Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Software Engineering (JSSE), June Edition, 2013
Volume 3, Issue 6

Elementary Cellular Automata (ECA) Research platform

K. Salman

***Abstract*— We present an educational and research platform for analysis and design of cellular automata (CA). The platform is intended as an aid to newcomers and interested scholars to this field to demystify the complexity associated with understanding the structure, behavior, dynamics and evolution of such systems. It is also an inspirational tool and engine for education that can uncover the immense power and the wide scope of applications that CA can encompass. The platform is designed to be extremely user friendly and flexible. It is believed to be unique in that it allows experimentation in a configurable mode for the CA rule space and lattice span design. Finite and bi-infinite lattice structures are allowed besides the classical periodic (cyclic) boundary conditions. Boundary conditions are comprehensively covered whereby the peripheral cells can be varied in a neighborhood adjacency mode that allows for a variety of time evolution and sampling. The problem of finite lattice is treated by the application of different rules at the boundaries. The entire rule space is utilized for the elementary cellular automata. The classification of the rule space can be studied in detail. Uniform and non-uniform (hybrid) rules can be implemented by simple pull down menus or switchable radio buttons. When the study is directed towards simulation or cryptology, the complexity and random behavior of the cellular automata is tested using "Diehard", the most stringent battery of tests. The results are automatically reported and a fail/pass criterion is established.**

Index Terms*— *Cellular Automata, Diehard, Periodic Boundary Conditions, Rule Space.

I. Introduction

THIS paper discusses the utilization of cellular automata in generating quality pseudo random numbers for use in cryptography, A cellular automaton is a decentralized computing model that provides an excellent platform for performing complex computation with the help of only *local information*. Cellular Automata (CA) is an emerging physical and mathematical structure that is extremely simple and consists of identical basic memory building blocks that are discrete in time and space. The whole structure evolves according to a local yet simple *transition rule* that is capable of evolving into an extremely complex and interesting structure. It was at first conceived around 1950 by the computer architecture inventor "von Neumann" [1] who used it to establish the possibility of creating replicating digital structures. In the early 1980s, Wolfram [2,16] realized the immense potential of the concept and carried out intensive research whereby he managed to rejuvenate the concept and soon inspired many others in various fields to research the matter extensively. Researchers, scientists and practitioners from diverse fields have exploited the CA model of local information, *decentralized control* and *universal computation* for modeling disparate applications [3,8,11,15,18]. The CA encompasses wide scope of applications. Representative examples can easily span wide fields such as modeling, cryptology, gaming, art, music, biology as well as computation, to name a few. An exploding field of applications of CAs is in *information security* which is the core motivation for the platform design. Cellular Automata has been suggested for Pseudo Random Sequence (PRS) generation [2,13,14,16] as a new, but characteristically different and more powerful alternative to the classical Feedback Shift Registers (FSR). Linear FSRs (LFSR), albeit simple in structure and design, were proven to have comparatively weak statistical features when utilized in the PRS generation [4-7]. The weakness in the LFSRs can be attributed to the linearity of the exclusive-or function used in the feedback network. The task of generating pseudorandom sequences that behave like random sequences is practically and theoretically impossible. The best approach is to generate a PRS so that it behaves like a random sequence for the application in hand [7,23]. This entails that the output of such a generator be tested to prove that it is indeed satisfactorily random. This is another problem that is not yet completely resolved [6, 7,23]. Again, theoretically, it cannot be solved. This leaves one currently widely accepted route is to subject the generated PRS to a well established and broadly recognized battery of statistical tests. The effort in this paper has culminated in adopting and incorporating the almost standard state of the art, the *Diehard* battery of tests [24]. The results of subjecting the outputs of LFSRs to the Diehard suite have shown that these generators cannot pass all the tests. On the other hand, one-dimensional (1-D) as well as two-dimensional (2-D) CAs have been suggested and are now beginning to be used to generate PRSs with good statistical features in Mont Carlo simulations, communications, cryptography and network security, to name a few [2, 8-11,13,14,16,17]. However, the physical limitations imposed on the lattice span of such CAs rendered *periodic boundary conditions*, originally proposed in [2,16] as the pertinent solution. The initial configuration will therefore be circular and the automaton will evolve cylindrically to the terminal time step. Such boundary conditions designs, notwithstanding simple, suffer from VLSI implementation constraints as long

K. Salman is with Middle Tennessee State University, Murfreesboro, TN 37132 USA (Phone: 615-898-2083, Fax: 615-898-5697).

bus lines, which are VLSI area intensive, are needed for hardware realization. Furthermore, the lattice span must be long enough to achieve required long periodic cycles and pass statistical randomness tests, i.e. *two inherently conflicting requirements but are both needed*. A 1-D CA where one rule is implemented throughout the spatiotemporal evolution of the CA has shown unique and useful characteristics and has been suggested in [2,16] for use in random sequence generation. A notable impediment, however, is the input to the boundaries of the CA where it is confined to a limited span. Hence, a long span will render the CA practically unrealizable, a shorter span, on the other hand, results in a shorter cycle length. The *periodic* configuration approach [2,16] partially resolves this issue where inputs are needed to feed the two extremities of the CA. An alternative technique is to feed the peripheral cells with external, uncorrelated inputs. These inputs can either be fixed, such as logical "0" or "1" or can be generated separately and independently. All these methods feeding fixed boundary values running under chaotic rules on uniform 1-D CAs have produced much shorter periods than the LFSR and drastically failed statistical battery of tests. The research platform described in this paper proposes alternative techniques that have passed the statistical tests and produced attractive parallelism and correlation properties. These will be discussed later. The rest of the paper is organized as follows: in the Preliminaries section the basic theory of cellular automata is presented, the following section will be devoted to the description of the CA platform followed by the summary and conclusions.

II. Theoretical Background

The cells of a 1-D CA are arranged on a linear finite lattice of identical cells of length $K, K \in \mathbb{N}$. Each cell c_k^t is indexed spatially by the variable $k \in K, K \in \mathbb{N}$ and temporally by the time variable $t \in T, T \in \mathbb{Z}$ and will be equipped with the ability to communicate with a number of its neighbors including itself and will collectively be referred to as size η-cells neighborhood such that $\eta = r_L + 1 + r_R$ where r_L is the left radius which is at distance r_L cells to the left of the center cell, where 1 refers to the center cell, and $r_R$is the right radius which is at distance $r_R$cells to the right of the center cell. Generally a one dimensional cellular automaton can be considered as a sextuple mathematical structure, $\{Q, K, \Phi, \phi_\theta, \eta(c_k^t), C_0\}$, wherein:

- Q is the finite set of states alphabet, from which the configurations of c_k^t cells take their values, $c_k^t: \mathbb{Z} \to Q$.
- $K \in \mathbb{N}$ is the span length of the cellular automaton.
- Φ is the global function that computes transformations between sets of configurations, $\Phi: Q^K \to Q^K$.
- $\Re_\theta$ is the local rule where $\theta \in 2^{Q^\eta}$ is the rule number, and $\Re_\theta : Q^\eta \to Q^1$.
- $\eta(c_k^t) \in \mathbb{N}$ is the size of the neighborhood defined by $\eta = r_L + 1 + r_R$.
- $\Gamma_0 = \{c_k^0\}, k = 0,1, \dots, K-1$ is the seed of the cellular automaton.

It follows that the cellular automaton dynamics consists of passing one configuration Γ_t to the next Γ_{t+1} in discrete time steps $\in \mathbb{Z}$. It should be pointed out that distant neighbor communication falls under the Cellular Neural Networks (CNN) research area [12] which is not the subject matter of this paper. Each cell will change its state according to a local transition rule $\Re_\theta$ where $\theta \in 2^{p^{(r_L+1+r_R)}}$ represents the rule number in the rule space $2^{p^{(r_L+1+r_R)}}$ and will take its state value from an alphabet $Q = \{0,1, \dots, p-1\}$. All cells are updated synchronously and the cells are restricted to local neighborhood interaction with no predetermined global means of communication. For ease of illustration we let the CA evolve according to one *uniform* neighborhood transition function and fixed radius r which is a local function (rule) $\Re_\theta: Q^{2r+1} \to Q$ where the CA evolves after a certain number of time steps T. In this case we have a total of p^{2r+1} distinct rules. It follows that a 1-D CA is a linear lattice or register of $K \in \mathbb{N}$ memory cells. Each cell is represented by c_k^t, where $k = [1{:}K], K \in \mathbb{N}$ and $t = [1{:}T], T \in \mathbb{Z}$ that describes the content of memory location k at evolution time step t. It can be seen that the rule space for the simplest automaton of radius $r = 1$ under $GF(2)$ will provide $2^{2^3} = 256$ distinct rules. On the other hand making $r = 2$ will increase the rule space to $2^{2^5} = 2^{32} = 4{,}294{,}967{,}296$ distinct rules! An exponential increase in the rule space! Therefore, finding a suitable rule for PRS application will be a daunting process in itself. In this paper we will confine ourselves to a finite binary field $F(2)$for which p= 2 such that each cell will be able to take one of two states "0" or "1" from $GF(2)$. This implies the applicability of binary Boolean algebra to the design of the rules over $GF(2)$. Therefore each cell will communicate with the two neighboring cells, one on the left and one on the right. This will render the size of the neighborhood $\eta = r_L + 1 + r_R = 2r + 1 = 3$. This CA will henceforth be referred to as Elementary Cellular Automaton (ECA), a name suggested by Wolfram [2,16]. Hence, the total number of 3-tuple configurations will be $2^\eta = 2^3 = 8$ and can be assigned the symbols $m_7, m_6, m_5, m_4, m_3, m_2, m_1, m_0$ as shown in figure 1.

$c_{k+1}^t c_k^t c_{k-1}^t$							
111	110	101	100	011	010	001	000
m_7	m_6	m_5	m_4	m_3	m_2	m_1	m_0

Figure 1 1-D ECA Rule configurations or minterms.

