

A PROPOSED DESIGN OF A STREAM CIPHER MODEL BASED ON FEEDBACK WITH CARRY SHIFT REGISTERS

Nabil Hamdy Shaker, Amal N.Asham, Ihab Talkhan, Tarek.A. Abdel Aziz
Cairo University, Faculty of Engineering, Cairo, Egypt.
Misr International Univ., Faculty of Engineering, Cairo, Egypt. nabil.hamdy@
miegypt.edu.eg , amalasham@hotmail.com, italkhan@cu.edu.eg

ABSTRACT

Stream ciphers are often used when it is necessary to encrypt large amounts of data very quickly; they are private-key encryption algorithms that operate on the plaintext one bit or one machine word at a time. Stream ciphers are considered secure if knowledge of a small number of bits of keystream cannot be used to recover the entire keystream. A good stream cipher should have good randomness, high period, linear span, and security against any known attack. Hence, the development of a good random number generator has been a hot topic in cryptology. Cryptographers have liked stream ciphers made up of shift registers: They are easily implemented in digital hardware. The Feedback with carry shift register (FCSR) is introduced as a source for long pseudo random binary sequences. Therefore, this paper provides the design of a new stream cipher model, it is an FCSR based stream cipher model, main building blocks and modules of the cipher system model are presented in details. The design considerations are provided. The initialization of the new cipher model and the operational scenario of the new model as a whole are introduced. Finally, the new cipher model is being implemented using VHDL.

Keywords: Stream ciphers, feedback shift registers, FCSR.

1 INTRODUCTION

In the stream cipher, the *running-key* (or *keystream*) is the sequence which is combined digit by digit to the plaintext sequence for obtaining the ciphertext sequence [1], [2]. The running key is generated by a finite state automaton called the *running-key generator* or the *keystream generator*. Stream ciphers based on shift registers have been the workhorse for cryptography since the beginnings of electronics [3], [4]. Shift registers [5], [6] are easily implemented in digital hardware. The simplest kind of feedback shift registers is the Linear Feedback Shift Register (LFSR) which is used for generating a wide variety of pseudo random sequences. The size of the smallest LFSR that generates a given periodic sequence is defined as the *linear complexity (span)* of a , it is an important measure of the cryptographic security of the sequence. Due to the linearity of LFSR, we can determine the LFSR which generates any output sequence using the Berlekamp–Massey algorithm [7] by knowing $2n$ output bits only.

Klapper and Goresky [8],[9] proposed a new type of pseudo random binary sequence generator called Feedback with Carry Shift Register (FCSR), FCSR has a shift register, feedback function, and a small amount of memory cells (to store the carry). The bits of the register are added together with the current contents of the memory to form sum. The parity bit ($\Sigma \text{ mod } 2$) of the sum is fed back into the first cell,

and the higher order bits ($\lfloor \Sigma / 2 \rfloor$) (where $\lfloor \cdot \rfloor$ denotes the floor or integer part) are retained for the new value of the memory. Fig.1 shows the main components of the Fibonacci structure of FCSR [10].

The main parameters related to the analysis of FCSR are: the connection integer " q ", the number of register cells. The connection integer is an odd positive integer $q \in \mathbb{Z}$, suppose $q = p^e$ with p an odd prime where 2 is primitive modulo p , and p^2 does not divide $(2^{p-1} - 1)$ and $e \geq 2$, the output period is given by:

$$\phi(q) = p^e - p^{e-1} \tag{1}$$

The relation between the connection integer and the number of register cells r [11] is given by:

$$r = \lfloor \log_2(q+1) \rfloor \tag{2}$$

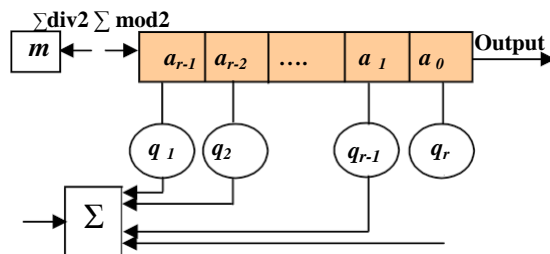


Figure 1: Fibonacci FCSR

FCSR has many properties that are similar to the properties of LFSR; it is equipped with a rich algebraic structure that enables its analysis in much the same way as the algebraic structure of LFSR. However, the analysis of FCSR sequences involves a completely different mathematical toolkit. The constructed sequences are balanced and have the de-Bruijn property which states that: in any single period, every nonzero binary string of length r (where r is the register length) occurs exactly once. In the other side, FCSR sequences can be synthesized analogue of the Berlekamp-Massey algorithm. As well as, it has been shown that the linear span of an FCSR is half of the period [12]. Consequently, an FCSR has much better linear span compared to that of an LFSR. The notion of the 2-adic complexity of the keystream is also an important measure of the security of a stream cipher [13], it is possible to increase the 2-adic complexity of FCSR sequences by introducing suitable boolean functions in the output of the FCSRs but this question has not received serious attention so far. Thus, the purpose of our paper is to explore the possibility to design a stream cipher model using FCSRs as our blocks for keystream generation [14], [15], [16].

2 THE PROPOSED STREAM CIPHER MODEL

The proposed model is a synchronous stream cipher based on FCSRs as the main building blocks, and a boolean function to combine the outputs of these FCSRs. The cipher system model is used as a keystream generator that outputs a pseudo random sequence of bits. Every output 8-bits of the keystream is XOR'ed with an equivalent byte of the plaintext data to produce the encrypted cipher data. Upon the receiving of the cipher data, it is XOR'ed with the same keystream bytes to recover the original plain data. The detailed structure of the proposed key stream generator is illustrated in Fig.2. Next, we will demonstrate the detailed internal structure and the design strategy of the model.

2.1 FCSRs Module

The FCSRs module uses 8-FCSRs as the main building blocks. For each FCSR we should determine the fixed parameters p , e , q , and r . Every time the model works one randomly c value (where c is an integer less than q , and coprime to p) for each FCSR is chosen. For every FCSR the fixed parameters are chosen as follow:

FCSR1 ($P1=11$, $e1=9$, $q1=2.3579e+009$, $r1=31$).
FCSR2 ($P2=13$, $e2=10$, $q2=1.3786e+011$, $r2=37$).
FCSR3 ($P3=19$, $e3=30$, $q3=2.3047e+038$, $r3=127$).
FCSR4 ($P4=11$, $e4=31$, $q4=1.9194e+032$, $r4=107$).
FCSR5 ($P5=3$, $e5=35$, $q5=5.0032e+016$, $r5=55$).
FCSR6 ($P6=13$, $e6=19$, $q6=1.4619e+021$, $r6=70$).

FCSR7 ($P7=61$, $e7=14$, $q7=9.8768e+024$, $r7=83$).
FCSR8 ($P8=61$, $e8=17$, $q8=2.2419e+030$, $r8=100$).

2.2 FCSRs' c Values and Initial States

Every time the model works; random c value for each FCSR should be chosen. Then, the model initializes the FCSRs' cells by calculating an initial state by the following procedure:

PROCEDURE INITIAL STATE (INTRGER p,r , $binq(,q)$)

```
// Get a random value for c where c coprime to p
1. SET index=q-1.
2. SET findc=false.
//Generates discrete uniform random number
3. WHILE findc=false.
  3.1 c=unidrnd(index)
  3.2 IF greatest common divisor of (c,p) =1
    Then findc=true
  ENDIF
ENDWHILE
4. DISPLAY c
// Get initial loading and initial memory
5. FOR i = 1 TO r
  5.1 IF i=1
    5.1.1 SET sum = c
  ELSE
    5.1.1 SET sum = 0
    5.1.2 FOR k=1 TO i-1
      5.1.2.1 sum + (binq(i-k)*a(k))
    ENDFOR
    5.1.3 sum=sum + m
  ENDIF
  5.2 SET a(i)= mod(sum,2)
  5.3 m=(sum - a(i))/2
ENDFOR
6. RETURN a , m
END PROCEDURE
```

2.3 FCSRs' Periods

The chosen FCSRs produce ℓ -sequences with period as follow:

$$\Phi(p^e)=p^{e-1}(p-1)=q(p-1)/p \quad (3)$$

which follows that they produce sequences with periods :

period1 = 2.1436e+009, period 2 = 1.2725e+011,
 period 3 = 2.1834e+038, period 4 = 1.7449e+032,
 period 5 = 3.3354e+016, period 6 = 1.3495e+021,
 period 7 = 9.7149e+024, period 8 = 2.2051e+030.

