ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

*An ISO 3297: 2007 Certified Organization* *Vol. 3, Special Issue 3, April 2014*

International Conference on Signal Processing, Embedded System and Communication Technologies and their applications for Sustainable and Renewable Energy (ICSECSRE '14)

Organized by

Department of ECE, Aarupadai Veedu Institute of Technology, Vinayaka Missions University,

Paiyanoor-603 104, Tamil Nadu, India

# FPGA Implementation of Cellular Automata Based Stream Cipher: YUGAM-128

K. J. Jegadish Kumar[1],S.Sudharsan[2], V.Karthick[3]

[1]Assistant Professor, SSN College of Engineering, Chennai, India

[2,3]PG Scholar, Dept. of ECE, SSN College of Engineering, Chennai, India

***Abstract*** **—** Ubiquitous computing is fetching a significant part in everyone's life. Few such examples are the mobile communication, personal computation and portable hand held devices. The growth in ultra-low power technology enabled the new development of small autonomous mobile devices. For the wireless communication systems with these portable mobile devices, security is a critical factor due to their impact on privacy.. Traditional cryptographic algorithms are much complex and power consuming thereby unfit for this resource constrained applications. In this paper, a novel stream cipher called YUGAM-128 is designed using one dimensional cellular automata (CA) rule 30 and linear feedback shift register (LFSR). The prime aspect of the stream cipher is to generate random 128 bit keystream. The proposed stream cipher is implemented and synthesized in Spartan-3 FPGA device using Xilinx 13.2.

***Keywords*** **–** Cellular Automata; Random number generator; LFSR; Stream Cipher

## I. INTRODUCTION

PSEUDORANDOM number generation by cellular automata (CA) has been an active field of research in the last decade [1], One of the underlying motivations stemming from the advantage offered by the CAs when considered from VLSI viewpoint: CAs are simple, regular, locally interconnected, and modular. These characteristic make them easy easier to implement in hardware than other models, thus making CAs as an attractive choice for on board applications. CA has been traditionally been used to implement RNGs in cryptographic devices [2] and in Built In Self-Test (BIST) circuits [3]. Random number generators play an import rule in several computational fields such as stochastic optimization methods. With the advent of massively parallel scientific computation, the parallel generation of pseudorandom numbers has become essential. With the advent massively parallel scientific computation, parallel generation of pseudorandom number has become essential.

The above domains depend critically on the quality of the random numbers as measured by appropriate statistical tests. Moreover, when very long sequences of random numbers are needed, computational efficiency is often of prime import, i.e., The sequence must be produced as rapidly as possible. CAs provide a good solution to this problem, able to produce rapid high-quality Random-number streams.

One-dimensional CA random number generators have been extensively studied in the past [1], [3], [4], [5]. These studies have shown convincingly the suitability of CA-generated pseudorandom numbers and their superiority with respect to other widely used methods, such as linear feedback shift registers (LFSRs), especially in the case of delay type faults which require pairs of patterns in a specified order [6]. In these works, CA RNGs were essentially handcrafted by studying the structure of the bit patterns generated over time, with theoretical results serving as a baseline offering guidance.

The mass use of hand-held devices/PDA has popularized the use of stream ciphers. Stream ciphers are much less power consuming, requires small space for their operations and are faster in operation than other cryptographic algorithms. Generally, in stream ciphers a secret key and a public IV are input. Key stream bits are generated by the cipher per cycle of operation. The plain-text is XORed on the encryption side with the generated key stream to produce the cipher-text. Decryption is carried out by simply XORing the cipher-text with the key stream.