It is obvious that the triplets $m_j, j = 0,1, \dots, 7$ are identical to and serve the same purpose as the *minterms* of a 3-variable function in *digital logic*, so is the *truth table*. Hence the truth table for all possible and distinct functions will have $2^8 = 256$ columns as depicted in Figure 2. The numbering scheme used for the rules as shown in the figure is attributed to Wolfram [2,16] and uses the values assigned to the minterms $m_j, j = 0,1, \dots, 7$ where m_0 is the least significant and m_7 is the most

significant digits. This rule number will be denoted by the symbol θ and mathematically represented by $\theta = \sum_{j=0}^{2^{\eta}} m_j\, 2^j$ as shown in figure 2 for the representative rules 0, 1, 30, and 255.

		Neighborhood			ECA Rules					
		c_{k+1}^t	c_k^t	c_{k-1}^t	$\Re_0$	$\Re_1$		$\Re_{30}$		$\Re_{255}$
minterms	m_0	0	0	0	0	1	---	0	---	1
	m_1	0	0	1	0	0	---	1	---	1
	m_2	0	1	0	0	0	---	1	---	1
	m_3	0	1	1	0	0	---	1	---	1
	m_4	1	0	0	0	0	---	1	---	1
	m_5	1	0	1	0	0	---	0	---	1
	m_6	1	1	0	0	0	---	0	---	1
	m_7	1	1	1	0	0	---	0	---	1

Figure 2, 1-D ECA rules truth table (rule space)

By means of logic minimization techniques such as Karnaugh-Mapping, the logical expression for the particular rule can be derived and realized. For example, the following figure 3 depicts $\Re_{30}$ expression derived from the Karnaugh -Map shown and represents the next state of the center cell of the minterm $m_j, j = 0,1,\ldots,7$. The current cell c_k^t is indexed by $k \in K$ and represents the center cell of the binary representation of the index $j = 0,1,\ldots,7$. $c_k^{t+1} = c_{k+1}^t \oplus (c_k^t + c_{k-1}^t)$, where $1 \le k \le K-1$.

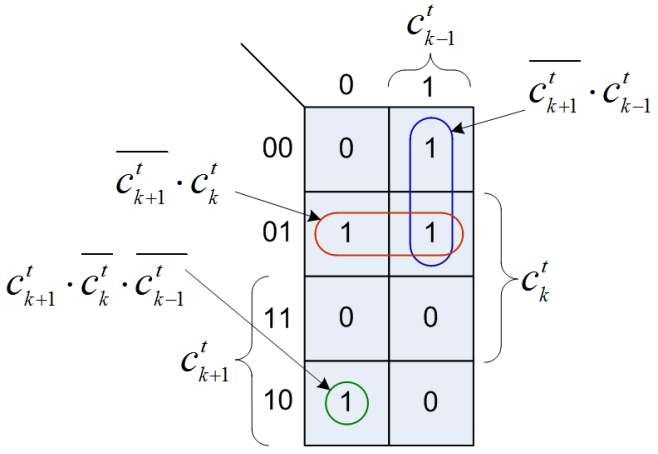


Figure 3, K-map of $\Re_{30}$

Similarly, the logic expressions for the other three rules shown in the truth table of figure 2 are: $\Re_0 \Rightarrow c_k^{t+1} = 0$, $\Re_1 \Rightarrow c_k^{t+1} = c_{k+1}^t \cdot c_k^t \cdot c_{k-1}^t$ and $\Re_{255} \Rightarrow c_k^{t+1} = 1$. It is clear that running the ECA under $\Re_0$ will result in the all *0-state* after just one time step evolution, $\Re_{255}$ will result in the all *1-state* after one time step evolution regardless of the seed contents and the boundary conditions of the cellular automaton. As explained previously that the rule for the ECA is a mapping $\{0,1\}^3 \rightarrow \{0,1\}^1$ which means that for a fixed span the end cells of the cellular automaton will have to be provided by an extra cell for each. The periodic configuration of Wolfram [2,16] wraps around the cellular automaton such that the peripheral cell on the extreme left would consider the peripheral cell on the extreme right as an adjacent cell while the peripheral cell on the extreme right would consider the peripheral cell on the extreme left as its adjacent cell. The configuration is illustrated in figure 4.

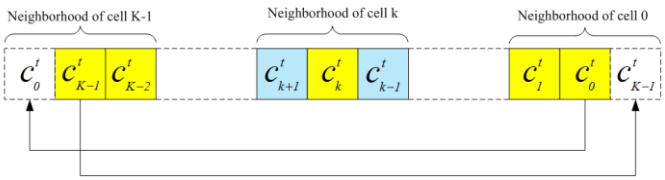


Figure 4; ECA periodic boundary configuration.

Thus, for a fixed span length $K, K \in \mathbb{N}$ the logical expressions for the next state for the extreme left and right cells will be represented respectively by:

$$c_{K-1}^{t+1} = \Re_{30}(c_0^t, c_{K-1}^t, c_{K-2}^t) = c_0^t \oplus (c_{K-1}^t + c_{K-2}^t)$$

And $c_0^{t+1} = \Re_{30}(c_1^t, c_0^t, c_{K-1}^t) = c_1^t \oplus (c_0^t + c_{K-1}^t)$ while for $1 \le k \le K-1$ the expression for the next state of the cell would be represented by: $c_k^{t+1} = \Re_{30}(c_{k+1}^t, c_k^t, c_{k-1}^t) = c_{k+1}^t \oplus (c_k^t + c_{k-1}^t)$. The hardware implementation of $\Re_{30}$ for $1 \le k \le K-2$ is depicted in figure 5.

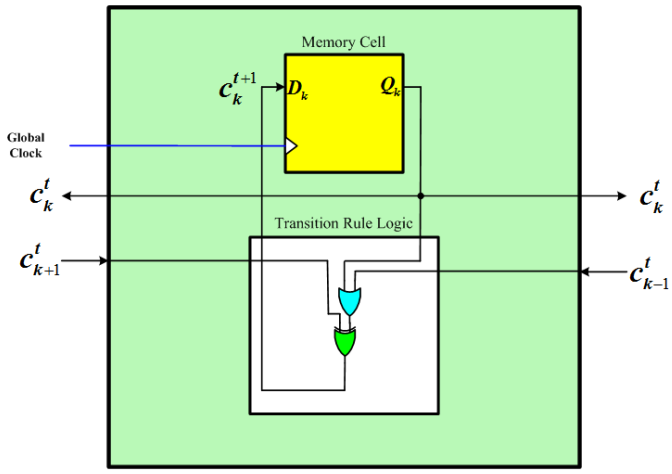


Figure 5, Detailed Structure of a typical Cellular Automaton Cell for rule 30.

It should be clear that only a subclass of the ECS rule space is actually useful for the generation of pseudo random numbers where the current paper is focused. Many authors have attempted to classify the ECA rule space [2,8,16]. The most common classification divides the rule space into four classes that are not necessarily distinct. The distinction is that the classification is phenomenological in nature and based on observations of the spatiotemporal patterns. Class I refers to the evolution that leads to homogeneous fixed points, class II where the evolution leads to periodic configurations, class III

leads to chaotic or aperiodic patterns and class IV produces persistent and complex localized structures. The only class that is suitable for the purpose of generating pseudo random numbers is class three. In fact it can be shown that only a sub-class of class III is suitable for this task. In order for any rule to be suitable for pseudo random number generation the output sequence has to satisfy one of the main conditions of randomness, i.e. the asymptotic distribution of ones and zeros have to be equal or the asymptotic entropy ξ of the output sequence is equal to 0.5 with good number of significant digits. Such condition refers to the so-called *balanced rules* and requires that the number of asserted minterms in any rule must equal the number of unasserted minterms. Therefore the total number of such rules is $\delta = \frac{2^{\eta}}{\left(\frac{2^{\eta}}{2}\right)!\left(2^{\eta}-\left(\frac{2^{\eta}}{2}\right)\right)!}$ .This result has also been described by the Langton's λ parameter, [6,7]. It has been found that only sixteen rules out of this number are actually chaotic and can tentatively be considered suitable for PRS generation. These rules are: (30,45,60,75,86,89,90,101,102,105,135,149,150,153,165,195). More extensive studies, for example the work of Andrew Wuensche and Mike Lesser, [8], have produced an atlas of Basin of attraction fields of one-dimensional cellular automata and showed that the ECA rule space can be clustered. The clusters are formed by the application of *rule equivalence*. The rule equivalence consists basically of three operations, *complementation*, *negation* and *reflection*, as illustrated in figures 6. The size of each cluster can be 8, 4 or 2 rules depending on the *minterms* truth table. Therefore studies of the dynamical behavior of the whole ECA rule space is not really necessary. Each cluster has a *rule leader* and the studies can be confined to this rule leader.

