International Journal of Computer Science and Applications, Vol. 6, No, 4, pp 43 - 62 , 2009

DESIGN OF RECONFIGURABLE IMAGE ENCRYPTION PROCESSOR USING 2-D CELLULAR AUTOMATA GENERATOR

Machhout Mohsen, Guitouni Zied, Zeghid Medien and Tourki Rached

Electronic and Micro Electronics Laboratory ,Faculty of Sciences of Monastir, Monastir ,Tunisia
http://<ziedguitouni@yahoo.fr>

Abstract. Cellular Automata (CAs) have been applied successfully to several physical systems, processes and scientific problems that involve local interactions, such as image processing, data encryption and byte error correcting codes. In this paper, we analyze the cellular automata, and we propose a reconfigurable cryptosystem based on 2-D Von Neumann cellular automata as an image protection technique. In our scheme we used 2-D cellular automata to generate a high quality of random number as key stream. The security analysis of our proposed cryptosystem by statistical approach shows the high quality of the encrypted image. A comparative result shows that the performances of our proposed system are superior in encryption time to those of MIE, the VC, and the N/KC. In order to have a fair and detailed evolution, we implemented an AES (Advanced Encryption Standard) processor. The proposed CA reconfigurable processor is compared to the AES. Performance metrics such as the throughput (Mbps), the area (slices), the power consumption, and the correlation results from these implementations (CA and AES processor) are computed and analyzed.

Keywords: Cellular automata, random number generator, Encryption image, Security analysis, Reconfigurable architecture, AES.

1. Introduction

With the ever growing data communication and multimedia application, the cryptography protocols have become an essential requirement for communication privacy and for the storage and transmission of digital images. Besides, special and reliable security is needed in many applications, such as Internet communication, multimedia systems, medical imaging, pay-TV and military communication.

Since 1990, many specific methods have been proposed, such as SCAN–based methods [1], chaos-based method [2], true structure-based methods [3], and other miscellaneous methods [4] for image encryption [5]-[6].

In this paper, we propose an image encryption method based on 2-D von Neumann cellular automata. This method consists in replacing the pixel values by Xoring them with a 2-D based CA key. Reasons for using 2-D cellular automata for image encryption/decryption are described as follows: (a) CA has been applied successfully to

several physical systems, processes and scientific problems that involve local interactions, like image processing, data encryption and byte error correcting codes: (b) The number of CA evolution rules is very large; (c) Recursive CA substitution only

requires integer arithmetic and/or logic operations [7]. These characteristics make them easier to implement in hardware than other methods.

The paper is structured as follows: The next section provides a brief overview of the CAs application in cryptography. Section 3 details the proposed architecture and the overall design of the 2-D cellular automata implementation. Section 4 evaluates the performance of the CA processor with respect to the security in image encryption. Issues concerning hardware performances of the cryptosystems under consideration are discussed in section 5. The illustration of the usefulness and the efficiency of the proposed processor through comparison with the AES processor are described in Section 6. Conclusions are drawn in Section 7.

2. Cellular Automata and previous work

2.1. *Cellular automata*

Cellular automata (CA) are dynamic systems in which space and time are discrete. A cellular automaton consists of any array of cells; each of which can be in one of a finite number of possible states. It can be defined as a d-dimensional Euclidean space (where d =1, 2 or 3 is used in practice), partitioned into cells of uniform size, each one embedding an identical elementary automaton (ea). Input for each ea is given by the states of the elementary automaton in the neighboring cells, where neighborhood conditions are determined by a pattern invariant in time and constant over the cells. At the time t = 0, eas are in arbitrary states and the CA evolves changing the state of all ea at discrete times, according to a local rule. Each cell can have any of a finite number of states. As mentioned before, the states of the cells in the lattice are updated according to a local rule called the state transition function. That is, the state of a cell at a given time depends only on its own state and the states of its nearby neighbors at the previous time step. In this subsection we presented the two dimensional grids CA(d=2)

2.2. *Two-dimensional Cellular Automata*

A 2-D cellular automaton consists of two-dimensional lattice of finite automata cells, which are connected by a rectangular network. The global state of the cellular automata evolves through local transitions. The cells are identical and the transition rule of one cell is usually named neighborhood. For a 2-D von Neumann 2-state/3-state CA, the evolution of the (i, j)th cell can be represented as a combinational logic of the present states of the (i-1,j)th, the (i, j-1)th, the (i,j)th, the (i,j+1)th, and the (i+1,j)th. In order to delineate our result, we introduce the following rule-number: Since there are 64 possible (additive) rules, we need 6 bits to describe a rule. Let $S_{i,j}^{t}$ be the state of the cell at row i and column j, at time t. the state at the next time is given by:

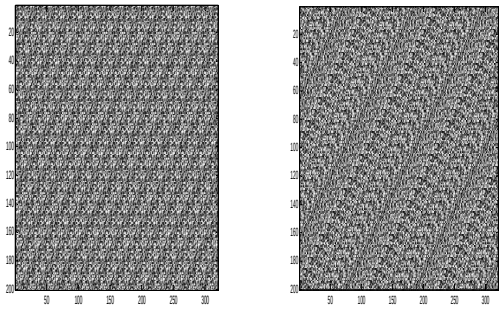
$S_{i,j}^{t+1}$ = X xor (C and $S_{i,j}^{t}$)) xor (N and $S_{i-1,j}^{t}$) xor (W and $S_{i,j-1}^{t}$)) xor (S and $S_{i+1,j}^{t}$) xor (E and $S_{i,j+1}^{t}$) (1)