2.4 FCSRs' Tapped Cells

The operation of each FCSR begins by adding the tapped cells to form the sum σ .

$$\sigma_n = \sum_{i=0}^{r-1} q_i a_{n-i} \quad (4)$$

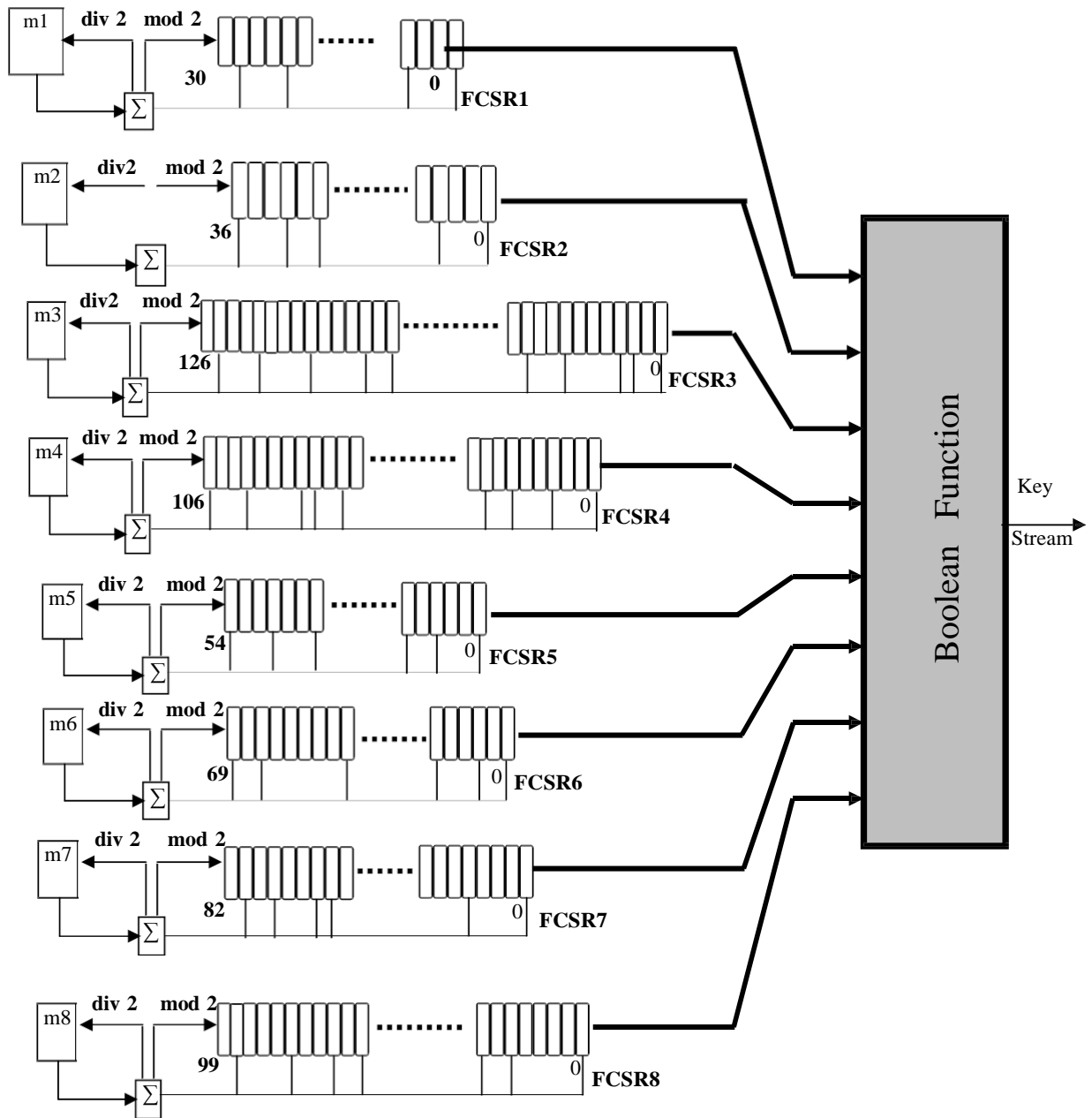


Figure 2: The detailed internal structure and the design strategy of the model.

