Ioana Dogaru and R. Dogaru, „FPGA implementation and evaluation of two cryptographically secure hybrid cellular automata", in Proceedings of the 2014 10th International Conference on Communications (COMM), final version of the paper is available on IEEE Xplore - index:
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6866740&queryText%3DFPGA+Implementation+and+Evaluation+of+two+Cryptographically

FPGA Implementation and Evaluation of two Cryptographically Secure Hybrid Cellular Automata

DRAFT

Dogaru Ioana[1] , Dogaru Radu[1,*]
[1]*University "Politehnica" of Bucharest, Natural Computing Laboratory, Applied Electronics and Information Engineering, Bucharest, Romania*
[*]Corresponding author (E-mail : radu_d@ieee.org)

***Abstract* — Two solutions for designing cryptographically secure pseudo-random number generators are considered from the perspective of FPGA implementation. Both are based on hybrid cellular automata (HCA) and are designed to maximize the efficiency/throughput ratio. While the first PRNG employs a simple HCA, the second solution builds a chain of HCA devices in order to improve cryptographic properties. Both solutions were implemented on FPGA-based platforms and their performances are evaluated in terms of complexity, speed and power. Pseudorandom sequences generated by the hardware implementations were extracted, saved and further analyzed using the standardized NIST statistical tests. All 188 tests were passed.**

***Keywords*-component; cryptography; Field Programmable Gate Array; random number generator; cellular automata; NIST statistical test suite**

I. INTRODUCTION

Hardware implementation for high quality pseudo-random number generators is required in many applications, such as cryptographic modules, stream ciphers, Built-In Self-Test (BIST) circuitry, optimization algorithms etc. In such terms, Filed programmable gate array (FPGA) devices are very common used for hardware implementation and development. Those circuits offer flexibility, rapid prototyping, and low power consumptions.

In order to prove and certify the cryptographic efficiency for a pseudo-random number generator, the standard battery of NIST statistical tests [1] is frequently used.

Various FPGA implementations of PRNG (pseudo-random number generators) for cryptographic applications based on chaotic maps and Cellular Automata were recently proposed, compared and evaluated. In [2] four discrete-time chaotic generators used in digital communications were discussed and the performance of their implementations using two FPGA circuits (Xilinx Virtex 6 and Altera Cyclone III) were compared in terms of estimated resource usage, maximum execution frequency of implementations and dynamic power consumption (mW) for circuits using different word lengths. For instance, the lowest complexity among the 4 solutions was reported for the Bernoulli map (with 92 LUTs for 32 bit word length of the state variable, i.e. about 2.8 LUT/bit). No information about NIST or other tests is given in [2]. In [3], a sophisticated one dimensional cellular automaton for pseudo-number generator and its FPGA hardware implementation is presented. The cells change the local rule based on the real-time clock of the system. Although their solution passes both the NIST and DIEHARD tests, the allocated resources are rather high (about 37.8 LE per cell). Note that LE (logic element) and LUTs (look-up-tables) are equivalent basic resources in FPGAs from different companies. Also one bit in the state variable is equivalent to one CA cell.

In [4], a cryptographically efficient (i.e. passing the DIEHARD and NIST tests) configuration (hybrid 37-bit Linear Feedback Shift Register - LFSR and 16-bit CA) is reported, requiring about 1.37 LUTs/cell. A similar approach is reported in [5] and [6] used in stream ciphers. Although they were reported as passing statistical test suites, they are difficult to scale-up for arbitrary numbers of cells and it is not very clear how they can ensure a large key space. In [7] a rather inefficient PRNG based on the classical CA rule 30 is reported, with a resource allocation of 234 LE for a 32-cell (bit) CA – i.e. 7.3 LE/cell.

An easy-scalable and VLSI efficient alternative to producing good PRNGs, namely the hybrid cellular automata (HCA) [8], was recently proposed by authors, detailed briefly in Section II. Two kind of nonlinear cells (standard or negated) are employed, each cell being controlled by a supplementary input assigned to a bit of a mask vector (also defining a key from a potentially huge space – its size equals the number of possible masks i.e. 2^n where n is the number of cells). Hardware implementations for hybrid cellular automata targeting the different FPGA families of circuits were presented in [9] and [10]. In [9] a convenient methodology based on algebraic normal form representations is presented and a software tool is introduced in order to easily generate the VHDL code description starting from the basic parameters of the PRNG (cellular automata rule, number of cells and the mask). In [10] a FPGA implementation of a cellular automaton defined by rule 101 and it's negate is presented. The VHDL description for

hybrid cellular automata counter is studied and compare for an arbitrary number of cells *n*.