Where C is the center, N is the north, S is the south, W is the west, and E is the east are binary variables [2]. They denote whether the respective neighboring cells states are taken into account or not. The binary variable X indicated linear (X = 0) from nonlinear (X = 1) additive rules. The genome of a cell is then given by a 6-bit string XCNWSE. For example, If XCNWSE = 001111 (Rule 15), equation (1) can be simplified as:

$$S_{i,j}^{t+1} = S_{i-1,j}^{t} \text{ xor } S_{i,j-1}^{t} \text{ xor } S_{i+1,j}^{t} \text{ xor } S_{i,j+1}^{t} \quad (2)$$

When XCNWSE =111111 (rule 63), equation (1) can be simplified as

$$S_{i,j}^{t+1} = 1 \text{ xor } S_{i,j}^{t} \text{ xor } S_{i-1,j}^{t} \text{ xor } S_{i,j-1}^{t} \text{ xor } S_{i+1,j}^{t} \text{ xor } S_{i,j+1}^{t} \quad (3)$$



(a) (b)

Fig.1. The evolved 8 x 8 cellular automata: (a) Rule 15, (b) Rule 63

In figure 1, two types of rule that correspond, respectively, to the Boolean equations (2) and (3) are presented. There is a tendency for some rules to “dominate” over others [8], i.e., the distribution of rules is not homogeneous; this is demonstrated in figure 1.

2.3. *Previous work*

Cellular Automata have been an active field of research during the last decade, one of the underlying motivations stemming from the advantages offered by CA when considered from a VLSI viewpoint: CA is simple, regular, locally interconnected, and modular [8]. These characteristics make them easier to implement in hardware than other models; Thus making CA an attractive choice for onboard applications. CA has traditionally been used to implement RNGs in cryptographic devices [9]. One dimensional CA RNG had been extensively studied in the past [10]-[11]-[12]. These studies have convincingly shown the suitability of CA generated pseudorandom numbers and their superiority in respect to other widely used methods. Cellular automata had

previously been used as encrypting devices by Wolfram [13] and by Nandi et al. [14]. Chowdhury et al. [9] described a methodology for producing pseudorandom numbers by two-dimensional cellular automata. Their results suggest that two-dimensional CA is superior to one dimensional of the same size in terms of the quality of the resulting pseudorandom numbers. HUA Li and C. N. ZHANG [15] described a reconfigurable crypto-architecture based on programmable cellular automata. Their results suggest that this VLSI architecture can be on line reconfigurable, and the ratio of throughput/area is much higher than that of the traditional FPGA methods. G. Alvarez and A. Hernandz [16], described a new graphic cryptosystem for encrypting images defined by any number of colors. It is based on a one dimensional reversible cellular automaton. R. J. Chan and J. L. Lai [17] presented a novel image security based on 2-D von Neumann CA. The encryption method is based on the replacement of the pixel values using a recursive CA substitution. Their results suggest that the system is economic in consuming computational resources because the encryption/ decryption scheme uses integer arithmetic and logic operations.

The goal of our work is to study and to conceive a crypto processor based on 2-D CA for image encryption. We have studied: the effect of the dimension of the CA on the quality of the encrypted image, the effect of the rule of CA on the quality of the encrypted image and the relation between the quality of the sequences generated and the quality of the encrypted image.

3. Cellular Automata Image Encryption processor design

3.1. Processor design

Figure 2 shows the block diagram of the different units of the proposed 2-D CA encryption processor.

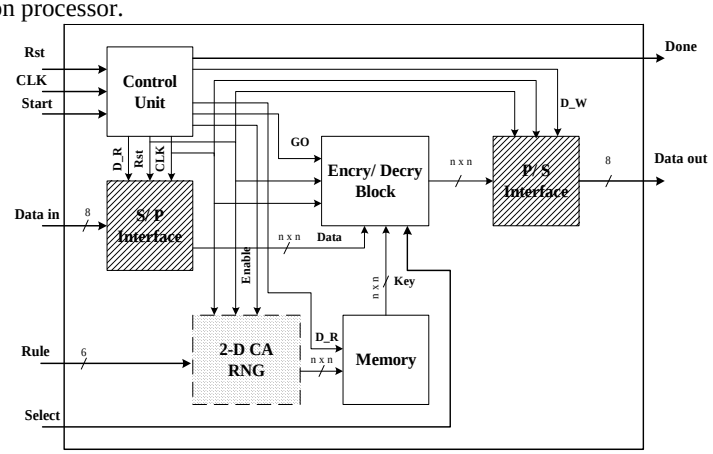


Fig.2 Proposed CA encryption processor design

The CA processor includes the following units.

- The Serial /Parallel and Parallel/ Serial interfaces take care of reading input data and writing encrypted output. They are controlled by the data ready (D_R), ciphertext ready and clk signals. When the bus puts a data to be read or written (D_W) this signal is selected and the data is taken.
- The control unit is used to generate control signals for all other units. Among other actions, the control unit determines when to reset the cipher hardware, to accept input data and to register output results.
- Encryption/decryption unit is used to encrypt input data. In our scheme, xor and xnor operations are used for data security upon the user need.
- 2-D CA RNG is used to generate a high quality of random sequence. Our proposed architecture for this block is presented in figure 4.
- Memory is used to store the current state of CA RNG (key).

3.2. Processor implementation

3.2.1. Image Encryption/ Decryption Process

The basic idea of image encryption with CA is to substitute the pixel values by XORing the plain data with a 2-D key generated. We have used the re-configurable 2-D von Neumann CA to perform image encryption/ decryption process. For image encryption; at the same time, the input is also a sequence of 8-bit (one Pixel) data and the output is a sequence of 8-bit encrypted data. The encryption method is defined as:

$$E(x) = CAT(S(x), K(x)), 1 \le x \le N$$

Where, the CAT (S(x), K(x)) means that S(x) and K(x) execute CA transform. The CA transform is logic operation. It can be expressed as:

CAT1: E(x) = S(x) xor K(x), when the input select = 1.

