Pseudorandom Number Generation With Self-Programmable Cellular Automata

Sheng-Uei Guan and Syn Kiat Tan

***Abstract*—We propose a new class of cellular automata, self-programming cellular automata (SPCA), with specific application to pseudorandom number generation. By changing a cell's state transition rules in relation to factors such as its neighboring cell's states, behavioral complexity can be increased and utilized. Interplay between the state transition neighborhood and rule selection neighborhood leads to a new composite neighborhood and state transition rule that is the linear combination of two different mappings with different temporal dependencies. It is proved that when the transitional matrices for both the state transition and rule selection neighborhood are nonsingular, SPCA will not exhibit nongroup behavior. Good performance can be obtained using simple neighborhoods with certain CA length, transition rules, etc. Certain configurations of SPCA pass all DIEHARD and ENT tests with an implementation cost lower than current reported work. Output sampling methods are also suggested to improve output efficiency by sampling the outputs of the new rule selection neighborhoods.**

***Index Terms*—Cellular automata, pseudorandom number generation.**

I. Introduction

RANDOM numbers are needed in a variety of scientific, mathematical, engineering, and industrial applications including cryptography, Monte Carlo simulations [11], etc. Mathematical measures are available to prevent wrong simulation results caused by inappropriate pseudorandom number generators (PRNGs). Statistical tests are conducted to ensure a PRNG produces numbers that are uniformly distributed, uncorrelated with extreme long periods. Still, finding a good PRNG is a difficult task [11], [12]; it is known that every PRNG has to fail in a certain simulation/statistical test, in certain setup models that interfere with the particular regularities of a given RNG and exhibits the hidden correlations between numbers [11]. Hence, PRNG must be carefully matched to the problem at hand. In the past decade, cellular automata (CA)-based PRNG were studied extensively [7] and found to be superior over traditional approaches in areas ranging from built-in self-test [8], [10] to cryptography [2]–[4], [12]–[18], etc.

The majority of research on CA-based PRNG has been focused on the one-dimensional (1-D) CA with nearest three-cell neighborhood, also known as elementary CA [1]. Recently, there is a research trend such that increased complexity from hybrid CA cell configurations and increased CA dimensionality can lead to better performance. Researchers have tried to improve their results by designing new configurations with hybrid cell rules [10], [14], [16] and new boundary conditions, as shown in [4]. Even though CA dimensionality is not directly related to the PRNG problem area as compared to image processing, spatial/temporal simulations [22], etc., researchers have also experimented with increasing the CA dimensionality to obtain better results [12]–[18]. The neighborhood of two-dimensional (2-D) CA leads to better performance is shown in [14], where the authors have improved randomness results over their original 1-D CA work [13].

While the above research direction is headed toward the exploration of increased dimensionality and rule combinations, we believe providing dynamic cell behavior can also produce interesting results. By changing the cell's state transition rules in relation to factors such as its neighboring cell's states, behavioral complexity can be increased and utilized in certain applications. In previous work [16], [17], we confirmed the feasibility of such an approach and achieved comparable results to the work in [14]. For the new approaches mentioned above, all the improvements come at a cost; the increased complexity is brought about by using more cells, additional external mechanisms, or using decimated sampling methods which reduce output throughput (see Section II-B). In this paper, we continue our work based on a reduced-complexity model using substantially fewer cells with 100% output sampling while maintaining comparable randomness test results.

In Section II, we review the work done on utilizing various forms of CA for PRNG applications. Section III explains the operations of our new proposal—the self-programming cellular automata (SPCA). The experimental setup and testing criteria used in this paper is shown in Section IV and the results obtained are examined in Section V. We show ways to further improve the output efficiency of SPCA in Section VI and conclude in Section VII.

II. Background

A. Cellular Automata

A 1-D binary CA is an array of n cells (registers) $\langle s_0(t), s_1(t), \ldots, s_n(t)\rangle$ where each cell's state $s_i \in \{0,1\}$ and $i \in [0,n]$. During each discrete time step, each cell of the CA updates its state using a transition rule based on a prespecified Boolean function applied to the current states of each cell's state transition neighborhood $s_i(t+1) = h_i(s_a(t), s_b(t), \ldots.)$. The conventional nearest three-cell state transition neighborhood, having a radius $r_s = 1$, consists of the cell itself s_i and its left/rightmost neighbors s_{i-1}/s_{i+1}. A CA can be uniform, with the same set of state transition neighborhood/rules are used for each cell, or hybrid, where each cell can use a different

Manuscript received June 9, 2003; revised October 30, 2003. This paper was recommended by Associate Editor N. K. Jha.

The authors are with the Electrical and Computer Engineering Department, National University of Singapore, Singapore (e-mail: sg_1_1@yahoo.com).

