Layered Cellular Automata for Pseudorandom Number Generation

Syn Kiat Tan[1] and Sheng-Uei Guan[2]
[1]Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260

[2]School of Engineering and Design
Brunel University, UK

Abstract—The proposed Layered Cellular Automata (*L*-LCA), which comprises of a main CA with *L* additional layers of memory registers, has simple local interconnections and high operating speed. The time-varying *L*-LCA transformation at each clock can be reduced to a single transformation in the set $\{A^f \mid f = 1, 2, \ldots, 2^n - 1\}$ formed by the transformation matrix A of a maximum length Cellular Automata (CA), and the entire transformation sequence for a single period can be obtained. The analysis for the period characteristics of state sequences is simplified by analyzing representative transformation sequences determined by the phase difference between the initial states for each layer. The *L*-LCA model can be extended by adding more layers of memory or through the use of a larger main CA based on widely available maximum length CA. Several *L*-LCA (*L=1,2,3,4*) with 10- to 48-bit main CA are subjected to the DIEHARD test suite and better results are obtained over other CA designs reported in the literature. The experiments are repeated using the well-known nonlinear functions f_{30} and f_{45} in place of the linear function f_{204} used in the *L*-LCA. Linear complexity is significantly increased when f_{30} or f_{45} is used.

Index Terms— cellular automata, programmable cellular automata, pseudorandom number generation**.**

I. INTRODUCTION

Random numbers are needed in a variety of scientific, mathematical, engineering and industrial applications including cryptography, built-in self test, artificial evolution such as genetic algorithm and simulated annealing, Monte Carlo simulations, etc. Mathematical measures are available to prevent wrong simulation results caused by using inappropriate pseudorandom number generators (PRNGs). Statistical tests can be conducted to ensure a PRNG produces numbers that are uniformly distributed, uncorrelated, with long periods, etc. Still, finding a good PRNG is a difficult task [5,6] - it is known that every PRNG has to fail in a certain simulation/statistical test, or in certain setups that interfere with the particular regularities of a given PRNG and thus exhibits the hidden correlations between numbers. Hence the PRNG must be carefully matched to the problem at hand. In the past decade, Cellular Automata (CA) based PRNGs are studied extensively [1,4,7-11] and found to be superior over traditional approaches in areas ranging from built-in self-test [3,4], cryptography [2,12,14], etc.

The majority of research on CA based PRNG has been focused on the binary one-dimensional (1-d) CA implemented using registers. Previously, researchers focused on configuring the individual registers' function in the CA such that the global evolution operator will generate maximum length sequences [2,17]. Although having a long period, these maximum length sequences are often found to be weak by randomness tests (see the results in Section IV). It is desirable to have low-cost CA based PRNGs that can generate sequences with desirable statistics. The DIEHARD test suite [15], comprising of 19 individual tests (detailed descriptions for these tests can be found in the given reference),

is often used for evaluating the randomness quality of random number sequences [4,7,9,10,11,21].

In Section II, we first explain the operations of a conventional CA and review some CA based PRNG designs. Section III explains the operations of our new proposal – the Layered CA (*L*-LCA) that uses time-varying transformations from a set derived from the transformation matrix of a maximum length CA. The experimental setup and results obtained are examined in Section IV. We conclude the paper in Section V.

II. BACKGROUND

A. Cellular Automata

A *n*-bit binary CA is an array of *n* registers. The CA state at time *t* is denoted $S^{(t)} = [s_0^{(t)}, s_1^{(t)}, ..., s_{n-1}^{(t)}]'$ where each register's state $s_j^{(t)} \in \{0,1\}$ and $0 \le j \le n-1$. Fig. 1 shows a 4-bit CA. During each discrete time step, each register of the CA updates its state using a pre-specified Boolean function *f* applied to the current states of each register's neighborhood, $s_j^{(t+1)} = f(s_a^{(t)}, s_b^{(t)}, ...)$. The conventional nearest-three-input neighborhood (having a radius of one) consists of the register itself s_j and its left/right neighbors s_{j-1} / s_{j+1}. When state updates are considered in terms of the entire CA, we have the global evolution operator Φ such that $S^{(t+1)} = \Phi(S^{(t)})$.

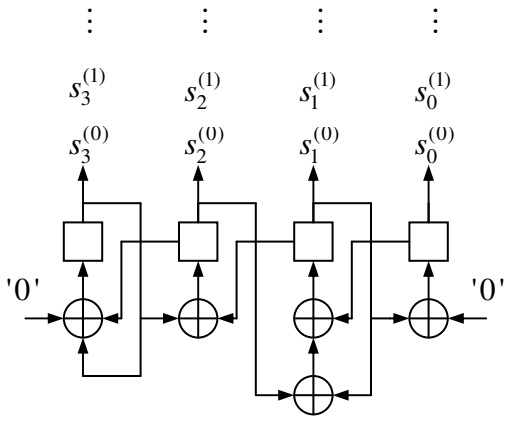

Fig. 1. A 4-bit cellular automata.

The 256 functions associated with the nearest-three-input neighborhood are usually denoted by the naming convention in [1]. For example, three well-known linear functions are listed below, where their associated function names can be calculated from their outputs in Table I. Each output is taken as a positional power of 2, in lexicographic order from top to bottom.

$$f_{90} : s_{j-1}^{(t)} \; XOR \; s_{j+1}^{(t)} \rightarrow s_j^{(t+1)} \tag{1}$$

$$f_{150} : s_{j-1}^{(t)} \; XOR \; s_j^{(t)} \; XOR \; s_{j+1}^{(t)} \rightarrow s_j^{(t+1)} \tag{2}$$

$$f_{204} : \; s_j^{(t)} \; \rightarrow s_j^{(t+1)} \tag{3}$$

Table I. Truth table for register functions

Input (s_{j-1}, s_j, s_{j+1})	f_{90}	f_{150}	f_{204}
(1,1,1)	0	1	1
(1,1,0)	1	0	1
(1,0,1)	0	0	0
(1,0,0)	1	1	0
(0,1,1)	1	0	1
(0,1,0)	0	1	1
(0,0,1)	1	1	0
(0,0,0)	0	0	0

We only consider CA with null boundary conditions (unless stated otherwise) where the leftmost/rightmost registers receive a fixed "0" input from its "supposed" left/right neighbors respectively. Details on boundary conditions can be found in [3]. A CA can be uniform - the same function is used for each register; or hybrid – where each register can use a different function. The example in Fig. 1 shows a 4-bit hybrid CA with the CA transformation $\Phi = \{f_{150}, f_{90}, f_{150}, f_{90}\}$. The states of a CA during each discrete time step can be successively sampled to form a pseudorandom n -bit sequence $\{S^{(1)}, S^{(2)}, S^{(3)}, \ldots\}$ or only one bit per clock is sampled from a particular register to form the single-bit sequence $\{s_j^{(1)}, s_j^{(2)}, s_j^{(3)}, \ldots\}$.

B. Linear Maximum Length CA

Linear maximum length CA (m-CA) are n-bit CA which can generate sequences of all possible non-zero states $S^{(t)}$ having a period $2^n - 1$. These are hybrid CA configured with f_{90} and f_{150} (the example in Fig. 1 is an m-CA) and configurations have been found for up to $n = 500$ registers in [17]. Using only f_{90} and f_{150}, these m-CA have very simple linear structure and low cost implementation associated with the nearest-three-input neighborhood. A linear CA, as well as any linear finite state machine such as linear feedback shift registers etc, can be represented by a so-called transformation matrix A; in other words the CA transformation Φ is equivalent to A. Each state is then given by

$$S^{(t)} = A^t \cdot S^{(0)},\ t \geq 0 \qquad (4)$$

The transformation matrix A of an m-CA is always non-singular and has an inverse $inv(A) \cdot A = I$. It can be shown [3] that this inverse is $inv(A) = A^{-1 \bmod (2^n - 1)} = A^{2^n - 2}$ and therefore the m-CA produces sequences of period $2^n - 1$ since $S^{(t+2^n-1)} = A^{2^n-1} \cdot S^{(t)} = I \cdot S^{(t)}$, for $t \geq 0$.