Herein we exploit the previous experience and discuss two convenient implementation solutions for cryptographically secure pseudo-random number generators. Particularly, we investigate the specific aspects of resource allocation and perform statistical tests (NIST) directly from the binary sequences generated by the physical implementations. In [12] these solutions were implemented in software and their parameters were optimized to match cryptographic requirements. Section II briefly introduces the definition of hybrid cellular automata, used as basic building block for the cryptographically secure PRNGs. Section III focuses on the specific aspects of FPGA implementation for the two PRNG types while Section IV presents the evaluation of binary sequences generated by the hardware implementation from the perspective of the standardized NIST statistical tests. Concluding remarks are summarized in Section V.

II. HYBRID CELLULAR AUTOMATA (HCA)

The HCA model [8] is based on the following equation:

$$x_i(t+1) = m_i \oplus Cell\left(x_{i-1}(t), x_i(t), x_{i+1}(t)), ID\right) \quad (1)$$

All HCA cells are updated simultaneously according to (1). Each cell has a position index i and a binary state $x_i(t)$. The index ranges from 1 to n while extreme cells are connected (ring topology). In (1) a neighbourhood with $m=3$ cells is considered, meaning three inputs per cell. A binary mask vector $\mathbf{m} = [m_1, m_2, .., m_n]$ is used, and providing the hybrid nature of the cellular automata. The particular rule function $y = Cell(x_{i+1}, x_i, x_{i-1}, ID)$ is specified by the cell ID. There are many possible choices for ID, but here we will refer to the rule ID=101. In [9, 11] an algebraic normal form (ANF) representation is associated with each particular ID. For hardware FPGA implementation that representation provides a very simple and efficient way for description. It is also used in designs reported herein. In the case of ID=101 it follows that:

$$x_i(t+1) = 1 \oplus m_i \oplus x_{i+1}(t) \oplus x_{i-1}(t) \oplus x_i(t) x_{i-1}(t).$$

An arbitrary mask vectors is considered. As shown in [8] the mask vector can be optimized in order to maximize the period of the operating cycle as proved in [8]. In Fig. 1 a simplified symbol for the HCA architecture is represented (also designed as a symbol in the FPGA design). The two designs discussed herein are described next:

PRNG-A corresponds to the basic HCA model with a large number of bits (63 in this case, easily extended to any arbitrary number of bits). For such large number of cells masks can be arbitrary chosen still ensuring good cryptographic properties.

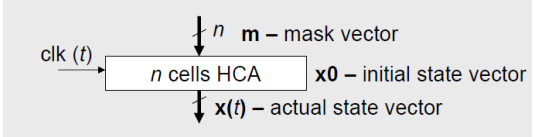

Figure 1. HCA architecture considered for the first PRNG proposed model

PRNG-B implements a chain of two HCAs [12] as shown in Fig. 2. One possible choice is to have 31 cells per each HCA, as detailed in Section III, but the design can be easily adapted to any arbitrary size.

III. FPGA IMPLEMENTATIONS AND EXPERIMETAL RESULTS

Practical implementation consists on using a Digilent Basys2 board having a Spartan 3E Xilinx FPGA target device.

Both proposed PRNG architectures (one HCA and chain HCA) were implemented in particular designs as a reuse module. The corresponding bitstream file will configure the FPGA device from the development board, using the Digilent Adept tool. The left most significant bit for the generated pseudo-random sequence is constraint to one of the board pins, in order to use the oscilloscope probe, as seen in the following section. Also, the clock will be constraint to another pin and can be visualize using the oscilloscope. Note that any other CA bit can be selected with equal results given the homogeneity of the CA cells.

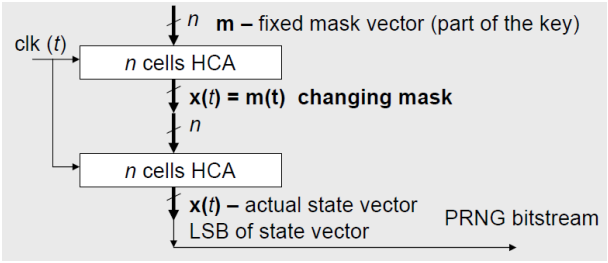

Figure 2. A chain of HCA architecture (PRNG-B model)

The software tool used in order to synthesize the VHDL source code, implement and generate the configuration bitstream file is *WebPack Xilinx ISE Project Navigator*. In order to simulate the design, the *Xilinx Simulator* tool is used. With the FPGA configured with the corresponding bitstream for each designs, real data output binary sequences were recorded and further evaluated using NIST tests. Also, using the dedicated constraints pins, the oscilloscope probe can be used in order to analyze the generated PRNG sequence and clock.