## II. CELLULAR AUTOMATA THEORY

A cellular automaton (CA) is dynamical systems in which space and time are discrete. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

*An ISO 3297: 2007 Certified Organization* *Vol. 3, Special Issue 3, April 2014*

International Conference on Signal Processing, Embedded System and Communication Technologies and their applications for Sustainable and Renewable Energy (ICSECSRE '14)

Organized by

Department of ECE, Aarupadai Veedu Institute of Technology, Vinayaka Missions University,

Paiyanoor-603 104, Tamil Nadu, India

synchronously in discrete time steps, according to a local, identical interaction rule. Here, we will only consider Boolean automata in which the cellular state, s, 2 f0; 1g. The state of a cell at the next time step is determined by the current states of a surrounding neighbourhood of cells. The cellular array (grid) is d-dimensional, where d. 1; 2; 3 is used in practice; in this paper, we shall concentrate on d. 2, i.e., On two-dimensional grids. The identical rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table (also known as the transition function), with an entry for every possible neighbourhood configuration of states. The cellular neighbourhood of a cell consists of itself and of the surrounding (adjacent) cells. For one-dimensional CAs, a cell is connected to r local neighbours (cells) on either side, where r is referred to as the radius (thus, each cell has 2r. 1 neighbours). For two-dimensional CAs, two types of cellular neighbourhoods are usually considered: five cells, consisting of the cell along with its four immediate non diagonal neighbours (also known as the von Neumann neighbourhood) and nine cells, consisting of the cell along with its eight surrounding neighbours (also known as the Moore neighbourhood). In this work, we only consider 5-neighbor grids, thus limiting the already large search-space size; moreover, results exist only for this neighbourhood type, which is also more amenable to hardware implementation.

When considering a finite-size grid, cyclic boundary conditions are frequently applied, resulting in a circular grid for the one-dimensional case and in a toroidal one for the two-dimensional case. Fixed, or null, boundary conditions can also be used, in which the grid is surrounded by an outer layer of cells in a fixed state of zero. This case of configuration is usually easier to implement in hardware.

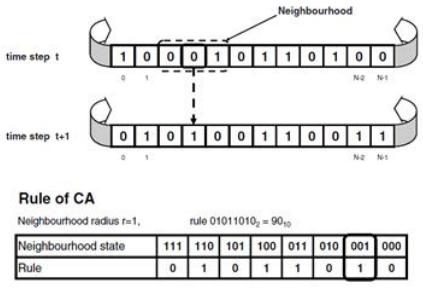


Fig. 1 1D Cellular Automata

## III. STREAM CIPHER

A stream cipher has a variable message input length, and it can be viewed as a small but changing secret substitution table that transforms plaintext bits at different positions with different substitution tables (the XOR operation between plaintext and key stream can be viewed as one-bit substitution determined by a key stream bit). A stream cipher consists of a state update function and an output function. The state of a stream cipher is updated continuously during encryption so that bits at different positions in a message are encrypted with different states. The output function generates key stream bits from the state and performs encryption or decryption. If the initial state of a stream cipher is not the same as the key, key setup is required to generate the initial state from the key. A key is used with different initialization vectors (IVs) is to generate key streams. The key/IV setup (resynchronization) is required to generate the initial state from the key and IV.

The criteria for good stream cipher are, long period with no repetitions statistically random, Large linear complexity (based on the size of equivalent LFSR), Correlation immunity (have the tradeoff with linear complexity), Confusion (output bits depend on all key bits) Diffusion and Use of highly non-linear Boolean functions.

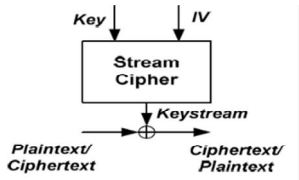


Fig. 2 Block diagram of stream cipher

## IV. DESIGN APPROACH

### *A. CA Rule Based Function*

Rule 30 is a one-dimensional binary cellular automaton rule introduced by Stephen Wolfram in 1983. Wolfram describes it as being his "all-time favourite rule" and details it in his book, A New Kind of Science. Using Wolfram's classification scheme, Rule 30 is a Class III rule, displaying a periodic, chaotic behaviour.