As simulation results have shown (see Section IV), the sequences generated by m-CA do not always pass all DIEHARD tests even when their period is above DIEHARD's testing requirement of 10M bytes of input, i.e. $2^n - 1 > 10M$, $n > 23$, thus these m-CA configurations still have to be carefully tested for their statistical properties before using them in applications. In short, long period is only one of the many considerations for a PRNG.

C. Previous Works

We now provide the published results of some reported works that passed all DIEHARD tests in the literature. In [21], the nearest-three-input neighborhood is expanded into a non-local neighborhood scheme with four inputs. A 64-bit CA passing all DIEHARD tests is then found through exhaustive testing. In [11], several 8-by-8 two-dimensional (2-d) CA are shown to pass all DIEHARD tests where each register XOR at least four inputs from surrounding registers to form the next state. A wide range of results is not available from the authors although it is mentioned the 2-d CA must be at least 7-by-7 to ensure satisfactory DIEHARD results. In [9], a 2-d array CA consisting of *m* arrays of *n*-bit 1-d

CA is proposed. Only registers near the left/right boundary can be connected to other boundary registers below or above it - complicated wiring that are usually present in normal 2-d CA are thus avoided. Again using only linear functions, multi-objective genetic algorithms are applied here to configure the inputs for registers near the boundary. 48- and 50-bit versions of this 2-d array CA are shown to pass all DIEHARD tests. Note that 2-d CA structures are actually equivalent to 1-d with an increased number of inputs and neighborhood radius. These reported examples suggest that by having register functions with increased inputs and/or over a non-local neighborhood, generated sequences are likely to have improved randomness quality to pass all DIEHARD tests. These represents the best results published from CA models using a fixed, time-invariant CA transformation Φ such that $S^{(t+1)} = \Phi(S^{(t)})$.

Avoiding the weakness of m-CA and other CA using fixed time-invariant transformations is an interesting direction to pursue. From (4), it is seen that each state $S^{(t)}$ is given as the successive application of the same transformation Φ on the initial state $S^{(0)}$. Each successive state $S^{(t+1)}$ is always given by the same transformation Φ on its preceding state $S^{(t)}$. Linear regularities and structures are thus unavoidable in these sequences.

The Programmable CA [3] first suggested switching between f_{90} and f_{150} in a CA such that the resulting configuration is always an m-CA. The required control signals are pre-determined and stored on a ROM. The Controllable Control CA [7,8] also suggested switching between f_{90} and f_{150} in a CA but the control signals are computed by an

additional uniform CA with f_{30}. A separate set of signals to change the registers' behavior is then supplied by another uniform CA with f_{105}. The authors only showed that a 50-bit Controllable CA (equivalent to 150 registers in use) passed all 19 DIEHARD tests. It is not known how the model will perform with less or more registers. The Self-Programmable CA [10] suggested using a uniform CA with either f_{90} or f_{150}, and switching to their complementary functions f_{165} or f_{105}. Switching between complementary functions has simpler implementation since the control signal is XOR-ed directly to f_{90} or f_{150}. The control signal is derived from a uniform set of functions (with non-local neighborhood) over an additional layer of memory registers. Several Self-Programmable CA (with 36 to 48 registers) are shown to generate *n*-bit sequences passing all 19 DIEHARD tests.

By incorporating time-varying CA transformations $\Phi^{(t)}$ such that at each clock a different transformation is used, i.e. $S^{(t+1)} = \Phi^{(t)}(S^{(t)})$, smaller CA can also generate sequences passing all tests. The improvement comes at a cost – increased complexity is brought about by using more registers, additional external mechanisms, etc. such that analysis is obscured. Scalability to smaller or larger models is not easily performed since the functions used are generally obtained through an exhaustive [10,21] or evolutionary search [7,8,11].

Besides increased cost and complexity, the above models are difficult to analyze. To circumvent analysis, the authors in [7,8,11] used evolutionary approaches [18] with the

fitness function defined as the results of some relevant randomness metrics such as entropy, correlation, DIEHARD, etc. to design a few specific CA models. However, for CA models to be confidently deployed as PRNG in many applications, rigorous testing needs to be conducted – this slows down evolutionary approaches tremendously because of the vast number of fitness evaluations to be performed over many iterations.

Since m-CA have properties that are well studied and tools for analysis have been developed [3], we now propose the Layered CA model built using m-CA such that improved randomness quality in generated sequences is achieved by having time-varying CA transformations $\Phi^{(t)}$ at each clock while analysis is still possible and the simple structure is easily scalable.

III. LAYERED CELLULAR AUTOMATA (*L*-LCA)

As shown in Fig. 2, an *n*-bit Layered CA (*L*-LCA) consists of an *n*-bit main CA with additional *L* layers of *n*-bit memory to store previous states of the main CA. Since these previous states require additional registers for storage, such approaches are commonly referred to as "with memory" [13]. For example, the Self-Programmable CA (SPCA) [10] mentioned in Section II is also a CA model "with memory" and can be considered the predecessor to the *L*-LCA. In the *L*-LCA, the main CA is chosen to be an m-CA (shown in Section II) having a hybrid combination of registers using f_{90} or f_{150} instead of the uniform CA with f_{90} or f_{150} used in the SPCA. This slight change does not appear to be

significant when the *L*-LCA is examined at the register level; the implication will become apparent at the CA level. Each main CA register function can then be switched between $f_{90} \leftrightarrow f_{165}$ or ($f_{150} \leftrightarrow f_{105}$) using a control bit $c_j^{(t)}$ (note that all arithmetic is performed over the binary field $GF(2)$), i.e. $s_j^{(t+1)} = s_{j-1}^{(t)} + s_{j+1}^{(t)} + c_j^{(t)}$, $0 \le j \le n-1$.

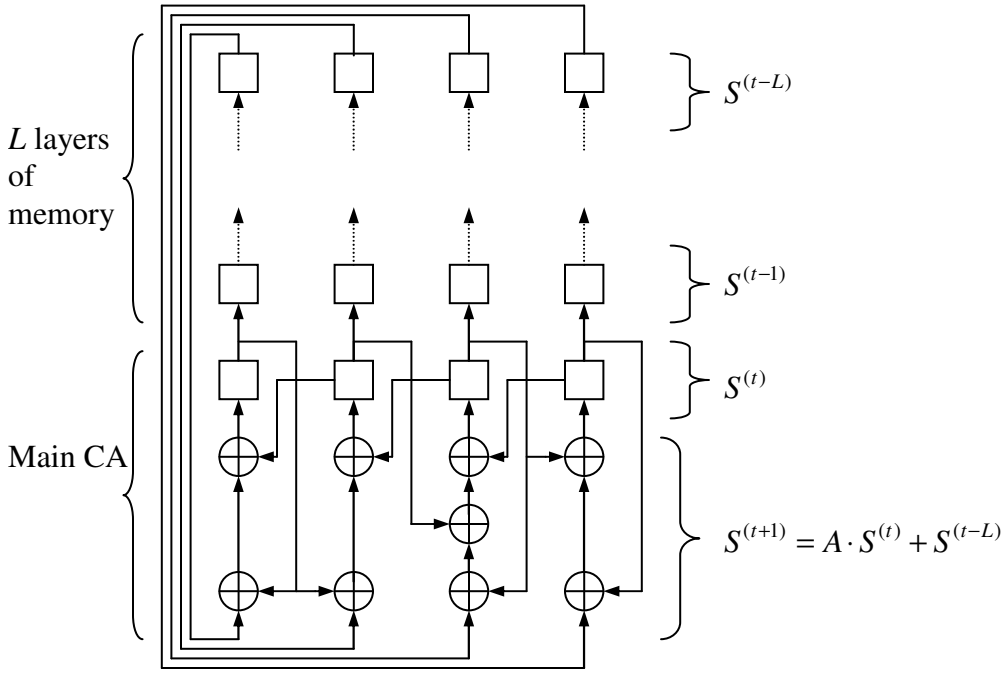

Fig. 2. The structure of a 4-bit *L*-LCA extended from the m-CA in Fig. 1

For the *L*-LCA, the *L* layers of memory are simply parallel arrays of registers with each layer holding a particular previous state $S^{(t-i)}$, $i = 1, 2, ..., L$ of the main CA. At each clock, the states of a lower layer are shifted up to its next higher layer - the main CA state is shifted into layer-*1* as $S^{(t-1)}$ while states $S^{(t-L)}$ from layer-*L* are used directly as control bits, i.e.