A. FPGA implementation for the PRNG-A

The pseudo-random generator based on a simple HCA with different number of cells is described in VHDL. As an example, the case of n= 63 cells is next considered. In the following table, the resource allocation using the Spartan 3E device (xc3s100e-5cp132) is presented. It can be remarked that the implementation offers a very compact solution and allocation of only 63 LUTs, i.e. 1 LUT per cell. It occupies 3% of all the available device resources of a low cost FPGA Xilinx Spartan 3E device.

TABLE I. DEVICE UTILIZATION SUMMARY (ESTIMATED VALUES) FOR SPARTAN 3E DEVICE

Module type implementation	Logic Utilization	Used	Available	% of all
	Number of Slices	36	960	3%

One CA module (63 cells)	Number of Slice Flip Flops	63	1920	3%
	Number of 4 input LUTs	63	1920	3%
	Number of bonded IOBs	65	83	78%
	Number of GCLKs	1	24	4%

Using the Xilinx ISE Simulator tool, in the following figures the pseudo-random number generator simulation is shown.

Figure 3. Snapshoot for simulated waveform for PRNG-A with 63 cells

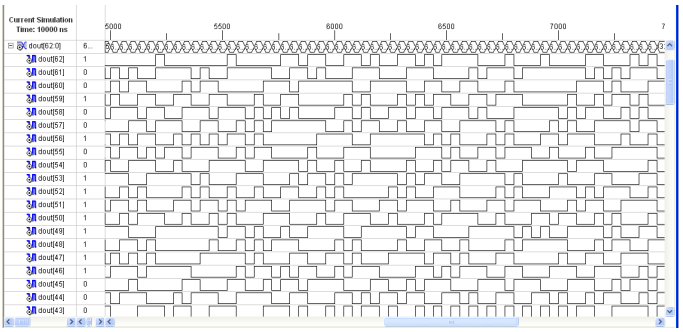

Figure 4. Snapshoot from the simulated process. Detailed simulated waveform for PRNG-A with 63 cells.

B. FPGA implementation for the PRNG-B

The architecture for PRNG-B based on the chain of HCA is shown in Fig.2. The synthesis is rather simple since it combines two HCA modules, described in VHDL, with their given parameters (previously saved as symbols). Schematic mode was chosen to design the top module PRNG-B as an interconnection of the two modules. The design can be easily extended to arbitrary number of HCA modules and cells per module. VHDL code implementation that model is considered. Using algebraic normal form representation as discussed in Section II, and $n = 31$ cells for each CA from the 2-stages chain results in a specific example discussed next. In Table II the device utilization summary (targeting the Spartan 3E device) is presented. The implementation is also very compact and efficient; using only one 1 LUT per cell (31 cells for each cellular automaton, 62 cells in total for the PRNG-B model).

TABLE II. DEVICE UTILIZATION SUMMARY (ESTIMATED VALUES) FOR SPARTAN 3E DEVICE

Module type implementation	Logic Utilization	Used	Available	% of all
Two CA module (31 cells each)	Number of Slices	36	960	3%
	Number of Slice Flip Flops	62	1920	3%
	Number of 4 input LUTs	62	1920	3%
	Number of bonded IOBs	33	83	39%
	Number of GCLKs	1	24	4%

It is very important to know for hardware implementation, the estimated power consumption, especially for the case of portable devices. In order to determine the estimated power consumption for the module, the *Xpower* tool (from the Xilinx ISE Project Navigator software) was used. The total estimated value for the above example is indicated to be 34 mW.

FPGA device for both proposed PRNG architectures was configured. For each design, a sequence of at least 100 million bits generated by the pseudo-random number generator was saved, evaluated and further analyzed with statistical tests.

Using an oscilloscope probe, real data from the FPGA corresponding to the PRNG was also captured and analyzed.

Figure 5. Oscilloscope probe capture from the PRNG sequence for chain HCA case

In Fig. 5 a capture from the generated sequence is presented. Also, in Fig 6 using the previous data acquisition, a spectral diagram is reported. The clock frequency of the PRNGs was set to 0.78 Mhz (i.e. 50Mhz / 64).

