"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2012
Brasov, 24-26 May 2012

USE OF RECONFIGURABLE CELLULAR AUTOMATA, IMPLEMENTED WITH FPGAs, IN CRYPTOGRAPHIC APPLICATIONS

Constantin GROZEA*, Gheorghe GUIMAN*, Daniel-Tiberius HRIȚCU*, Ionuț RĂDOI*, Florent-Mircea ROMAN*

*Military Equipment and Technologies Research Agency, Bucharest, Romania

Abstract: *Cellular automata (CA) are dynamic systems, successfully used in mathematics, biology, chemistry or physics. The random generated by these systems has also been used in computer science (e.g. games industry) as well as in cryptographic applications (e.g. random number generators). As in the case of more well-known LFSR (Linear Feedback Shift Register), the usefulness of these systems in cryptographic applications depends on how it is accomplished, as well as the reconfiguration of the sequence used (seed). In this paper presents the principle of functioning of cellular automata, their usefulness in cryptographic applications, as well as a reconfiguration mechanism is implemented using FPGA (Field-Programmable Gate Array).*

Keywords: *cellular-automata, LFSR, FPGA*

1. INTRODUCTION

Providing encryption data flows through the channels of communications exchanged has become an essential requirement in the objectives realization context on which both large companies and institutions as well as civil and military proposed.

Currently, cryptography is no longer a specific branch of the military scope, which is why more and more companies are investing in civil and financial resources for the acquisition of significant human or implementation of cryptographic solutions to ensure the desired privacy.

Although cryptographic algorithms are based on different deployment scheme *(DES, 3-DES, AES, etc.)*, all make use of a shared-key encryption.

Thanks to the random that you generate, cellular automata have been used in order to obtain some strings of random numbers, [1] either individually or in combination with other mathematical algorithms, such as *LFSR* [2].

The notion of "cellular automata" appeared for the first time in his work titled: *„John von Neumann Theory of Self-Reproducing Automata"*, which represent a model defined a biological system self-regenerating.

Until now, cellular automata have been used successfully in various fields such as physics, mathematics, biology or chemistry. As can be seen in *Figure 1*, cellular automata is formed from a number of cells (for practical implementations consider a finite number is)

interconnected with neighboring cells after a mathematical rule.

Depending on the existing connections between neighboring cells, cellular automata can be one-dimensional or multidimensional [3].

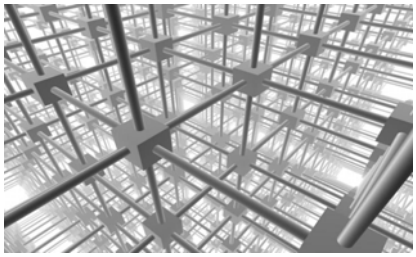

Figure 1 The structure of multidimensional cellular automata

Every cell of an automatic cell is characterized by *3 (three)* parameters:

- The current status *(e.g.: „1L" or „0L")*;
- Status of the neighboring cells;
- Math rule after this will change the current status.

How mode to use an automatic rule specific mathematical cell is described in detail in [4].

In the next section will show how to deploy a cellular automata are implemented in *FPGA* technology, as well as a changing automatic mechanism of mathematical rule.

II. IMPLEMENTATION

In *Figure 2* is presented block diagram of the proposed architecture to examine the possibility of implementing cellular automata reconfigurable. Block control meets second functions. The first function is to count the number of random bits and generate comparison with maximum length allowed for a random sequence. The second function consisted in testing the online statistics of random sequences generated. In the experiments, we used the „Monobit test".

For the purposes of a more stringent statistical tests can add a series of tests, such as tests specified in *FIPS 140-2 (Poker, Longest Run, Runs)*. When the control block passed the *CE* output signal in the *„1L"*, on a length of *32* impulse clocks takes place after the change of the rule that runs the mathematical cellular automata, cells at the level of the module to change the rule.

The new rule will be passed on to mathematics through the serial signal *CDI* by random number generator based on cellular automata, *AC*.

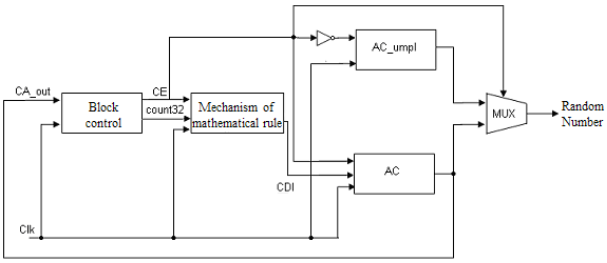

Figure 2 Random Number Generator based on reconfigurable cellular automata

The operation mode is similar to random number generators based on *LFSR*. Assuming that the *LFSR* is composed of *N* cells, maximum length of the random sequence on which he can generate is 2^N. Thus, once the 2^N numbers generated, *LFSR* must be reconfigured.

