Programmable Cellular Automata Encryption Algorithm Implemented in Reconfigurable Hardware

Petre Anghelescu

***Abstract* — This article presents an encryption system based on the PCA (Programmable Cellular Automata) theory and the implementation in reconfigurable hardware in order to achieve high speed communication for real time applications. The proposed encryption algorithm belongs to the class of symmetric key and the entire model was implemented on a reconfigurable hardware in FPGA (Field-Programmable Gate Arrays) device of type Spartan 3E XC3S500E in order to take the full advantage of the inherent parallelism of the PCA. Based on PCA state transitions certain fundamental transformations are defined which represents block ciphering functions of the proposed enciphering scheme. The experimental results prove that the proposed enciphering scheme provides high speed, good security and it is ideally for hardware implementation in FPGA devices.**

***Keywords*—Block ciphers, Cellular automata, Cryptography, Programmable cellular automata, Reconfigurable hardware.**

I. Introduction

Data security for many internet based applications is becoming more and more important with the rapid growth of the quantity of the information transmitted using network communications.

In present, promising applications for cryptographic algorithms may be classified into two categories: *processing of large amount of data at real time* (potentially in a high speed network) – examples include telephone conversations, telemetry data, video conferencing, streaming audio or encoded video transmissions and so forth – and *processing of very small amount of data at real time* (in a moderately high-speed network transmitted unpredictably) – examples include e-commerce or m-commerce transactions, bank account information, e-payments and micro-browser-based (WAP-style), HTML page browsing and so forth. In both cases, cryptography is the best solution against the unauthorized use of the information.

In the recent years, researchers have remarked the similarities between bio-inspired systems – as cellular automata (CAs), chaos and cryptography [1], [2]. Several of the CAs features can be correlated with the cryptographic properties. A relevant relationship between cellular automata and cryptography was revealed by Shannon in his fundamental early work [3]. In [3], Shannon discusses about a system composed from simple components that interact between them – with a transparent local comportment – but the global comportment of the entire system unsuspected, things that are well known in the cellular automata theory.

The essence of the theoretical and practical efforts which are done in this new field is represented by the idea that CAs cryptosystem is capable to have similar performances regarding the classic methods based on computational techniques.

Also, technologic evolution in the field of communication using reprogrammable hardware structures (FPGA and CPLD), gives appropriate solutions for the implementation of the cryptographic modules in high speed applications.

The cryptosystem presented in this paper uses four one-dimensional PCAs arranged in pipeline and a SRAM memory that store the evolution rules used by the PCAs. The entire cryptosystem is implemented in hardware on a FPGA of type Xilinx Spartan 3E XC3S500E and the plaintext/ciphertext is received/transmitted using User Datagram Protocol (UDP) connection.

The paper is organized as follows. The following section presents basic theoretical foundations of the proposed work. We describe some basics of CA, PCA and reconfigurable hardware. Section III shows how the PCA theory was used in order to construct a block encryption technique. In this section it is presented the structure of the entire PCA based encryption system. Section IV contains experimental results and the analysis of results. In this section the proposed encryption method was tested and verified on a FPGA board and using UDP connection protocol. Conclusions and future research directions will end the paper.

II. Concept and Theory of CA, PCA and Reconfigurable Hardware

A. Cellular Automata (CA)

CAs, introduced by J. v. Neumann [4] and further popularized by S. Wolfram [5], are computational models that can perform complex computation with only local information. The simple structure of CA has attracted researchers from different fields of interests and has undergone rigorous theoretical and experimental analysis.

CA represents a particular class of dynamical systems that

Manuscript received October 25, 2012, revised March 08, 2013. This work was supported by CNCSIS UEFISCSU, project number PN II-RU PD 369/2010, contract number 10/02.08.2010.

Petre Anghelescu – University of Pitesti, Department of Electronics, Communications and Computers. Str. Targu din Vale, No. 1, 110040, Pitesti, Arges, Romania. Corresponding author phone: +4 0724193051 and e-mail: petre.anghelescu@upit.ro.

doi: 10.11601/ijates.v2i2.12

enable to describe the evolution of complex systems with simple rules, without using partial differential equations. A CA consists of a regular uniform n-dimensional array of cells where every cell can take values either 0 or 1. Each cell evolves in each time step (discrete steps) depending on some combinational logic on itself and its neighbors as shown in Fig. 1.