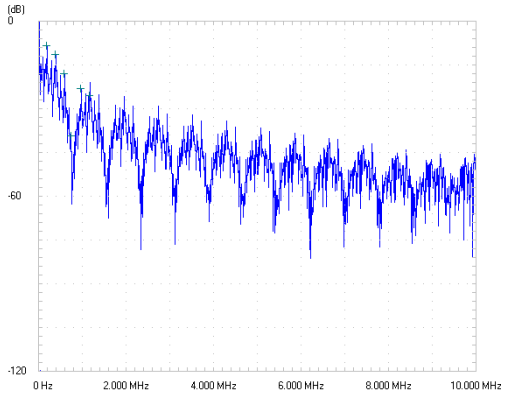

Figure 6. Spectral diagram for the generated data. Note the wide spectrum of the pseudorandom sequence

IV. ASSESEMENT USING THE NIST BATTERY OF TEST

In order to evaluate the proposed architecture for PRNGs based on HCA, the real data pseudorandom sequences were recorded and further evaluated running the NIST statistical tests [1]. The test results were saved in a file "finalAnalysisReport.txt" showing the two relevant values for each of the 188 tests. One value is the proportion P of sequences passing the test and the second is the distribution of p-values. The parameter significance level $\alpha=0.01$ is considered for all the applied tests. Having good cryptographic PRNG corresponds to passing all considered 188 NIST tests, as confirmed by results in Table III. The decision on passing or not the test is based on 100 sequences with 1 million of bits each. A detail of the results obtained in the case of PRNG-A is given in Fig. 7

TABLE III. NIST TEST RESULTS FOR 2 TYPES OF PRNG ARCHITECTURE – SINGLE HCA AND CHAIN HCA

Details of the PRNG architecture		***# OF FAILED TESTS (in 188)***
Type of architecture	***PRNG parameters***	
PRNG-A	n=63	NONE
PRNG-B	n=31 x 2	NONE

A specific test is considered passed if for that test $P > 0.96$ and $p-value-T > 10^{-4}$.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
10	11	9	9	11	14	8	9	12	7	0.924076	99/100	Frequency
10	7	12	10	9	8	11	7	15	11	0.798139	96/100	BlockFrequency
11	12	9	9	10	4	13	13	12	7	0.595549	98/100	CumulativeSums
12	6	14	6	13	6	13	7	12	11	0.350485	100/100	CumulativeSums
8	11	7	14	5	16	9	8	14	8	0.236810	99/100	Runs
15	11	8	11	9	13	8	4	11	10	0.514124	97/100	LongestRun
10	10	15	15	6	8	10	8	9	9	0.574903	100/100	Rank
9	7	12	15	9	8	9	11	15	5	0.383827	100/100	FFT
6	13	9	12	12	13	7	7	5	16	0.202268	99/100	NonOverlappingTemplate
11	8	10	8	14	11	13	8	10	7	0.851383	100/100	NonOverlappingTemplate
14	13	8	6	7	7	13	10	14	8	0.419021	100/100	NonOverlappingTemplate
....												
11	13	7	13	12	7	8	9	12	8	0.798139	98/100	OverlappingTemplate
12	7	9	15	13	4	10	12	7	11	0.366918	99/100	Universal
14	11	9	10	14	6	9	8	10	9	0.779188	99/100	ApproximateEntropy
8	7	7	4	4	4	6	9	7	1	0.249284	57/57	RandomExcursions
9	7	7	1	5	2	7	11	5	3	0.028817	56/57	RandomExcursions
7	8	5	2	5	6	6	8	7	3	0.514124	56/57	RandomExcursions
.....												
7	9	6	16	7	15	10	13	10	7	0.249284	99/100	Serial
8	13	7	10	6	12	9	9	12	14	0.699313	99/100	Serial
11	11	13	8	13	10	11	3	13	7	0.419021	99/100	LinearComplexity

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 96 for a sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately = 54 for a sample size = 57 binary sequences.

Figure 7. Complete results of the NIST tests applied to a real-data binary sequence obtained from the PRNG-B FPGA implementation. Some lines were ommited due to space reasons.