Digital Object Identifier 10.1109/TCAD.2004.829808

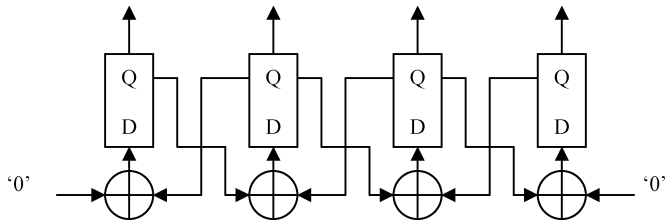


Fig. 1. Four-bit UCA 90 implemented in hardware.

TABLE I
DETAILS OF LINEAR/ADDITIVE RULE 90, 165, 150, 105

Rule Name	Possible Input Configurations								Boolean Representation
	111	110	101	100	011	010	001	000	
90	0	1	0	1	1	0	1	0	$S_{i-1} + S_{i+1}$
165	1	0	1	0	0	1	0	1	$\overline{S_{i-1} + S_{i+1}}$
150	1	0	0	1	0	1	1	0	$S_{i-1} + S_i + S_{i+1}$
105	0	1	1	0	1	0	0	1	$\overline{S_{i-1} + S_i + S_{i+1}}$

set. For convenience, we denote a uniform CA with rule X as "UCA X," while a hybrid CA with rule X and Y is "HCA X/Y." For example, UCA 90 in Fig. 1 represents a uniform CA with rule 90.

The n-bit global state of a CA at time t can be denoted as vector $S^{(t)}$. The states of a CA during each discrete time step is successively sampled to form a pseudorandom number stream $\langle S^{(0)}, S^{(1)}, S^{(2)}, \ldots \rangle$. This approach qualifies the CA as an iterative PRNG. In this paper, by dealing with only linear/additive CA rules, more specifically only rules 90, 150, 165, and 105 shown in Table I (see [1] for rule naming convention), we can use the matrix algebra tools developed in [5]. We can then define a state transition matrix for a CA, denoted as T. It is an n-by-n square matrix, with each row representing the state transition neighborhood dependencies for each cell, an entry of "1" means dependency and "0" otherwise. The next global state vector is then calculated uniquely by $S^{(t+1)} = T.S^{(t)}$. For example, if $S^{(0)} = [1\ 1\ 0\ 0]^T$, $S^{(1)} = T.S^{(0)} = [1\ 1\ 1\ 0]^T$ with the state transition matrix defined as

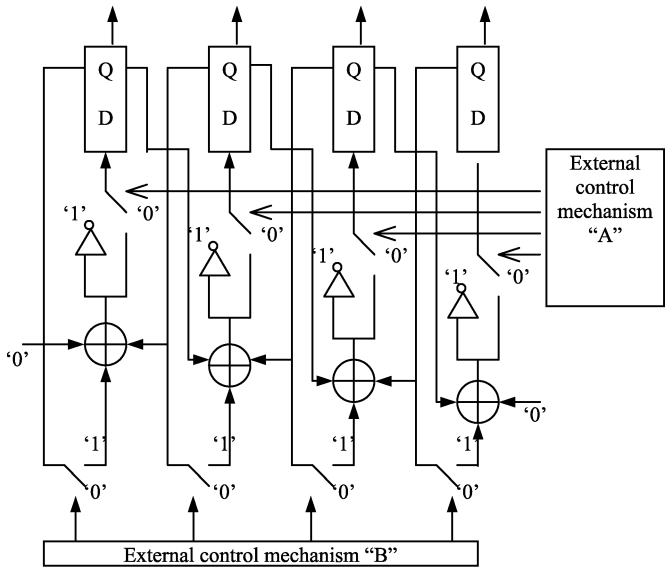
$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

All arithmetic is performed over $GF(2)$.

In this paper, we only consider CA with null boundary conditions where the leftmost/rightmost cells receive a fixed "0" input from its "supposed" left/right neighbors, respectively. Boundary conditions details are found in [9].

B. Programmable Cellular Automata

Programmable cellular automata (PCA) [6], [12]–[18] allows spatial and temporal variations in the state transition rules within a CA, according to some external control scheme. This equates to dynamically changing the state transition matrix T. Through an appropriate selection of state transition rules and the trio of logic gate/connection/control signal wirings, a number of rules can be programmed into the operation of the PCA. Fig. 2 shows the PCA shown with four programmed state transition rules,



Switch A	Switch B	Rule Implemented
0	0	90
0	1	150
1	0	165
1	1	105

Fig. 2. PCA programmable with rule 90/150/165/105.

denoted as PCA 90/150/165/105. A cell uses rule 90 when its switch A is open and switch B is open. The PCA architecture is also flexible in emulating different hybrid CA configurations instead, by applying a fixed control signal.