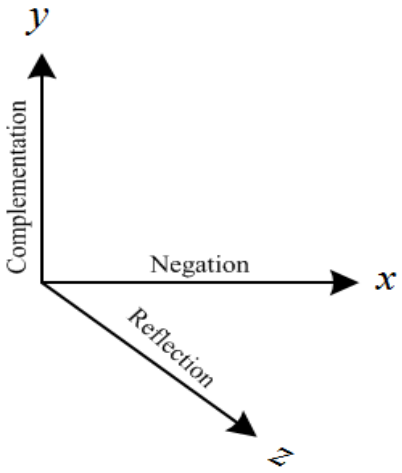


Figure 6, Rule equivalence basic operations

The clusters that contain the chaotic rules listed above are: (30,86,135,149), (45,75,89,101), (60,102,153,195), (90,165) and (105,150), as depicted in figure 8. The leaders of the five clusters are 30, 45, 60, 90, and 105, respectively. Except for the complementation of rule 105 that results in rule 150, the complementation of the other four cluster rule leaders do not produce chaotic rules that are useful for PRS generation. For example for rule 30, the process of negation is achieved by complementation followed by re-ordering of the minters, thus: $\Re_{30} \xrightarrow{negation} \Re_{135}$, while reflection is achieved by mirror reflection of the rule as well as the mirror reflection of the minterms followed by reordering of the minterms, thus: $\Re_{30} \xrightarrow{reflection} \Re_{86}$, and $\Re_{30} \xrightarrow{reflection} \Re_{86} \xrightarrow{negation} \Re_{149}$, which is the same as $\Re_{30} \xrightarrow{negation} \Re_{135} \xrightarrow{reflection} \Re_{149}$. The rules generated by complementation do not usually produce chaotic rules. For example, rule 30 undergoing complementation produces rule 225 which is not in class III and therefore it is not chaotic and is not usually useful for PRS when used in a uniform *periodic* configuration. The same applies to the other three rules, rule 120 produced from rule 135, rule 169 produced from rule 86 and rule 106 produced from rule 149. These rules are colored gray in figure 8. Figure 7 depicts the rule equivalence actions for a generic rule $\Re_{\theta}$. If this rule is in class III then the three rules produced $\Re_{\theta_N}$,$\Re_{\theta_R}$ and $\Re_{\theta_{RN}}$will be in the same class while those rules produced by complementation, $\Re_{\theta_C}$,$\Re_{\theta_{CN}}$,$\Re_{\theta_{RC}}$ and $\Re_{\theta_{RNC}}$do not fall in class III. The exception to this is the complementation of rule 105 which produces rule 150, $\Re_{105} \xrightarrow{complementation} \Re_{150}$ and they are both in class III but both of them are *linear* chaotic rules. Due to the structure of the minterms placement the three rule equivalence operations, rule 105 wraps around one equivalent rule, i.e. rule 150. On the other hand, rule 90 has only one equivalent rule which is rule 165, and this is achieved by Negation, $\Re_{90} \xrightarrow{negation} \Re_{165}$. The minterm structure of rule 60 produces three rules by Negation, Reflection as well as the combined Negation-Reflection operation. Complementation action wraps around the three equivalent rules. All these three rules $\Re_{60}, \Re_{90}, \Re_{105}$have *linear* logical expressions and consequently, their suitability for PRS is limited. Figure 8 illustrates the formation of the 16 chaotic rules while figure 9 depicts in details these actions for generic and cluster rule 30.

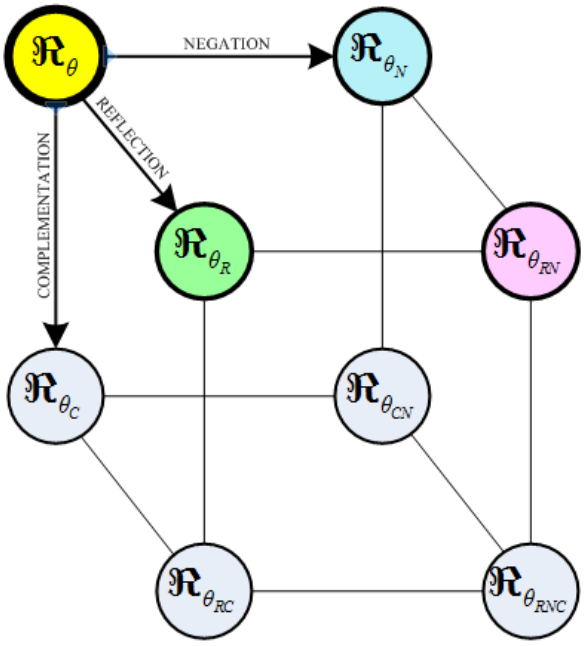


Figure 7, Cube representation of a generic rule cluster.

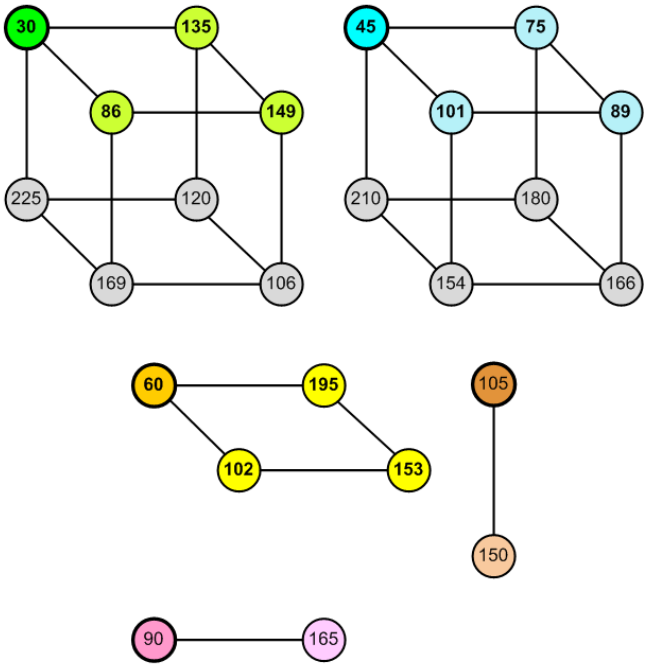


Figure 8, Cube representation of balanced chaotic rule clusters.

$\Re_x$	m_7 111	m_6 110	m_5 101	m_4 100	m_3 011	m_2 010	m_1 001	m_0 000
⇩	Complementation							
$\Re_{x_c}$	m_7 000	m_6 001	m_5 010	m_4 011	m_3 100	m_2 101	m_1 110	m_0 111
⇩	Order Restoration							
$\Re_{x_{cr}}$	m_0 111	m_1 110	m_2 101	m_3 100	m_4 011	m_5 010	m_6 001	m_7 000

Generic Rule NEGATION

$\Re_{30}$	m_7 0	m_6 0	m_5 0	m_4 1	m_3 1	m_2 1	m_1 1	m_0 0
⇩	Complementation							
$\Re_{225}$	m_7 1	m_6 1	m_5 1	m_4 0	m_3 0	m_2 0	m_1 0	m_0 1
⇩	Order Restoration							
$\Re_{135}$	m_0 1	m_1 0	m_2 0	m_3 0	m_4 0	m_5 1	m_6 1	m_7 1

Rule 30 NEGATION

$\Re_{30}$	m_7 111	m_6 110	m_5 101	m_4 100	m_3 011	m_2 010	m_1 001	m_0 000
	0	0	0	1	1	1	1	0
Reflection	Unsymmetrical Minterms Mirror Reflection							
$\Re_{86}$	m_7 111	m_3 110	m_5 101	m_1 100	m_6 011	m_2 010	m_4 001	m_0 000
	0	1	0	1	0	1	1	0

Rule 30 REFLECTION

Figure 9, illustration of rule Complementation, Negation and Reflection for cluster rule 30.

All these rules evolve into chaotic time-space dynamics but the linear rules are so called because of the use of the linear logical exclusive-or primitive in their logic expressions. Hence the utilization of such rules in cryptography is expected to be limited. On the other hand, the two non-linear groups (rules 30 and 45), have non-linear primitives in their logical expressions and therefore can be expected to be more suitable for cryptographic applications:
$c_k^{t+1} = \Re_{30}(c_{k+1}^t, c_k^t, c_{k-1}^t) = c_{k+1}^t \oplus (c_k^t + c_{k-1}^t)$ and
$c_k^{t+1} = \Re_{30}(c_{k+1}^t, c_k^t, c_{k-1}^t) = c_{k+1}^t \oplus (c_k^t + \overline{c_{k-1}^t})$. Rule 30 has already been used to generate the first and most powerful random number generator by Wolfram, [2,16] and was utilized in the well known Mathematica software engine. The drawback is its complexity and low output yield. Other research attempted using genetic algorithms to find optimum combination of rules to produce strong PRS in a two dimensional cellular automata setting, [13,14]. Obviously, this approach restricts the flexibility of rule choice and distribution. The emphasis in this paper is to test the one dimensional ECA in various heuristic configurations that can efficiently yield outputs to be considered viable for cryptographic applications.

III. Output Test Data Testing

Each run or every binary sequence can be tested either individually or independently or as part of a sequence of operations. The part that tests the text data generated by the selected rule and CA structure undergoes certain data transformation in the background that is necessary for testing. The problem of pseudo random bit stream of data testing is resolved by adopting the Diehard test suite which is widely accepted in the art and represents the stringent battery of statistical tests suite available, as of this writing, [24]. There are 19 individual and independent statistical tests within the

Diehard test suite that can be individually selected for testing. Each test requires a minimum of 80Mbits of test data. They are listed in Table1.

Table 1, List of original Diehard test suite.

#	Test Name	#p-values
1	BIRTHDAY SPACINGS	9+1
2	tough BIRTHDAY SPACINGS	1
3	BINARY RANK for 31x31 matrices	1
4	BINARY RANK for 32x32 matrices	1
5	BINARY RANK for 6x8 matrices	25+1
6	BITSTREAM	20
7	Overlapping-Pairs-Sparse-Occupancy	23
8	Overlapping-Quadruples-Sparse-Occupancy	28
9	DNA	31
10	COUNT-THE-1's on a stream of bytes	1
11	COUNT-THE-1's for specific bytes	25
12	PARKING LOT	10+1
13	MINIMUM DISTANCE	10+1
14	3-D SPHERES	20+1
15	SQUEEZE	1
16	OVERLAPPING SUMS	10+1
17	UP-DOWN RUNS	3
18	CRAPS	2
19	CRAPS with different dice	2

The Diehard test suite produce what the statisticians refer to as *p*-values. Some of these 19 tests produce just one *p*-value while others produce more than one *p*-value as shown in the above table. The tests produce 223 original *p*-values. Some tests add Kolmogorov-Smirnov Goodness-of-Fit tests (marked in red with the + symbol) and the new total sums to 229 *p*-values. It also adds the *overall* Kolmogorov-Smirnov *p*-value that results in a grand total of 230 different *p*-values. This last *p*-value can be considered as a pass/fail criteria for testing. Each *p*-value is considered to have failed at the 0.05 level when its value falls in the range $p \leq 0.025$ or $p \geq 0.975$. More recently three more tests (GCD, Gorilla and Overlapping 5-Permutation) were added to the suite. They are claimed to be hard to pass tests and require at least 2.7Gbits of test data. It follows that when the output data is greater than 80Mbits and less than 2.7Gbits, the Diehard test suite skips the new tests and carry out the testing on the original 19 tests, otherwise the total 21 tests will be attempted and produce a new total of 269 *p*-values. The Diehard test cannot run on data of less than 80Mbits. When the Diehard test suite program is activated by the ECA Research Platform it undergoes several operations. The first of which is to test the size of the data output. If the size of the data is less than 80Mbit it will exit otherwise it carries out data transformation where the text data is transformed into suitably formatted Binary data. The minimum size of the binary file should be equal to or exceed 10Mbytes in order for the test suite to be activated.