CAT2: E(x) = S(x) xnor K(x), when the input select = 0.

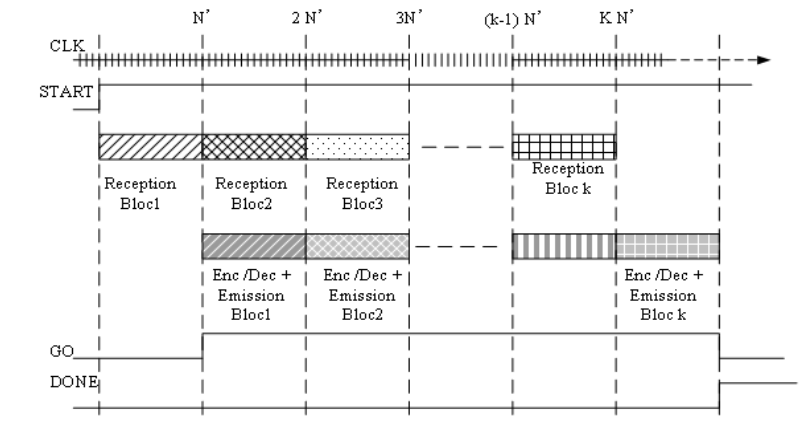


Fig.3 Timing diagram for encryption process sequence

The proposed encryption/ decryption process is shown in figure 3, where N' is the number of input blocks (numbers of cells divided by 8) and K= the size of image divided N'

The start signal is asserted at the start of each message. The CA processor is ready to accept data when start is asserted. Each 8-bit is clocked into the core on the rising edge of clk when start is asserted. The end of the message is indicated by a low-state of the start signal. After a feeding of block of N' octets at the input, the signal go is asserted as the encryption-decryption unit which encrypted the data.

3.2.2. 2-D CA Generator Implementation

The reconfigurable architecture of 2-D Cellular Automata Generator (CAG) is presented in Figure 4. The rules selections are commonly described as 6-bit words. The selected rules refer to the local rules and types of run, and then it is possible to change the rule while the encryption/decryption processing is in progress. The CAG architecture consists of: a Serial /Parallel Converter for storing the initial state, a 2-D CA, a Memory for storing the current state of CA (key), a Control Unit and a Parallel / Serial Converter. The CAG generates different key streams. The generation of random numbers is synchronous with the main clock signal (CLK).

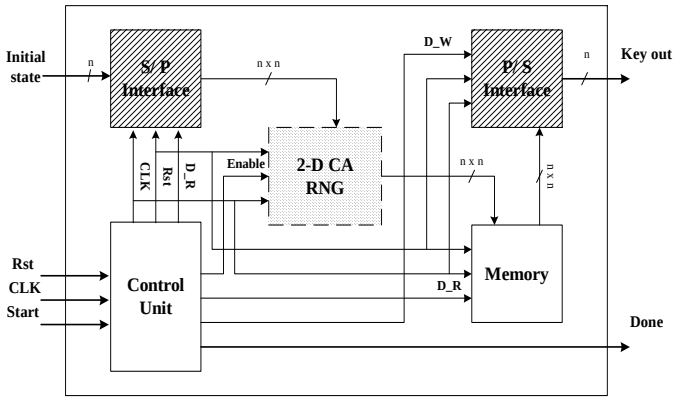


Fig.4 Proposed architecture of CAG

4. Security Analysis of our proposed architectures

4.1. Security analysis of CA random number generator

4.1.1. Cycle length

The length of the CA's state cycle is very important in determining the suitability of the CA as a generator of random number [8]. Ideally, an arbitrary n-cell CAG should have a maximum cycle length of about 2n-1.In table 1 we present the ratio of maximum to average cycle lengths for Various Small Two- Dimensional CAs for different sizes of CA.

Table1. Average and maximum cycle length for various 2-D CA

No. cells	Avg cycle Length	Max cycle Length
12 (3 x 4)	1261	4095
15 (3 x 5)	8079	32767
16 (4 x 4)	14286	65535
18 (3 x 6)	44671	262143
20 (4 x 5)	139681	1048575
21 (3 x 7)	203166	2097151
24 (4 x 6)	1409529	16777215
25 (5 x 5)	2458786	33554431
28 (4 x 7)	13308964	268435455

4.1.2. Diehard Test suite

Table2. Diehard Test result of a 2-D CAG

Test name	**4 x 4 CA**		**8 x 8 CA**		**16 x 16 CA**		**32 x 32 CA**	
	CA1	CA2	CA3	CA4	CA5	CA6	CA7	CA8
Birthday spacing	Fail	Fail	Fail	Pass	Pass	Fail	Pass	Pass
Binary rank 31*31	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
Binary rank 32*32	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
Binary rank 6*8	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
Count the1	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
Parking lot	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
Minimum distance	Fail	Fail	Pass	Fail	Pass	Pass	Pass	Pass
3D sphere	Fail	Pass	Pass	Pass	Pass	Pass	Pass	Pass
the SQUEEZE	Fail	Pass	Pass	Fail	Pass	Pass	Pass	Pass
Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Run up 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Run up 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Run down 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Run down 2	Pass	Fail	Pass	Pass	Pass	Pass	Pass	Pass
Craps of throws	Fail	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Craps of wins	Fail	Pass	Pass	Pass	Pass	Pass	Pass	Pass

The evolved and constructed 2-D cellular automata generator is subjected to an extensive battery of statistically randomness tests. The output sequence of the generator has to go through standard statistical tests as specified in Fips 140-2 for a test of randomness. These tests are performed on files of 10 Mega Bits (MB) for the Diehard test suites values [18]. Table 2 summarizes the test result of different sequences generated by the CAG generator. In this table four families of CA are presented, These CA consist of 16 cells (4 x 4), 64 cells (8 x 8), 256 cells (16 x 16) and 1024 cells (32 x 32), respectively.