Our motivation is to design a CA with a lesser cell count that can pass DIEHARD. The 1-D nearest three-cell neighborhood rule 90 and rule 150 are among the most well-studied combinations in the literature for PRNG purposes. We build on top of rule 90 or 150 CA a more elegant form of PCA, with control signals derived from the PCA itself via a new different neighborhood, for a dynamic change of state transition rules in each cell. Varying the behavior of a cell on the conditions of its neighborhood allows for a more dynamic behavior to emerge. This provides a new direction of research in increasing the complexity of CA, aside from the conventional approaches of increasing the dimensionality and neighborhood size [14], [16]. We will show in this paper there are several ways to derive control signals from within the same CA, and in quite a few examples they produce good PRNG with good ENT and DIEHARD results (see Section V).

III. SELF-PROGRAMMABLE CELLULAR AUTOMATA

Fig. 3 shows an SPCA cell programmable with rules 90/150/165/105. Similar to a conventional PCA, it has a localized state transition neighborhood for updating cell states and a new rule selection neighborhood for selecting rules. For a state transition neighborhood, it uses the nearest three-cell neighborhood having a radius $r_s = 1$ and linear rule combinations. The rules are programmable by the control lines (dashed).

The increased complexity of SPCA lies in the presence of the additional rule selection neighborhood (could be wider than the nearest three-cell, which is nonlocal here, in the general sense),

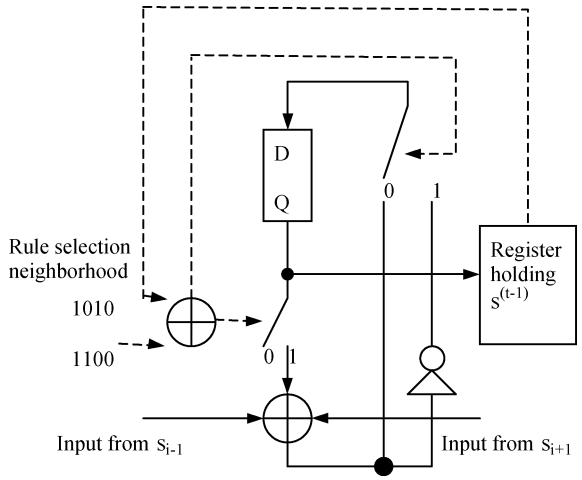


Fig. 3. SPCA cell.

for the purpose of dynamic rule selection. This new rule selection neighborhood can be defined with any linear/additive rule. The output of the rule selection neighborhood controls the behavior of the cell, switching it between the two state transition rules programmed.

By using rule 150/105 (and similarly for 90/165 and other complementary rule combinations), the state transition neighborhood connections differ by only one NOT gate at the output. At the same time, the state transition neighborhood connections of rule 150/105 and 90/165 differ only by the self-dependency on the cell itself. This similar structure with elegance and simplicity in changing between these four rules motivates us to study the effect of switching between rules 90 and 165 (or rules 105/150) on the CA's state transitions. If all four rules are programmed into an n-length PCA, a total of $4n$ different hybrid CA rule combinations can be produced.

In this paper, we considered two SPCA having uniform cells with the following two rule combinations: SPCA 90/165 and SPCA 150/105. The other possibilities such as hybrid SPCA or four-rule SPCA have a higher search space associated for locating better configurations; hence, they are better considered in our future work using genetic algorithms.

Let $s_0^{(t)}, s_1^{(t)}, \ldots, s_n^{(t)}$ denote the cell states at time interval t, and $h_{f_i^{(t)}}(.)$ is the selected state transition rule for cell i at time t; $f_i^{(t)} \in \{0,1\}$ is the rule index for cell i at time t and we denote $F^{(t)}$ as the binary rule vector at time t. Then, $s_i^{(t+1)} = h_{f_i^{(t)}}(s_{i-1}^{(t)}, s_i^{(t)}, s_{i+1}^{(t)})$ and $h_{f_i^{(t)}} \in \{\text{Rule 1, Rule 2}\}$ is selected using the output of each cell's rule selection rule. Every cell's state transition rule then changes for each time interval, due to the rule selection mechanism.

Each SPCA can be characterized by its state transition matrix and rule selection matrix. When we switch between a pair of complementary state transition rules for each cell, the next state of the CA is obtained as $S^{(t+1)} = T.S^{(t)} + F^{(t)}$. In this case, $f_i^{(t)}$ has a value of "0" for each cell using the linear rule and a "1" for each cell using the complemented rule. The rule vector $F^{(t)}$ is then updated as $F^{(t)} = P.S^{(t-1)}$. This leads to the definition of another transition matrix P for the SPCA. To differentiate and avoid confusion between these two matrices, P is defined as the rule selection matrix for determining the next rule vector,

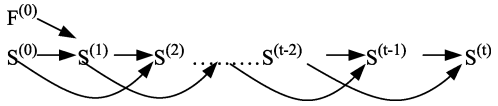


Fig. 4. Mapping of states of the SPCA.

while the state transition matrix defined previously is denoted T. Furthermore, we have the following initialization factors.