IV. THE ADVANCED CA RESEARCH PLATFORM (ACARP)

Figure 10 shows the form of the ACARP. The form was written in visual Basic for convenience while the applications invoked within the form are written in C++ and visual C for computational speed. The platform has evolved over long time of development. The increase in the number of configurations and functionalities of the CA made it pertinent to redesign the platform and distribute individual programs in a form of tabs. Ten tabs appear in the current version. It is envisaged that more tabs will be added as the research in this platform ensues. One dimensional binary cellular automaton (or the so-called Elementary Cellular Automata (ECA)) is the main vehicle for this platform. The contents of the form change as the tab changes. However, there are some groups of operations that will be present with all tabs. For example eight of these tabs, i.e. all of them with the exceptions of the two tabs, the ***Misc*** and the ***Cycle***, each produce an output text file. Also, these tabs contain the programs that actually run the CA with specific configurations. On the other hand, the ***Misc tab*** and the ***Cycle tab*** carry out certain operations on the text files produced by the other eight tabs. The details will be explained later on in this section. The tabs can be divided into two categories. The first category encompasses the different configurations of the CA to be analyzed, numbered 1 to 8 and the second category lists the additional functionalities numbered 9 to 10 as described in Table 2. The tabs can be divided into two categories. The first group encompasses the different configurations of the CA to be analyzed, numbered 1 to 8 and the second lists the additional functionalities numbered 9 to 10 in Table 2. The visual form is populated with radio buttons, clickable buttons and open text windows. Selection of certain operations can be done by clicking appropriate radio buttons. Active buttons, when clicked, invoke certain programs or permit the accessing of certain files. A file can be accessed through an Open button that invokes the browsing action to the specific location of the file. The file will then be displayed in the appropriate window. Active windows can be filled either manually and/or through up/down arrows. It the opinion of the author that the best approach to explain the functionalities of the ECA research platform is to start with the ***CA Tab***.

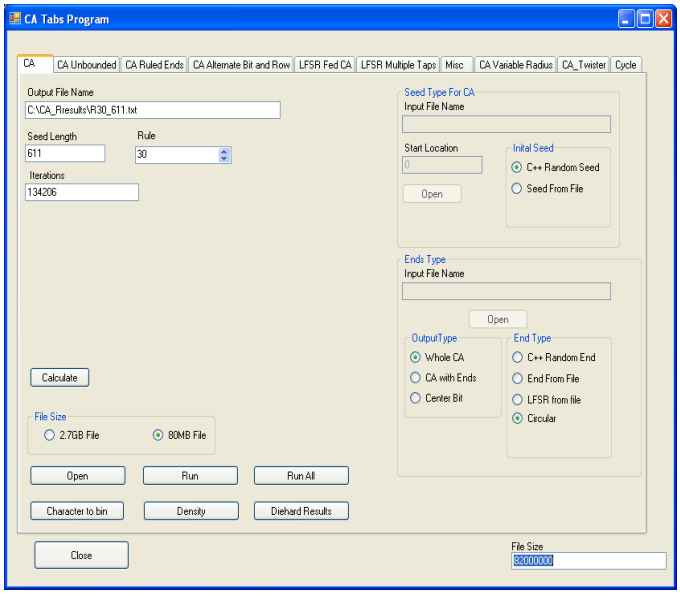


Figure 10, the Advanced CA Research Platform (ACARP).

Table 2: Advanced CA Research Platform Tabs

Tab #	Tab Description
1	CA
2	CA Unbounded
3	CA Ruled Ends
4	CA Alternate Bit and Row
5	LFSR and CA
6	CA Variable Radius
7	CA Twister
8	LFSR Fed CA
9	Miscellaneous
10	Cycle

The ***CA tab*** is the core of the platform. Under this tab the ECA is run in the classical *periodic* boundary conditions mode with additional configurations including perturbations to the boundaries from various independent sources. The functions in the main body of the platform change with the choice of each tab in such a way that the platform is provided with the necessary functionalities to allow the choice accomplishes its objectives. Although the platform may look different with each tab selection, some parts of the form are essentially the same and therefore repeat when switching between tabs and consequently warrants some explanation. With the exception of the four configurations Tabs, the ***CA Unbounded***, the **CA Twister**, the ***Misc*** and the ***Cycle***, all the other six Tabs yield cylindrical Cellular Automata of span length $K \in \mathbb{N}$ and evolution time $T \in \mathbb{Z}$, from a seed Γ_0 that can be viewed as a vector of $\{0,1\}^K$ variables occupying the cells of the ECA lattice at time step $t = 0$ such that $\Gamma_0 = \{c_k^0\}^K$ for $k = 0,1,\dots,K-1$. Figure 11 shows how the cylindrical ECA collects its data with time evolution while figure 12 depicts the manner the data is collected as the ECA evolves in time and yields the output text file of string of one-bit data. The data concatenation that takes place prior to testing by the Diehard engine is carried out as depicted in figure 12(a). This way of concatenation ensures minimum correlation dependence between the data of the cellular automatons at consequent time steps. The concatenation of data according to the image of figure 12 (b) gives poor tests results because of the dependence between the end cells and the consequent negative correlation. This data, represented by the symbol Δ is in fact the text data to be presented to the Diehard test suite for testing, $\Delta = \{\Gamma_t\}^T, t = 1,2,\dots,T$. The same approach has been used during the unbounded CA data output generation (Tab #2) for the same reasoning.

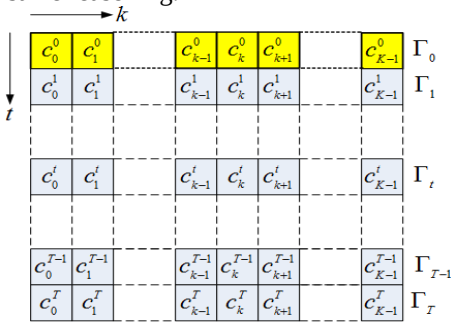
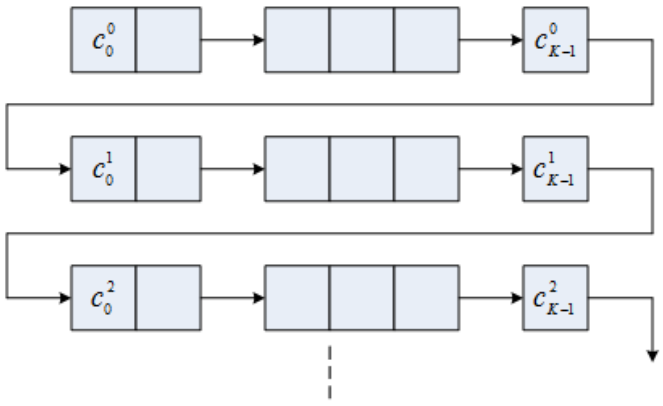
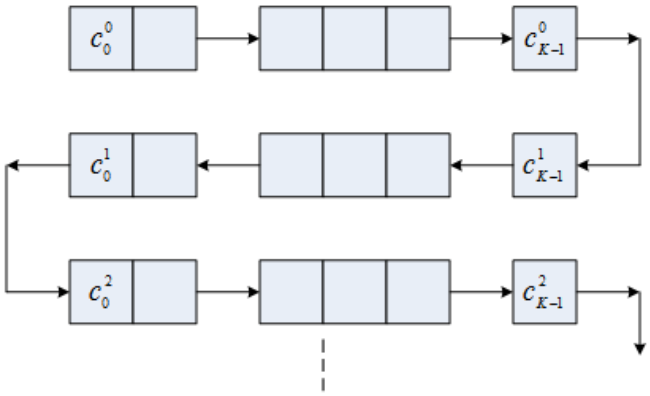


Figure 11, Cylindrical ECA data.

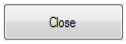


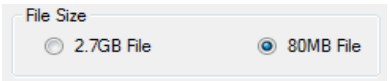
(a)

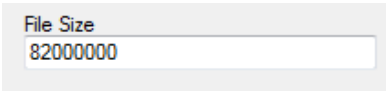


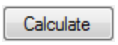
(b)

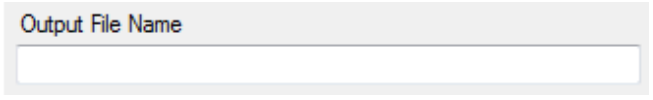
Figure 12, Two methods of concatenating the data bit stream.

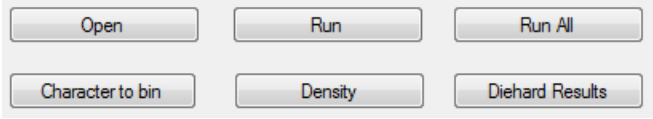
All tabs share the ***Close*** button  which is used to terminate any program running and closes the platform. With the exception of the ***Cycles tab***, all the other 9 tabs share the ***File Size*** section and the following buttons: ***Calculate, Output File Name, Open, Run, Run Al, Character to bin, Density,*** and ***Diehard Result***.



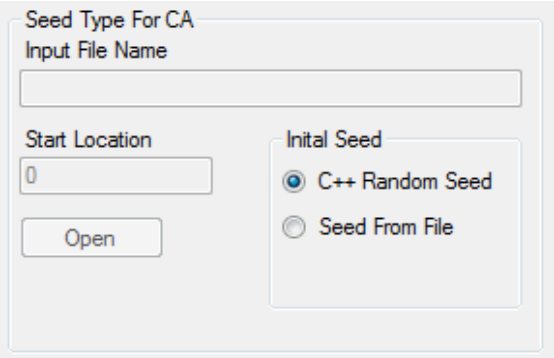




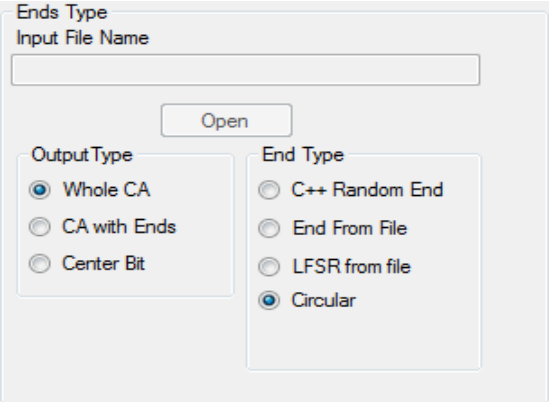




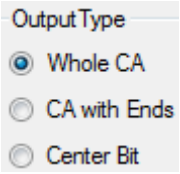
The ***Seed Type For CA*** group is shared by the following Tabs 1, 2, 3, 4, 7, and 8.