According to table 2, our results show the low quality of the 4 x 4 CA This quality is due to the weak period of the generator and the initial value. Otherwise, the High quality of the 16 x16 and the 32 x 32 CAG is better than that the 8 x8 CA.

In table 3 we present the sensitivity of the 16 x 16 CA and the 32 x 32 CA for different rules. We can see that the quality of the generated sequence depends on the initial state, the number of cells and the transition function (rule).

Table3. CAG Sensitivity rules

Test name	**16 x 16 CA**			**32 x 32 CA**		
	Rule15	Rule47	Rule63	Rule15	Rule47	Rule63
Birthday spacing	Fail	Pass	Pass	Fail	Pass	Pass
Binary rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass
Binary rank 32*32	Pass	Pass	Pass	Pass	Pass	Pass
Binary rank 6*8	Pass	Pass	Pass	Pass	Pass	Pass
Count the1	Pass	Pass	Pass	Pass	Pass	Pass
Parking lot	Pass	Pass	Pass	Pass	Pass	Pass
Minimum distance	Pass	Pass	Pass	Pass	Pass	Pass
3D sphere	Pass	Pass	Pass	Pass	Pass	Pass
the SQUEEZE	Pass	Pass	Pass	Pass	Pass	Pass
Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass
Run up 1	Pass	Pass	Pass	Pass	Pass	Pass
Run up 2	Pass	Pass	Pass	Pass	Pass	Pass
Run down 1	Pass	Pass	Pass	Pass	Pass	Pass
Run down 2	Pass	Pass	Pass	Pass	Pass	Pass
Craps of throws	Fail	Pass	Fail	Pass	Pass	Pass
Craps of wins	Pass	Pass	Pass	Pass	Pass	Pass

4.2. Security analysis of the CA encryption processor

In order to choose the size of the CAG, we carried out a statistical survey of the ciphered image. This survey was based on the variation of the correlation of two vertical and horizontal adjacent pixels of the ciphered image and by a test on the histograms of the ciphered image.

4.2.1. Correlation of two Adjacent Pixels

We tested the correlation between two vertically adjacent pixels, and two horizontally adjacent pixels respectively, in a ciphered image. First, we randomly selected n pairs of two adjacent pixels from an image. Then, we calculated the correlation coefficient of each pair by using the following formula.

$$Cov(x,y) = E((x - E(x))(y - E(y))).$$

Where x and y are grey-scale values of two adjacent pixels in the ciphered image. Figure 5 and figure 6 present the variation of the correlation value of two horizontal and vertical adjacent pixels in function of the different rules of CA. In this experiment, we chose two tests images known as “Lena” image (256 pixels x 256) and Clown image (200 pixels x 320).

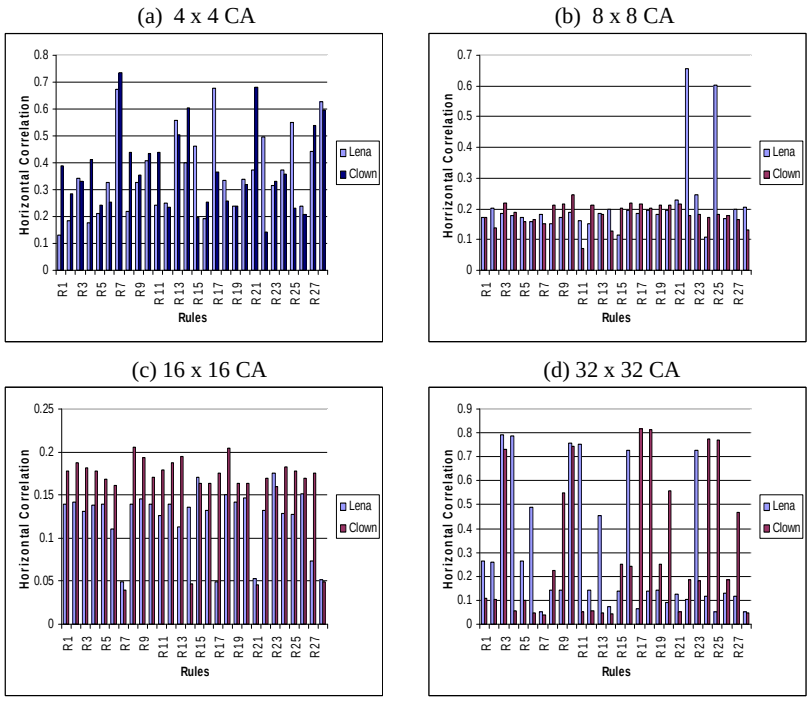
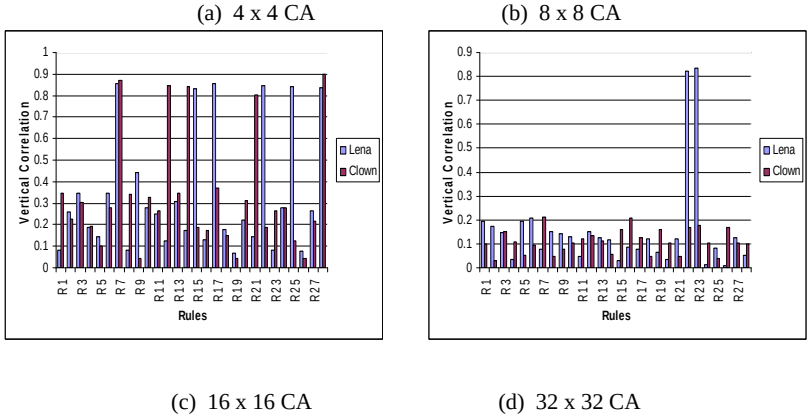