This rule is of particular interest because it produces complex, seemingly random patterns from simple, well-defined rules and offers reversible property. Because of this, Wolfram believes that Rule 30, and cellular automata in general, are the key to understanding how simple rules produce complex structures and behaviour in nature. Rule 30 has also been used as a random

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

*An ISO 3297: 2007 Certified Organization* *Vol. 3, Special Issue 3, April 2014*

International Conference on Signal Processing, Embedded System and Communication Technologies and their applications for Sustainable and Renewable Energy (ICSECSRE '14)

Organized by

Department of ECE, Aarupadai Veedu Institute of Technology, Vinayaka Missions University,

Paiyanoor-603 104, Tamil Nadu, India

number generator in Wolfram's program mathematical and has also been proposed as a possible stream cipher for use in cryptography.

In all of Wolfram's elementary cellular automata, an infinite one-dimensional array of cellular automaton cells with only two states is considered, with each cell in some initial state. At discrete time intervals, every cell spontaneously changes state based on its current state and the state of its two neighbors. For Rule 30, the rule set which governs the next state of the automaton is given in table I (6)

**TABLE I. Rule 30 Neighborhood State**

| Current state Neighbourhood | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| New state for centre cell | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

The following pattern emerges from an initial state in a single cell with state 1 (shown as black) is surrounded by cells with state 0 (white). Time increases down the vertical axis.

The evaluated function for CA rule 30 is

$$f(x) = \overline{x}_{i-1}x_{i+1} + \overline{x}_{i-1}x_i + x_{i-1}\overline{x}_i\overline{x}_{i+1}$$

For Rule 45, the rule set which governs the next state of the automaton is evaluated as the function,

$$f(x) = \overline{x}_{i-1}\overline{x}_{i+1} + x_{i-1}\overline{x}_i x_{i+1} + \overline{x}_{i-1}x_i$$

For Rule 57, the rule set which governs the next state of the automaton is evaluated as the function,

$$f(x) = \overline{x}_i\overline{x}_{i+1} + x_{i-1}\overline{x}_i + \overline{x}_{i-1}x_i x_{i+1}$$

An LFSR consists of clocked storage elements (flip-flops) and a feedback path. The number of storage elements gives us the said to be of degree *m*. The feedback network computes the input for the last flip-flop as XOR-sum of certain flip-flops in the shift register Simple LFSR We consider an LFSR of degree $m = 3$ with flip-flops *FF*2, *FF*1, *FF*0, and a feedback path as shown in Fig. 3. The internal state bits are denoted by $s_i$ and are shifted by one to the right with each clock tick. The rightmost state bit is also the current output bit. The leftmost state bit is computed in the feedback path, which is the XOR sum of some of the flip-flop values in the previous clock period. Since the XOR is a linear operation, such circuits are called linear feedback shift registers. If we assume an initial state of ($s_2$= 1, $s_1$= 0, $s_0$= 0), Table 2.2 gives the complete sequence of states of the LFSR. Note that the rightmost column is the output of the LFSR. One can see from this example that the LFSR There is a simple formula which determines the functioning of this LFSR. Let's look at how the output bits $s_i$ are computed, assuming the initial state bits $s_0$, $s_1$, $s_2$:

$$s_3 \equiv s_1 + s_0 \bmod 2$$

$$s_4 \equiv s_2 + s_1 \bmod 2$$

$$s_5 \equiv s_3 + s_2 \bmod 2$$

In general, the output bit is computed as $s_{i+3} \equiv s_{i+1} + s_i \bmod 2$ Where $i = 0,1,2, \ldots$

*B. Mathematical Description of LFSRs*

The general form of an LFSR of degree *m* is shown in Fig. 2.4. It shows *m* flip-flops and *m* possible feedback locations, all combined by the XOR operation. Whether a feedback path is active or not, is defined by the feedback coefficient $p_0, p_1, \ldots, p_{m-1}$:

- If $pi = 1$ (closed switch), the feedback is active.
- If $pi = 0$ (open switch), the corresponding flip-flop output is not used for the feedback.