The ***Ends Type*** group is shared by Tabs: 1, 4, and 8.



The ***Output Type*** group is shared by Tabs 1, 3, 4, and 8.



Due to the wide varieties and functionalities, the author is of the opinion that the best route to explain the platform, in the nutshell, is to go through each Tab in turn with examples.

A. The CA Tab

Here we will first demonstrate the use of the bounded ECA running under chaotic Rule 30 in the classical periodic boundary conditions. It will generate data size suitable for testing by the Diehard *standard* tests suite. The data is intended for cryptographic applications; therefore it would be desirable to have the data pass all Diehard tests. It should be pointed out here that the whole data output is being used in contrast to the approach of Wolfram [2,16] which utilizes the output of just one cell. Figure 13 shows the selections made in the CA Tab. A seed length of 611-bit was selected for rule 30 and calculated a total of time iterations of 134206 for a data size of 82Mbits. Random seed was selected which is derived from C++. The *periodic* boundary conditions are selected and this is referred to as "Circular" in the End Type section. The whole data is used by selecting the "Whole CA" button in the Output Type section. In order to obtain the final Diehard results the "Run All" button was selected. This button encompasses the running in turn of the other buttons except the Open button.

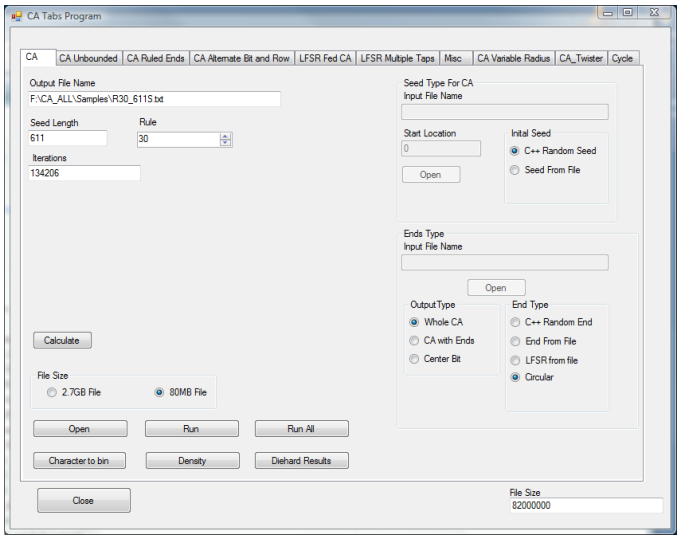


Figure 13, CA Tab selection details.

The Diehard results, shown in the snap of the results below, show that all the 229 *p*-values have passed as well as the Overall Kolmogorov-Smirnov test *p*-value. The density of the whole data set is 0.500011indicating very good distribution.

***** TEST SUMMARY *****

All p-values:
0.2262,0.8282,0.5494,0.9644,0.8122,0.6090,0.6494,0.8748,0.2978,0.2321,
0.0672,0.4666,0.1703,0.8049,0.4028,0.6675,0.2351,0.0635,0.9024,0.0425,
0.6110,0.3006,0.9952,0.2172,0.8532,0.6533,0.7121,0.9179,0.8561,0.8313,
0.9745,0.0064,0.8217,0.1175,0.0509,0.1394,0.1516,0.0738,0.1892,0.5332,
0.9778,0.1947,0.4783,0.0688,0.2271,0.0317,0.9140,0.8023,0.8308,0.7364,
0.8055,0.2126,0.0894,0.6524,0.9563,0.8506,0.6489,0.0898,0.6961,0.6995,
0.7777,0.8175,0.5775,0.2614,0.6676,0.6601,0.0510,0.2247,0.1510,0.2785,
0.1695,0.3699,0.3266,0.8273,0.2785,0.6763,0.7067,0.3033,0.5064,0.3699,
0.2558,0.2937,0.6735,0.0935,0.8717,0.2496,0.2518,0.6575,0.9697,0.5441,
0.7388,0.8369,0.8688,0.6194,0.7707,0.2853,0.6361,0.4806,0.0479,0.0158,
0.4296,0.3940,0.4806,0.0940,0.7758,0.0669,0.5401,0.4887,0.9814,0.5881,
0.8375,0.4773,0.0720,0.5629,0.3223,0.8164,0.0460,0.7556,0.7928,0.1278,
0.5137,0.8964,0.2195,0.7782,0.3025,0.8203,0.8132,0.4422,0.1723,0.5173,
0.2292,0.2862,0.7834,0.6906,0.5733,0.5929,0.5826,0.1716,0.1354,0.3384,
0.5895,0.1785,0.3314,0.1781,0.4632,0.1066,0.0680,0.5170,0.0139,0.6239,
0.4846,0.2960,0.1479,0.1007,0.7525,0.8073,0.6922,0.1831,0.6746,0.8886,
0.8581,0.6967,0.2298,0.4085,0.0261,0.4284,0.3782,0.9901,0.4636,0.5725,
0.2919,0.1268,0.0854,0.2325,0.6426,0.8072,0.7081,0.9821,0.7093,0.8891,
0.6806,0.8599,0.1898,0.9521,0.9996,0.9212,0.5955,0.7424,0.0046,0.2034,
0.2505,0.5803,0.2752,0.7529,0.2423,0.5523,0.9966,0.7258,0.3638,0.0679,
0.6739,0.7031,0.8274,0.3415,0.3220,0.0929,0.8170,0.6220,0.1496,0.8303,
0.5319,0.8150,0.1778,0.5253,0.6933,0.8973,0.0059,0.3641,0.2218,0.4741,
0.1551,0.7464,0.3723,0.6137,0.7747,0.3341,0.4803,0.1616,0.0604,

Overall p-value after applying KStest on 229 p-values = 0.545818

A supporting display program is also used to display the space-time diagram of part of the output data as shown in figure 14 for the span length $K = 611$ and iteration $T = 500$.

The typical *self similar* triangular shapes (*fractals*) usually associated with periodic rule 30 are apparent in the image.

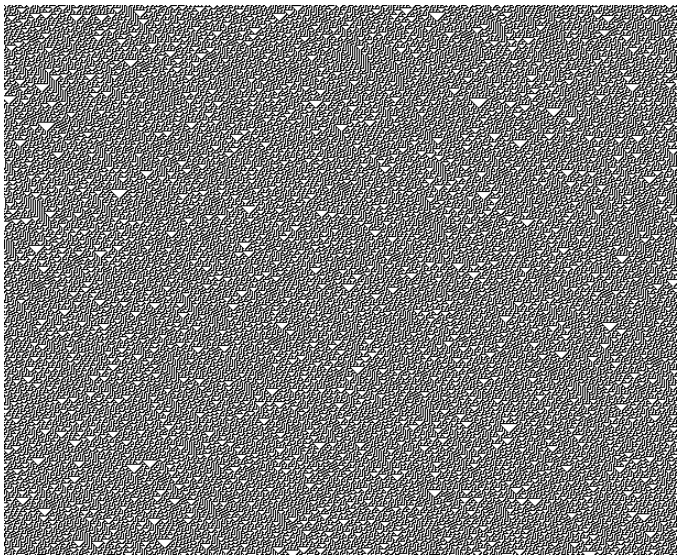


Figure 14, Space-Time image of ECA under *periodic* rule 30.

On the other hand, running the *CA Tab* under the same rule 30 for a smaller span length $K = 131$ requiring different iteration $T = 625954$ but failed quite a number of the Diehard tests. It passed 201 p-values and failed the Overall *Kolmogorov-Smirnov* test *p*-value. Repeating the same operation on rule 45 produced even less number of passes, 200 p-values. The time-space images for the two rules are shown in figure 14 for span length $K = 131$ and iteration $T = 200$. Note again that the self similar *fractals* associated with the two rules are apparent.

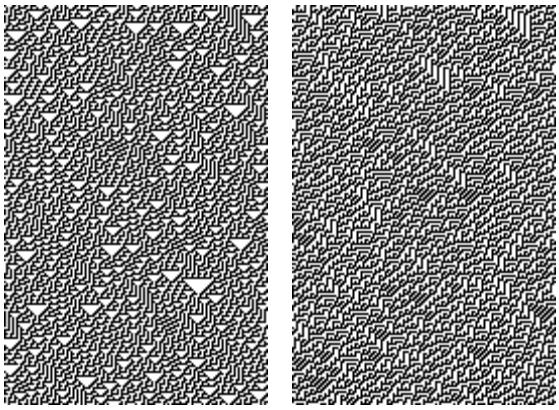


Figure 15, Space-Time images of the *periodic* rules 30 (left) and 45 (right) running in periodic configurations.