Fig.5 Correlation coefficients of horizontally adjacent pixels in the ciphered tests images (Lena and clown) using different rules



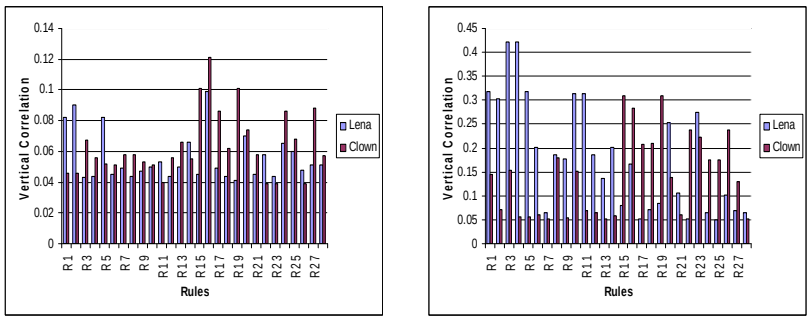


Fig.6 Correlation coefficients of vertical adjacent pixels in the ciphered tests images (Lena and clown) using different rules

Table 4. Arithmetic Average and Standard Deviation of the correlation

Image		**4 x 4 CA**	**8 x 8 CA**	**16 x16 CA**	**32 x32 CA**
Lena (256 x 256)	Avg Horz Correlation	0.360	0.211	0.124	0.289
	Avg Vertical Correlation	0.340	0.156	0.055	0.180
	STD Hor Correlation	0.152	0.121	0.035	0.270
	STD Verti Correlation	0.283	0.198	0.015	0.116
Clown (200 x 320)	Avg Horz Correlation	0.370	0.183	0.158	0.303
	Avg Vertical Correlation	0.334	0.111	0.063	0.142
	STD Hor Correlation	0.152	0.037	0.048	0.288
	STD Vertical Correlation	0.263	0.050	0.020	0.085

In table 4, we present the arithmetic average and standard deviation of the horizontal and vertical correlation for different sizes of the CA processor.

According to table 4, we can see that the arithmetic Average of the correlation of two vertical adjacent pixels is about 0.055 for 16 x 16 CA with a STD value about 0.015 for “Lena” encrypted image, and 0.048 with STD value about 0.02 for “Clown” encrypted image. The average of correlation of two Horizontal adjacent pixels is about 0.124 with a STD value about 0.035 for encrypted image “Lena”, and 0.158 with STD value about 0.048 for encrypted image “Clown”. These values are best (ones).

We conducted many experiments with different initial value, and different rules. We noticed that rules 15, 31, 47 and 63 gave a better performance results than other rules.
In figure 7 we presented the sensitivity of these rules for different standard images such as “lisaw,” “mouse,” “cheetah” and “clown”.

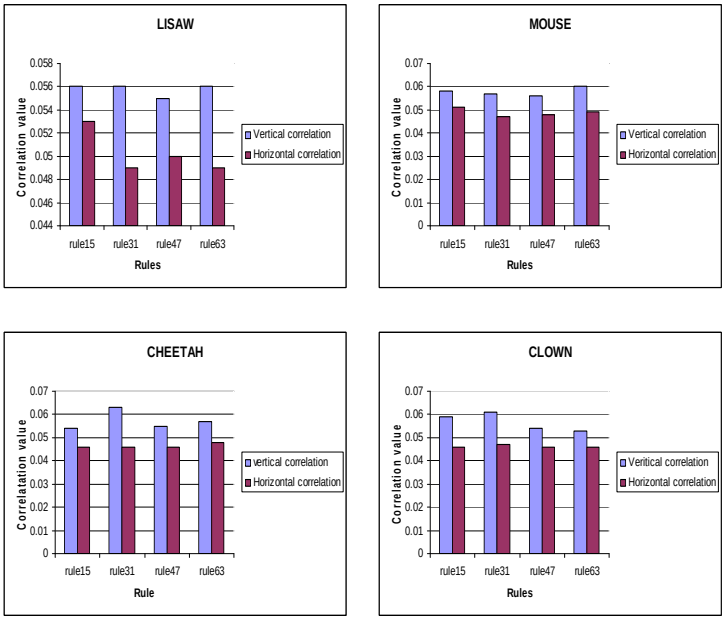
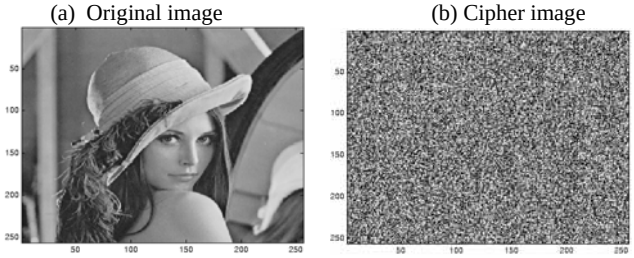


Fig.7 CA processor Sensitivity rules for different images

4.2.2. Histograms of Encrypted Image

We selected several grey-scale images (256 x 256) having different contents, and we calculated their histograms. One typical example among them is shown in Figure 8. We can see that the histogram of the ciphered image is fairly uniform and is significantly different from that of the original image. Therefore, it does not provide any indication to employing any statistical attack on the image under consideration. Moreover, there is no loss of the image quality after performing the encryption/ decryption steps.