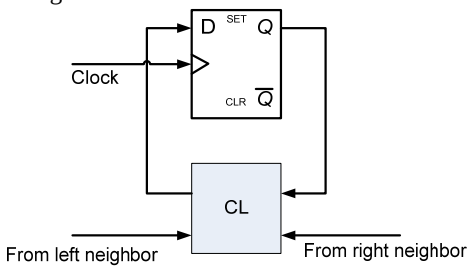

Fig. 1. The component of a cellular automata cell.

Such a CA is called three-neighborhood CA. The combinational logic is called the *rule* of the CA. The next state function for a three-neighborhood CA cell can be expressed as follows:

Say,

i – position of an individual cell in an one dimensional array,

t – time step,

$a_i(t)$ – output state of the central cell (i-th cell) at the t-th time step.

Then,

$$a_i(t+1) = f[a_i(t), a_{i+1}(t), a_{i-1}(t)] \quad (1)$$

where f denotes the local transition function known as a rule of the CA.

In the CA theory, there are two classic types of neighbourhoods: the Moore neighbourhood that comprises 3 cells for one-dimensional CA and 9 cells for two-dimensional CA (Fig. 2a); the von Neumann neighbourhood with 3 cells for one-dimensional CA and 5 cells for two-dimensional CA (Fig. 2b).

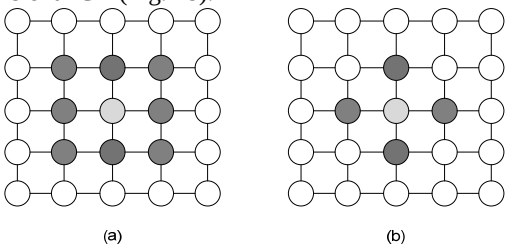

Fig. 2. Classical neighborhood (a) – Moore Neighborhood, (b) – von Neumann Neighborhood

S. Wolfram has studied the relationships between CA and different dynamical systems and suggested a classification of CA behavior in this context. According to [5] there are four classes of CA:

Class I – the CA evolution reaches a certain final state and stays there (limit points).

Class II – the CA encounters simple or cyclic structures (limit cycles).

Class III – the majority of initial states lead to arbitrary patterns (chaotic behavior of the kind associated with strange attractors).

Class IV – generates global complex structures (very long transients with no apparent analog in continuous dynamic systems).

This classification of the CA is done by means of empirical observations and simulations (space-time patterns) and mainly refers to 1-D CAs, but similar ones can be made for 2-D or 3-D cases.

In case of 1-D, three neighborhoods, two states (0 and 1) CA, the number of all possible uniform rules is 2^8. These rules are enumerated using Wolfram's naming convention [5] from rule number 0 to rule number 255 and can be represented by a 3-variable Boolean function. Among the rules, rule 51, rule 60 and rule 102 are used in this paper to design the encryption algorithm. The three rules are presented in Table I.

TABLE I
AN EXAMPLE OF CA NUMBERING RULES

Rules name	7 111	6 110	5 101	4 100	3 011	2 010	1 001	0 000
51	0	0	1	1	0	0	1	1
60	0	0	1	1	1	1	0	0
102	0	1	1	0	0	1	1	0

Each CA rule corresponds to a unique combinational logic. For example, using Veitch-Karnaugh diagram, rule 60 specifies an evolution from the neighborhood configurations to the next state as:

$$\text{Rule 60: } a_i(t+1) = a_i(t) \oplus a_{i-1}(t). \quad (2)$$

That is, the next state of the i_{th} is obtained by XORing the present states of the current cell and its left neighbor. In this case, XOR yields true if exactly one, but not both, of two conditions is true.

In a CA, different cells may have different evolution rules. If all cells have the same CA rule, then this CA is called a uniform CA; otherwise it will be called a hybrid CA. if all cells rules involve XOR or XNOR only, like rule 60, then this CA is called additive CA. If in a CA the rules only involve XOR operation, then it is called a non-complemented CA and the corresponding rules are referred to as non-complemented rules. If the rules only involve XNOR operations, then the CA is called a complemented CA. The corresponding rules are called complemented rules.

B. Programmable Cellular Automata (PCA)

The programmable cellular automata (PCA) was firstly introduced in [6] and are modified CA structures, where the combinational logic of each cell is not fixed but controlled by a number of control signals such that different functions (evolution rules) can be realized on the same structure. As the matter of fact, PCA are essentially a modified CA structure. We can say that a CA is a PCA if it employs some control signals that implement various functions dynamically in terms of different rules.