$$g \equiv f_{204} : s_j^{(t-L)} \rightarrow c_j^{(t)} \qquad (5)$$

The SPCA used a single layer of memory for storing $S^{(t-1)}$ and the g used to derive the control bit $c_j^{(t)}$ has the form $g: s_{j\pm a}^{(t-1)} + s_{j\pm b}^{(t-1)} \to c_j^{(t+1)}$, $0 \le a, b \le 3$, $a \ne b$. This g to derive the control bit from previous states must be separately determined for each SPCA of different size thus hindering scalability. The function g is simplified in the L-LCA so that the search for an optimal form of g is avoided, as well as simplification of both analysis and wiring complexity of the memory layers. The L-LCA can then be easily scaled by adding more layers of memory or through the use of a longer main CA based on widely available maximum length CA [17]. Putting all together, the L-LCA transformation is given by

$$S^{(t+1)} = A \cdot S^{(t)} + S^{(t-L)} \tag{6}$$

At clock $t=0$, $S^{(0)}$ and $S^{(-l)}, l = 1, 2, ..., L$ are initialized. Since (6) is actually a recurrence equation, the new state $S^{(t+1)}$ can always be expressed in a linear combination of transformations on the initial states $S^{(0)}$ and $S^{(-l)}, l = 1, 2, ..., L$. Denote the transformation on the initial states $S^{(0)}$ and $S^{(-l)}, l = 1, 2, ..., L$ at time t as $A_0^{(t)}$ and $A_{-l}^{(t)}, l = 1, 2, ..., L$ respectively. These transformations on the initial states are actually identical except for a time shift of one, i.e. $A_{-L}^{(t+1)} = A_0^{(t)}$ and $A_{-l}^{(t+1)} = A_{-l-1}^{(t)}, l = 1, 2, ..., L-1$. This can be shown via induction. At $t = L-2$, we have

$$S^{(L-1)} = A \cdot S^{(L-2)} + S^{(-2)}$$

$$S^{(L-1)} = A \cdot (A \cdot S^{(L-3)} + S^{(-3)}) + S^{(-2)}$$

$$S^{(L-1)} = A \cdot (A \cdot (A \cdot (\ldots (A \cdot (A \cdot S^{(0)} + S^{(-L)}) + S^{(-L+1)}) \ldots) + S^{(-4)}) + S^{(-3)}) + S^{(-2)}$$

$$S^{(L-1)} = A^{L-1} \cdot S^{(0)} + A^{L-2} \cdot S^{(-L)} + A^{L-3} \cdot S^{(-L+1)} + \ldots + S^{(-2)} \quad (7)$$

At $t = L-1$, $S^{(L)} = A \cdot S^{(L-1)} + S^{(-1)}$ can be expressed in terms of the initial states by similar treatment,

$$S^{(L)} = A^{L} \cdot S^{(0)} + A^{L-1} \cdot S^{(-L)} + A^{L-2} \cdot S^{(-L+1)} + \ldots + S^{(-1)} \quad (8)$$

For $t \geq L$, the transformation $A_0^{(t)}$ (and $A_{-l}^{(t)}, l = 1, 2, \ldots, L$) will be of the generalized form $A_0^{(t)} = c_t^{(t)} \cdot A^t + c_{t-1}^{(t)} \cdot A^{t-1} + \ldots + c_0^{(t)} \cdot A^0 = C^{(t)} \cdot A^{*(t)}$ where the coefficients $c_t^{(t)} \in [0,1]$, $C^{(t)} = [c_t^{(t)} c_{t-1}^{(t)} \ldots c_0^{(t)}]$ and $A^{*(t)} = [A^t A^{t-1} \ldots A^0]$. The corresponding transformations on $A_{-l}^{(t)}, l = 1, 2, \ldots, L-1$ can then be similarly expressed as $A_{-l}^{(t)} = C^{(t+l-L-1)} \cdot A^{*(t+l-L-1)}$.

Theorem 1 Each *L*-LCA state can be expressed as a linear combination of $S^{(0)}$ and $S^{(-l)}, l = 1, 2, \ldots, L$, and we have the following

$$S^{(t)} = \left(C^{(t)} \cdot A^{*(t)} \right) \cdot S^{(0)} + \sum_{l=1}^{L} \left(C^{(t+l-L-1)} \cdot A^{*(t+l-L-1)} \right) \cdot S^{(-l)} \quad (9)$$

Hereafter, we will illustrate some aspects of *1*-LCA while *L*-LCA with more layers can be studied by extending the methods directly. In *1*-LCA, the states are obtained by $S^{(t+1)} = A \cdot S^{(t)} + S^{(t-1)}$, that is the control bits $S^{(t-1)}$ from memory layer-*1* is XOR-ed with the newly computed main CA states. To start, we have the two initial states $S^{(0)}$ and

$S^{(-1)}$ to obtain $S^{(1)} = A \cdot S^{(0)} + S^{(-1)}$. The first few transformations for *1*-LCA are given in Table II.

Table II. Transformations on initial states

1-LCA	$S^{(0)}$	$S^{(-1)}$
$t=0$	1	0
1	A	1
2	A^2+1	A
3	A^3	A^2+1
4	A^4+A^2+1	A^3
5	A^5+A	A^4+A^2+1
6	A^6+A^4+1	A^5+A
7	A^7	A^6+A^4+1

A. Transformation Sequence of L-LCA

To analyze *L*-LCA, we propose an approach based on the properties of transformation matrices A from m-CA. Consider the state sequence generated in a single period by an *n*-bit m-CA, we can write using (4) (note that $A^{2^n-1} = I$)

$$\{S^{(1)}, S^{(2)}, \ldots, S^{(2^n-1)}\} = \{A^1, A^2, \ldots, A^{2^n-1}\} \cdot S^{(0)} \qquad (10)$$

We then have an ordered sequence of increasing exponents of A multiplied with $S^{(0)}$, an arbitrary initial state. Each exponent of A appears exactly once in a period. This transformation sequence $\{A^1, A^2, \ldots, A^{2^n-1}\}$ can be viewed as a fixed process inherent to the m-CA. Regardless of the initial state $S^{(0)}$ used, the m-CA always use this ordered sequence of transformations to generate the successive states in a single period. In

general, for CA models using a fixed transformation Φ, the corresponding transformation sequence will be of the form $\{\Phi^1, \Phi^2, \ldots, \Phi^p\}$. Since m-CA generates cyclically equivalent sequences with all possible non-zero initial states [3], we have the following.

Corollary 1 The transformation sequence $\{A^1, A^2, \ldots, A^{2^n-1}\}$ of an m-CA produces a cyclically equivalent state sequence $\{S^{(1)}, S^{(2)}, \ldots, S^{(2^n-1)}\}$ given any non-zero $S^{(0)}$.

Corollary 2 Any non-zero state $S^{(t)}$ can be given by the multiplication of another non-zero state $S^{*(t)}$ with a unique $A^d \in \{A^f \mid f = 1, 2, ..., 2^n - 1\}$ (called a phase difference).

Lemma 1 Matrix addition modulo two over the set $\{A^f \mid f = 1, 2, ..., 2^n - 1\}$ forms a group. Matrix addition is associative. The element A^{2^n-1} is the identity matrix I and for each element A^t, its inverse is $A^{-t \bmod (2^n-1)}$. It is well known that the modulo-2 summation of any two cyclic equivalent m-CA sequences results in another shift of the same sequence, which can be written as $A^\theta \cdot S^{(t)} = A^\alpha \cdot S^{(t)} + A^\beta \cdot S^{(t)}$. Since each exponent of A only appears once in a period of the sequence, each relation of the following form $A^\theta = A^\alpha + A^\beta$ is closed.