Where the tapped cells are the following tap positions in each FCSR:

FCSR1= [31, 27, 26, 23, 19, 17, 16, 14, 13, 11, 10, 8, 5, 3, 1].

FCSR2 = [37, 28, 27, 24, 16, 15, 13, 10, 8, 7, 6, 3].

FCSR3= [127, 125, 123, 122, 120, 118, 117, 113, 110, 104, 103, 99, 98, 96, 92, 90, 87, 86, 85, 83, 81, 79.]

FCSR4= [107, 104, 102, 101, 100, 98, 97, 95, 93, 91, 89, 87, 85, 82, 81, 80, 79, 78, 76, 74, 72, 65, 64, 60, 57, 56].

FCSR5= [55, 53, 52, 48, 47, 45, 44, 43,42, 41, 40, 39, 38, 36, 34, 33, 31, 30, 25, 24, 22, 21, 18, 17, 16].

FCSR6= [70, 67, 66, 65, 64, 62, 53, 52, 51, 45, 44, 42, 41, 39, 33, 31, 30, 26, 22, 21].

FCSR7= [83, 77, 75, 73, 72, 70, 69, 68, 67, 66, 65, 62, 61, 58, 57, 55, 52, 51, 50, 49, 47, 46, 42, 41, 39, 38].

FCSR8= [100, 99, 98, 94, 91, 89, 88, 87, 86, 85, 84, 83, 77,76, 74, 73, 72, 68, 67, 66, 63, 62, 57, 56, 53,52, 51, 49, 48, 36, 35,33, 32, 31].

2.5 Boolean Function Used

We added to the cipher system design is an 8-bit variable combining Boolean function f_d of dimensionality (256x1).the Boolean function we used has been chosen to be balanced and highly nonlinear. The truth table of the Boolean function in hexadecimal format is given as follows:

6F4FC635EE280B7135159C4BB472512B
CA8A932DD2E4A84D90D0C977CABEF217

3 OPERATIONAL SCENARIO OF THE PROPOSED MODEL

After the previous details, we will form the whole scenario of our proposed model. The system's operation starts by the keystream generator initialization process at the beginning of every new call setup or data transmission session. Then the system continues to repeat a group of sequential steps to generating an output stream of bits. The steps to produce one output keystream byte processed as follows:

1) Each FCSR operates the following steps to produce 1-bit

- form the sum $\sigma_n = \sum_{i=0}^{r-1} q_i a_{n-i} .$

- Output the rightmost bit a_0 by shifting the FCSR one right step.
- The parity bit $\sigma \pmod 2$ is feedback into the first cell of the FCSR (leftmost cell a_r).
- The higher order bits $\lfloor \sigma / 2 \rfloor$ are retained as a new memory value.

2) The output bits from the FCSRs' are used as inputs to the Boolean function f_d to output a 1-bit.

3) Repeat the steps (1), (2) eight times to produce one output keystream bytes.

The flowchart of the key Generator operational scenario is shown in Fig.3.

4 THE PERIOD OF THE OUTPUT SEQUENCE

The feedback function of the FCSR is highly nonlinear, and hence FCSR sequences are resistant to linear attacks such as the Berlekamp-Massey algorithm. They state that therefore linear functions are adequate to mask the 2-adic structure of the FCSR and to protect against 2-adic attacks such as de Weger's algorithm. Further, using a simple Boolean function as shown previously to combine the FCSRs provide some measure of immunity against certain correlation attacks, beside the simplicity to implement .The period of the combining FCSR

sequences using the Boolean function yields a sequence whose period, is approximately the product of the individual FCSR sequences[15]. For the proposed stream cipher model the observed period is given as follows:

$Total\ Period = 1.0021e+184$

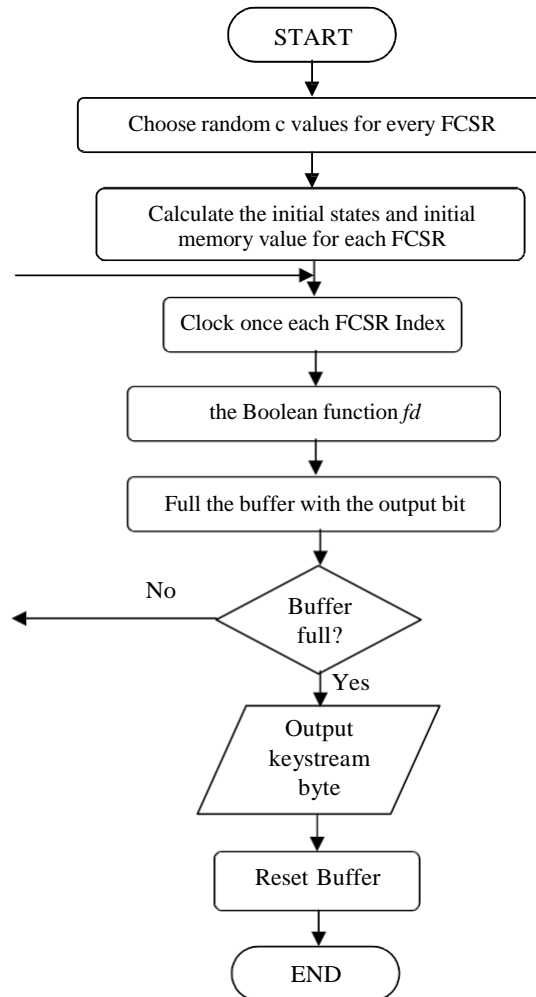


Figure 3: The Flowchart of the Key Generator operational scenario

5 VHDL SIMULATIONS

The proposed stream cipher model is represented by blocks and re-usable components connected by signals, buses or bundles, then, each block is defined using HDL text views that defines the behavior of each component design unit. Fig.4 illustrates the block diagram of the proposed stream cipher model. To illustrate the functionality of the designed stream cipher model the designed test bench has been run and the available signal was applied as the input with clock period equals to 200 ns. Fig.5 illustrates the

simulation results of the FCSRs' model. We have implemented the design using Xilinx FPGA XC4000XV, device 40110XVHQ240, which contains 4000k gates. Fig.6 shows the longest path (critical path) in the FCSR's model. Besides, the RTL schematic is provided in Fig.7. Finally, the Report Area and Report Delay as follows:

Device Utilization for 40110XVHQ240
 ***** Resource
 Used Avail Utilization

Used	Avail	Utilization
IOs	11 178	6.18%
FG Function Generators	593 192	7.24%
H Function Generators	35 4096	0.85%
CLB Flip Flops	677 8192	8.26%