For each ECA lattice span length K, there are three ways the output can be selected, the *Whole CA*, as was displayed in the two figures 14 and 15. The second is the *CA with Ends* which will include the boundaries in the data output. The third choice is the *Center Bit* which means using only the center bit in the span and discarding the rest. This choice is in effect copies the method of Wolfram [2], and is currently adopted in the mathematical engine Mathematica. This method represents a very strong pseudo random number generator although it lacks both speed and efficiency. With each of the above output choices the user can select any of the four End Types. The End types refer to the boundary conditions. The *Circular* type is another name of the usual *periodic* boundary condition. The *C++ Random End* uses the embedded C++ random number generator to provide the two boundaries. The *End From File* can use any other external source for the boundaries. When the respective radio button is selected the file that contains the boundary data is determined by browsing and populates the *Ends Type* window which is automatically activated. The *LFSR from file* uses any size of register with predetermined taps selected from a list of primitive polynomials. This type in fact shares the functionalities of another tab, *LFSR Multiple Taps* that will be described later. For each of the above choices, two sizes of data can be selected, 80MB or 2.7GB. Therefore it can be seen that the choices available for each span length K of the ECA is in fact equal to $2^{(2+2+3+4)}$! For a one dimensional ECA of lattice span length K, the initial seed space is composed of the upper bound of $2^K, K-tuple$ words. Hence, Due to the exhaustive logical complexity of the ECA, the computational complexity of the *periodic* boundary conditions scheme running uniformly under a single local rule for total time evolution T can be estimated as utmost $\mathcal{O}(2^{(K+T)})$. When the ECA is injected with two independent external boundaries, the new injected variables would enhance the computational complexity of the bounded ECA by 2^{T^2}. If the external boundaries are derived from two LFSRs, the computational complexity of the system would be enhanced by utmost $\mathcal{O}(2^N - 1)^2 * \varphi(N)$ where $N \in \mathbb{N}$ is the size of the memory cells of the LFSR and $\varphi(N)$ is the Euler's *totient* function that determines the number of primitive polynomials available. To eliminate the possible introduction of cycle repetition in the external boundary inputs, the cycle length can be made equal or greater than the total time evolution of the ECA. This implies that the LFSR span length can be computed from $N \geq \lceil \log_2 T \rceil$.

Rule Equivalence Examples

The following display by way of examples how the ECA research platform can be used to examine the dynamical behavior resemblance of one equivalent group, namely group rule 30. It has been found that best way to compare these rules is through the choice of a center bit seed. In this way the initial seed is made an odd number and assert only the center bit, the rest of the seed consist of zeros. A small size of the span, $K = 31$, and evolution time $T = 100$ is selected and the ECA is run under the 4 rules: rule 30, rule 86, rule 135 and rule 149 that constitute the chaotic rules of group rule 30. The following space-time images shown in figure 16 illustrate the similarities in the dynamical behavior of these equivalence rules in turn. It is evident from the space-diagrams that rule 86 is a mirror reflection of rule 30 while rule 135 is its negation and rule 149 is the negation of rule 86 and is also the combined reflection and negation of rule 30. It is also evident that the randomness in the patterns is the same for all the four rules and the usual fractals of rule 30 are also repeating in the other three rules.

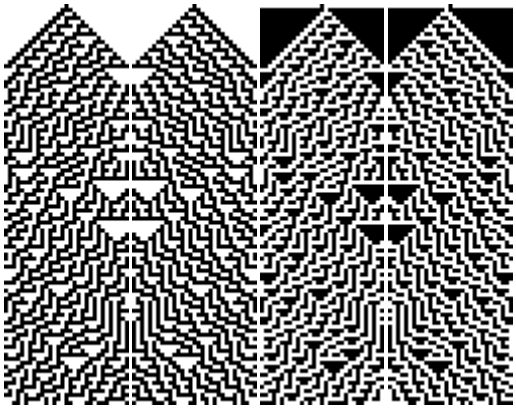


Figure 16, Rule equivalence example or cluster rule 30.

A. *CA Unbounded Tab*

The CA Unbounded configuration is a departure from the classical finite periodic ECA configuration. The concept is based on an attempt to eliminate the effect of propagation inwards at the so-called *speed of light* of correlated patterns and dependencies caused by the wrapping around action of the periodic configuration. Note that the next state of a particular cell at $t+1$ is dependent on the local rule or function of the present state at time t of the three cells at the previous time evolution step $c_k^{t+1} = \mathcal{R}(c_{k+1}^t, c_k^t, c_{k-1}^t)$. It can be seen that the next state of the extreme cell at one end c_K^{t+1} depends on the three adjacent cells at the previous time-step, c_1^t, c_K^t and c_{K-1}^t. It is evident that the cell at the other extreme has influence on the next state of the cell at the other extreme. Therefore this dependency will cause correlation effect that will propagate at the speed of light and the correlation effect will reach the other extreme end after K time steps. However, the movement of the correlation effect from both sides will meet at $T = (K+1)/2$ Time-steps. The structure of the Unbounded CA is illustrated in figure 17 and the correlation effect and propagation is illustrated in figure 18 for three time steps where the green and blue cells represent the boundary cells and the yellow cells carry the correlation effect of the wrapping around action of the *periodic* configuration.

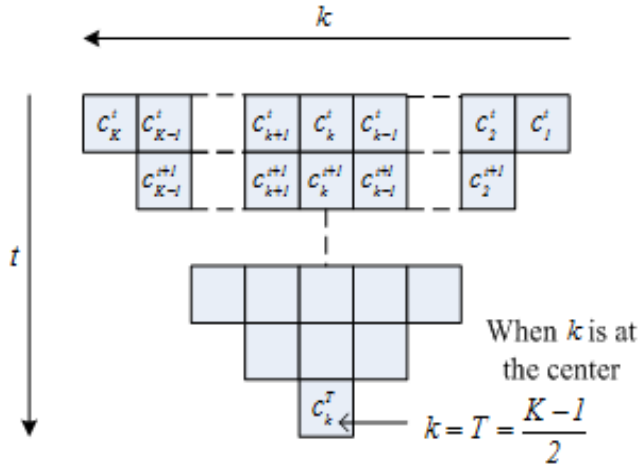


Figure 17, structure of the unbounded ECA.

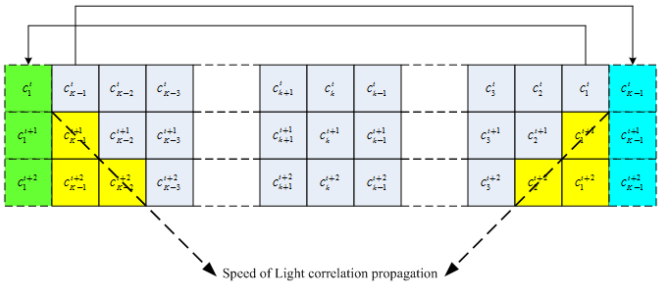


Figure 18, the correlation effect is shown in yellow.

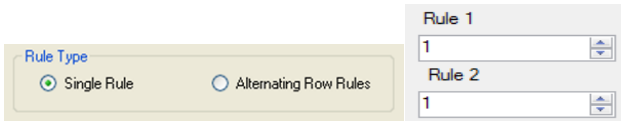
The following gives an example for the above discussion. The space-time image shown in figure 19 is for the ECA periodic configuration running under rule 30 for $K = 101$ and $T = 51$. The cells marked yellow represent the correlation effect carried out due to the periodic boundary action. Removing these cells will yield the ECA shown in figure 20 and can be considered free of correlation dependencies. The output of figure 20 is the space-time image obtained from the Unbounded ECA configuration running under rule 30 and using the same random seed. It is worthwhile noting that the dependencies referred to above are not the self similar shapes (fractals) usually associated with some CA rules. These self similar shapes are generated with the particular rule used by the CA and the shapes may vary as the rule is changed which is obvious by observing the difference in the self similar shapes produced for example by Rule 30 and Rule 45 as well as their equivalences.

Figure 19, Periodic ECA under rule 30, K=101 and T=51 from a random seed.

Figure 20, space-time image of the Unbounded ECA with the same rule and seed of figure 19.

Starting from a random seed this unbounded configuration has passed all Diehard tests suite for almost all the balanced chaotic rules. Rule mixing in this configuration, whether

temporally or spatially or the mixture of both, has also easily passed the Diehard tests suite. The only possible but minor drawback to this configuration is the large size of the seed. However, with the current advance in VLSI area density such drawback becomes asymptotically negligible. This configuration has the added advantage that the serious VLSI drawback of the periodic configuration wire wrap around routing is overcome. Note that the form for this Tab is somehow different from the ***CA Tab***. The ***Calculate*** button is missing because the time evolution steps, T is related to the ECA lattice span K by the expression $T = (K + 1)/2$. A new section ***Rule Type*** is introduced and contains *Single Rule* and *Alternating Row Rules* buttons, as shown in figure 21(a). When the Alternating Row Rules button is activated another text window for rule number 2 is added to the form and situated below the first rule as shown in figure 21 (b) .



(a) (b)

Figure 21, Unbounded ECA Tab additions.

A. *CA Ruled Ends Tab*

This Tab implements a novel introduction to the CA bounded configurations by changing the rules at the extremities of the CA with a two-cell rule so that we do away with the standard and costly method of *periodic* boundary conditions. While the ***End Type*** section is reduced to part of the ***Output Type*** sub-section that includes only the ***Whole CA*** and the ***Center Bit***, there is an addition of two pull down windows for the ***Left Rule*** and the ***Right Rule***, as shown in figure 22. Since we have 2^{2^2} two variable rules at each end and that 2*3 of these are actually single variable rules, namely the ground, the V_{cc}, two buffers and two invertors, we end up with 10 two- cell (or variable) rules for each end. Note that those six rules actually represent the fixed boundaries usually used in some CA applications. Therefore the two variable rules function for the left hand side peripheral cell is derived from the function $f(c_K^t, c_{K-1}^t)$ and similarly the two variable rules function for the right hand side peripheral cell will be derived from the function $f(c_2^t, c_1^t)$. Table 3 lists the rules for both sides using the numbering terminology discussed in a previous section. It can be seen that we are finally left with $(2^{2^2} - 2 * 3)^2$ configurations to try in order to uncover viability of any or some to produce quality pseudo random sequences.

Figure 22, CA Ruled Ends Tab sub-sections

Table 3

	Rules									
LHS	17	34	68	102	119	136	153	187	221	238
RHS	3	12	48	60	63	192	195	207	243	252

B. *CA Alternate Bit and Row Tab*

This is another novel approach to improve the statistical features and enhance the complexity of the data output of the ECA. The platform now has a new group added, ***Single or Multiple Rule per File***, and replaces the ***Rule*** window in the previous Tabs, as shown in figure 23. This is in addition to the functions described earlier for the bounded *CA Tab*.

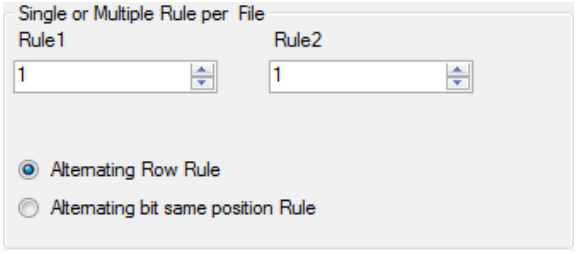


Figure 23, CA Alternate Bit and Row Tab addition.