With this notation, we obtain an elegant mathematical description for the feedback path.

***The maximum sequence length generated by an LFSR of degree m is 2m−1.***

If we *multiply* the output of flip-flop *I* by its coefficient $p_i$, the result is either the output value if $p_i$= 1, which corresponds to a closed switch, or the value zero if $p_i$=0, which corresponds to an open switch. The values of the feedback coefficients are crucial for the output sequence produced by the LFSR.

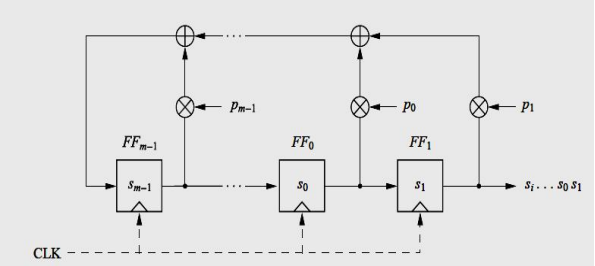


Fig.3 Block diagram of LFSR with tapping

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

*An ISO 3297: 2007 Certified Organization* *Vol. 3, Special Issue 3, April 2014*

**International Conference on Signal Processing, Embedded System and Communication Technologies and their applications for Sustainable and Renewable Energy (ICSECSRE '14)**

**Organized by**

**Department of ECE, Aarupadai Veedu Institute of Technology, Vinayaka Missions University,**

**Paiyanoor-603 104, Tamil Nadu, India**

Let's assume the LFSR is initially loaded with the values $s_0, \ldots, s_{m-1}$. The next output bit of the LFSR $s_m$, which is also the input to the leftmost flip-flop, can be computed by the XOR-sum of the products of flip-flop outputs and corresponding feedback coefficient:

$$s_m \equiv s_{m-1}p_{m-1} + \cdots + s_1p_1 + s_0p_0 \bmod 2$$

The next LFSR output can be computed as:

$$s_{m+1} \equiv s_m p_{m-1} + \cdots + s_2p_1 + s_1p_0 \bmod 2$$

In general, the output sequence can be described as

$$s_{i+m} \equiv \sum_{j=0}^{m-1} p_j . s_{j-1} \bmod 2$$

Clearly, the output values are given through a combination of some previous output values. LFSRs are sometimes referred to as umber of recurring states, the output sequence of an LFSR repeats periodically. Moreover, an LFSR can produce output sequences of different lengths, depending on the feedback coefficients. The following theorem gives us the maximum length of an LFSR as a function of its degree.

It is easy to show that this theorem holds. The state of an LFSR is uniquely determined by the minterm register bits. Given a certain state, the LFSR deterministically assumes its next state. Because of this, as soon as an LFSR assumes a previous state, it starts to repeat. Since an $m$-bit state vector can only assume $2_{m-1}$ nonzero states, the maximum sequence length before repetition is $2_{m-1}$. Note that all zero state must be excluded. If an LFSR assumes this state, it will get "stuck" in it, i.e., It will never be able to leave it again. Note that only certain configurations $(p_0, \ldots, p_{m-1})$ yield maximum length LFSRs. We give a small example for this below.

## V. PROPOSED STREAM CIPHER ARCHITECTURE

The figure represents the simple the simplest architecture of the proposed steam cipher using cellular automata. In this architecture, the initial key 128 bit is transformed into unidentifiable form by the cellular automata (CA) rule. The 128 bit in initialization is applied to the linear feedback shift register (LFSR) and then its output is xored with CA rule based update key to generate a key stream per clock cycle.