Theorem 2 From (9), the *L*-LCA transformation at each clock consists of matrix additions. By Corollary 2 and Lemma 1, these composite transformations can be reduced

to a single $A^{(t)} \in \{A^f \mid f = 1, 2, ..., 2^n - 1\}$ and (9) is reduced to the following simple form and a corresponding transformation sequence (see (10)) of the form $\{A^{(t)}\}_{t=1}^{t=p}$ can be obtained.

$$S^{(t)} = A^{(t)} \cdot S^{(0)} \tag{11}$$

Next, we proceed by obtaining a Cayley table [16] with the members of $\{A^f \mid f = 1, 2, ..., 2^n - 1\}$. A Cayley table is a table with row and column indices being the members of a group. The entries of the Cayley table are the results of the group operation - matrix addition on any two members, and each result is also a member of the group. The complete set of Cayley relations for the CA in Fig 1 is shown in Table III, where the row/column indices indicate the exponents of A, and each table entry is the relation $A_{f,t} = A^f + A^t$.

Table III Cayley relations ($A_{f,t} = A^f + A^t$) for primitive polynomial $x^4 + x^3 + 1$

	t=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f=1	-	13	10	5	4	11	9	14	7	3	6	15	2	8	12
2	13	-	14	11	6	5	12	10	15	8	4	7	1	3	9
3	10	14	-	15	12	7	6	13	11	1	9	5	8	2	4
4	5	11	15	-	1	13	8	7	14	12	2	10	6	9	3
5	4	6	12	1	-	2	14	9	8	15	13	3	11	7	10
6	11	5	7	13	2	-	3	15	10	9	1	14	4	12	8
7	9	12	6	8	14	3	-	4	1	11	10	2	15	5	13
8	14	10	13	7	9	15	4	-	5	2	12	11	3	1	6
9	7	15	11	14	8	10	1	5	-	6	3	13	12	4	2
10	3	8	1	12	15	9	11	2	6	-	7	4	14	13	5
11	6	4	9	2	13	1	10	12	3	7	-	8	5	15	14
12	15	7	5	10	3	14	2	11	13	4	8	-	9	6	1
13	2	1	8	6	11	4	15	3	12	14	5	9	-	10	7
14	8	3	2	9	7	12	5	1	4	13	15	6	10	-	11
15	12	9	4	3	10	8	13	6	2	5	14	1	7	11	-

The above suggests that the entire Cayley table has to be computed in order to obtain the transformation sequence in reduced form $\{A^{(t)}\}_{t=1}^{t=p}$ for an *L*-LCA. However, given a particular relation $A^a + A^b = A^c$ and by multiplying it with A^f for $1 \le f < 2^n - 1$, we can get the other relations $A^{a+f} + A^{b+f} = A^{c+f}$ where the exponents are taken modulo $2^n - 1$. We then proceed by first pre-computing relations of the form

$$A + A^i = A^{k_i} \tag{12}$$

where $2 \le i, k_i \le 2^n - 1$. All the remaining relations of the form $A^a + A^b = A^c$ are easily obtained using $i = a - b + 1$ and $c = k_i + b - 1$. The above pre-computed $n-1$ relations of the form $A + A^i = A^{k_i}$, can be re-used for analysis of all *L*-LCA with different layers of memory as well as other CA designs using matrix addition over the same set $\{A^f \mid f = 1, 2, ..., 2^n - 1\}$.

B. Period analysis

For convenience, we refer to the *L*-LCA as an *n*-bit CA model since only *n*-bit words will be sampled at each clock from the main CA states, although the actual number of registers *m* contained is *m=(L+1)*n* registers while an *n*-bit CA contains exactly *n* registers. In configurable generators, the number of registers used is more than the bits sampled for output because some registers hold the system configuration bits which determine the actual generator being used. For example, the Programmable CA [3] has

configuration bits held in a ROM and these bits can be fixed for a particular session of generating outputs. In keystream generators, these configuration bits are considered as part of the secret key. Hence these registers cannot be sampled for output to prevent from leaking the secret key. Generally, such generators with *m* registers ($m > n$) may generate cyclically different *n*-bit sequences with different periods depending on the *m*-bit initial state used. Furthermore, some *n*-bit states may appear more than once during a single period and/or may appear in more than one sequence. To find the period characteristics for these state sequences, all possible *m*-bit initial states may have to be attempted.

The *1*-LCA uses *m=2n* registers for $(S^{(t)}, S^{(t-1)})$ while only *n*-bit outputs are taken at each clock as $S^{(t)}$. We can write (9) as $S^{(t)} = (A_0^{(t)} + A_{-1}^{(t)} \cdot A^d) \cdot S^{(0)}$ using Corollary 2, where A^d is the phase difference between the initial state pair $(S^{(0)}, S^{(-1)})$.

Theorem 3 All state sequences $\{S^{(1)}, S^{(2)}, ..., S^{(p)}\}$ generated by the *1-LCA* can be categorized into similar groups based on the phase difference A^d between the initial state pair $(S^{(0)}, S^{(-1)})$. Each A^d will result in a different type of sequence. Let $\{A_0^{(t)} + A_{-1}^{(t)} \cdot A^d\}_{t=1}^{t=p}$ represent the transformation sequence for an arbitrary initial state pair $(S^{(0)}, S^{(-1)})$ with $S^{(0)} = A^d \cdot S^{(-1)}$. There all $2^n - 1$ such initial state pairs $(S^{*(0)}, S^{*(-1)})$ having the same phase difference A^d , i.e. $S^{*(0)} = A^k \cdot S^{(0)}$ and $S^{*(-1)} = A^k \cdot S^{(-1)}$ $(1 \le k \le 2^n - 1)$, and their transformation sequences are given by $\{A_0^{(t)} + A_{-1}^{(t)} \cdot A^d\}_{t=1}^{t=p} \cdot A^k$.

The resulting state sequences $\left(\{A_0^{(t)} + A_{-1}^{(t)} \cdot A^d\}_{t=1}^{t=p} \cdot A^k\right) \cdot S^{(0)}$ have the same period since each transformation is multiplied by the same A^k.

For example, with the 4-bit *1*-LCA whose $A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ and substituting $S^{(1)} = A^2 \cdot S^{(0)}$ into (6), we obtained the following transformation sequence with period $15 = 2^4 - 1$, $\{A^{15}, A^{13}, A^{11}, A^9, A^7, A^5, A^3, A^1, A^{14}, A^{12}, A^{10}, A^8, A^6, A^4, A^2\} \cdot S^{(0)}$. Notice that the transformation sequence has all members of $\{A, A^2, ..., A^{2^4-1}\}$ appearing exactly once and is a "permuted" version of the transformation sequence for a 4-bit $\text{HCA}_{f_{90}/f_{150}}$ thus indicating a time-varying transformation $A^{(t)}$ is being used at each clock.

Other types of sequences can be generated by using initial state pairs having different phase differences. However, not all of these sequences will contain all the members of $\{A, A^2, ..., A^{2^4-1}\}$ in a single period. For example, if we use $S^{(-1)} = S^{(0)}$, then the resulting transformation sequence for the above *1*-LCA is $\{A^{15}, A^{12}, A^7, A^{11}, A^2, A^9, A^8, 0, A^8, A^9, A^2, A^{11}, A^7, A^{12}, A^{15}\} \cdot S^{(0)}$.

After determining the phase difference A^d that yields the desired sequence type, appropriate initial state tuples can be used to generate more such sequences. The task of analyzing all possible state sequences is then made simpler by analyzing only each representative transformation sequence $\{A^{(t)}\}_{t=1}^{t=p}$. Being a deterministic generator, *1*-LCA

then generates a periodic sequence with period p when the successive transformations $(A^{(t)}, A^{(t-1)})$ and $(A^{(t+p)}, A^{(t+p-1)})$ are identical.