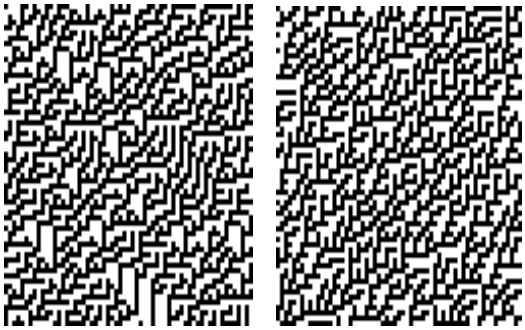
Therefore, there exists the choice of two rules out of the symmetric and chaotic rule sub-space. Two rules are selected and alternate either temporally, whereby it is referred to as ***Alternating Row Rule*** or selected by the activation of the relative radio button. Or alternate spatially in which case it is referred to as ***Alternating bit same position Rule*** or likewise it is selected by its relative radio button. The ***Alternating Row Rule*** means that if row index i is applying ***Rule 1*** then the two rows index $i + 1$ *and* $i - 1$ will apply ***Rule 2***, etc. On the other hand the ***Alternating bit same position Rule*** means that for any row, a cell indexed by column j will apply ***Rule 1*** while the two cells adjacent to it, i.e. indexed by $j + 1$ and $j - 1$ will apply ***Rule 2***. This rule alternation will apply to every row throughout the time evolution of the CA. Note that the increment/decrement ***Rule*** window in the CA Tab is removed and substituted by the two increment/decrement windows ***Rule 1*** and ***Rule 2*** as shown in the above figure. For example we choose the two main chaotic rules Rule 30 and Rule 45 and run the CA for the two modes, Alternating Row and Alternating Bit for the following seed lengths: 61, 73, 611and 671. Figure 24 (a) shows the spatiotemporal image for the Alternating Bit while (b) shows spatiotemporal image for the Alternating Row and both images are for seed length 61-bit and 61 time steps. It can be seen that the self similar shapes (fractals) are different from the native fractals of either rules 30 or 45. The two combinations also demonstrate different fractals from each other. Table 5 shows the Diehard results for both modes as well as the *periodic* configuration, for comparison purposes, for the seed lengths 61, 73, 611, and 671. The Kolmogorov-Smirnov Goodness- of- Fit *p*-values are given in table 5. It is obvious that the Alternating Row mode behaved superior to the Alternating Bit mode. In fact the Alternating Row scored even better than the example in the bounded CA Tab for seedlength 671. Of course this comes with the bonus added complexity advantage.

Table 4, for rules 30 and 45, out of 229 *p*-values

Seed Length	Number of tests passed		
	Alternating Row	Alternating Bit	Periodic Rule 30
61	134	137	123
73	228	223	228
611	229	212	229
671	223	193	215

Table 5, for rules 30 and 45

Seed Length	Overall KS *p*-value		
	Alternating Row	Alternating Bit	Periodic Rule 30
61	0	0	0
73	0	0	0
611	0.177297	0	0.990502
671	0.013958	0	0



(a) (b)

Figure 24, Space-Time images of Alternating Bit and Alternating Row, respectively.

C. *LFSR Fed CA*

This Tab performs an operation that is unique and different from the normal use of the CA. It makes use of the non-linear rules of the CA in order to modify the output of a LFSR in a manner analogous to the action of a *T*-function used in cryptography. Hence, the form must provide necessary information for the LFSR (i.e. span length and the taps from a primitive polynomial) and the CA Rule. Figure 25 shows the addition of the ***LFSR Input File Name*** window where the information about the LFSR initial seed and the taps from an appropriate primitive polynomial reside, the associated ***Open*** button and the ***LFSR Type*** section that in houses two radio buttons, the ***Moving LFSR*** and the ***Over lapping***. The ***Seed Length*** window refers to the span of the LFSR. This Tab provides two modes of operation, one is to decimate the serial generation of the bit stream of the LFSR so that no overlapping of data takes place, and the second one that permits overlapping of the bit stream and therefore does not decimate the data. The two operations are illustrated in figure 26 where a clock rate division by three takes place at the output unit for the ***Moving LFSR*** case.

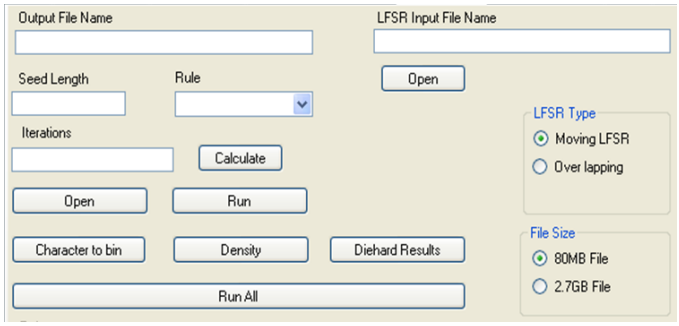


Figure 25, LFSR Fed CA form addition.

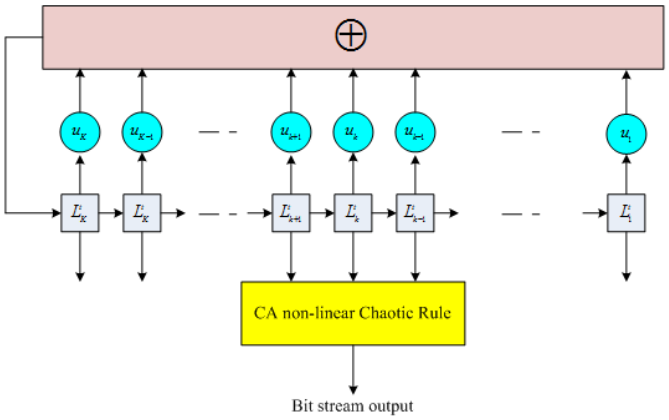


Figure 26, Block Diagram of the LFSR Fed CA design.

D. *LFSR Multiple Taps Tab*

This Tab is intended primarily to prepare text files to be used for CA seeds as an alternative to the C++ random numbers, as mentioned earlier. It is also used as a LFSR based pseudo random sequence generator for comparison with the CA. The platform form contains the windows shown in figure 27 in addition to the usual ***File Size*** section, Calculate and the other six ***Run*** buttons. It also contains the same three windows as used with the other Tabs (the ***Output File Name***, ***Seed Length*** and the ***Iterations***) in addition to a new window the ***Input File Name for Seed and Mask*** where information about the size of the LFSR span and the taps are extracted from a text file that is pointed at in this window.

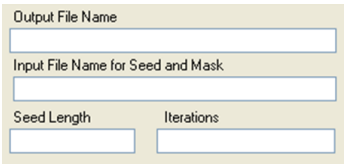


Figure 27, LFSR main input text windows.

E. *Misc. Tab*

This is intended to provide a collection (or ménage) of facilities to manipulate and condition further the output bit data stream. For example, the inverse of the data matrix can be obtained and passed to the Diehard for testing. It has been established that such transformation has almost always passed more tests than the original data. The reason is also

understandable in that the transpose operation in fact conditions the data in such a way that dependencies of adjacent cells are diminished or separated and consequently less correlation is encountered. Column splitting is another more detailed conditioning of the data output. Rearranging the data such that cell adjacency dependencies are reduced is bound to improve on the testing with the Diehard. Various similar operations on the rows are also provided. The variations in the placement of the columns and the rows provide convenient means to decimate the data both spatially as well as temporarily. The use of this Tab helped in determining that the results collected from such operations have shown that temporal decimation is less powerful than spatial sampling. Details of the available operations in this Tab are shown in the two figures 28 and 29.

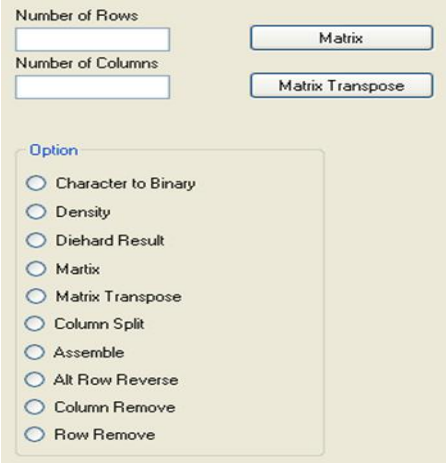


Figure 28, Options in the Misc. Tab.

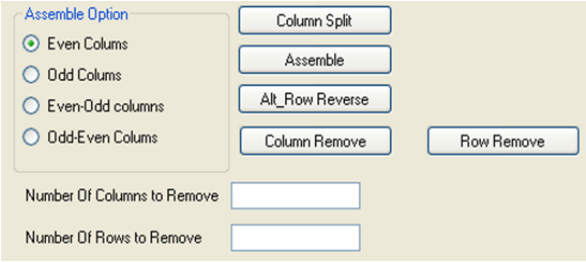


Figure 29, Assemble Options in the Misc. Tab.

F. CA Variable Radius

As for the *Unbounded CA* and the *CA Ruled Ends* described above this new configuration is yet another attempt to remove the dependency on the periodic boundary condition. Instead of wrapping around the cell at the extreme end to the other cell at the other extreme end in the *periodic* configuration explained previously, the boundary cell is imported from within the CA lattice. This is illustrated in figure 30. The next state of the left hand extreme end cell will be generated thus:

$$c_K^{t+1} = f(c_{K-R_L}^t, c_K^t, c_{K-1}^t)$$

and similarly the next state of the right hand extreme end cell will be generated thus:

$$c_1^{t+1} = f(c_2^t, c_1^t, c_{K-R_R}^t)\,.$$

Mixed results were obtained from running some variations of the radius. Some values of the left and right radii did produce test results that indeed matched the results from the standard periodic configuration. This configuration requires extensive amount of variations to obtain an optimum or near optimum values for the two radii. The size of both radii was found to vary depending on the rule used. The platform has the addition of two windows ***Left Radius*** and ***Right Radius*** and the ***Ends Type*** is reduced to the ***Output Type*** sub-section having three buttons, ***Whole CA***, ***CA with Ends*** and ***Center Bit***, as shown in the inserts of figure 31.