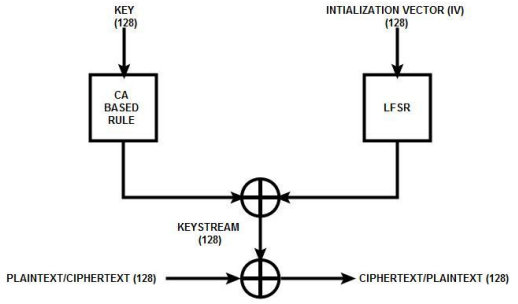


Fig. 4 Architecture of proposed stream cipher

## VI. HARDWARE IMPLEMENTATION AND SYNTHESIS RESULTS

The proposed stream cipher is implemented in SPARTAN-3 xc3vs50-5 pq208 device using Xilinx 13.2. The hardware implementation of the algorithm is very simple as the operator used in the design of stream cipher is flip-flops based hardware circuits. The nonlinearity of the algorithm is decided by the rule 30 CA based pseudo random number generator. The results of the Xilinx Spartan 3 FPGA implementations are shown in Table II.

TABLE II. RESULTS OF THE XILINX SPARTAN 3 FPGA IMPLEMENTATION

| Stream Cipher | Maximum Clock Frequency (MHz) | Maximum Throughput (Mbps) | Area (Slices) | Throughput/ Area (Mbps/Slice) |
|---|---|---|---|---|
| YUGAM-128 | 343 | 6255 | 320 | 19.55 |
| DECIM v2 | 185 | 46.25 | 80 | 0.58 |
| DECIM 128 | 174 | 43.5 | 89 | 0.49 |
| Edon 80 | 130 | 130 | 1284 | 0.10 |
| F-FCSR-H v2 | 138 | 1104 | 342 | 3.23 |
| F-FCSR-16 | 134 | 2144 | 473 | 4.53 |
| Grain v1 | 196 | 196 | 44 | 4.45 |
| Grain v1(X16) | 130 | 2080 | 348 | 5.98 |
| Grain128(X32) | 133 | 4256 | 534 | 7.97 |
| Mickey 128 2.0 | 223 | 223 | 176 | 1.27 |
| Moustique | 225 | 225 | 278 | 0.81 |
| Pomaranch | 49 | 49 | 648 | 0.08 |

The Xilinx static timing analysis tool is used to determine the maximum clock frequency. Brief overviews of each cipher implementation are given in the following. The ECIM ciphers produced low area implementations due to the simple LFSR structure; however, the through-put was low due to the decimation factor of four. Edon80 was the largest design of the implemented ciphers. The F-FCSR family of ciphers were fairly large (342 slices and 473 slices) compared to the smallest ciphers, but due to the high data radix (8 bits/cycle and 16 bits/cycle), the throughput and

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

*An ISO 3297: 2007 Certified Organization* *Vol. 3, Special Issue 3, April 2014*

**International Conference on Signal Processing, Embedded System and Communication Technologies and their applications for Sustainable and Renewable Energy (ICSECSRE '14)**

**Organized by**

**Department of ECE, Aarupadai Veedu Institute of Technology, Vinayaka Missions University,**

**Paiyanoor-603 104, Tamil Nadu, India**

through-put/area was relatively high. Grain ranks top in terms of small area and good throughput/area ratio[19]. It was the smallest cipher and the parallelized versions of Grain produced higher throughput/area ratios. Mickey had a medium size area but a good throughput/area ratio; the main disadvantage Mickey had in Xilinx FPGAs were that the S and R registers could not be inferred into Xilinx primitive shift register blocks; thus Mickey in an ASIC implementation may yield better results when compared to the other small ciphers. The same could be said with the F-FCSR family of ciphers. Moustique was of medium-to-large area with a less than one ratio of throughput/area from our design. Moustique was the only self-synchronizing cipher so this should be mentioned in the comparison. Pomaranch was the slowest design and yielded a high area. An implementation using a lookup table of the S-Box was faster (68 MHz) but also larger (1155 slices)[19].

## VII. CONCLUSION

Multimedia information transmission like high quality videos and color still images requires high speed processor for fast processing and transmission over the communication channels. As a result, designing a high speed processing security algorithm has become a challenging issue for the portable computing applications. As a challenge, the proposed YUGAM-128 stream cipher is designed in a simple manner with mere shift registers, whose basic element is flip-flops, XOR and CA functions. This promises efficient implementation in reconfigurable FPGA with high throughput owing to parallelism nature. Hence, the algorithm suits well for the portable computing devices that are facilitated with GHz processors.

