and applications of cellular automata in cryptography," *IEEE Transactions on Computers*, vol. 43, pp. 1346–1357, 1994.

[2] I. Kokolakis, I. Andreadis, and Ph. Tsalides, "Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators," *Microprocessors and Microsystems*, vol. 20, pp. 643–658, 1997.

[3] S. Wolfram. Cellular Automata and Complexity. Addison-Wesley, New York, 1994.

[4] S. Wolfram, "Cryptography with cellular automata," in Proceedings of the Conference, CRYPTO 85 on Advances in Cryptography, Lecture Notes in Computer Science, vol. 218, pp. 429–432, 1985.

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

[6] 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, pp. 291–305, 1999.

[7] M. Tomassini, and M. Perrenoud, "Cryptography with cellular automata," *Applied Soft Computing*, vol. 2, no. 1, pp. 151–160, 2001.

[8] F. Seredynski, P. Bouvry, and A. Y. Zomaya, "Cellular automata computations and secret key cryptography," *Parallel Computing*, vol. 30, pp. 753–766, 2004.

[9] S. Wolfram, "A New Kind of Science," Champaign, IL: Wolfram Media, pp. 90, 55, 870, and 952, 2002.

[10] L. Kotoulas, D. Tsarouchis, G. Ch. Sirakoulis, and I. Andreadis, "1-d Cellular Automaton for PseudoRandom Number Generation and its Reconfigurable Hardware Implementation," in Proc. of ISCAS'2006, pp. 4627–4630, May 2006.

[11] S. Das, and D. R. Chowdhury, "Cryptographically suitable maximum length cellular automata," *Journal of Cellular Automata*, vol. 6, no. 6, pp. 439–459, 2011.

[12] S. Karmakar, D. Mukhopadhyay, D.R. Chowdhury, "CAvium – Strengthening Trivium stream cipher using Cellular Automata," *Journal of Cellular Automata*, vol. 7, no. 2, pp. 179–197, 2012.

[13] G. Marsagglia, Diehard, http://stat.fsu.edu/ geo/diehard.html, 1998.

[14] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks,A. Heckert, J. Dray, and S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST, http://csrc.nist.gov/rng/, April 2010.

[15] P. L. Ecuyer, and R. Simard. "Testu01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 22/1–40, August 2007.

[16] J. Bath and A. J. Turberfield, "DNA nanomachines," *Nature Nanotechnology*, vol. 2, pp. 275–284, 2007.

[17] B. Yurke, A. J. Turberfield, A. P. Mills, Jr. F. C. Simmel, and J. L. Neumann, "A DNA-fuelled molecular machine made of DNA," *Nature*, vol. 406, pp. 605–608, August 2000.

[18] Z.-G. Wang, J. Elbaza, F. Remacle, R. D. Levinea, and I. Willner, "All-DNA finite-state automata with finite memory," *Proc. Natl. Acad. Sci. USA*, vol. 107, no. 51, pp. 21996–22001, December 2010.

[19] P. Yin, S. Sahu, A. J. Turberfield, and J. H. Reif, "Design of Autonomous DNA Cellular Automata," in Proc. of the 11th international conference on DNA Computing (DNA'05), pp. 399–416, 2006.

[20] P. Yin, A.J. Turberfield, S. Sahu, and J.H. Reif, "Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion," In Proc. 10th International Meeting on DNA Computing (DNA'05), pp 344–356, 2004.

[21] H. Yan, T.H. LaBean, L. Feng, and J.H. Reif, "Directed nucleation assembly of DNA tile complexes for barcode patterned DNA lattices," *Proc. Natl. Acad. Sci. USA*, vol. 100, no. 14, pp. 8103–8108, 2003.