"Looking for the Lost Noise"

Gheorghe Ştefan
Department of Electronics, Polytechnica University of Bucharest,
E-mail: stefan@agni.arh.pub.ro

Abstract

The noise can be very well approximated using a simple circuit and a sophisticated rule. We propose a simple, recursively defined, big circuit that, starting from own autonomy, generates pseudo-noise. To the well known method of using cellular automaton in order to generate pseudo-random sequences some new features are added in order to improve the "random" behavior. The randomness results in a chaotic process, very sensible to the **initial state** *of a simple machine working after a strange rule. The proposed structure claims a huge amount of work in order to find an appropriate initial state according to the noise "characteristics". The initial state must be selected from a space having* 2^{256} *points. Only the* **genetic algorithms** *offers us the illusion of finding the best point in this huge space.*

1 Introduction

Generating the noise is a paradoxical action. Simulating the noise can be an interesting process in which the rule is simple but must be hidden to the "noise receiver". A simple rule can act generating an *apparent complex* behavior that hide the rule. A very good example is a fractal rule generating a geometrical form without any suggestion about the simple rule that grounds the process.

The cellular automata (CA) are also a very spectacular support for simulating noise. S. Wolfram was the first who used CA as pseudo-noise generator [Wolfram '86]. He used a uniform linear CA having two states cells. The CA is initialized with one cell in the state 1 and the rest in the state 0. The transition function of each cell is the same: f_{30} (the 30-th three input logic function). Another variant, consists in a non-uniform CA, is presented in [Hortensius '89] where two types of cell are used arranged in a specific order. The next step was made by J. R. Koza by his evolutionary approach [Koza '92]. He used genetic programming to the evolution of a LISP expression representing the rule for an uniform CA. M. Sipper and M. Tomassini studied two state non-uniform CA. An evolutionary process is performed on this CA in a completely local manner. Each genetic operator is applied only between directly connected cells. They used this approach for building well performant pseudo-noise generators [Sipper '96].

Our proposal is to use, for a pseudo-random sequence generator, a uniform programmable CA consists in 256 two state cells over which a *global loop* is closed. The initial state of this system is the main responsible for the randomness. Genetic algorithms are used in order to find the appropriate initial states. This paper is a "programmatic" one because proposes only the structure, some system configurations and the investigation way. Indeed, because "The Solution" sunk in an exponentially expanded space, the path toward the "real" noise (*the lost noise*) is long, difficult and, maybe, roundabout.

2 The Structure

The proposed system is characterized by two structures: the circuit structure and the structure of the initial state. The first is a physical structure and the second is an informational structure. The first is a big sized simple circuit and the second is an algorithmic complex binary sequence must be selected from an exponentially expanded space.

2.1 The simplest two-input programmable automaton

The simplest automaton has two states. If it must be connectable in the simplest network, then it has two one bit inputs. In order to be programmable all the 256 three input combinational function must be selectable to perform the state transition function. Finally, the output and the state are the same.

Definition 1 *The simplest two input programmable automaton is defined by:*

$$SPA_2 = (X \times P, Y, Q, f, g)$$

where:

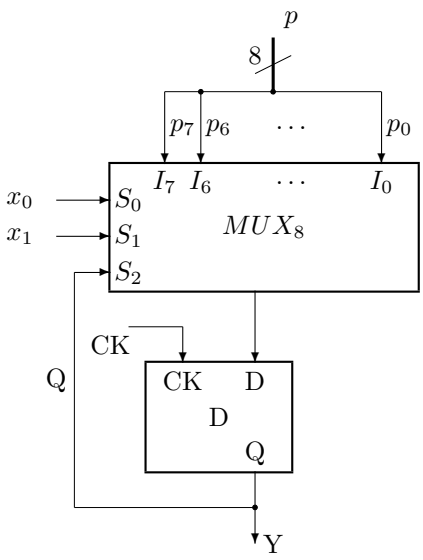

Figure 1: The simplest two-input programmable automaton.