For example, using such a cell structure as in Fig. 3, all possible non-complemented additive rules can be achieved through the combinations of the control signals of C_1, C_2 and C_3 switches.

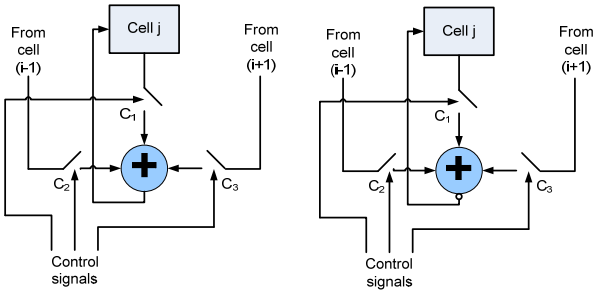

Fig. 3. An example of a cell of PCA.

In this paper one dimensional PCA defined over binary state alphabet (state 0 or 1) with neighborhood size three and dynamically combination of rules 51, 60 and 102 is used.

In conclusion, the very large phenomenology of the CA and PCA models, its apparently big complexity and parallel, regular, cascadable and local interconnections (however, this parallelism, when emulated in software or in sequential hardware, disappears) offer a good basis for applications in cryptography.

C. *Reconfigurable Hardware*

The reconfigurable devices, firstly introduced by G. Estrin in 1960, consist on a hybrid machine composed by a general purpose microprocessor interconnected with programmable logic devices [7].

The most popular reconfigurable hardware devices are FPGAs. FPGA circuits represent a compromise between circuits with microprocessor and ASIC (Application Specific Integrated Circuits) circuits [8]. *On one hand*, they present flexibility in programming, called here reconfiguration, which is a feature for microprocessors. Even if FPGA cannot be programmable while operation, they can be configured anytime is needed, having a structure based on RAM programmable machines. *On the other hand*, they allow parallel structures implementation, with response time less than a system with microprocessor.

FPGAs are programmable semiconductor devices introduced by Xilinx in the mid 1980s that are based around a matrix of configurable logic blocks connected via programmable interconnects. A number of tools are available for synthesizing logic designs such as Hardware Description Languages (HDL) Verilog, and especially, VHDL, are the two most widely spread hardware languages.

Cryptographic realizations in hardware offer high speed and bandwidth providing real-time encryption if needed [9], [10]. Besides cryptography, applications of FPGAs can be found in the domains of evolvable and biologically-inspired hardware, network processors, real-time systems, rapid ASIC prototyping, digital signal processing, interactive multimedia, machine vision, computer graphics, robotics, embedded applications, and so forth. In general, FPGAs tend to be a good choice when dealing with algorithms that can benefit from the high parallelism offered by the FPGA fine-grained architecture.

FPGAs offer advantages for reducing time to design, power consumption, flexibility, high-speed and security.

III. PCA Encryption Algorithm

The encryption method proposed in this paper is based on the PCAs that exhibit periodic behavior (each state lies in some cycle). In these cases, their evolution depends essentially of the initial state, but we can say that after a while the initial state is “forgotten”, in sense that the initial state cannot be retrievable through analyses of the current configuration.

The encryption system is composed from four one-dimensional PCA arranged in pipeline. The block diagram of the proposed PCA encryption system is presented in Fig. 4.

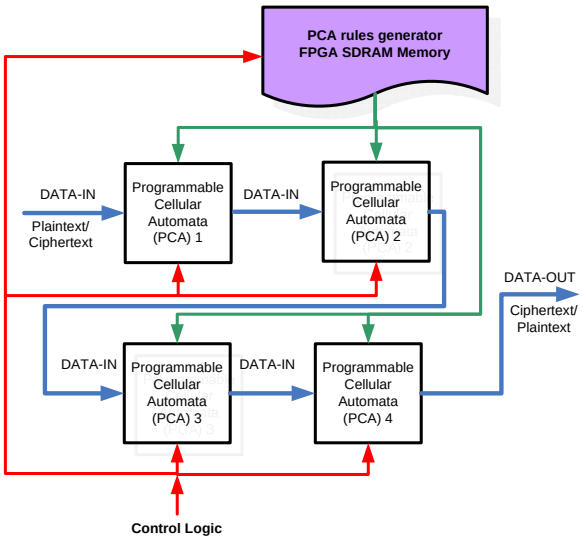

Fig. 4. Block diagram of PCA encryption system.