- $S^{(0)}$ is the initial value loaded into the global states of the n-cell SPCA.
- $F^{(0)}$ is the corresponding initial rule vector (note that $F^{(0)}$ can also be equal or derived from $S^{(0)}$).

Execution of the SPCA is as follows.

1) An initial n-bit global state $S^{(0)}$ is loaded into the n-cell CA.
2) An initial n-bit rule vector $F^{(0)}$ is loaded (or derived from $S^{(0)}$) into the n-cell CA.
3) Each cell calculates and updates its next internal state synchronously, based on its neighborhood's current states and the current state transition rule.
4) Each cell then calculates and selects its next transition rule synchronously, based on its neighborhood's current states and the rule selection matrix P.
5) Repeat step 3) and 4).

In SPCA, we have used temporal dependencies to create more behavioral complexity. There is a separate set of registers that holds the value of $S^{(t-1)}$, which is used for the derivation of $F^{(t)}$. The general formula is as follows [note that all arithmetic is performed over the binary field $GF(2)$]:

$$\begin{aligned} S^{(0)} &= \text{IV1} \\ F^{(0)} &= \text{IV2} \\ F^{(t)} &= P.S^{(t-1)} \\ S^{(t)} &= T.S^{(t-1)} + F^{(t-1)} \\ S^{(t)} &= T.S^{(t-1)} + P.S^{(t-2)}. \end{aligned}$$

IV1 and IV2 are initialization values, also known as initial seeds. The initial two values for $S^{(0)}$ and $F^{(0)}$, are one-time values used for initializing the sequence. For $t > 1$, the sequence value at each time step $S^{(t)}$ is obtained as a linear combination of two separate mappings on two lagged values of S, as depicted in Fig. 4.

In PRNG applications, one important criterion is for generators to produce long nonrepeated sequences and there is a need to avoid CA with graveyard states [7] $S^{(g)} = T.S^{(g)}$, which leads to a sequence of constant numbers being produced. Also known as nongroup CA, they have a singular transition matrix T, i.e., $\det(T) = 0$. For a linear/additive CA to have group properties, that is $S^{(t+c)} = S^{(t)}$ for all t such that the cycle length is c, the transition matrix must be nonsingular, i.e., $\det(T) \neq 0$ such that $S^{(t+c)} = T^c.S^{(t)}, T^c = I$. A maximal length CA has a nonsingular state transition matrix and the characteristic polynomial of $|T + I\lambda|$ is irreducible and primitive.

Theorem 1: If T and P are nonsingular matrices, then the corresponding SPCA will not exhibit nongroup behavior.

Proof: Assume g is the first time instance where $S^{(g)}$ takes on a graveyard state such that $S^{(g+t)} = S^{(g)}$ for all $t > 0$. A graveyard state has ≥ 1 possible preceding state, besides

being self-looping. Thus, if $S^{(g)}$ is the first time instance of the graveyard state, then $S^{(g-1)} \neq S^{(g)}$ denotes a possible preceding state leading to the graveyard state $S^{(g)}$

$$\begin{aligned} S^{(g+1)} &= T.S^{(g)} + P.S^{(g-1)} \\ S^{(g)} &= T.S^{(g)} + P.S^{(g-1)} \\ (I+T).S^{(g)} &= P.S^{(g-1)} \end{aligned} \tag{1}$$

(Note: I is the identity matrix.)

$$\begin{aligned} S^{(g+t+2)} &= T.S^{(g+t+1)} + P.S^{(g+t)} \\ S^{(g)} &= (T+P).S^{(g)} \\ I+T &= P \end{aligned} \tag{2}$$

So from (1) and (2), $P.S^{(g)} = P.S^{(g-1)}$.

Since P is nonsingular, then $P.S^{(g)} = P.S^{(g-1)}$ leads to a contradiction; hence, graveyard states will not exist in the model. In this paper, T is defined with rule 150/105 or 90/165. As an initial investigation of SPCA properties, P is defined with the following rule: XOR *of any two time-lagged cell states* $s_i^{(t-1)}$ *over a neighborhood radius of 3*. Formally stated, $f_i^{(t)} = s_a^{(t-1)} + s_b^{(t-1)}$, $a, b \in [i-r_r, i+r_r]$, and $a \neq b$, $r_r = 3$. This leads to a total of 21 rule selection neighborhoods (rule selection matrix P) to be explored.

