CNNA'96: Fourth IEEE International Workshop on Cellular Neural Networks and their Applications, Seville, Spain, June 24-26, 1996

PSEUDO-RANDOM SEQUENCE GENERATION USING The CNN UNIVERSAL MACHINE With Applications to Cryptography

Kenneth R. Crounse *Tao Yang* *Leon O. Chua*
Electronics Research Laboratory,
Cory Hall, University of California, Berkeley, CA 94720 USA
crounse@eecs.berkeley.edu
taoyang@eecs.berkeley.edu
chua@eecs.berkeley.edu
phone: 1 (510) 642-5311
FAX: 1 (510) 643-8869

ABSTRACT: *A good source of reproducible random-looking data is important in many applications ranging from simulation of physical systems, communications, and cryptography. It is demonstrated that the CNN Universal Machine [1] (or the Discrete-time CNN [2]) is capable of producing a two-dimensional pseudo-random bit stream at high speeds by means of cellular automata (CA). First, the random properties of some irreversible two-dimensional CA rules, selected by applying mean-field theory, are analyzed by a battery of statistical tests. Second, a special class of reversible CA are considered for random number generation and are shown to have some of the desirable properties of physics-like models. Finally, as an example application for random number generation on the CNNUM, some cryptographic schemes are proposed.*

1. Introduction

High-speed high-quality random number generation is a significant problem, especially in cryptography[3]. Here we present methods for random number generation on the CNNUM for two reasons: First as a native source of good random numbers for use in CNNUM algorithms for such areas as image processing and modeling physical systems, and second as a basis for implementing exceptional, and possibly novel, cryptographic algorithms.

Random numbers and encryption schemes will be implemented on the CNNUM by simulation of Cellular Automata [4, is a nice introduction]. The randomness of CA have been investigated in other places, but have only been well-analyzed for a certain one-dimensional rule[5]. Some two-dimensional CA have been noted to be random[6, 4], but have not been sufficiently analyzed. However, the two-dimensional CA show great promise as random number generators. Due to its massive parallelism, the CNNUM offers the possibility of simulating two-dimensional CA at high speeds while offering very large (> 1000 bits) key/state space.

1.1. CA and the CNNUM

The cellular automata are a class of dynamical systems which are discrete in state, space, and time. The states of a cellular automaton evolve synchronously according to a state transition rule which operates within a local neighborhood of some radius, r, of the state being updated, typically in a time and space-invariant manner. The number of states a cell can take on is denoted by k. The order of the CA is the number of previous times the states are required to compute the next state. In this paper we will discuss both first-order and an important class of second-order CA with $k = 2$ (binary), $r = 1$ (3×3 neighborhood), in two-dimensional space.

The first-order CA dynamical equations can be written

$$s_{i,j}(n+1) = f\left[s_{\mathcal{N}_{i,j}}(n)\right] \tag{1}$$

where $\mathcal{N}_{i,j}$ are the indices for the 9-neighborhood of cell (i, j) and f is a boolean function of the states in the neighborhood. A total of $2^9 = 512$ neighborhood configurations are possible. The boolean function which defines the transition rule must specify a next state for each of these neighborhoods. Since each can have two possible next states there are therefore $2^{512} \approx 10^{154}$ possible cellular automata rules in the considered class. It has been demonstrated that the CNN Universal Machine is capable of implementing an arbitrary cellular automaton from this class[7]. In addition, algorithms have been developed for designing such templates[8].

For the second order CA, the state transition function for the next states can in general be any boolean function of the current and previous states. However, boolean functions of 18 variables are a bit unwieldy, so we restrict ourselves to the special form:

$$s_{i,j}(n+1) = g\left[f\left[s_{\mathcal{N}_{i,j}}(n)\right], s_{i,j}(n-1)\right] \tag{2}$$

where g is a boolean function of two variables, the center state at time n and time $n-1$. This form also has the advantage of being directly implementable on a CNNUM by using an addition local logic memory and one additional local logic operation to perform the extra boolean function.

1.2. Tests of Randomness

Two-dimensional sequences generated by our methods may be considered effectively random if no possible procedure can identify a pattern in them, or allow their behavior to be predicted. We need be content with accumulating circumstantial evidence by trying various statistical procedures and finding that they reveal no regularities. The basic idea is to compare statistical results on the generated sequences with those calculated for sequences whose elements occur purely according to probabilities [5, paraphrased].