With Lemma 4.5, the period can then be obtained using direct computation of the transformations. By Theorem 1, transformation sequence on $S^{(-1)}$ lags that of $S^{(0)}$ by one clock. Thus we only need to compute the transformations $A_0^{(t)}$, $t = 1, 2, ...$ on an arbitrary $S^{(0)}$. After accounting for the phase difference A^d, each transformation in the overall transformation sequence is then given by

$$A^{(t)} = A_0^{(t)} + A_0^{(t-1)} \cdot A^d \qquad (13)$$

IV. EXPERIMENTAL SETUP AND RESULTS

A. DIEHARD Randomness Test Suite

We examined the sequences from *L*-LCA using the DIEHARD randomness test suite [15] (detailed descriptions for each of the 19 tests can be found in the given reference). We start with an *n*-bit (*n=10 to 48*) *1*-LCA, and more layers are subsequently added, i.e. *L=2,3,4*. The m-CA configurations are also tested as ordinary CA for comparison benchmark. Testing requires a minimum of 10Mbytes of random numbers and is conducted on two types of sequences obtained from each PRNG tested. At each clock, we put all *n* bits from the *n*-bit main CA into an *n*-bit sequence, so each CA is executed for $10^7 * 8/n$ clocks. Next we examine the randomness quality of single-bit sequences generated by each register in the main CA. At each clock, the bit from each register is put into *n* single-sequences, so each CA is executed for $10^7 * 8$ clocks. Each sequence is then

tested and finally the average results over all single-bit sequences are computed. The above procedure is then repeated for 20 different initial states and the averaged results plotted in Fig 3-10 - the vertical axis shows the number of DIEHARD tests passed (maximum score 19) and the horizontal axis shows the length of the main CA.

Based on these results, the performance of *L*-LCA exceeds that of the conventional m-CA for PRNG purposes. The results here also include two variants of the *L*-LCA with nonlinear functions f_{30} and f_{45} [1] (instead of f_{204}) used to derive the control bit because these functions are used to examine if the *L*-LCA structure increases linear complexity in the next section,

$$g \equiv f_{30} : s_{j-1}^{(t-L)} \; XOR \; \left(s_j^{(t-L)} \; OR \; s_{j+1}^{(t-L)} \right) \rightarrow c_j^{(t)} \tag{14}$$

$$g \equiv f_{45} : s_{j-1}^{(t-L)} \; XOR \; \left(s_j^{(t-L)} \; OR \; \overline{s_{j+1}^{(t-L)}} \right) \rightarrow c_j^{(t)} \tag{15}$$

In Fig. 3, the DIEHARD results for single-bit sequences generated by the m-CA are observed to be very inconsistent – some m-CA with more registers pass very few DIEHARD tests but a slowly increasing trend is still observed. The results for m-CA with $48 < n \leq 64$ (not shown due to space constraints) show the same inconsistency - weak results (passing only 3 tests) are still obtained from some m-CA (51-, 53-, 63-bit). For the *1*-LCA with f_{204}, there are 16 cases that passed all DIEHARD tests. However, there are several cases failing all tests. Interestingly, these results seem to "track" those for the m-CA – for the same main CA used if m-CA result is weak, the *1*-LCA fails all tests while if the m-CA result is good, *1*-LCA passes all tests. The randomness quality of sequences

generated by the *1*-LCA is thus strongly affected by the m-CA configuration used for its main CA. Unlike the inconsistent results from m-CA and *1*-LCA with f_{204}, the results of *1*-LCA with f_{30} and f_{45} showed less fluctuation. More tests are passed as more registers are used, but no sequences passing all DIEHARD tests are found by us. The results based on f_{45} are consistently better than f_{30}.

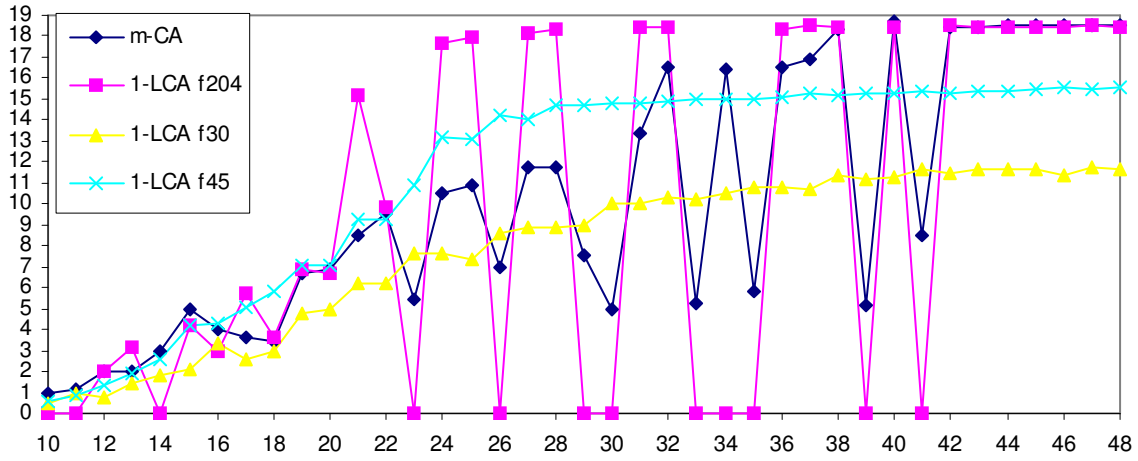

Fig. 3. Number of DIEHARD tests (max. 19) passed by single-bit sequences from 10- to 48-bit m-CA and *1*-LCA

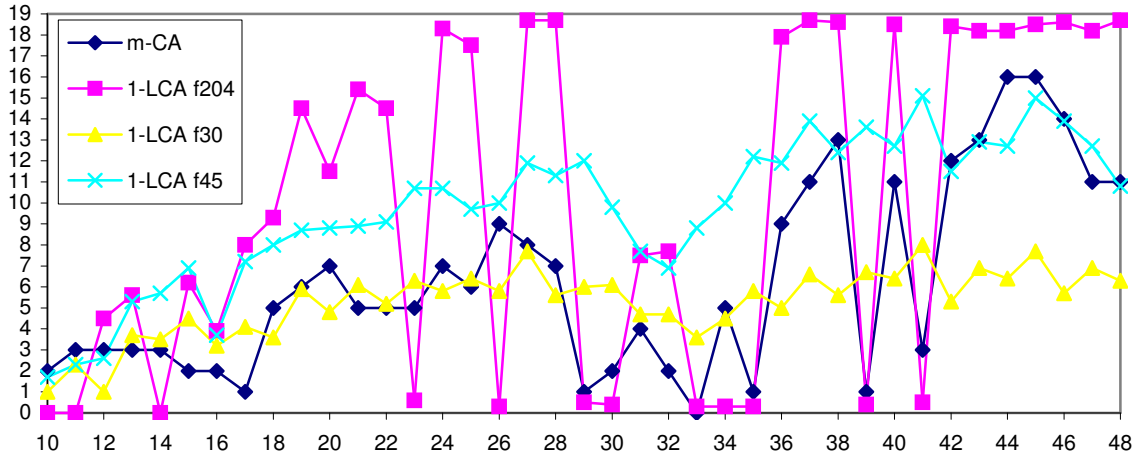

Fig. 4. Number of DIEHARD tests (max. 19) passed by *n*-bit sequences from 10- to 48-bit m-CA and *1*-LCA

In Fig. 4, relative comparison of the DIEHARD results from *n*-bit sequences generated by all four methods show similar trends as observed in the DIEHARD results from single-bit sequences. Compared to the previous DIEHARD results, the results from *n*-bit sequences generated by m-CA and *1*-LCA with f_{30} and f_{45} are visibly weaker. *1*-LCA with f_{45} still produces better sequences than with f_{30}. Also, we did not find any m-CA generating sequences that can pass at least 18 tests. The DIEHARD results from *1*-LCA with f_{204} are quite similar to the previous test cases – if the single-bit sequences passed at least 18 tests, the corresponding *n*-bit sequences are likely to do as well (except the 31- and 32-bit cases).

In Fig. 5, the *2*-LCA f_{204} with longer than 15-bit main CA generate single-bit sequences that can pass all DIEHARD tests and for all register lengths, the number of DIEHARD tests passed is larger than the m-CA. The *2*-LCA f_{30} and f_{45} results are now very similar and consistent, more than 16 tests are passed by those with more than 23-bit main CA; but no sequences are observed to pass all DIEHARD tests.