Data required time (default specified - setup time)
 199.70

Data required time 199.70
 Data arrival time 137.10

slack 62.60

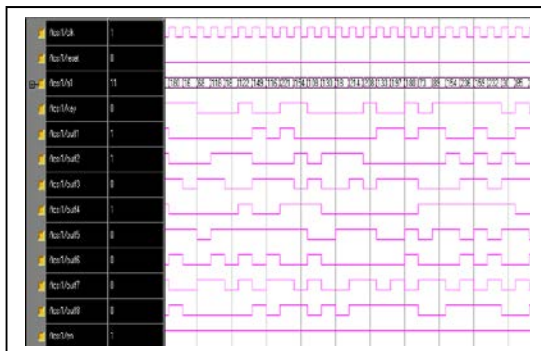


Figure 5: Simulation Results of FCSRs' Model

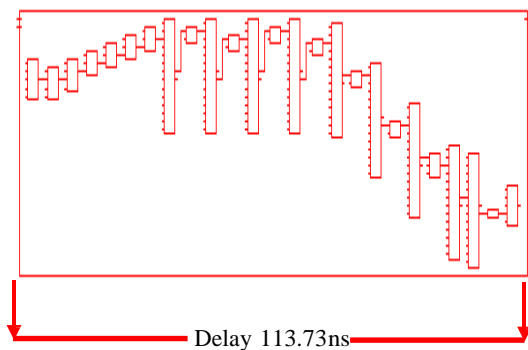


Figure 6: Critical path of the proposed stream cipher model

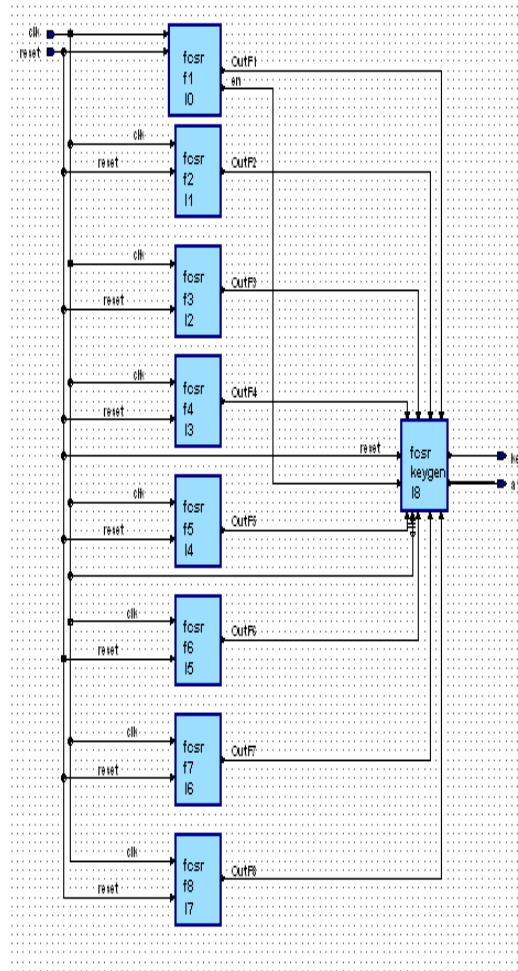


Figure 4: The VHDL block diagram of the proposed stream cipher model

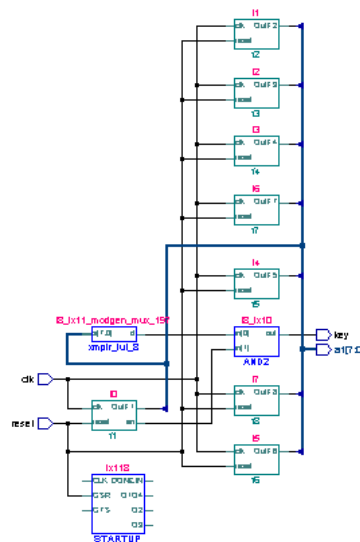


Figure 7: RTL Schematic

6. RANDOMNESS STATISTICAL PROPERTIES

We applied the five basic standard statistical tests [FIPS]: Frequency test, serial test, poker test, runs test, and autocorrelation test to the samples of the generated sequences. 100 sample sequences were used, each sample has 500Kbits for level of significance $\alpha=0.05$, and $\alpha=0.01$ respectively.

Hardware specifications: we applied these tests at Intel Pentium processor 1.86 GHz, 782 MHz, 504MB of RAM. MATLAB Release 14 with Service Pack 1 (R14SP1)

In case of $\alpha =0.05$ the tested samples passed as follows:

- 100 samples passed the frequency test, and 0 samples failed.
- 100 samples passed the serial test, and 0 samples failed.
- 98 samples passed the poker test, and 1 sample failed.
- 100 samples passed the runs test, and 0 samples failed.
- 100 samples passed the autocorrelation test, and 0 samples failed.

In case of $\alpha =0.01$ the tested samples passed as follows:

- 100 samples passed the frequency test, and 0 samples failed.
- 100 samples passed the serial test, and 0 samples failed.
- 100 samples passed the poker test, and 0 samples failed.
- 100 samples passed the runs test, and 0 samples failed.
- 100 samples passed the autocorrelation test, and 0 samples failed.