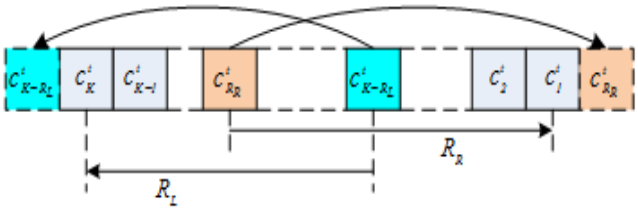


Figure 30, Variable Radius Configuration.

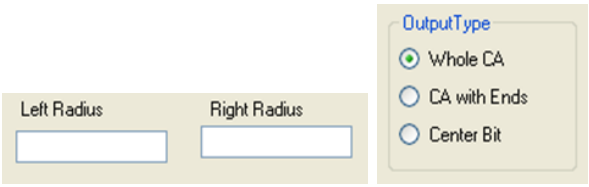


Figure 31, Variable Radius Tab addition.

J. CA Twister

This configuration is fundamentally different from those detailed in the above three examples. Here a *parent* CA that can be generated by any of the above discussed configurations is used to provide the seeds for identical *Offspring* CAs. The number of the offspring CAs is equal to the seedlength of the parent CA, i.e. K as depicted in figure 32. The seedlength of the offspring CAs is equal to the time evolution of the parent CA, i.e. T_1. The time evolution number of steps for the offspring CAs, i.e. T_2 can be determined from the size of the final data output of the whole CA configuration. The size of the data output bit string is equal to $T_1 T_2 K$. We have arranged the output bit string by concatenating the **rows** of the offspring CAs in the same manner as was discussed in the example of the *CA Tab* for the periodic finite CA. This configuration has the obvious main two advantages of expandability and improved complexity and has produced superior test results compared to the standard periodic CA and is the subject matter of another future publication. The form for this Tab is different from the form for the CA Tab in that the ***Calculate*** button is not included. In addition the ***Ends Type*** section is reduced to the ***Input File Name*** window, the ***Open*** window and the ***Circular*** radio button.

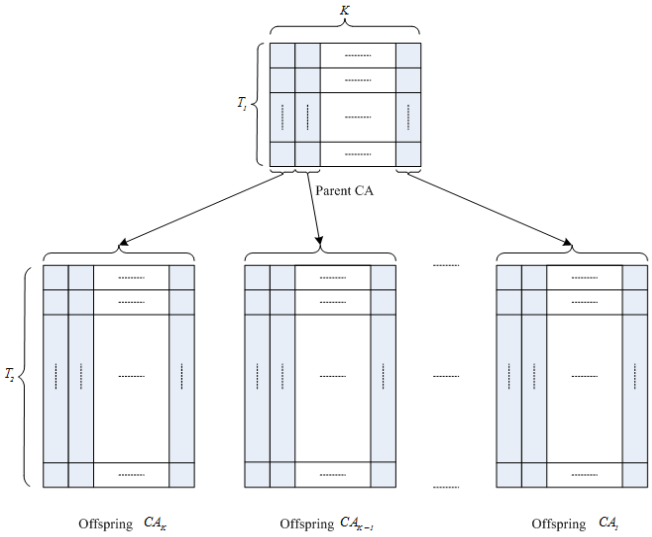


Figure 32, CA Twister structure.

G. *Cycle Tab*

The main objective of this Tab is to determine the cycle length, if it exists, and the transient trajectory to the cycle. Usually the cycle and the trajectory can be determined for relatively small sizes of the seedlength. For larger sizes where the bit stream data output is determined *a priori*, the cycle cannot be captured otherwise the data will not stand a chance of passing the Diehard tests suite. This Tab is useful during analysis of a CA configuration. The platform in this Tab has only one window, the ***Input File Name*** and two buttons, the ***Open*** and the ***Run***, the duties of which are identical to the other Tabs that have been explained previously. It is only applicable to the data output in the form of a matrix with the number of columns representing the seedlength K and the number of rows representing the evolution time T . Therefore, it can be seen that it is applicable neither to the *Unbounded CA* nor to the *LFSR Fed CA* configurations.

V. CONCLUSION

We have presented the details of a software-based platform targeted towards advanced research in cellular automata. The motivation for this platform was the lack of a research engine and tool in the open literature using cellular automata as an emerging technology for the generation of high quality and cryptographically viable pseudo random sequence generation. We have demonstrated the use of the platform to carry out a number of novel cellular automata configurations. Apart from the standard periodic boundary conditions configuration the rest seven configurations are new introduction to the binary one dimensional cellular automaton of neighborhood size 3. The drawbacks of the periodic boundary conditions, such as problems in VLSI hardware implementation and weak statistical features due to the wrapping around effect of cell adjacency dependencies are removed with the new proposed configurations. By injecting externally and independently generated inputs adjacent to the CA lattice end cells, problems due to the wrapping around of the lattice are overcome and gargantuan increase in the complexity is achieved. The Twisted CA configuration introduces a new concept of a continuously expandable CA structure. The complexity increases as new layers of offsprings are added. Data derived from the Unbounded CA configurations have produced excellent results with Diehard tests suite. The scope of variability is illustrated with the variable radius and the end rules configurations. When the application is targeting pseudo random generation, results have shown that the periodic boundary scheme can be effectively substituted with the variable radius configuration. The advantage of this achievement is double fold, more amenable to VLSI hardware implementation and increase in data complexity. Decimations in space or/and time showed the potential for superior data output as well as improved complexity. In addition to the various configurations proposed and tested, the platform offers a lot more facilities to search for yet other means of achieving better results with the Diehard test suite and added complexity to the data output.

REFERENCES

[1] J. Von Neumann, “The Theory of Self-Reproducing Automata”, University of Illinois Press, Urbana, Ill, 1966.

[2] STEPHEN WOLFRAM, "Random Sequence Generation by Cellular Automata", Advances in Applied Mathematics 7,123-169 (1986).

[3] Erica Jen, “Global properties of Cellular Automata”, Journal of Statistical Physics, Volume 4, Issue 1-2, April 1986, pp 219-242.

[4] S. K. Park and K. W. Miller, “*Random number generators: Good ones are hard to find”*, Communications of the ACM, 31 (1988), pp. 1192-1201.

[5] Leon O. Chua, “Cellular Neural Networks: Applications”, IEEE Transactions on Circuits and Systems, Vol. 35, No. 10, October 1988.

[6] Chris G. Langton, “Computation at the edge of chaos: Phase transitions and emergent computation”, Physica D, 42:12-37, 1990.

[7] Wentian Li, Norman H. Packard, Chris Langton, "Transition Phenomena in Cellular Automata Rule Space", Physica D, 45, 77-94 (1990)

[8] Andrew Wuensche and Mike Lesser, "An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata", (1992), Addison-Wesley, ISBN: 0-201-55740-1.

[9] K. Das and P. P. Chaudhuri, “Vector space theoretic analysis of additive cellular automata and its application for pseudo exhaustive test pattern generation,” *IEEE Trans. Comput.*, vol. 42, pp. 340–352, Mar. 1993.

[10] I. Vattulainen and T. Ala-Nissila, *“Mission Impossible: Find a Random Pseudorandom Number Generator”*, Computers in Physics, September 1995, Volume 9, Issue 5, pp. 500-510.

[11] Murray Gell-Mann, "Let's Call It Plectics", Complexity, Vol. 1, no. 5 (1995/96).

[12] Donald Knuth, “The Art of Computer Programming, Seminumerical Algorithms”, Volume 2, 3rd edition, Addison Wesley, Reading, Massachusetts, 1998.

[13] Marco Tomassini, Moshe Sipper, Mosé Zolla, Mathieu Perrenouda, "Generating high-quality random numbers in parallel by cellular automata", Future Generation Computer Systems 16 (1999), 291–305.

[14] Tomassini, Marco, “On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata”, IEEE transactions on computers, Volume 49, Number 10, October 2000, pp 1146-1151.

[15] A. Llachinski, “Cellular Automata: A Discrete Universe”, World Scientific Publishing, Singapore, September 2001.

[16] S. Wolfram, “A New Kind of Science”, Champaign, IL: Wolfram Media, 2002, ISBN: 1579550088.

[17] Barry Shackleford, Motoo Tanaka, Richard J. Carter, and Greg Snider (2002), "FPGA Implementation of Neighborhood-of-Four Cellular Automata Random Number Generators", Proc. FPGA 2002.

[18] Pierre L’Ecuyer, and Richard Simard, “TestU01: A C Library for Empirical Testing of Random Number Generators" ACM Transactions on Mathematical Software, Vol. 33, No. 4, Article 22, August 2007.

[19] Wang Qianfeng, Songnian Yu, Wang Ding, & Ming Leng (2008), "Generating High-Quality Random Numbers By Cellular Automata With PSO", Fourth International Conference on Natural Computation, IEEE Computer Society, pp. 430-433. DOI 10.1109/ICNC.2008.560.

[20] Alonso-Sanz Ramón, & Larry Bull (2009), "Elementary Cellular Automata with Minimal Memory and Random Number Generation", Complex Systems, 18, pp. 195-213.

[21] H. K. Hoe David, Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, & Mukul V. Shirvaikar (2012),"Cellular Automata-Based Parallel Random Number Generators Using FPGAs", International Journal of Reconfigurable Computing, Volume 2012, Article ID 219028, 13 pages, doi:10.1155/2012/219028.

[22] J.M. Comer, Cerda, J.C., Martinez, C.D., & Hoe, D.H.K. (2012), "Random number generators using Cellular Automata implemented on FPGAs", 44th Southeastern Symposium on System Theory (SSST), 44, pp. 67-72.

[23] Charles Wright, “So You Need a Random Number Generator”, Available at the following URL: http://islab.oregonstate.edu/koc/ece399/f04/final/Wright.pd

[24] George Marsaglia**, “**DIEHARD Statistical Tests”: http://www.stat.fsu.edu/pub/diehard/