There are many tests that can be used to help establish the random characteristics of a bit-stream [9–11]. We have implemented both the χ^2-test and the Kolmogorov-Smirnov (K-S) test to compare various properties of our pseudo-random sequences to those that would be expected from a 'truly' random sequence. Typically the bit-stream is blocked to form a sequence of random numbers. We have tested the following properties of these sequences of numbers: uniform distribution, repeated occurrences of number pairs (serial test), length of sequences that are within a certain number range (gap test), and the number of unique numbers in a sequence of five (poker test).

The three-dimensional bit stream (space and time) produced by the CA must be blocked in a form appropriate for these tests. Since correlations seem more likely to be local effects, we apply the tests to the CA states in two ways: small blocks in space for a given time, and small blocks in time for a given cell.

We have also investigated the use of autocorrelation as a measure of randomness and spectral characteristics, which is important in such applications as spread-spectrum communications.

2. Randomness in Two-dimensional Cellular Automata

Previous research on using cellular automata for random sequence generation has focused on one-dimensional (line) irreversible automata [5, 12, 13]. Typically, the state of a particular cell as a function of discrete-time is chosen for the random bit sequence. There are two widely used rules, which have been characterized as chaotic, or 'Class 3', rules [14]. The resulting random sequences have been shown to compare favorably with conventional techniques for random number generation.

Although, the CNNUM could implement such one-dimensional CA, it seems likely that the greater degree of freedom found in two dimensions would allow a greater amount of 'mixing' to occur among the states, giving a more random sequence. Only a few Class 3 two-dimensional rules have been identified in the literature, although it has been stated that most 2-D rules are chaotic[6].

We are investigating the random properties of 2-D CA rules by the battery of techniques discussed in Section 1.2. The space of CA state bits from which to choose the random sequence is three dimensional. Choosing is a difficult matter, and depends on the application and the quality of randomness required. For instance, it is sometimes useful to think of the random sequence as one-dimensional, in which case high-quality can be obtained by massively sub-sampling the CA array in space and time. Whereas, in other situations it is useful to produce random images at high rates, for instance in simulation of physical processes or video encryption. In this case, we would like both spatial and temporal correlations to be small and therefore the CA rule must have exceptional qualities.

2.1. First Order Two-dimensional Cellular Automata

The enormous number of possible CA make it a difficult task to choose among them for ones exhibiting 'good' randomness. We are investigating the use of CA mean-field theory [15] to quickly screen candidate rules and, ultimately, as a synthesis tool.

To demonstrate, two first-order candidate CA rules were chosen. The first (Rule 1) is the outer-totalistic rule 143954 which was shown in [6] to have Class 3 behavior. Using the method of [8], a sequence of 8-templates was found by which a single step of this CA rule can be implemented on a CNNUM. The second (Rule 2) under investigation is a slight variation of a RNG rule used for simulation of physical systems[4] and can also be considered a type of two-dimensional generalization of Wolfram's one-dimensional rule. The dynamics are given

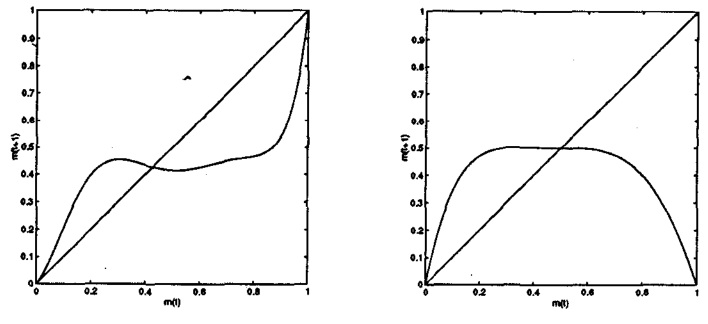

Figure 1: The mean-field maps for CA Rule 1 (left) and Rule 2 (right). Note that Rule 2 has a fixed point exactly at the equiprobable density.

by

$$s_{i,j}(n+1) = \left(s_{i+1,j}(n) \text{ OR } s_{i,j+1}(n)\right) \oplus s_{i-1,j}(n) \oplus s_{i,j-1}(n) \oplus s_{i,j}(n) \tag{3}$$

where '$\oplus$' denotes the exclusive-or (XOR) boolean function of two inputs. A sequence of four CNN templates can implement this rule.