Similarly, in this article, it is proposed to change the mathematical rule after either generated a random number sequence of maximal length, either after a random test statistic sequences detects a determinist sequence.

For generating random numbers were used two generators (*AC* and *AC_umpl*), the first of which was active during the period of time during which the rule change takes place

by operation of cellular automata in the composition of the second.

The objective of this work is to implement a strong random number generator, it resorted to the method previously mentioned in order to avoid the appearance at the exit of architecture presented in *Figure 2* has a long string of bits in the State due to its connections to *„0L"* during the period of time during which the rule change takes place for *AC* generator functioning.

A more practical alternative might be to use a memory *FIFO (First In First Out)* to withhold only valid outs of random number generator.

In this way you can give up using random number generator *AC_umpl*.

For the implementation of the reconfigurable cellular automata components were used for the basis of programmable logical areas supplied by *Xilinx*, known as *LUT (Look Up Table)*, in the reconfigurable version.

The principle of operation of these components is described in [5].

In *Figure 3* is a block diagram of *LUT* for two-dimensional cellular automata adapted with four neighbors for each cell.

The inputs I_0, I_1, I_2 and I_3 are connected with their four neighbors of a cell.

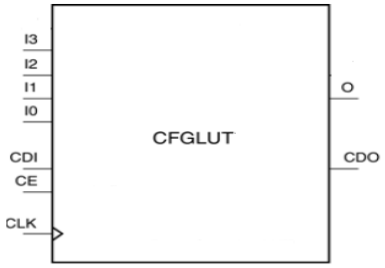

Figure 3 The cellular automata basic cell

The output of a cell depends on the *CA* table of truth, operated on the basis of the four entrances and mathematical rule to change cell condition, as can be seen in *Figure 4*, to rule *CA06990*:

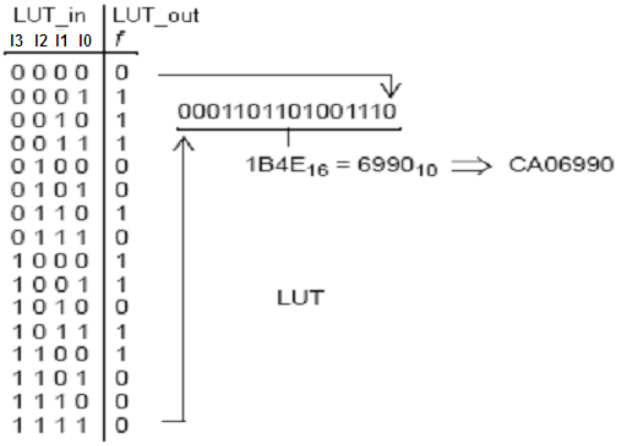

Figure 4 Table of truth

Determination of the neighbors of a cell is made according to the method described in [4] for two-dimensional configuration *{2n2w, c, ne, s}*, illustrated in Figure 5.

For example, starting from the second cell number *7* positions in the direction of North and West in the direction of two positions *(2n2w)*, you will reach the cell number *53*, representing the first neighbor cell number *7*.

The mechanism is similar to the cell *7* inputs *2* and *3 (ne and s)*.

In contrast, entry *1 (c)* is connected to its output, so that one of the neighbors of each cell is considered even the cell oneself.

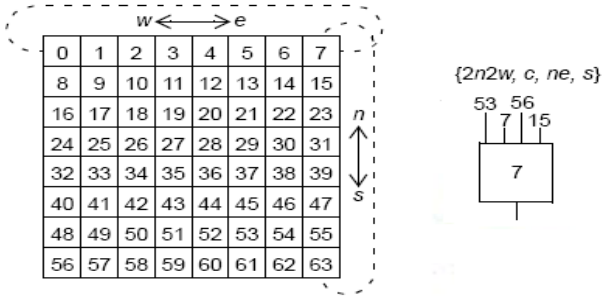

Figure 5 Two-dimensional configuration {2n2w, c, ne, s}

In the next section are presented the results of the implementation of the mechanism for reconfiguration of the mathematical rule change whit cells.

III. RESULTS

The main aspects considered in this paper aims at the implementation of the control block and module to change the rule, both shown in *Figure 2*.

The mechanism of functioning of the control block is illustrated in the following two figures.

In *Figure 6* is presented if it reaches a maximum length of random sequence generated.

For demonstration purposes we considered as maximum length of a random sequence of bits is *60,000*.

In reality, this maximum length sequences depends on the number of cells used.

Count_sw signal pulses counted *20,000* clocks, so when it exceeds the value *2*, the signal goes in *„1L”*, generating *60,000* bits random bits sequences, out-classing reconfiguration process.

Parallel to the length of random testing is accomplished a sequence statistics test for *20,000* bits.