In the cipher scheme, one 8-bit message block is enciphered by one enciphering function. The PCAs control signals are activated with the help of the signals that are stored in the FPGA SDRAM memory rules. For the sake of simplicity, the enciphering function has four fundamental transformations FTs (PCA = 4) to operate on 8-bit data. It is obvious that for high security applications, more fundamental transformations are to be used.

The block cipher (decipher) procedure can be defined as follows:

1. Load the PCA_1 with one byte plaintext (ciphertext) from I/O. The initial block of the message is the initial state of the PCA_1. The global configuration of the PCA_4 represents the encrypted message.

2. Load a rule configuration control word from memory rules file into the PCA_1 … PCA_4.

3. Run the PCA (1, 2, 3 and 4) for 1 … 7 cycles (in the next paragraph I will explain why must have 1…7 cycles).

4. Repeat steps 2 and 3 for four times.

5. Send one byte ciphertext (plaintext) to I/O (from the PCA_4). If not end of the plaintext (ciphertext) go to step 1. Otherwise, stop the process.

In the block cipher algorithm four 8-cell PCAs are cascaded to form a pipeline CA. With the pipeline, four CA fundamental transformations (FTs) can be performed simultaneously. That means one enciphering function can be done in a single pipeline.

The PCA use for evolution a combination of rules 51, 60

and 102. The rules specify the evolution of the CA from neighborhood configuration to the next state and these are presented, as numerical values, in Table I.

The corresponding combinational logic of rules 51, 60, 102 for PCA can be expressed as follows:

$$\text{Rule 51: } a_i(t+1) = \overline{a_i(t)}. \quad (3)$$

$$\text{Rule 60: } a_i(t+1) = a_i(t) \oplus a_{i-1}(t). \quad (4)$$

$$\text{Rule 102: } a_i(t+1) = a_i(t) \oplus a_{i+1}(t). \quad (5)$$

The PCA configured with the rules 51, 60 and 102 has a state-transition diagram that consists of *equal circles of even length*. As an example, 8-cell PCA with rule configuration <51, 51, 60, 60, 60, 60, 51 and 51> generates cycles as depicted in Fig. 5.

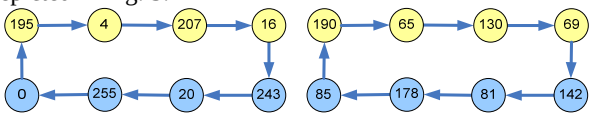

Fig. 5. The state transitions diagram of a non-maximum-length PCA.

Any PCA transformation takes two input parameters. The first one is the seed of the PCA and the second one is the number of clock cycles that needs to be run. We have found that for a PCA with a combination of rules 51, 60 and 102 the initial seed of the PCA reappear after an even number of evolution cycles (see Fig. 5).

In Fig. 5, the PCA has two equal length cycles and each cycle has a cycle length 8. Considering this PCA as an enciphering function and defining a plaintext as its original state it goes to its intermediate state after four cycles which is enciphering process. After running another four cycles, the intermediate state returns back to its original state which deciphers ciphertext into plaintext ensuring deciphering process.

Table II shows the number of 8-cell CA configurations, each generates cycles of length 2, 4, 8 or 16.

TABLE II
CA HAVING EVEN CYCLES LENGTH

Rules applied to cells	8-cell CA having 2 length cycles	8-cell CA having 4 length cycles	8-cell CA having 8 length cycles	8-cell CA having 16 length cycles
51, 60 (or 102)	7	327	**156**	2

In this encryption algorithm are used only the configurations of the rules that generates cycles of length 8. So the system designer is free to take any number in the 156 combinations (see Table II) to enhance the security of the system. The rules with 8-cycle length are presented in detail in my previous papers [11] and [12].

The proposed PCA encryption method has many differences in main concepts in comparing with previous proposed methods [11], [12]. One of the main differences is the nature of method. In the proposed encryption algorithm, we have four PCA's arranged in pipeline in order to achieve good security, but in [11] and [12] we have only one PCA. Also, the communication interface was serial RS232 and here we use TCP/IP connection (UDP protocol) in order to achieve high speed and encrypt/decrypt data sent over the Internet.