Fig. 6 shows that *n*-bit sequences from many *2*-LCA with f_{204}, f_{30}, f_{45} can pass at least 18 DIEHARD tests, and all sequences have better performance than m-CA sequences. For main CA with more than 25 registers, *2*-LCA with f_{204}, f_{30}, f_{45} have almost similar performance.

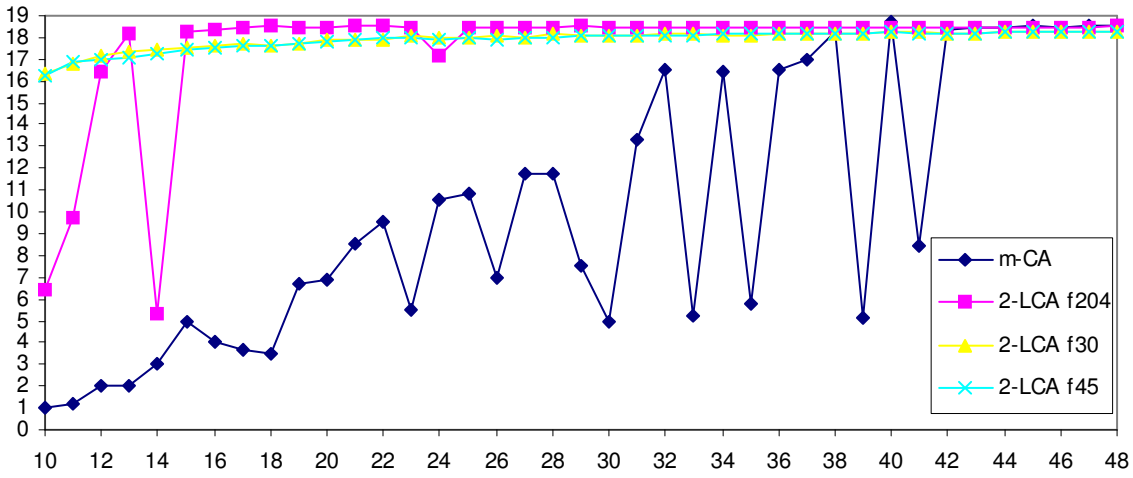

Fig. 5. Number of DIEHARD tests (max. 19) passed by single-bit sequences from 10- to 48-bit m-CA and *2*-LCA

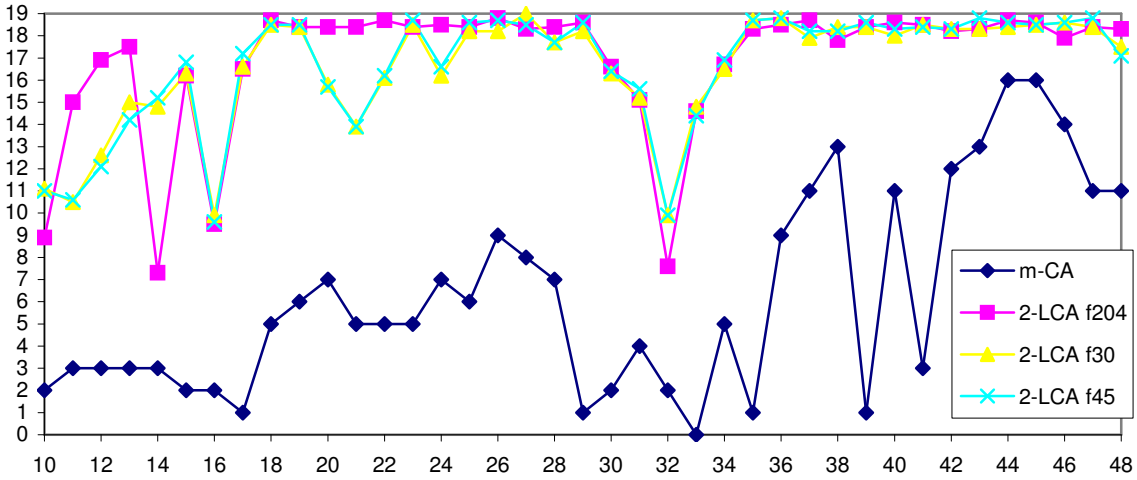

Fig. 6. Number of DIEHARD tests (max. 19) passed by *n*-bit sequences from 10- to 48-bit m-CA and *2*-LCA

In Fig. 7 and 9, all single-bit-sequences from *3*-LCA and *4*-LCA passed at least 18 DIEHARD tests. The results in Fig. 8 and 10 for *n*-bit sequences from *3*-LCA and *4*-LCA repeat the trends we have seen for *2*-LCA. With 16- and 32-bit main CA, *3*-LCA and *4*-LCA results are still relatively weak although the results are slightly improved with

increased layers of memory. With main CA of other lengths, the improvement in results can also be seen by using more layers of memory such that *4*-LCA with (small) 10-bit main CA and *3*-LCA with 11-bit main CA can pass 19 DIEHARD tests.

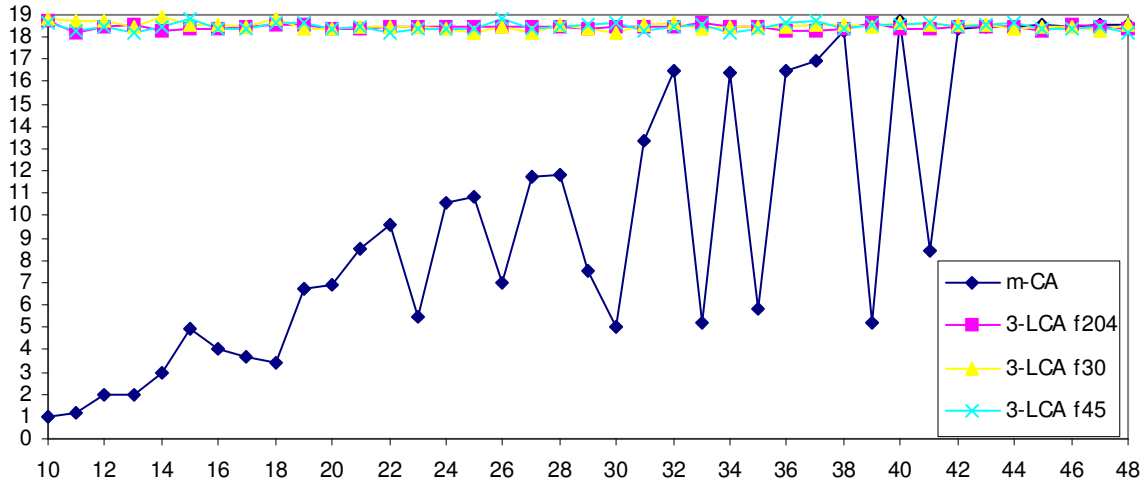

Fig.7. Number of DIEHARD tests (max. 19) passed by single-bit sequences from 10- to 48-bit m-CA and *3*-LCA

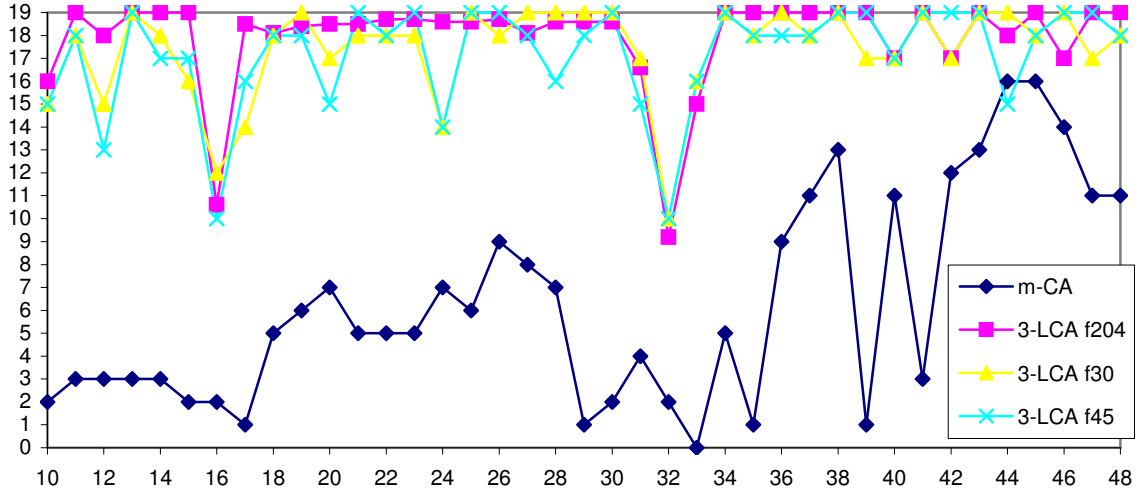

Fig. 8. Number of DIEHARD tests (max. 19) passed by *n*-bit sequences from 10- to 48-bit m-CA and *3*-LCA