V. CONCLUDING REMARKS

FPGA implementations for two high quality PRNG architectures are considered. Both solutions are based on hybrid cellular automata and proved to be highly efficient in terms of resources (1 LUT/cell, better than most state of the art solutions) and throughput (equal to clock frequency, limited only by the specific FPGA technology). The first PRNG is based on a single HCA architecture with a large number of cells. The design can be easily expanded to any number of cells. The second PRNG implements a cryptographically more sophisticated chain solution, having the same low complexity and a high throughput. Real data streams obtained from implementations on a Spartan 3E low cost FPGA Xilinx device were applied to the NIST statistical tests battery suite, and all test passed. Both PRNG architectures are highly nonlinear (a plus in many cryptographic applications), and are easily scalable to arbitrary number of cells.

So far, we developed tools to produce automatically the specific VHDL code, given the PRNG parameters (cell ID, mask, initial-states, number of chain stages) [10]. Further research will focus on embedding the high quality PRNG in various applications. Particularly, they are of interest in the context of smart sensing based on chaotic scan [14]. Since the transfer of binary sequences to the PC and the running of NIST tests were found computationally intensive, the possibility to implement some of the NIST testing procedures on the same FPGA which implements the PRNG [13] will be also considered in the future in order to facilitate the on-line monitoring of performances for the proposed PRNG using different parameters.

ACKNOWLEDGMENT

This research was partially supported by the SCOUTER PN-II-IN-DPST-2012-1-0034 project.

REFERENCES

[1] A. Rukhin et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications", NIST Special Publication 800-22 (with revisions dated April, 2010).

[2] Pascal Giard, Georges Kaddoum, et al., "FPGA Implementation and Evaluation of Discrete-time Chaotic Generators Circuits", In Proceeding of: IECON, 2012 - 38th Annual Conference on IEEE Industrial Electronics Society.

[3] Leonidas Kotoulas, et al., "1-d Cellular Automaton for PseudoRandom Number Generation and its Reconfigurable Hardware Implementation " IEEE International Symposium on Circuits and Systems, ISCAS, 21-24 May, 2006.

[4] Juan C. Cerda, et al., "An Efficient FPGA Random Number Generator using LFSRs and Cellular Automata", IEEE 55th International Midwest Symposium Circuits and Systems (MWSCAS), 5-8 Aug 2012.

[5] Jonathan M. Comer, et al.,"Random Number Generators using Cellular Automata Implemented on FPGAs" 44th IEEE Southeastern Symposium on System Theory University of North Florida, Jacksonville, FL March 11-13, 2012

[6] Lakshman Raut and David H. K. Hoe. "Stream Cipher Design using Cellular Automata Implemented on FPGAs", 45th Southeastern Symposium on System Theory Baylor University, Waco, TX, USA, March 11, 2013

[7] Ding Jun, et al., "A high-performance pseudo-random number generator based on FPGA", Proceeding WNIS '09 Proceedings of the 2009 International Conference on Wireless Networks and Information Systems, pages 290-293.

[8] R. Dogaru, "Hybrid Cellular Automata as Pseudo-Random Number Generators with Binary Synchronization Property", in Proceedings of the International Symposium on Signals Circuits and Systems, Iasi Romania, July 2009, pp. 389-392.

[9] I. Dogaru and R. Dogaru, "Algebraic Normal Form for Rapid Prototyping of Elementary Hybrid Cellular Automata in FPGA", in Proceedings ISEEE 2010 (September 2010, Galati, Romania), pp. 277-280.

[10] I. Dogaru, R. Dogaru, C. Damian, "FPGA Implementation Of Chaotic Cellular Automaton with Binary Synchronization Property", in

Proceedings of COMM2010, 8th Intl. Conference on Communications, Bucharest, June 10-11, Vol. 1, pp. 45-48.

[11] R. Dogaru and Ioana Dogaru, "Applications of Natural Computing in Cryptology: NLFSR based on Hybrid Cellular Automata with 5-cell Neighborhood", PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A, Volume 14, Special Issue 2013, pp. 365–372.

[12] Radu Dogaru and Ioana Dogaru "Efficient and Cryptographically Secure Pseudorandom Number Generators based on Chains of Hybrid Cellular Automata Maps", in Proceedings of COMM2014 - Intl. Conference on Communications, Bucharest, May 2014.

[13] F. Veljkovic, V. Rozic, I. Verbauwhede, “Low-cost implementations of on-the-fly tests for random number generators”, in Design Automation and Test in Europe – DATE2012. EDAA, 2012.

[14] R. Dogaru, “A low complexity image sensing method using pseudo-random scan and recursive reconstruction with radial basis functions”, in Proceedings ISEEE 2013 (September 2013, Galati, Romania).