Because of the fact that the PCA does not generate sequences of maximum-length for all the possible combinations (512) of the rules we must apply from the FPGA RAM memory only the combinations (156) that generate cycles of length 8.

As is presented in my previous paper [13] and according with the CA theory, a single basic PCA cell was designed (as is depicted in Fig. 6).

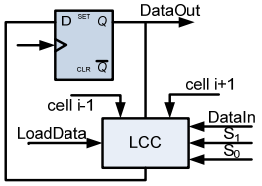

Fig. 6. The structure of the PCA cell.

The cell consists of a D flip-flop and a logic combinational circuit (LCC). The LCC includes multiplexers and XNOR logic gates to implement the rules of CA and to control the loading of data and operation of the CA. When the load control signal (LoadData) is "logic 1", data is loaded into D flip-flop. When LoadData is "logic 0", data is run into the cell according to the rules applied to the rule control signals (S1, S0) and the states of neighborhoods. After an established number of cycles (1 to 7), the data on the Q output of the flip-flop is sent out and new data is loaded in.

In this research are connected together eight cells in order to build an 8-cell PCA as is presented in Fig. 7.

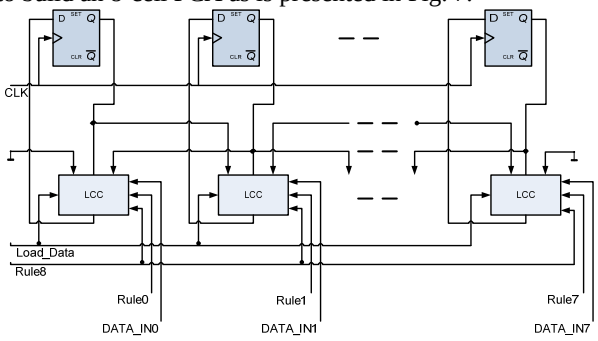

Fig. 7. PCA structure.

We note that the PCAs evolution rules must be downloaded into the FPGA RAM memory before start the encryption/decryption process. When the encryption process begins, rules are read out in sequence and applied to the four pipelined PCAs.

IV. TESTING, RESULTS AND SECURITY ANALYSIS

The general structure of the system is presented in Fig. 8.

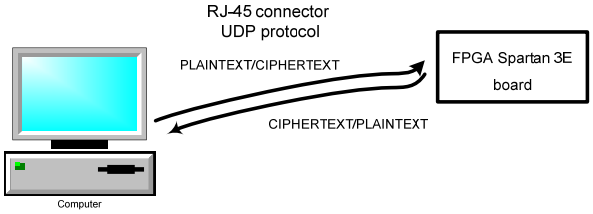

Fig. 8. General system architecture.

The hardware project implements the four pipelined PCA, the memory for storing the evolution rules and the UDP protocol (Fig. 9).

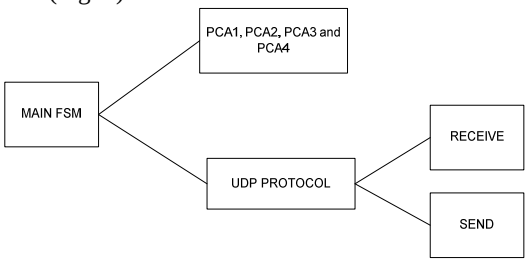

Fig. 9. Hardware design.

The PCA encryption system was implemented in hardware in a Spartan 3E XC3S500E FPGA board from Xilinx [14] (Fig. 10).

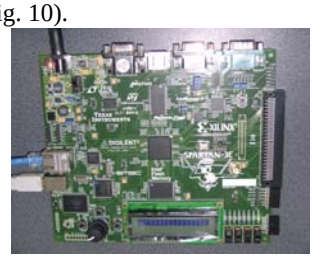

Fig. 10. Spartan 3E XC3S500E FPGA.

In hardware, the PCA cryptosystem was developed using VHDL, which is a standard language for hardware description. Using VHDL we tested the application modules in order to verify that the results obtained through software programming (using C# language) agree with hardware simulation. Because a lot of simulation and research has been carried out using 8-bit PCAs in this research, an 8-bit four PCAs was chosen for our design.

The FPGA board is interfaced with a host computer using RJ-45 connector and using UDP protocol (sees Fig. 11).