Applying the mean-field theory to these rules results in the maps shown in Figure 1. It is clear that Rule 1 has an equilibrium distribution which does not exhibit the desired equiprobable output values. Rule 2, on the other hand has this ideal property. And, the mean-field theory predicts very fast convergence to the equiprobable density. These conclusions were validated by applying the statistical tests of Section 1.2 to the direct simulation of these CA. Our results show that Rule 1 is unusable as a spatial generator of random bits (although the time sequence may be usable), whereas Rule 2 has excellent statistical properties. Rule 2 also demonstrates the spontaneous generation of randomness from simple initial conditions, as shown in Figure 3.

2.2. Reversible Second-order Cellular Automata

Irreversible CA have been shown to have some nice properties, but it is very difficult to prove things about cycle length, etc. A problem with irreversibility is that it implies convergence of trajectories to a fixed point or limit cycle. That is, as time gets large, the state space which is explored is decreased. This could have some dangerous consequences.

Reversible CA on the other hand are guaranteed not to converge to a fixed point. The behavior of a reversible system is like that of a gas. The counting arguments of statistical mechanics suggest that the reversible system will generally be found in a random-looking state. And, even when starting from a highly ordered initial condition, the image quickly tends to a maximal entropy state [16]. This is analogous to putting all the air of a room in one corner at letting it go – it is very unlikely that it will return to that configuration.

In general, it is difficult to tell whether a particular CA rule is reversible. But, by using a clever technique attributed to Ed Fredkin [16], a very large class of them can be generated constructively. This kind of 2-D reversible CA has the following form:

$$s_{i,j}(n+1) = f\left[s_{\mathcal{N}_{i,j}}(n)\right] \oplus s_{i,j}(n-1) \tag{4}$$

where "$f(\cdot)$" is an arbitrary (potentially irreversible) boolean function. Since XOR is reversible, from Equation 4 we have the following relation:

$$s_{i,j}(n-1) = f\left[s_{\mathcal{N}_{i,j}}(n)\right] \oplus s_{i,j}(n+1) \tag{5}$$

which means that if we know state $s_{i,j}(n+1)$ at time $n+1$ and the states $s_{\mathcal{N}_{i,j}}(n)$ at time n then we can find the state $s_{i,j}(n-1)$ at time $n-1$. And so, this kind of CA is reversible.

The tendency of the array towards maximal entropy can be understood in terms of the XOR function. Namely, let $0 < p, q < 1$ be the average density (the probability a cell is in the '1' state) of two arrays which were produced

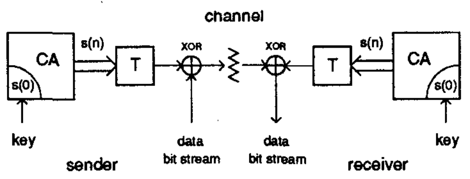

Figure 2: A typical arrangement for a stream cipher using a random number generator. Here the CNNUM would be used to implement a CA based RNG. The transformation T is a non-invertible mapping used to isolate that state during a plain-text attack.

by independent Bernoulli random variables (unfair coin toss). Applying the XOR operation between elements in the same position of the two arrays produces a third array which can easily be shown to have average density $r = p + q - 2pq$. Now

$$|r - \frac{1}{2}| = |-2(p - \frac{1}{2})(q - \frac{1}{2})| = 2|p - \frac{1}{2}||q - \frac{1}{2}| \leq |p - \frac{1}{2}| \tag{6}$$

which demonstrates that the resulting array has average density closer to $\frac{1}{2}$ than either p or q. It can therefore be expected that the second order CA will tend to an average density of $\frac{1}{2}$ as long as as $f(\cdot)$ provides enough 'mixing' to justify the approximation of statistical independence. We are attempting to apply the thinking of mean-field theory to the second order cellular automata to further support this argument.

We have analyzed several boolean functions f. Since the exact properties of the mixing function is not important, we are trying some very simple ones including boolean functions that can be implemented by a single template (threshold logic). Our statistical analysis indicates that not only are the time sequences sufficiently random, the spatial sequences are as well indicating the these CA may be useful for generating random images at high speed. For even better statistical qualities, we are exploring the possibility of using several iterations of the random irreversible CA rules for the mixing function. This can be considered to be a rule which uses a larger neighborhood radius.