- $X \times P$ *is the input set, where:*
  - X*, the input variable, is codified by* x_1x_0*, with* $x_1,\ x_0 \in \{0,1\}$
  - P*, the* **program***, is codified by* $p_7, p_6, \ldots, p_0$*, with* $p_i \in \{0,1\}$*, for* $i = 0, \ldots, 7$
- $Y = Q$ *is the output set coded by* $q \in \{0,1\}$
- Q *is the state set coded by* q
- $f(q, x_1, x_0, p_7, \ldots, p_0) = qx_1x_0p_7 + qx_1x_0'p_6 + \ldots + q'x_1'x_0'p_0$ *is the programmable state transition function*
- $g(q) = q$ *is the output function.*

The structure of SPA_2 *is presented in Figure 1.* $\diamond$

2.2 The simplest programmable cellular automaton

Using a linear network of the simplest programmable automata, the simplest programmable CA is obtained. The program of the CA is given by the code P that selects the transition function of each automata.

Definition 2 *The simplest programmable cellular automaton* PCA_n *consists in* n SPA_2^i *units,* $i = 0, \ldots, n$*, linearly connected, so as for each cell* i*,* $x_0 = q_{(i-1) mod\ n}$*,* $x_1 = q_{(i+1) mod\ n}$*. The state of the cellular automaton is given by the* n *bits binary code* $q_{n-1}q_{n-2}\ldots q_0$*. The program is given by the code* P *(see the previous definition).* $\diamond$

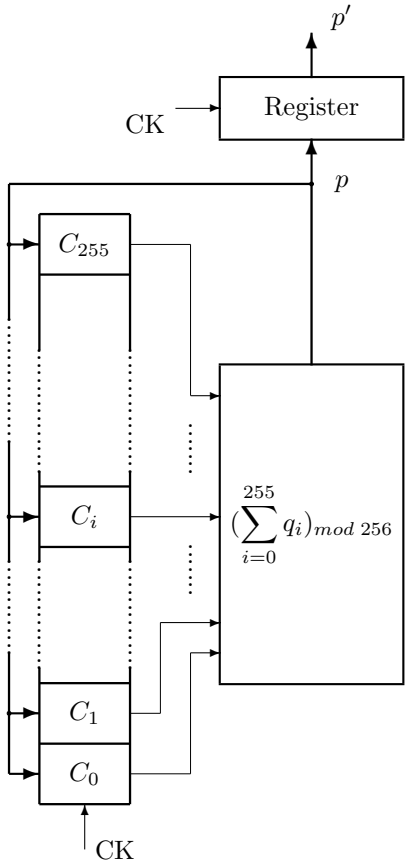

Figure 2: One of the simplest programmable cellular automaton with a global loop closed through an adder.

The evolution of this structure is controlled by: the initial state of cells and by the transition function P that can be modified in each clock cycle, if needed.

2.3 Adding a global loop over a cellular automaton

The first new feature added to a programmable CA is a *global loop*. This loop "says something" about the global state of CA improving the circuit autonomy. Our expectance is that the "positive feedback" of the added loop improves the random behavior of the system.

Definition 3 *Let be a* PCA_n *with the global state* $\mathbf{Q}$*, codified by* $q_{n-1}q_{n-2}\ldots q_0$*, and the program* P*. The* global loop *is introduced through the circuit*

that perform the function $F : \mathbf{Q} \to P$. *The resulting system is a PCA with a* **global loop** *(PCAL).* $\diamond$

The loop function makes a partition of $\mathbf{Q}$ in a very small number of equivalence classes. Thus the "positive feedback" is very weak, because the dimension of P is much smaller than the dimension of $\mathbf{Q}$. The formal experiments made by simulations will confirm or not if this loop is useful in order to improve the random behavior.

Depending on the particular form of the function F there are many actual PCALs.