IV. Experimental Setup

In this section, we describe the CA and SPCA used for testing with the ENT [19] and DIEHARD [20] randomness test suites. The test index-name lookup is provided in Table II, and detailed descriptions of each test can be found in the given references. Tests 1–3 are from the ENT suite while tests 4–22 are from DIEHARD.

In each of the tests conducted, we draw n bits from the n-cell CA at each time step. The DIEHARD test suite requires a minimum of 10 Mb of random numbers, so each CA is executed for $10^7 * 8/n$ time steps. The ENT test suite requires fewer numbers, but we execute the test with the same 10 Mb sequence for convenience. Two sets of ten initial seeds are used with each CA generating 20 different sequences for testing. Set A contains ten seeds with some "structure" and Set B has ten "pseudorandom" seeds. This allows us to examine if the CA exhibits autoplectic behavior [10].

The ENT and DIEHARD results of some UCA and maximal length HCA are also presented here for comparison. To the best of our knowledge, there is no explicit study in the literature comparing the results of randomness tests such as ENT [19] and DIEHARD [20] on sequences produced by maximal-length CA and nonmaximal length CA. Maximal-length CA have advantages such as long period and start value independence.

V. Experimental Results

In Tables III–V, the ENT results and DIEHARD results are averaged over 20 sequences. Standard deviation values are also given for the DIEHARD results of SPCA. Based on these empirical results from the ENT and DIEHARD randomness test suite, an SPCA's performance can exceed that of a conventional CA for PRNG purposes.

TABLE II
INDEX OF ENT AND DIEHARD TESTS USED

Index	Test Name	Grading
1	Entropy	Max = 8.00
2	Serial correlation coefficient (SCC	Max for SCC: 0.00
3	Chi-square	Max for Chi-square: 0.50
4	Overlapping sum	
5	Runs test	
6	3D sphere	
7	A parking lot	A pass is considered when the p-value is not 0 or 1.
8	Birthday spacing	
9	Count the ones 1	
10	Binary rank 6*8	
11	Binary rank 31*31	
12	Binary rank 32*32	
13	Count the ones 2	
14	Bitstream test	
15	Craps test	
16	Minimum distance	
17	Overlapping permutation	
18	Squeeze	
19	OPSO test	
20	OQSO test	
21	DNA test	

Table III shows us some surprising results. Maximal length property of the HCA 90/150 is not sufficient to ensure good performance in DIEHARD—the sequences consistently fails 16 out of 18 DIEHARD tests. Increasing the CA length, which increases the sequence's cycle length exponentially does not help, suggesting that those 16 DIEHARD tests are evaluating on other aspects of randomness beside cycle length alone. For ENT, entropy is maximum as expected—these sequences contain all nonzero numbers in the 2^n range. Serial correlation coefficient (SCC) values are also very good, close to zero. The chi-square results with values very close to 0.0000 or 1.0000 are poor, indicative of a nonrandom sequence.

UCA 90 and 150 also produces poor results in both DIEHARD and ENT. All sequences failed to pass a single DIEHARD test. The ENT results shown are poor as well. From the ENT results, uniform rule 90 and 150 CA are found to be dependent on the initial values used for generating sequences.

Table IV shows the best results attainable using SPCA 90/165 and SPCA 150/105. Results were much better compared to the rest of the experiments. Entropy is at a maximum, while SCC shows very low correlation in the sequences. Other than the 16-cell SPCA, the rest of the configurations have chi-square results near the maximum value. This coincides with the DIEHARD results where SPCA with more than 16 cells passes 18 DIEHARD tests.

Several configurations passing all 18 DIEHARD tests and having excellent ENT results were shown in Table V. Although we used an averaging method for ENT results and majority voting for DIEHARD, we did not observe any significant deviation in results produced by the 20 different sequences for each CA. Standard deviation values were low (<0.4) for the good performing SPCA passing >17 DIEHARD tests, while the

TABLE III
ENT AND DIEHARD RESULTS OF MAX-LENGTH HCA 90/150, UCA 90, AND UCA 150

Test No.	HCA 90/150				UCA 90			UCA 150		
	16-cell	20-cell	24-cell	32-cell	16-cell	18-cell	20-cell	16-cell	18-cell	20-cell
1	8.000	8.000	8.000	8.000	4.850	7.124	6.406	4.471	7.064	5.988
2	0.00004	0.00003	0.00003	0.00005	0.1946	0.0090	0.0607	0.1264	0.0043	0.1269
3	0.9999	0.9999	0.9999	0.9999	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
4	Pass	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
5-12	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
13	Pass	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
14-21	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
Total	2.00	2.00	2.00	2.00	0	0	0	0	0	0