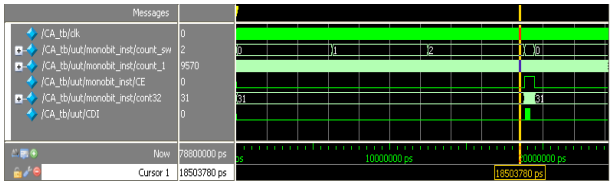

Figure 6 The changing mechanism of mathematical rule if case of maximum sequence random

It has also implemented a simplified variant of the module test statistics, relying only on the *„Monobit test”*.

This module is intended to trigger the process of changing the current mathematical rule that runs after the cellular automata, every time when it detected a sequence of *20,000* bits that cannot be considered random, statistical testing.

You can see in *Figure 7* that, although the signal is less than *count_sw 2* (sequence generated by the random number generator has a length less than *60,000* bits), the signal that has gone in *„1L”*, out-classed the changing mechanism of mathematical rule.

This is due to the amount of the registered signal after *20,000* bits generate *count_1*. The role of signal *count_1* is to count the number of 1 bits in a sequence of *20,000* bits.

The *„Monobit test”*, if the number of *1* bits is within governmental forecasts, $9725 < n_1 < 10275$, test is passed. In the situation illustrated in *Figure 7*, you notice that it was only counted a number of *5385* bits in *„1L”*, in the *20,000* bits sequence frame that was tested.

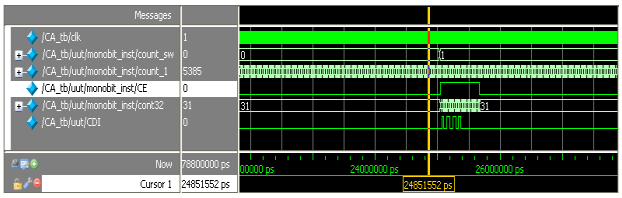

Figure 7 Changing mathematical rule imposed by statistical testing result

Figure 8 shows the gear-changing mathematical rule that runs after the automata cellular cell.

Note that when the signal becomes active for *32* cycles, counted by *cont32* (counter), takes place a change of the signal *CDI*.

In the case presented have used mathematical rules represented by *16-bit* binary functions, for which reason the signal change takes place in the early *CDI 16* pulses of tact.

You can also notice that the mathematical rule change, getting random number generator is provided, it fill-in signal representing getting the generator *AC_umpl* in *Figure 2*.

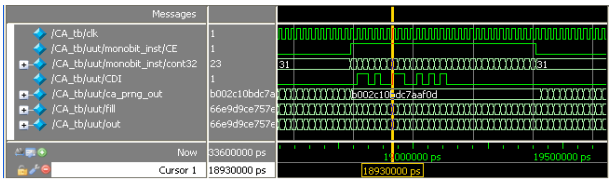

Figure 8 Changing mechanism of mathematical rule

Note also that while the process of mathematical rule change would require only *16 cycles* (in the case of a mathematical rule on *16-bit* representation), due to its structure and *CFGLUT* mode of operation, the *32* cycles are required.

This is a limitation of the mechanism proposed in the event that uses a mathematical rule on represented less than *32*-bit, but if you want to use different mathematical rules represented more than *32*-bit.

IV. CONCLUSIONS

A *RNG* (random number generator) based on components whose operation is controlled by a mathematical algorithm takes reinitializing periodicals.

This mechanism has been used in the case of random number generators based on *LFSR* modules.

In this paper we proposed a way of re-initialization that can be used to implement some random number generators that have automatic membership.

Thus, using the *CFGLUT* component areas programmable *Virtex 5* logic, the re-initialization could not accomplish in *32* cycles.

Re-initialization process is triggered automatically on the basis of tests that are performed during the selection of random number generator.

Thus, the re-initialization mechanism of described can be used to implement the random number generators that require online testing performance.

One of the limitations of the proposed method consists in the fact that the length of the cellular automata sequence re-initialization (mathematics) cannot exceed *32*-bit.

This constraint is due to the structure of the CFGLUT modules that may be initialing with fixed-length sequences *(32*-bit*)*.

REFERENCES

1. R. Santoro, S. Roy, O. Sentieys, „Search for Optimal Five-Neighbor FPGA-Based Cellular Automata Random Number Generators”, University of Rennes.

2. T. Tkacik, „A Hardware Random Number Generator”, Motorola (2009).

3. M. Woudenberg, „Using FPGAs to Speed Up Cellular Automata Computations”, Master thesis for Grid Computing, University of Amsterdam.

4. B. Shackleford, M. Tanaka, R.J. Carter, G. Snider, „FPGA Implementation of Neighborhood-of-Four Cellular Automata Random Number Generators”, HP Laboratories (2001 November 14).

5. „Virtex-5 Libraries Guide for HDL Designs”, UG621(v 11.3) 2009 September 19.4. Shihong Wang, Huaping Lü, Gang Hu, “A new self-synchronizing stream cipher” (paper presented at Beijing, China, January 11, 2005), Chapter III, Principles to construct efective SSSC, pp. 9-12.