(a) Original image (b) Cipher image



(c) Histogram of the Original image (d) Histogram of the ciphered image

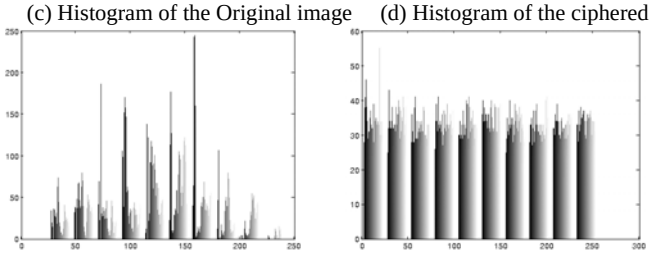
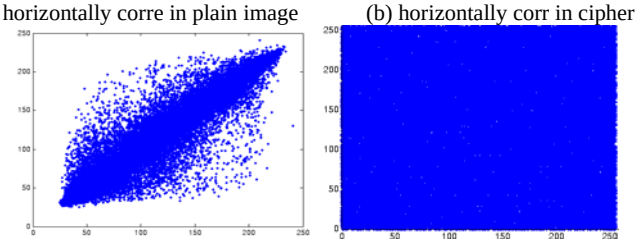


Fig.8 Histograms of the original image and ciphered image

(a) horizontally corre in plain image (b) horizontally corr in cipher image



(c) Vertically corr in plain image (d) Vertically corr in cipher image

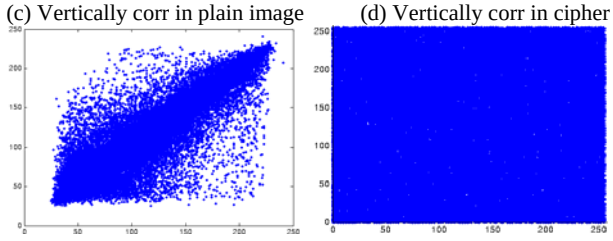


Fig.9 Correlation of two Horizontally and Vertically adjacent pixels :(a-c) in the plain image, and (b-d) in the ciphered image

5. Experimental results

The described architecture was implemented in VHDL using the Model technology Modelsim simulator and synthesized, placed, and routed using a target device of Xilinx (xilinx virtex XC2V1000 FPGA). Four performances metrics such as the clocking frequency (Mhz), the throughput (Mbps), the area (slices) and the total power consumption are computed. The results of the FPGA implementation are illustrated in table 5. To our knowledge, there are no published hardware implementations results for the 2-D CA generator, which we can compare with our respective implementations.

Table 5. FPGA synthesis results

Our Design	**Performance metrics**			
	Freq (Mhz)	Area (slices)	Power (mW)	Throughput (Mbps)
CA-processor (16x 16) XC2V1000	176.32	1184	516.61	1259.72
RNG- 16 cells (4 x 4) XC2V1000	184,16	95	428.83	736.64
RNG-36 cells (6 x 6) XC2V1000	184,16	151	433.56	1104.96
RNG- 81 cells (9 x 9) XC2V1000	163,94-	228	449.36	1473.28
RNG- 100 cells (10 x 10) XC2V1000	163,94	338	477.55	1639.46
RNG- 144 cells (12 x 12) XC2V1000	183,57-	463	499.36	2202.92
RNG- 196 cells (14 x 14) XC2V1000	164,62	610	507.39	2304.68
RNG- 256 cells (16 x 16) XC2V1000	163,703	773	523.19	2619.26

From table 5, the following comments can be drawn:

- Implementations of all RNG schemes (4 x 4, 6 x 6, 9 x 9, 10 x 10, 12 x 12, 14 x 14, 16 x 16) ranging from 95 to 773 of the total number of CLB slices available in the Virtex XC2V1000 device have been used in our design. That means, when the number of cells is changed slightly the area becomes slightly different.
- Additionally, we noted the low total power consumption by our proposed cryptosystem. It is about 426.76 mw for 4 x 4 CA, 449.64 mw for 8 x 8 CA,

516.61 mw for16 x 16 CA and it is about 704.04 mw for the 32 x 32 CA. Thanks to the best results of the 16 x 16 CA which are the low power consumption, the high quality of random number sequences generated and the security of the 16 x 16 CA processor by statistical approach, we consider the 16 x 16 CAG for encryption/ decryption images.

- As we can see, from the sensitivity to throughput, which is the main metric of CA the CA cells increase, as the throughput increases. As a result, the throughput goes from 1104.96 (RNG-generator (4 x 4)) to 2619.26 (RNG-generator (16 x 16)).
- As illustrated in table 5, our design is economic in consuming computational resource because the encryption/decryption scheme uses integer logic operations (xor and xnor). These characteristics make this crypto-processor able to be implemented in an embedded system.

6. CA processor and AES processor Performance comparison

In this section, we analyze the AES, and we compare our proposed CA processor to AES. In order to have a fair and detailed evaluation, we implemented AES-128 encryption-decryption.

6.1. AES processor

Rijndael is a block cipher with a variable key length and block length. The AES standard has specified the block size of 128-bit and key size of 128-bit, 192-bit, and 256-bit respectively. There are three kinds of choices for the cipher key of the AES: 128-, 192- and 256-bit, called AES-128, AES-192, and AES-256, respectively. The AES algorithm is a block cipher that can process data blocks of 128 bits [18]. Each 128-bit data block (plaintext) can be represented as a two-dimension 4×4 array of bytes called the state. The only differences are the number of rounds performed and round keys needed. Its key setup time is excellent, and its key agility is good. AES requires a very low memory which makes it very well suitable for restricted-space environments, where it also shows an excellent performance.