TABLE IV
ENT AND DIEHARD RESULTS OF SPCA 90/165 AND SPCA 150/105

Test No.	SPCA 90/165					SPCA 150/105				
	16-cell	18-cell	20-cell	22-cell	24-cell	16-cell	18-cell	20-cell	22-cell	24-cell
1	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000
2	0.0002	0.0002	0.0003	0.0002	0.0003	0.0002	0.0002	0.0002	0.0002	0.0002
3	0.5225	0.5325	0.5463	0.3825	0.5000	0.9889	0.4663	0.5445	0.5862	0.5150
4 - 10	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
11	Fail	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
12	Fail	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
13	Fail	Fail	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
14	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
15, 16	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
17	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
18	Fail	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
19	Fail	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
20	Fail	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
21	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Total	11.65	17.95	17.85	17.95	17.95	11.90	17.95	17.90	18.00	17.95
Std Dev	0.574	0.218	0.357	0.218	0.224	0.300	0.218	0.300	0.000	0.224

poor performing SPCA have higher varying standard deviation values.

No time spacing or regular site spacing schemes [10] were used during output sampling yielding no reduction in output efficiency. This is, however, offset by the possible lowered speed of operation due to the longer cell connection wiring in certain configurations. One possible remedy is rearranging cells in a 2-by-n structure, where the state transition neighborhood uses the nearest three cells in the same row while the rule selection neighborhood uses the nearest three cells in the other row.

For the SPCA 90/165, performance is not proportional to increased CA length and this is due to the singularity of the state transition matrix formed using rule 90/165 as the length of the matrix changes. However, for certain lengths, configurations passing all ENT and DIEHARD tests exist and such configurations can be cost effective compared to the SPCA 150/105 counterparts in terms of throughput and area (see Table VI).

The cycle length of a PRNG determines its suitability for a particular application. We have tested for minimum cycle length using the best SPCA configurations shown in Table V. Twenty different 1-Gb sequences were generated from each SPCA listed and no cycles were detected in any of the sequences.

In Table VI, we show the throughput performance and area requirements of several SPCA configurations. The results are obtained from Synopsys Design Analyzer [23] with the c35_CORELIB library. For both SPCA and LFSR, the area requirement is proportional to the length of the required PRNG. The maximum clock speed for the SPCA is dependent on the type of transition rules used as well as the propagation delay through cascades of logic gates and is independent of the CA length. As expected, SPCA 90/165 has both higher throughput and lower area due to its two-input transition rule.

Although SPCA 90/150 provides better throughput and lower area, it does not pass DIEHARD for all length 18 to 24 cells. However, SPCA 150/105 consistently produces good performance in both DIEHARD and ENT. Suitable SPCA rule selection neighborhoods for good DIEHARD/ENT performance do not necessarily come at a performance–cost tradeoff. Con-

TABLE V
CONFIGURATIONS PASSING ALL DIEHARD AND ENT TESTS

SPCA	Length	Rule selection neighborhood*
150/105	16	No such CA
150/105	17	No such CA
150/105	18	(-3, +1)
150/105	19	(0, +3)
150/105	20	(-3, 0)
150/105	21	(-1, 0)
150/105	22	(+1, +3)
150/105	23	(0, +3)
150/105	24	(0, +3)
90/165	16	No such CA
90/165	17	No such CA
90/165	18	No such CA
90/165	19	No such CA
90/165	20	(-3, 0)
90/165	21	No such CA
90/165	22	(-2, +1)
90/165	23	No such CA
90/165	24	(-2, +3)

*(x, y) are the indices of the 2 cells included in the rule selection neighborhood. E.g. 18-cell SPCA 150/105's rule selection neighborhood consists of (s_{i-3}, s_{i+1}).

TABLE VI
THROUGHPUT AND AREA OF SPCA

PRNG Type	Throughput (Mbit/s)	Area (nm^2)
16bit SPCA 90/165	3376	11672
20bit SPCA 90/165	4220	14580
24bit SPCA 90/165	5064	17489
16bit SPCA 150/105	2240	13998
20bit SPCA 150/105	2800	17487
24bit SPCA 150/105	3360	20979

sider the 21-cell SPCA with rule 150/105 (refer to Table V). This CA has a similar performance to the longer 22, 23, and 24-cell SPCAs, yet it only requires a shorter $(-1, 0)$ rule selection neighborhood compared to the longer cell SPCAs' longer $+3$ connections and the associated complexity of wiring. The rule selection neighborhood yielding good DIEHARD/ENT performance varies differently with the SPCA type and length and contributes some cost in terms of wiring area. Multiobjective genetic algorithms can then be further utilized to locate a suitable configuration [17].

Finally, we compare in Table VII some of the best CA-based PRNG reported in the literature. Due to differences in experimental setups amongst researchers, we cannot claim a totally fair comparison, e.g., no standard deviation results given for some of these CA. The SPCA has advantages of efficiency in terms of size and throughput over the other CA.