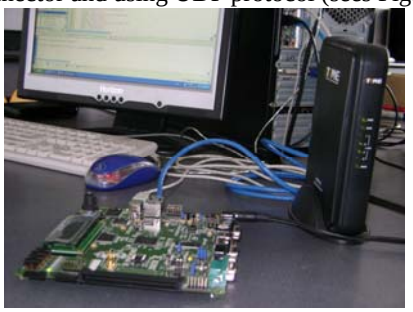

Fig. 11. The application of the encryption system.

The UDP allows high speed data transfer from the PC to the cryptosystem. The message split into 1KB packages is sent to the FPGA board using the UDP client – server connection (Fig. 12)

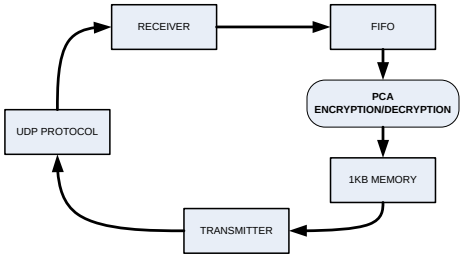

Fig. 12. UDP protocol.

As the bytes reach destination they are immediately encrypted using the correspondent bytes of the PCA's state and then saved into the 1KB RAM memory of the board. In the FPGA, the message received is treated character by character as we explained above and the encryption/decryption dates are sent by the FPGA to the PC to be displayed and stored. In hardware, the encryption rules are downloaded to the RAM before encryption. When the encryption process begins, rules are read out in sequence and sent to the PCA. The process of read of the RAM rules does not introduce delays in the process of encryption because are read in parallel with the encryption of a block of message.

An illustrative example for the encryption-decryption process applied to a short text file is presented in Fig. 13.

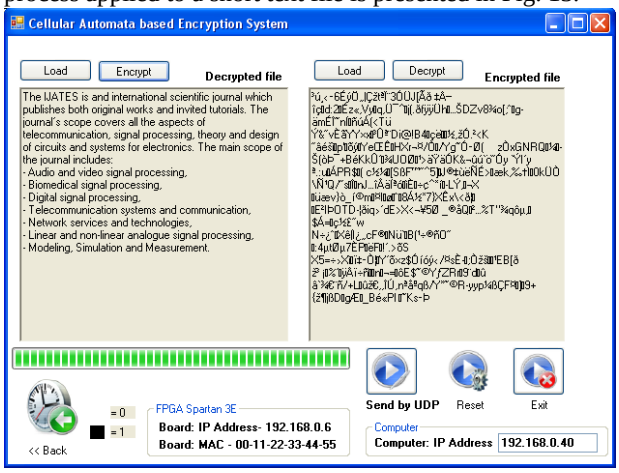

Fig. 13. Spartan 3E XC3S500E FPGA.

It is relevant to note that the distribution of the encrypted text is uniform in all ASCII intervals and not only in zone of alphanumeric intervals (as is depicted in Fig. 14 and Fig. 15).

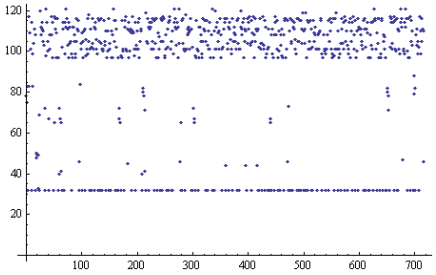

Fig. 14. Plaintext distribution.

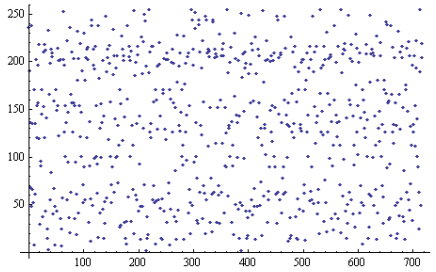

Fig. 15. Ciphertext distribution.

On x-axis we have the number of characters that compose the message (plaintext in Fig. 14 and ciphertext in Fig. 15), and on y-axis we represent the distribution of the

plaintext/ciphertext.

The PCA encrypted sequences was tested using a set of 16 statistical tests conceived by the National Institute of Standards and technology (NIST) [15]. The NIST test generates probabilistic results with respect to some characteristics that describe the pseudo-random number generators. The encrypted sequences pass the NIST tests and the system is accepted as possible random.