3. Cryptographic Applications

There are many potential uses for a CNN based pseudo-random number generator, for instance, random location selection, annealing algorithms, image dithering and quantization, stochastic resonance, Gaussian noise generation and spread spectrum communications. Random numbers also play an important role in many types of cryptographic schemes[17], an application we are now considering.

3.1. Key Stream Ciphers

Perhaps the simplest use of pseudo-random bit generation for cryptographic purposes is in a key stream cipher. In this scheme, a stream of 'random' bits from a *key stream generator* is used to encode the plain-text data stream bit-by-bit by applying an XOR operation (which is invertible) between them. At the receiver, the same sequence of bits produced by an identical key-stream generator is then used to decode the plain-text. Security is provided by the use of a key, which is a shared secret, to somehow affect the bit-stream produced.

Typically, the key stream generator consists of an internal state, a function for iteratively updating the state, and an output function which is applied to the state. For serious applications the pseudo-random bit-stream generated must be cryptographically secure against a plain-text attack. In this scenario, a interloper is assumed to have access to an arbitrary amount of plain-text and corresponding cipher-text. He can easily recover the bit-stream for this part of the sequence. So, we require that even by knowing an arbitrarily long output sequence it must be infeasible to determine the following output sequence.

The key can affect any of the components of the key stream generator. This is done in a way as to insure that the each key gives a unique sequence. The length of the key (given in bits) therefore determines the number of possible output sequences. Ideally the strength of the encryption rests entirely on the key. That is, the best possible attack is to try every possible key and see if the known plain-text encrypts to the given cipher-text.

There are many ways of using two-dimensional CA as a key stream generator for stream ciphers. The basic configuration would be to use the initial state of a CA array as the key, and the update function would be the rule of a non-invertible random number generating CA. On a 32×32 array this would allow a massive 1024 bits of key. Less key bits could be used however by placing them in zero-padded box in the array. The output function would subsample the state in space and time. See Figure 2.

The strength of such a method relies on the difficulty in calculating possible previous states based on the partial information revealed by the output function. This could be even more difficult by more complicated output functions such as moving the sampled location around according to another key or some aspect of the current state or even the cipher text, or using another CA rule as an output function. Invertible CA rules may also be useful, but more care will need to be taken to isolate the state from the output stream.

3.2. Block Ciphers

A block cipher transforms blocks of the plain-text during encryption. This type of approach has many advantages including the facts that the sender can easily mix random data into the plain-text stream so that sending the same message will result in a different cipher-text and the key or key stream is not easily found through a plain-text attack.

In our proposed scheme, under investigation, we use the reversible CA as the basis for the algorithm. The initial states $s(-1)$ and $s(0)$ are set to be an arbitrary random image and the data image respectively. Then, the CA is run for a previously defined number of steps, N, and $s(N-1)$ and $s(N)$ are transmitted. The trick is to introduce some secret (shared) information into the CA dynamics. This could be many different things including switching between CA rules according to a secret ordering, flipping state bits in certain secret places and times, or specifying boundary value information. The receiver can then run the CA system backwards to recover the initial data.

4. References

[1] T. Roska and L. O. Chua, "The CNN Universal Machine: An analogic array computer," *IEEE Transactions on Circuits and Systems – II*, vol. 40, pp. 163–173, Mar. 1993.

[2] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems – I*, vol. 40, pp. 147–156, Mar. 1993.

[3] T. Ritter, "The efficient generation of cryptographic confusion sequences," *Cryptologia*, vol. XV, pp. 81–139, Apr. 1991.

[4] T. Toffoli, *Cellular Automata Machines*. Cambridge, MA: The MIT Press, 1987.

[5] S. Wolfram, "Random sequence generation by cellular automata," *Advances in Applied Mathematics*, vol. 7, pp. 123–169, 1986.

[6] N. H. Packard and S. Wolfram, "Two-dimensional cellular automata," *Journal of Statistical Physics*, vol. 38, no. 5/6, pp. 901–946, 1985.

[7] K. R. Crounse and L. O. Chua, "The CNN Universal Machine is as Universal as a Turing machine," *IEEE Transactions on Circuits and Systems – I*, vol. 43, pp. 353–355, Apr. 1996.

[8] K. R. Crounse, E. Fung, and L. O. Chua, "Hardware implementation of Cellular Automata via the Cellular Neural Network Universal Machine," Memorandum UCB/ERL M95/111, University of California at Berkeley Electronics Research Laboratory, Nov. 1995. Being revised for publication in IEEE Trans. Circuits and Systems I.