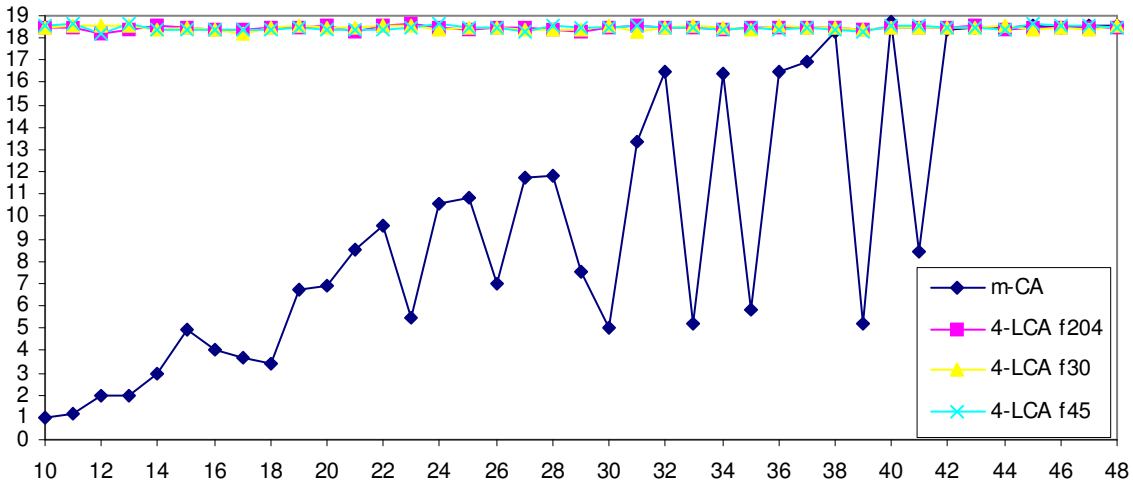

Fig. 9. Number of DIEHARD tests (max. 19) passed by single-bit sequences from 10- to 48-bit m-CA and *4*-LCA

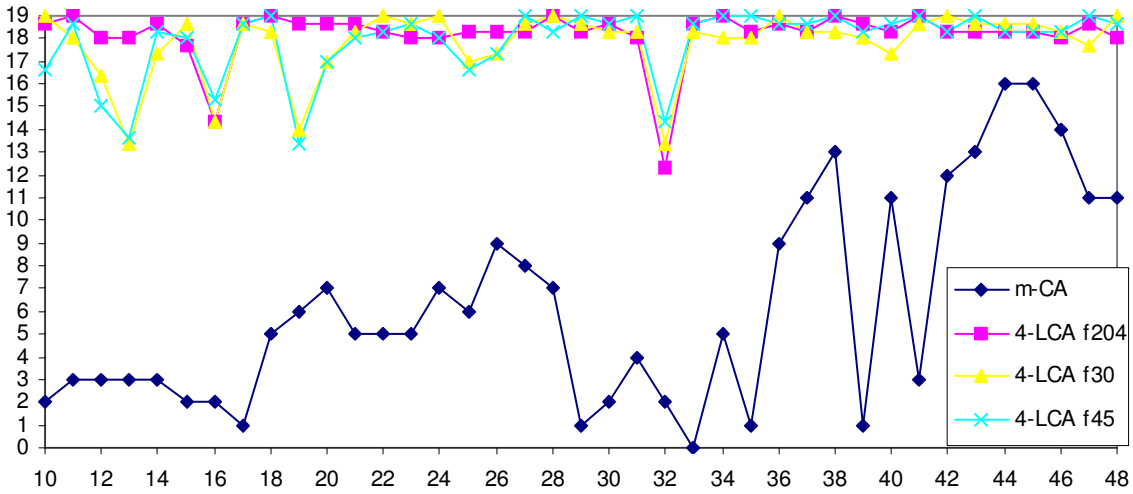

Fig. 10. Number of DIEHARD tests (max. 19) passed by *n*-bit sequences from 10- to 48-bit m-CA and *4*-LCA

From these results, we can see that for all the different main CA used, adding more layers of memory will improve the DIEHARD results for the *n*-bit sequences. With $L>1$, most single-bit sequences passed at least 18 tests regardless of the main CA used. Thus, simple

scalability is achieved. At the same time, it is important to check the minimum overall count of registers used to generate both single-bit and *n*-bit sequences passing all DIEHARD tests to arrive at fair implementation cost comparison - *1*-LCA requires at least 28*2 = 56 registers, *2*-LCA requires 18*3 = 54 registers, *3*-LCA requires 11*4 = 44 registers, *4*-LCA requires 10*5 = 50 registers. The m-CA requires 38 registers minimum for single-bit sequences while no *n*-bit sequence is observed to pass at least 18 DIEHARD tests.

Comparatively, the Self-Programmable CA is shown to pass all DIEHARD tests with 36 to 48 registers (see Section II.C). This saving in register count is offset by longer propagation delay through the set of functions *g* with non-local neighborhood to derive each control bit, causing layout complications as well as non-scalability of design. In contrast, the *1*-LCA has simplicity and speed advantages because the previous main CA states are shifted without any processing such that the $S^{(t-L)}$ state is used directly as the control signal (see (6)). Furthermore, the results for *L*-LCA showed that consistent results can be obtained from a range of different main CA used while adding more layers of memory led to improved results. The simplifications made to allow simplified analysis and scalability for the *L*-LCA outweighs this minor disadvantage. The *L*-LCA only used time-varying transformations $\Phi^{(t)}$ to improve randomness quality of generated sequences – this possibly explained the examined *L*-LCA require more registers than the SPCA to pass all DIEHARD tests. The *L*-LCA can also be studied with a linear function *g* with non-local neighborhood to derive the control bit.

B. Linear Complexity

Besides the above statistical tests, it is also important to check the linear complexity of the sequences generated by the *L*-LCA. Linear complexity of an arbitrary sequence is defined as the number of registers required in an equivalent linear system to reproduce that sequence. The Berlekamp-Massey algorithm [13] is used to measure linear complexity from the sequences generated. Linear complexity for *n*-bit sequences from an *n*-bit m-CA is simply $LC = n^2$, which is why m-CA sequences are usually used with post-processing to remove inherent linearity.

Linear complexity tests for *n*-bit sequences are then conducted using smaller *1*-LCA and *2*-LCA with 5- to 14-bit main CA (ten randomly selected sequences are tested for each case due to resource limitations). The averaged results are shown in Fig. 5.9 - the vertical axis shows the linear complexity for the various *L*-LCA and the horizontal axis shows the length of the main CA. Since we have only used linear operations in the *L*-LCA, the linear complexity is equivalent to the number of registers used, $n^2 \times (L+1)$.

Given that the *L*-LCA structure has been shown to improve DIEHARD results, we like to see if linear complexity can also be improved through the *L*-LCA structure. The well-known nonlinear functions f_{30} and f_{45} are used to replace f_{204} in (5). The nonlinear functions f_{30} and f_{45} are called simple because they only have three inputs. When used in normal uniform CA, the linear complexity of sequences generated with f_{30} and f_{45} do not show a substantial increase compared to the purely linear m-CA. When used in *1*-

LCA and *2*-LCA, the linear complexity increases rapidly with the number of registers used. It is observed that *1*-LCA f_{45} generated sequences with higher linear complexity than *1*-LCA f_{30} while for *2*-LCA the linear complexity of sequences generated by both functions are similar. This is very similar to their DIEHARD results.