AES consists of four transforms (SubBytes, ShiftRows, MixColumns, AddRoundKe) operating on bytes, rows and columns of the 4 x 4 byte array, called the state that represents the data block. The transformation of the plain text to the cipher text takes N_r round of operations. N_r is a number associated with the key length. Most of the rounds take identical structure except the last one which doesn't use the MixColumns transform. Before the cipher operation takes place, a key schedule is generated. A round key is required for each round of the cipher algorithm. The round key for the first round is the private cipher key. For a given round, the first round key is obtained by first rotating once

the last round key, then substituting each byte using the S-box in the SubBytes function. Thereafter XORing the result with a given constant and finally XORing the result with the first round key of the previous round. The subsequent round keys of the current round are computed using a XOR of the previous key in the current round and the one inversely respective from the previous round. The details of the AES algorithm can be found in [18].

6.2. Area, Power, and Throughout analysis

Table 6 compares our implementations known as a CA processor and an AES processor. The comparison is based on performances in terms of frequency, area, power and throughput.

Table 6. Results of the iterative AES-128 design and CA processor

Our Design	**Performance metrics**			
	Freq (Mhz)	Area (slices)	Power (mW)	Throughput (Mbps)
AES-128-encryption XC2V1000	159	1743	537,50	1850
AES-128-encry/decryp XC2V1000	82	3555	637,50	1049
CA processor XC2V1000	176.32	1184	516.61	1259.72

It is shown that the CA processor requires the largest amount of throughput, so its performances are superior in its throughput is about 210 Mbps, while its area occupation is about 2371 slices and a reduction in power above 120 mW. That we have a reduction in area occupation of 2371 slices, a decrease of about 120 mw in power consumption and an increase of throughout of 210 Mbps.

Table 7.Average time required by CA processor for different images

Image (size)	**Encryption time (ms)**
Lisaw (256 x 256)	0.786
Lena (256 x 256)	0.786
Cheetah (200 x 320)	0.767
Clown (200 x 320)	0.767

Table 7 shows the average time required by CA processor for different images (Lena", "Lisaw," "Cheetah" and"Clown"). Our results are compared with these obtained by existing image encryption algorithms.

Table 8 examines quantitatively the encryption time of the MIE, the VC, the N/KC, the AES [19] and the 2-D CA proposed scheme. We can note clearly that our proposed method is much faster then its counterpart.

Table8. Encryption time using different algorithms with Lena as test image

Algorithm	**Encryption (s)**
MIE	0.27
VC	1.98
N/KC	0.15
AES	0.03175
2- D CA	0.000785

6.3. Security analysis

A good encryption scheme should be sensitive to the secret keys, and the key space should be large enough to prevent brute-force attacks. For AES-128, the key space size is 2^{128}. The experimental results demonstrate that AES is very sensitive to the secret key as well. This is shown by a test on the correlation of adjacent pixels in the ciphered image (see Figure 10).

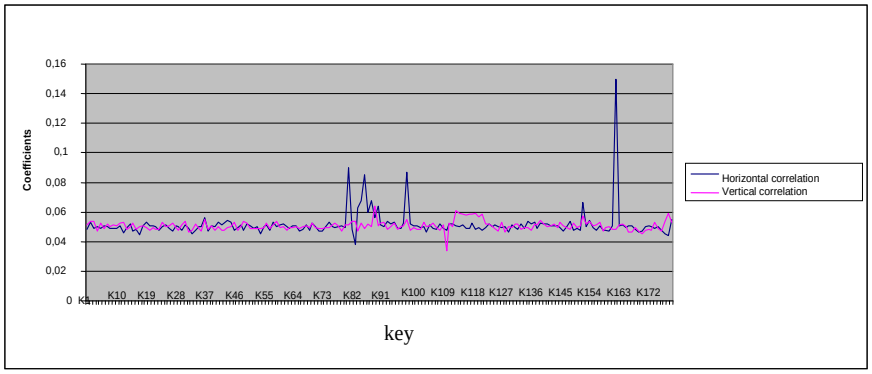


Fig.10 Correlation coefficients of adjacent pixels in the Lena ciphered image using different Ki

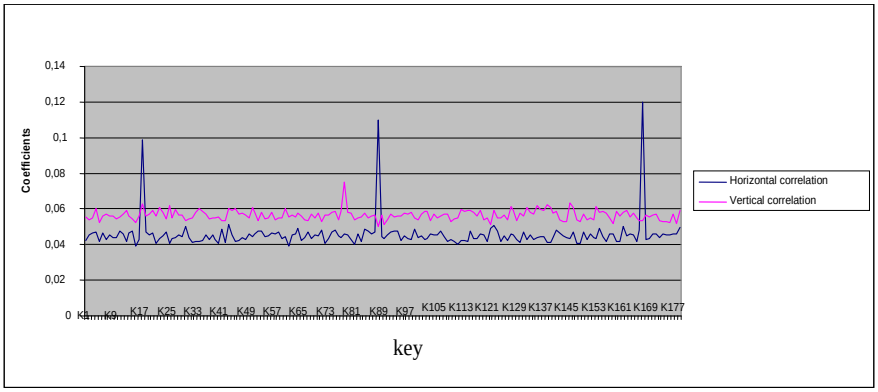


Fig.11 Correlation coefficients of adjacent pixels in the Clown ciphered image using different Ki

Figures 10-11 show the simulation results of the correlation coefficients of two adjacent pixels in two images (ciphered Lena and Clown test images) using different secret keys, Ki. As can be seen, when Ki changes slightly, the ciphered image becomes absolutely different: By studying the strength of the confusion and diffusion properties, and the security against statistical attack, AES ensures a high security for ciphered images.
Table 9 compares our implementations (CA and AES processor) in terms of an arithmetic average (Avg) and standard deviation (STD)of the horizontal and vertical correlation.