VI. OUTPUT EFFICIENCY

To increase output efficiency, one approach is by sampling the output of the rule selection neighborhood $F^{(t)}$, resulting in two times the original throughput. $F^{(t)}$ does not require $S^{(t)}$ as input for all $t > 1$ and is usually generated at t^+, immediately after t and before $t+1$. Thus, it is feasible to provide increased output throughput with the SPCA. Using SPCA 150/105, we repeated

TABLE VII
DIEHARD RESULTS OF OTHER CA-BASED PRNGS

CA Type	No. of cells	*Site /Time spacing used	Bits sampled per iteration	Tests passed
CCA0 [17]	50	2, 1	16	18
CCA2 [17]	50	2, 1	16	18
PCA 90/150 [17]	50	2, 1	16	17
HCA 15/63/31/47 [14]	64	0, 0	64	18
SPCA 150/105	21	0, 0	21	19

*Time/Site spacing is a form of regular bit-decimation sampling which reduces the throughput [10].

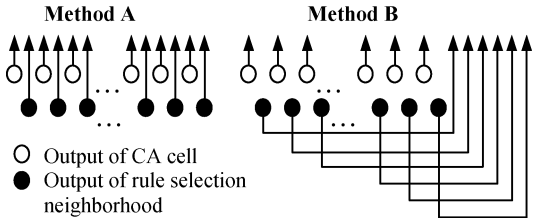


Fig. 5. Improved efficiency using sampling output method A and B.

TABLE VIII
ENT AND DIEHARD RESULTS OF OUTPUT SAMPLING METHODS A AND B

Output type	Length	Ave DIEHARD results	Std Dev	Best Neighborhood
A	16	5.20	0.6782	-3, +2
A	17	9.80	0.6000	-3, 0
A	18	13.15	0.6538	-3, 0
A	19	15.65	0.4770	-2, 0
A	20	16.70	0.4583	-3, +1
A	21	17.85	0.3571	-3, 0
A	22	17.90	0.3000	-2, +1
A	23	17.95	0.2179	0, 3
A	24	17.75	0.4443	0, 3
B	16	10.85	0.4770	-2, +3
B	17	16.75	0.7665	-2, +3
B	18	16.65	0.4770	0, +1
B	19	17.85	0.3571	-2, +3
B	20	16.40	0.5831	0, +1
B	21	16.00	0.0000	-2, +3
B	22	16.00	0.0000	-3, 0
B	23	17.95	0.2179	-3, -1
B	24	16.95	0.2236	-2, +1

the DIEHARD experiment in the previous section. We tried two different output methods depicted in Fig. 5, and the results in Table VIII show some of the best performing minimal-cost rule-selection neighborhoods of each method using cell lengths from 18 to 24.

Method A samples the output of each CA cell followed by the output of its rule selection neighborhood. Using method A, all 18 DIEHARD tests can be passed using a minimum of 21 cells (which yield a 42-bit stream at each time step). Output type B achieves comparative results only at lengths 19 and 23, which samples all the CA cells followed by all the rule selection neighborhood outputs. Most of the rule selection neighborhoods for both output types A and B require the longer $+/-3$ cell connections. Such a scheme should only be considered for applications with lowered security requirements, since the output of the rule

selection neighborhood can be considered as the internal state of the PRNG and hence ideally should not be exposed.

VII. Conclusion

We have shown how complex behavior can be produced by introducing dynamic state transition rules via an additional rule selection neighborhood. For an SPCA, the interplay between the state transition neighborhood and rule selection neighborhood leads to a new composite neighborhood and state transition rule that is the linear combination of two different mappings with different temporal dependencies. It is proved that when the transition matrices for both the state transition and rule selection neighborhood are nonsingular, the SPCA will not exhibit nongroup behavior. All possible 21 rule selection neighborhoods were examined in this paper, and it can be concluded that good performance can be obtained using simple neighborhoods with certain interdependent conditions: CA length, state transition neighborhood/rules. The SPCA is shown to produce pseudorandom number sequences that pass all DIEHARD and ENT tests with an implementation cost lower than other works reported in the literature. The CA uses either decimated sampling methods which reduced the PRNG throughput or a high cell count (64-cell) compared to SPCA 150/105's 21 cells. Output sampling methods were also suggested to improve output efficiency by sampling the outputs of the new rule selection neighborhoods. The future direction of this work lies in the development of SPCA for other possible rule combinations and hybrid cell makeup within the SPCA.

References

[1] S. Wolfram, *Theory and Applications of Cellular Automata: Including Selected Papers 1983–1986*. River Edge, NJ: World Scientific, 1986.

[2] M. Matsumoto, "Simple cellular automata as pseudorandom m-sequence generators for built-in self-test," *ACM Trans. Modeling Comput. Simulation*, vol. 8, no. 1, pp. 31–42, 1998.