[9] D. Z. Knuth, *The art of computer programming*, vol. 1. Canada: Addison-Wesley, 2 ed., 1981.

[10] D. Z. Knuth, *The art of computer programming*, vol. 2. Canada: Addison-Wesley, 2 ed., 1981.

[11] J. Dagpunar, *Principles of random variate generation*. New York: Oxford University Press, 1988.

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

[13] W. Meier and O. Staffelbach, "Analysis of pseudo random sequences generated by cellular automata," in *Advances in Cryptology - EUROCRYPT '91*, (Berlin), pp. 186–199, Springer-Verlag, 1991.

[14] S. Wolfram, "Universality and complexity in cellular automata," in *Cellular Automata: Los Alamos*, (Amsterdam), pp. 1–35, North Holland Physics Publishing, 1984.

[15] Z. Burda, J. Jurkiewicz, and H. Flyvbjerg, "Classification of networks of automata by dynamical mean-field theory," *Journal of Physics A*, vol. 23, pp. 3073–3081, 1990.

[16] N. Margolus, "Physics-like models of computation," *Physica*, vol. 10D, no. 1, pp. 81–95, 1984.

[17] B. Schneier, *Applied Cryptography*. New York: John Wiley & Sons, 1994.

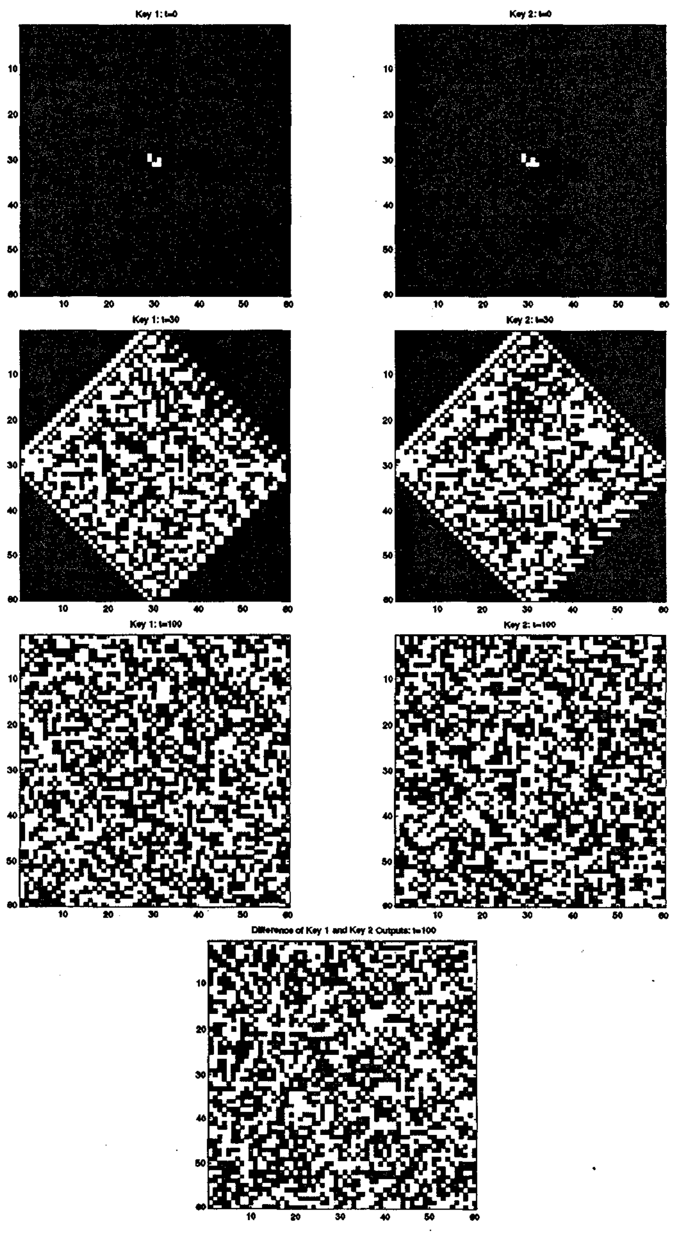

Figure 3: Two simple initial states (keys) differing by one pixel and their time evolution using Rule 2 on a 60 × 60 array with reflecting boundary conditions. At bottom: the difference between the states at time step 100. This demonstrates that the random sequences which start with similar keys quickly diverge.