Example 1 *Let be* PCA_{256} *with the global state* $q_{255}q_{254}\dots q_0$ *and the program* P *given as an eight bit number* p. *The global loop is introduced using a 256 one bit numbers adder so as*

$$p = (\sum_{i=0}^{255} q_i)_{mod\ 256}.$$

The resulting system is presented in Figure 2. $\diamond$

Example 2 *Let be* PCA_{256} *with the global state* $\mathbf{Q}$ *given as the 256 bits number* $\mathbf{q}$, *codified by* $q_{255}q_{254}\dots q_0$, *and the program* P *given as the eight bit number* p. *The global loop is introduced by the function:*

$$p = \begin{cases} log_2\ \mathbf{q} & for\ \mathbf{q} \neq 0 \\ 0 & for\ \mathbf{q} = 0 \end{cases}$$

$\diamond$

2.4 Adding the input

The previous defined class of circuits have no input, evolving on the output as pure generators. In order to add an input the structure is modified so as the value from P is related with the input value from a set M.

Definition 4 *The* cellular noise generator, $CNG_{log_2\ n}$, *is defined starting from a* PCA_n *with the global state* $\mathbf{Q}$ *and the program* P, *over which a* global loop *is introduced through the circuit that perform the function* $F : \mathbf{Q} \times M \to P$, *where* M *is the input set.* $\diamond$

The program executed by each cell depends by the global state of CNG and by the input value. A new source of unpredictable behavior can be introduced in system on this way.

Example 3 *Let be the system from Exemple 1. If* M *consists in eighth bits numbers* m *then a variant of PCA with input is presented in Figure 3 where an eight bit* mod 256 *adder is added on the system presented in Figure 2.* $\diamond$

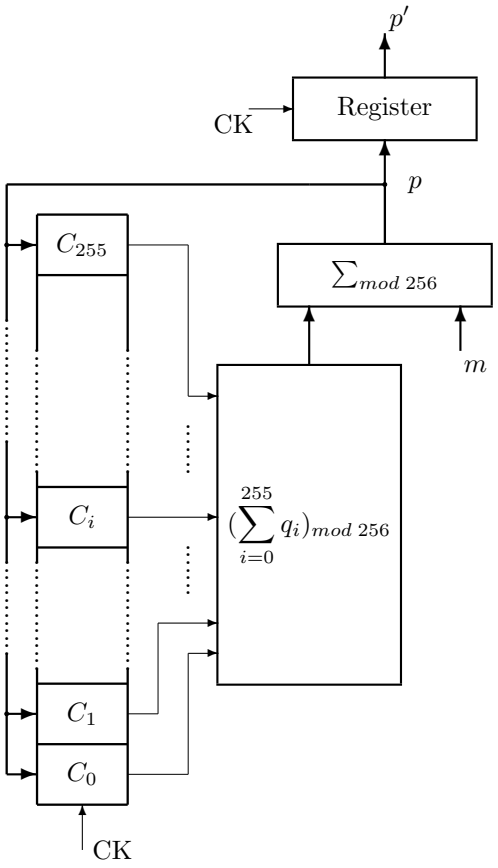

Figure 3: The cellular noise generator with 8 bits input CNG_8.

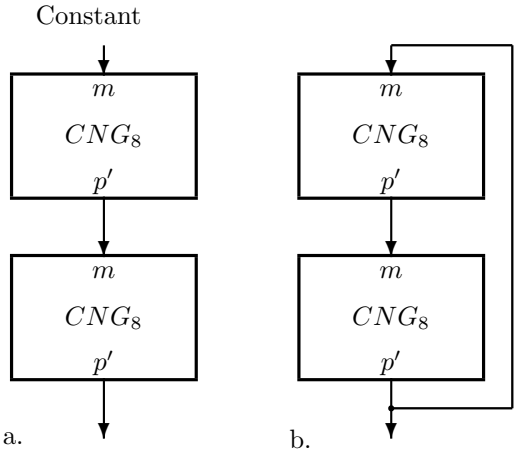

Figure 4: Interconnecting CNG_8 units. a. Serial connection with *Constant* on the input. b. Loop connecting two units.