7. CONCLUSIONS

Stream ciphers are an important class of encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time. What makes them useful is the fact that the encryption transformation can change for each symbol of plaintext being encrypted. In situations, where transmission errors are highly probable, stream ciphers are advantageous because they have no error propagation. They can also be used when the data must be processed one symbol at a time (e.g., if the equipment has no memory or buffering of data is limited). A stream cipher attempts to capture the spirit of the one-time pad by using a short key to generate the keystream which appears to be random. Such a keystream sequence is often described as *pseudorandom sequences*, and deciding what constitutes a pseudorandom sequence

forms much of the work in the field of stream ciphers.

In this paper, a proposed stream cipher algorithm based on the FCSR architecture has been proposed. It provided the detailed description of the model design with the necessary considerations for the model components. The proposed stream cipher model consists of 8-FCSRs with different lengths as well as different initial states and initial memories values. A boolean function is used to combine the outputs of these FCSRs. The proposed stream cipher algorithm is used as a keystream generator that outputs a pseudo random sequence of bits. The model initialization process, total output cycle length, functional and operational scenarios are provided. Next, the recommended statistical tests are being applied to 100 samples of the generated keystream, each sample has a length of 500kb, to determine how secure the proposed cipher model is. As a final step, the hardware implementation of the proposed model using VHDL coding is introduced as follows: the block diagram of the proposed model, the simulation results, RTL schematic and critical path to provide the synthesis results.

8 REFERENCES

- [1] M.J.B.Robshaw,"Stream Ciphers", RSA Laboratories Technical Report TR-701, Version 2.0.July 25, 1995.
- [2] "Applied Cryptography", Bruce Schneier, Wiley & Sons, 1997.
- [3] A.Menezes, P.van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.
- [4] "Data & Computer Communications", William Stallings, IEEE Press, 2002.
- [5] R.A.Rueppel, "Analysis and Design of Stream Ciphers", Springer Verlag Berlin, 1986.
- [6] S. Golomb, "Shift Register Sequences", Laguna Hills, CA: Aegean Park Press, 1982.
- [7] J.L. Massey, "Shift Register Synthesis and BCH Decoding," IEEE Trans. Inform. Theory, Vol. IT-15, pp. 122-127, 1969.
- [8] A. Klapper and M. Goresky."2-adic shift registers in Fast Software Encryption", Cambridge Security Workshop, Lecture Notes in Computer Science, volume 809. Springer-Verlag, December 1993.
- [9] A. Klapper and M. Goresky." Feedback shift registers, 2-adic span and combiners with memory". Journal of Cryptology, 10:111-147, 1997.
- [10] Mark Goresky and Andrew Klapper, "Fibonacci and galois representations of feedback-with-carry shift registers", IEEE Transactions on Information Theory, 48(11):2826-2836, 2002.
- [11] M. Goresky and A. Klapper." Large period nearly de Bruijn FCSR sequences". In Advances in Cryptology EURO-CRYPT'95, Lecture Notes in Computer Science, volume 921, pages 263-273. Springer, NewYork, 1995.
- [12] Changho Seo, Sangjin Lee, Yeoulouk Sung, Keunhee Han, and Sangchoon Kim , "A Lower Bound on the Linear Span of an FCSR",IEEE Transaction on Information Theory, VOL. 46, NO. 2 MARCH 2000.
- [13] B. M. M. de Weger. "Approximation lattices of p-adic numbers". Journal of Number Theory, 24:70-88, 1986.
- [14] J. Xu. "Stream Cipher Analysis Based on FCSRs". Ph.D. dissertation, University of Kentucky, Lexington, Kentucky, 2000.
- [15] S.Anand, Gurumurthi V. Ramanan. "Periodicity, Complementary and Complexity of 2-adic FCSR Combiner Generators", ASIACCS '06, March 21-24, 2006, Taipei, Taiwan Copyright 2006 ACM 1-59593-272-0/06/0003.
- [16] M. Mittelbach and A. Finger. "Investigation of FCSR-based pseudo-random sequence generators for stream ciphers". In Proceedings of the 3rd. International Conference on Networking, February 2004.