[3] M. Mihaljevic, "Security examination of a cellular automata based pseudorandom bit generator using an algebraic replica approach," in *Proc. Applied Algebra, Algorithms and Error Correcting Codes, Lecture Notes Computer Science*, vol. 1255, 1997, pp. 250–262.

[4] M. Mihaljevic and H. Imai, "A family of fast keystream generators based on programmable linear cellular automata over GF(q) and time-variant table," *IEICE Trans. Fund.*, vol. E82-A, no. 1, pp. 32–39, 1999.

[5] A. K. Das and P. P. Chaudhuri, "Vector space theoretic analysis of additive cellular automata and its application for pseudoexhaustive test pattern generation," *IEEE Trans. Comput.*, vol. 42, pp. 340–352, Mar. 1993.

[6] S. Nandi, B. K. Kar, and P. P. Chaudhuri, "Theory and applications of cellular automata in cryptography," *IEEE Trans. Comput.*, vol. 43, pp. 1346–1357, 1994.

[7] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chattopadhyay, *Additive Cellular Automata Theory and Applications*. Los Alamitos, CA: IEEE Computer Soc. Press, 1997, vol. 1.

[8] D. R. Chowdhury, I. S. Gupta, and P. P. Chaudhuri, "A class of two-dimensional cellular automata and applications in random pattern testing," *J. Electrical Testing: Theory Applicat.*, vol. 5, pp. 65–80, 1994.

[9] S. Nandi and P. P. Chaudhuri, "Analysis of periodic and intermediate boundary 90/150 cellular automata," *IEEE Trans. Comput.*, vol. 45, Jan. 1999.

[10] P. D. Hortensius, R. D. Mcleod, W. Pries, D. M. Miller, and H. C. Card, "Cellular automata-based pseudorandom number generators for built-in self-test," *IEEE Trans. Comput.-Aided Design*, vol. 8, no. 8, pp. 842–859, 1989.

[11] P. Hellekalek, "Good random number generators are (not so) easy to find," in *Math. Comput. Simulation*, 1998, vol. 46, pp. 485–505.

[12] M. Sipper and M. Tomassini, "Co-evolving parallel random number generators," in *Parallel Problem Solving From Nature IV*, 1996, pp. 950–959.

[13] ——, "Generating parallel random number generators by cellular programming," *Int. J. Modern Phys.*, vol. 7, no. 2, pp. 181–190, 1996.

[14] M. Tomassini, M. Sipper, and M. Perrenoud, "On the generation of high-quality random numbers by two-dimensional cellular automata," *IEEE Trans. Comput.*, vol. 49, pp. 1146–1151, 2000.

[15] M. Tomassini and M. Perrenoud, "Cryptography with cellular automata," *Appl. Soft Computing*, vol. 1, pp. 151–160, 2001.

[16] S.-U. Guan and S. Zhang, "A family of controllable cellular automata for pseudorandom number generation," *Int. J. Modern Phys.*.

[17] ——, "An evolutionary approach to the design of controllable cellular automata structure for random number generation," *IEEE Trans. Evolutionary Computation*, vol. 7, no. 1, pp. 23–36, Feb. 2003.

[18] ——, "Incremental evolution of cellular automata for random number generation," *Int. J. Modern Phys.*.

[19] ENT Test Suite*http://www.fourmilab.ch/random* [Online]

[20] Diehard, S. Marsaglia. (1998). *http://stat.fsu.edu/~geo/diehard.html* [Online]

[21] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 3rd ed. Reading, MA: Addison-Wesley, 1998.

[22] R. Subrata and A. Y. Zomaya, "Evolving cellular automata for location management in mobile computing networks," *IEEE Trans. Parallel Distributed Syst.*, vol. 14, Jan. 2003.

[23] Synopsys Design Analyzer.

Sheng-Uei Guan received the M.Sc. and Ph.D. degrees from the University of North Carolina, Chapel Hill.

He worked in a prestigious R&D organization for several years, serving as a Design Engineer, Project Leader, and Manager. He has also served as a member on the R.O.C. Information and Communication National Standard Draft Committee. After leaving the industry, he joined Yuan-Ze University in Taiwan for three and one-half years. He served as Deputy Director for the Computing Center and the Chairman for the Department of Information and Communication Technology. Later, he joined the Department of Computer Science and Computer Engineering, La Trobe University, where he helped to create a new multimedia systems stream. He is currently with the Electrical and Computer Engineering Department, National University of Singapore, Singapore.

Syn Kiat Tan received the B.Eng. degree in electrical and computer engineering from National University of Singapore, Singapore. He is currently pursuing the Ph.D. degree at the same university.

His research interests include cellular automata, evolutionary computation, and neural networks.