Table 4: Arithmetic Average and Standard Deviation of the correlation

Image		**16 x16 CA**	**AES processor**
Lena (256 x 256)	Avg Horz Correlation	0.124	0.05
	Avg Vertical Correlation	0.055	0.05
	STD Hor Correlation	0.035	0.0014
	STD Verti Correlation	0.015	0.0013
Clown (200 x 320)	Avg Horz Correlation	0.158	0.045
	Avg Vertical Correlation	0.063	0.058
	STD Hor Correlation	0.048	0.0024
	STD Vertical Correlation	0.020	0.002

According to table 9, we can see that the AES processor assures more security than the CA processor. We can see that the STD horizontal and vertical deviation for AES is

about 0.0014 and 0.0013, respectively, for Lena ciphered image. So, the correlation deviation is constant for AES algorithm, which is the main metric for AES under CA (16 x 16) processor. Therefore, the application can be divided into two groups: (i) The first group requires the largest security and the medium throughput, so the AES processor is recommended; (ii) The second group requires the largest throughput and the medium security, therefore the CA (16 x 16) processor is recommended.

7. Conclusion

In this paper, the image encryption method based on 2-D von Neumann cellular automata is proposed. The reconfigurable design and the hardware implementation of the cryptosystem based on 2-D cellular automata are described. The security analysis of our proposed cryptosystem shows the high quality of 16 x 16 CA for encryption image. The comparative results show, that the performance of the proposed system is superior in encryption time to that of other algorithms MIE, the VC, the N/KC and AES. The processor is economic in consuming computational resource because the encryption/decryption scheme uses integer logic operations. The security of this system depends on the high quality of the key stream generated by the cellular automata generator.

References

[1] N. Bourbakis, C. Alexopoulos,:Picture data encryption using SCAN patterns, Pattern Recognition, vol. **25** (6), pp567-581, 1992

[2] J. Scharinger, :Fast encryption of image data using chaotic Kolmogorov flows:, Electronic Imagining, Vol. **17** (2), pp 318-325, 1998.

[3] L. Chang,:Large encrypting of binary images with higher security:, Pattern Recognition Letter, Vol. **19** (5), pp 461-468, 1998.

[4] T. Chuang, J. Lin, :New approach to image encryption:, Electronic Imaging, Vol. **4**, pp 350-356, 1998.

[5] H. K. -C Chang, J. –L. Liu, :A linear quad tree compression scheme for image encryption, Signal Process.: Image Commu., Vol. **4**, pp 279-290, 1997.

[6] D. Jones, :Application of splay trees to data compression:, Commun. ACM, pp 996-1007, 1988.

[7] R. J. Chen and J. L. Lai,:Image security system using recursive cellular automata substitution, Pattern Recognition Vol. **40**, 2006, pp. 1621-1631.

[8] M. Tomassini, M. Sipper, M. Perrenoud,:On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata, IEEE Transactions on computers, vol. **49** (10), 291-305, October 2000.

[9] D.R. Chowdhury, I.S. Gupta, and P.P. Chaudhuri, :Aclass of Two-Dimensional Cellular Automata and applications in Random Pattern Testing, J. Electronic Testing: Theory and Applications, vol.**5**, pp.65-80, 1994.

[10] P.P. Chaudhuri, D.R. Chowdhury, :Additive Cellular Automata Theory and application, vol.**1**. Los Alamitos, Calif.: IEEE CS Press, 1997.

[11] P.D. Hortensius, R.D.McLeod, and H.C. Card, :Parallel Random Numbers VLSI Systems using Cellular Automata, IEE Transactions on Cmputers, vol. **38**, no. 10,pp.1,466-1,473 October 1989.

[12] P. Tsalides, T.A. York, and A. Thanailakis, :Pseudorandom Number Generators for VLSI Systems Based on Linear Cellular Automata, IEE Proc. E. Computers and Digital Technology, vol. **138**, pp. 241-249, 1991.

[13] S. Wolfram, :Cryptography with cellular automata, Advances in Cryptography: Crypto'85 Proceedings, Lecture Notes in Computer science, Vol. **218**, Springer, pp 429-432, 1986.

[14] S. Nandi, B. K. Kar, :Theory and applications of cellular automata in cryptography, IEEE trans. Comput. **43** (12), pp 1346-1357, 1994.

[15] HUA LI and C. N. ZHANG, :A Cellular Automata Based Reconfigurable Architecture for Hybrid Cryptosystems, the Computer Journal, Vol.**47**, No.3, 2004.

[16] G. Alvarez, A. Hernandez, :A New Graphic Cryptosystem Based on One-Dimensional Memory Cellular Automata, IEEE, 2005.

[17] G. Marsaglia, http://stat.fsu.edu/~geo/diehard.html,

[18] National Institute of Standards and Technology (NIST), :Advanced Encryption Standard (AES), Federal Information Processing Standards Publications (FIPS PUBS) 197-26 (2001).

[19] Z. Medien, M. Mohsen, K. Lazhar, B. Adel, and T. Rached,:A Modified AES Based Algorithm for Image Encryption, Int. Journal of Computer Science and Engineering, Vol.**1**, pp.70–75,2007.