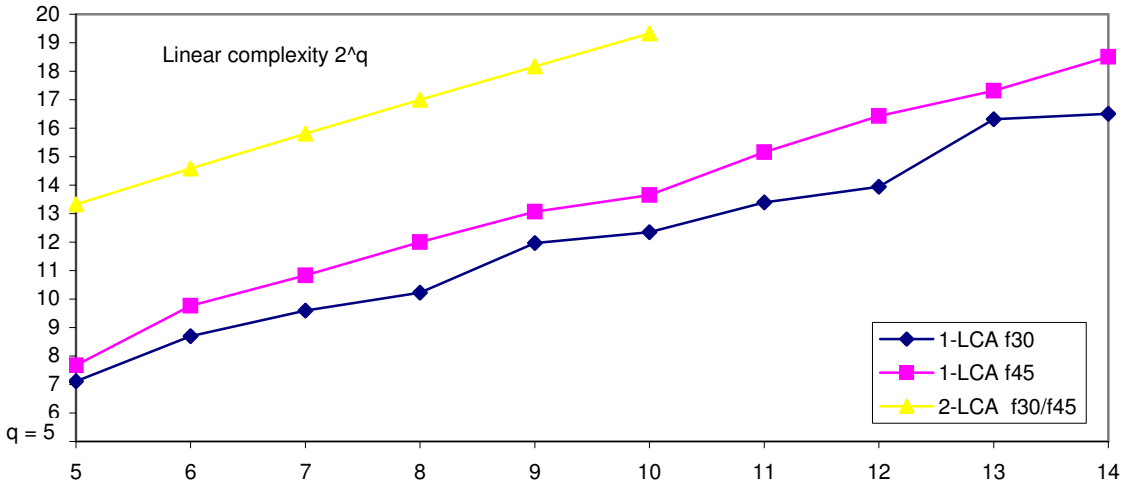

Fig. 11 Linear complexity of *1*-LCA and *2*-LCA

From the DIEHARD results and linear complexity measured, it is clear that the *L*-LCA structure does improve the DIEHARD results substantially when f_{204}, f_{30} or f_{45} is used. The linear complexity is also increased when f_{30} or f_{45} is used.

V. CONCLUSION

In this paper we proposed the Layered CA (*L*-LCA), consisting of a main CA with additional *L* layers of memory for storing previous states of the main CA. The operating speed for *L*-LCA is fast as interconnections in the main CA are local and only shifting of

states is associated between each memory layer without any additional processing. These previous states are used directly to switch the main CA registers' function while the main CA is chosen from the widely available configurations of maximum length CA [17]. We are able to reduce the time-varying transformations used at each clock to an equivalent transformation sequence that facilitates analysis.

Extensive experiments on *L*-LCA *(L=1,2,3,4)* showed improved randomness quality such that at least 18 DIEHARD tests are passed. As more layers of memory are added or the main CA's length is increased, more tests are passed. The *1*-LCA and *2*-LCA structures are shown to increase linear complexity when simple non-linear functions f_{30} and f_{45} are used in place of the linear function f_{204} to select the function for each main CA register.

Additional memory layers can also be used with other forms of linear finite state machines such as the linear feedback shift registers (LFSR) [20] which are widely used in hardware pseudorandom number generators. LFSR have well-known disadvantage of highly correlated single-bit sequences since each adjacent single-bit sequence is "time-shifted" by one clock [19]. It will be interesting to be study the randomness quality of sequences generated by LFSR with additional memory layers.

REFERENCES

[1] S. Wolfram, "Theory and applications of Cellular Automata: Including Selected Papers 1983-1986", World Scientific publishing Co., Inc., River Edge, NJ. 1986.

[2] S. Nandi, B. K. Kar, and P. Pal Chaudhuri, “Theory and Applications of Cellular Automata in Cryptography”, IEEE Transactions on Computers, Vol. 43, pp. 1346-1357, 1994.

[3] P. Pal Chaudhuri, D. Roy Chowdhury, S. Nandi and S. Chattopadhyay, “Additive Cellular Automata Theory And Applications”, Volume 1, IEEE Computer Society Press, Los Alamitos, ISBN 0-8186-7717-1, California, 1997.

[4] P. D. Hortensius, R. D. Mcleod, Werner Pries, D. Michael Miller and H. C. Card, “Cellular Automata-Based Pseudorandom Number Generators for Built-in Self-Test”, IEEE Transactions on Computer-Aided Design, Vol. 8, No. 8, pp. 842-859, 1989.

[5] D. E. Knuth, “The Art of Computer Programming, Vol. 2: Seminumerical Algorithms”, 3rd ed., Reading, Mass.: Addison-Wesley, 1998.

[6] P. Hellekalek, “Good Random Number Generators Are (Not So) Easy to Find”, In Mathematics and Computer in Simulation, 46, pp. 485-505, 1998.

[7] Sheng-Uei Guan and Shu Zhang, “An Evolutionary Approach to the Design of Controllable Cellular Automata Structure for Random Number Generation”, IEEE Transactions on Evolutionary Computation, Vol. 7, No. 1, pp. 23 -36, Feb. 2003.

[8] Sheng-Uei Guan and Shu Zhang, “Incremental Evolution of Cellular Automata for Random Number Generation”, International Journal of Modern Physics C, Vol. 14, No. 7, pp. 881-896, Sep. 2003

[9] Sheng-Uei Guan, Shu Zhang, and Marie Therese Quieta, "2-d CA Variation with Asymmetric-Neighborship for Pseudorandom Number Generation", pp. 378-388, IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, Vol. 23, No. 3, Mar. 2004.

[10] Sheng-Uei Guan and Syn Kiat Tan, "Pseudorandom Number Generation with Self Programmable Cellular Automata", IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, Vol. 23, No. 7, pp. 1095-1101, July 2004.

[11] M. Tomassini, M. Sipper and M. Perrenoud, “On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata”, IEEE Transactions on Computers, 49, pp. 1146-1151, 2000.

[12] M. Tomassini and M. Perrenoud, “Cryptography With Cellular Automata”, Applied Soft Computing, Vol. 1, pp. 151 – 160, 2001.

[13] R. A. Rueppel, “Analysis and Design of Stream Ciphers”, Springer Verlag, 1986.

[14] M. Mihaljevic and Hideki Imai, “A Family of Fast Keystream Generators based on Programmable Linear Cellular Automata over GF(q) and Time-variant Table”, IEICE Trans. Fundamentals, Vol. E82-A, No. 1, pp. 32-39, 1999.

[15] Marsaglia, “Diehard”, http://stat.fsu.edu/~geo/diehard.html, 1998.

[16] Joseph A. Gallian, “Contemporary Abstract Algebra”, Houghton Mifflin Company, 2002.

[17] Kevin Cattell and Shujian Zhang, “Minimal Cost One-Dimensional Linear Hybrid Cellular Automata of Degree Through 500”, Journal of Electronic Testing: Theory and Applications, Kluwer Academic Publishers, Boston, Vol. 6, pp. 255-258, 1995.

[18] N. Chaiyaratana and A.M.S. Zalzala, “Recent Developments in Evolutionary and Genetic Algorithms: Theory and Applications”, In Proc. of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, pp. 270-277, 1997.

[19] Mrugalski, G., Rajski, J. and Tyszer, J., “Cellular Automata-Based Test Pattern Generators with Phase Shifters”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol 19, Issue 8, pp. 878 - 893, Aug. 2000.

[20] Solomon W. Golomb and Soloman Golomb, “Shift Register Sequences”, Aegean Park Press, 1981.

[21] B. Shackleford, M. Tanaka, R. Carter, and G. Snider, “High-Performance Cellular Automata Random Number Generators for Embedded Probabilistic Computing Systems”, in Proc. NASA/DOD Conference on Evolvable Hardware, pp. 191-200, 2002.