## REFERENCES

[1] P.P. Chaudhuri, D.R. Chowdhury, S. Nandi, and S. Chattopadhyay, "Additive Cellular Automata: Theory and Applications", vol. 1. Los Alamitos, Calif.: IEEE CS Press, 1997.

[2] S. Nandi, B.K. Kar, and P.P. Chaudhuri, "Theory and Application of Cellular Automata in Cryptography", IEEE Trans. Computers, vol. 43, pp. 1,346-1,357, 1994.

[3] Bouganim.L and Guo.Y, "Database encryption," in Encyclopedia of Cryptography and Security. *Springer*, 2010, 2nd Edition.

[4] Carlet.C, Dalai.D.K, Gupta.K.C and Maitra.S, "Algebraic Immunity for Cryptographically Significant Boolean Functions: Analysis and Constraction," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 3105-3121, 2006.

[5] Coppersnith D, Halevi S, Lutla C.S. "Cryptanalysis of stream cipher with linear masking." In Yung M, eds. Advances in Cryptology-Crypto 2002. LNCS 2442, Berlin: Springer-Verlag, 2002. 515-532.

[6] Douglas A. Pucknell and Kamran Eshraghian, "Basic VLSI design", 3rd Edition, Prentice Hall of India, 2004. pp. 118-274.

[7] Ekdahl "On LFSR Based Stream Ciphers (Analysis and Design)," Ph.D. Thesis, Lund Univ. (November 2003).

[8] Gammel B.M, Gottfert.R and Kniffler.O, "An NLFSR-based stream cipher," in ISCAS, 2006.

[9] Good.T, and Benaissa.M, "ASIC hardware performance," New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986, pp. 267–293, 2008.

[10] Grocholewska-Czurylo, " Random generation of Boolean Function with high degree of correlation immunity," Journal of Telecommunication and Information Technology, pp. 14-18, 2006.

[11] Ju Young KIM and Hong Yeop SONG "A Nonlinear Boolean Function With Good Algebraic Immunity "IEEE Proceeding Of IWSDA '07, 2007, pp. 94-98.

[12] Kitsos, Sklavos.N, Papadomanolakis.K and Koufopavlou.K, "Hardware Implementation of Bluetooth Security", IEEE Pervasive Computing, vol. 2, no.1, pp. 21-29, January-March 2003.

[13] Maximov, "Some Words on Cryptanalysis of Stream Ciphers," Ph.D. dissertation, Lund Univ., Lund, Sweden, 2006.

[14] Menezes.A, van Oorschot.P, and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996. pp. 482-504.

[15] Paris Kitsos, "OntheHardwareImplementationofthe MICKEY-128 Stream Cipher," eSTREAM, ECRYPTStreamCipher Project, Report 2006/059, 2006.

[16] Rukhin, Soto, Nechvatal, Smid, Barker, Leigh, Levenson, Vangel, Banks, Heckert, Dray, VO, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." NIST Special Publication 800-22, May 15, 2001, 1–153.

[17] Rizomiliotis.P, "On the Resistance of Boolean Functions Against Algebraic Attacks Using Univariate Polynomial Representation," IEEE Trans. Inf. Theory, vol.56, no. 8, pp. 4014-4024, 2010.

[18] Rose.G.G and Hawkes.G Turing "A Fast Stream Cipher" In Fast Software Encryption FSE 2003, pages 290-306. Springer-Verlag, 2003.

[19] Hwang, David, Mark Chaney, Shashi Karanam, Nick Ton, and Kris Gaj. "Comparison of FPGA-targeted hardware implementations of eSTREAM stream cipher candidates." The State of the Art of Stream Ciphers (2008): 151-162.