The timing analyzer was used to determine the maximum operating frequency (approximately 5Mbps at 50MHz FPGA – XC3S500E). To improve this value further application can use larger RAM memories in order to store more encrypted UDP packages into the FPGA before starting back to PC transmission phase.

V. Conclusions and Future Research Directions

The paper presents a symmetric key block encryption algorithm based on PCA theory. The main contribution is the design, the implementation and the analysis of the pipelined PCA encryption algorithm in reconfigurable hardware using UDP communication protocol.

As PCA achieves high parallelism and only local interconnections we simplify the implementation and with low cost. Also, the encryption and decryption devices share the same module, and could be implemented efficiently in hardware due to simple structure of PCA.

A prototypal hardware realization of this module was realized and described, and the modules presented are programmed by means of a VHDL language.

Future works include larger storage memories (for higher speed), more flexible parameters for system initialization and the implementation in FPGA of both UDP and TCP/IP protocol (for increased transmission safety).

References

[1] A. Fuster-Sabater, P. Cabalerro-Gil, "Chaotic Cellular Automata with Cryptographic Application", 9th International Conference on Cellular Automata for Research and Industry, Springer-Verlag Berlin Heidelberg, LNCS 6350, pp. 251–260, 2010.

[2] C. S. Rao, S. R. Attada, M. J. Rao, K. N. Rao, "Implementation of object oriented encryption system using layered cellular automata", *International Journal of Engineering Science and Technology (IJEST)*, ISSN : 0975-5462, Vol. 3, No. 7, July 2011, Available: http://www.ijest.info/docs/IJEST11-03-07-163.pdf.

[3] C. Shannon, "Communication Theory of Secrecy Systems", Bell Sys. Tech. J. 28, pag. 656–715, 1949, Available: netlab.cs.ucla.edu/wiki/files/shannon1949.pdf.

[4] J. von Neumann, *Theory of self -reproducing automata*, edited and completed by Burks, A.W. (Ed.), Univ. of Illinois Press, London, 1966.

[5] S. Wolfram, *A new kind of science*, Wolfram Media Inc., ISBN: 1-57955-008-8, 2002.

[6] S. Nandi, B. K. Kar, P. P. Chaudhuri, "Theory and applications of cellular automata in cryptography", *IEEE Transactions on Computers*, 43(12), 1994, pp. 1346-1356.

[7] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of applied cryptography*, CRC Press, ISBN: 0-8493-8523-7, 1996.

[8] T. Fogarty, J. Miller, and P. Thompson, "Evolving digital logic circuits on Xilinx 6000 family FPGAs," in Soft Computing in Engineering Design and Manufacturing, P.Chawdhry, R. Roy, and R. Pant (eds.), Springer: Berlin, pp. 299–305, 1998.

[9] E. Jamro, P. Russek, A. Dabrowska-Boruch, M. Wielgosz, "The implementation of the customized, parallel architecture for a fast word-match program", *International Journal of Computer Systems Science and Engineering*, Volume 26, Issue 4, pp. 285-292, 2011.

[10] F. Rodriguez-Henriquez, N. A. Saqib, A. Diaz-Perez, C.K. Koc. *Cryptographic algorithms on reconfigurable hardware*, Springer – Verlag ,ISBN 978-0-387-33883-5, 2007.

[11] P. Anghelescu, S. Ionita, E. Sofron, "Encryption technique with programmable cellular automata (ETPCA)", *Journal of Cellular Automata*, ISSN 1557-5969, Volume 5, Issue 1-2: 79-106, 2010.

[12] P. Anghelescu, S. Ionita, E. Sofron, "FPGA implementation of hybrid additive programmable cellular automata encryption algorithm", The 8th International Conference on Hybrid Intelligent Systems, HIS 2008, pp. 96-101, 2008.

[13] P. Anghelescu, "Security of Telemedical Applications over the Internet using Programmable Cellular Automata", International Journal of Intelligent Computing Research, IJICR, Volume 3, Issue 1/2, ISSN: 2042–4655, pp. 245-251, 2012.

[14] Spartan 3E Starter kit board data sheet downloaded from http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf.

[15] A. Rukhin, J. Soto, J. Nechvatal1, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, "A Statistical Test Suite for Random and Pseu-doRandom Number Generators for Cryptographic Applications", NIST (National Institute of Standards and Technology) Special Publication 800-22, (2005&2010), http://csrc.nist.gov/rng/.