2.5 Interconnecting the simplest big autonomous circuits

The main utility of the input is due to the possibility to interconnect two or more CNGs in order to improve the "noise quality". In figure 4 are suggested two possible interconnections. The first is a serial connection. The second CNG_8 "amplifies" the noise generated by the first CNG_8. The second connection is a loop closed over two serial connected CNG_8. The loop acts as a "positive feedback" generating conditions for long cycles. (An interesting variant is a loop closed over a single CNG_8.)

The role of the output *Register* is now obvious: allows to close properly the loops or ensures the pipeline path on the serial connection.

If more complex interconnections are needed supplementary modulo 256 adder can be used. Thus two input CNGs can be built.

A very interesting question arises: two "composed" pseudo-noises have the chance to manifest a tendency toward order? If have, then the physical structure of the generator must remains simple and the key is only the initial state of CA, else we have at disposal a strange deterministic method to improve the noise quality.

3 The Noise Generator

The behavior of a CNG hide the simplicity of the own structure. Because we have no formal methods to describe simply the output behavior the system seems to behave chaotically. In fact we are faced only with an apparent complexity.

There are formal methods to evaluate the quality of the resulting noise. Some of few initial CA's states provide a very good random behavior to the CNG's output. The noise can be received in many forms from the CHG's output. The entire number p' or only one bit of the CA can be received at each clock cycle, thus generating a random sequence. There are standard testing procedure, described in [Knuth '81], for estimating the random characteristics of the generated sequences.

4 The Maine Problem: the Initial State

Because the transfer function on the loop is very simple, the random behavior depends only by the initial state of the CA. In the design process the main step is to find a "noisy" initial state.

The maximal expected length of a cycle in a CNG_8 has $2^{2^8}-1$ states. It is too much for current applications. We can be content in most of the applications with shortest sequences. For example, a sequence having "only" 2^{64} elements generated with the frequency of 1MHz has the length of many thousand century. This means that we are not interested in finding the optimal solutions. It is enough to find a locally optimal solution.

The only way to look for an appropriate initial state in this huge space is to use *genetic algorithms*. The chromosomes consist in a set of random chosen initial state of CA, i.e., a set of 256 bits numbers. The fitness function results applying standard testes for randomness. The searching process efficiency is due to the ability to imagine efficient cross-over rules.

5 Conclusions

This paper is only a programmatic one, maybe a challenging one, because starts an unending research process in order to find the best noise generator, if exists.

We have no formal methods to describe the actual evolution of the proposed machine. The only way to present the behavior of this machine is to put them to work using a simulator. The formal method to be used is the experimental mathematics.

References

[Hortensius '89] P. D. Hortensius, R. D. McLeod, H. C. Card: "Parallel Random Number Generator for VLSI Systems Using Cellular Automata", *IEEE Trans. on Comp.*, 38(10): 1466-1473, October 1989.

[Koza '92] J. R. Koza: *Genetic Programming*, The MIT Press, 1992.

[Knuth '81] D. E. Knuth: *The Art of Computer Programming: Volume 2, Seminumerical algorithms.* Addison-Wesley, 1981.

[Sipper '96] M. Sipper, M. Tomassini: "Coevolving Parallel Random Number Generator" in H. M. Voigt, W. Embeling, I. Rechenberg, H. P. Schwefel (editors): *Parallel Problem Solving from Nature IV (PPSN IV)*, volume 1141 of *Lecture Notes in Computer Science*, Heidelberg, 1996. Springer-Verlag.

[Wolfram '86] S. Wolfram: "Random Sequence Generation by Cellular Automata", *Advances in Applied Mathematics*, 7:123 169, June 1996.