International Journal of Reconfigurable and Embedded Systems
Vol. 7, No. 3, November 2018, pp. 186~194
ISSN: 2089-4864, DOI: 10.11591/ijres.v7.i3.pp186-194

Design and Implementation of LCG-Trivium Key Stream Generator into FPGA

Tchahou Tchendjeu A. E.[1], Tchitnga Robert[2], Fotsin Hilaire B.[3]
[1] Department of Electrical and Power Engineering, University of Bamenda, Cameroon
[2,3]Department of Physics, Universityof Dschang, Cameroon

Article Info

Article history:

Received Aug 7, 2018
Revised Oct 8, 2018
Accepted Oct 22, 2018

Keywords:

Cryptography
FPGA
Key stream
Linear congruential
Trivium

ABSTRACT

This paper presents the Design and implementation into Field Programmable Gate Array (FPGA) of a combine stream cipher and a simple linear congruential generator circuit to produce key stream. The LCG circuit is used to produce initialization vector (IV) each 2^{64} clock cycle to the cipher trivium in other to strengthen the complexity of the cipher to known attacks on trivium. The LCGTrivium is designed to generate 2^{144} bits of keystream from an 80-bits secret and a variable 80-bits initial value. To implement the LCG-Trivium on FPGA, we use VHDL to build a simple LCG and Trivium and a state machine to synchronize the functioning of the LCG and Trivium. The number of gates, memory and speed requirement on FPGA is giving after analysis. The design is simulated, synthesized and implemented in Quartus II 10.1, ModelSim-Altera 6.5 and Cyclone IV E EP4CE115F29C7N.

Corresponding Author:

Tchahou Tchendjeu A. E.,
Department of Electrical and power Engineering,
University of Bamenda,
P.O. Box 39 Bambili, Cameroon.
+237 677103754.
Email: tchahoutchendjeu@yahoo.fr

1. INTRODUCTION

Cryptography provides tchniques, mechanisms and tools for secure private communication and authentication on Internet and other open networks. It is almost certain that in the coming years every bit of information owing through a network of any kind will have to be encrypted and decrypted. All devices connected to a network should therefore incorporate mechanisms that implement cryptographic function to ensure safe transfers. With this in mind, it is necessary to design and implement hardware structures which are suitably efficient in terms of area, operating frequency and power consumption. An additional challenge is that the implementation must be constructed to withstand cryptographic attacks launched against them by adversaries who may have access to communication channels.

Trivium is a synchronous stream cipher designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initial value (IV). Algebraic attacks [1] are commonly applied to stream ciphers based on shift registers. To attack Trivium, Raddum [2] used an algebraic relabeling technique, where the state-update bits are represented using new variables, instead of nonlinear combinations of initial state bits [3]. This prevents equations of high degrees from being generated. For key stream generators which use a linear output function (as Trivium-like cipher do), Berbain et, al.[4] expressed new feedback bits of a Non-Linear Feedback Shift Register (NLFSR) as linear combinations of key stream bits and internal state bits.

From this, we see that, the attacks on Trivium build a system of Boolean equations with unknown and the solution of the system of equations is used to recover the secret key. In the present paper, we have chosen to combine a linear congruential generator with the Trivium to generate up to 2^{144} bits of key stream to reinforce the complexity of the system of Boolean equations to be generated during the algebraic attacks.

The remainder of the paper is organized as follows. Section 2 deals with theory of LCG and Trivium specification algorithm. The design circuit of LCG-Trivium for FPGA implementation and nets connections are covered in section 3. Section 4 provides some implementation data and analysis. Finally, the conclusions are viewed in section 5.

2. LCG AND TRIVIUM SPECIFICATION

2.1. Linear Congruential Generator

There is a popular and most used method to generate random number called linear congruential generator. The idea was introduced by Lehmer [5] according to sequential formula in equation (1), where *a* is the multiplier, *c* the increment factor, and *m* the modulus. Parameters *a, c* and *m* have to be chosen carefully in order to avoid repetition of similar numbers before *m* [6-8]. Park & Miller suggested a good results will be obtained by choosing c=0 [9]. The modulus *m* should be a large prime integer, multiplier *a* must be an integer in the range *2, 3 … m-1*. The cycle length of LCG would never exceed modulus *m*, but it could be maximized using three following conditions [7, 9, 10]:

a. *c is relativily prime to modulus m,*
b. *multiplier a-1 is a multiple of every dividing modulus m,*
c. *multiplier a-1 is a multiple of four when modulus m is a multiple of four too.*

$$X_{n+1} = (aX_n + c) \bmod m \tag{1}$$

2.2. Trivium

As shown in Figure 1, at the heart of Trivium are three shift registers, A, B and C. The lengths of the registers are 93, 84 and 111, respectively. The XOR-sum of all three register outputs forms the key stream S_i. A specific feature of the cipher is that the output of each register is connected to the input of another register. Thus, the registers are arranged in circle-like configuration.

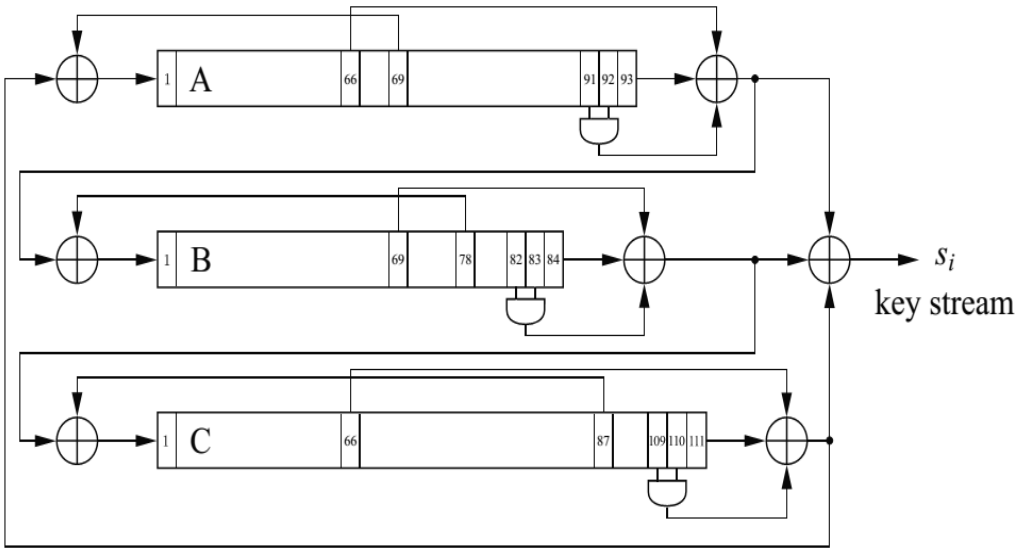

Figure 1. Internal structure of the stream cipher Trivium

The cipher can be viewed as consisting of one circular register with a total length of 93 +84 +111 = 288. Each of the three registers has similar structure as described below.
The input of each register is computed as the XOR-sum of two bits:

a. The output bit of another register according to Fig. 1
b. One register bit at a specific location is fed back to the input

The positions are giving in Table 1. The output of each register is computed as the XOR-sum of three bits:

a. The rightmost register bit,
b. One register bit at a specific location is fed forward to the output. The positions are given in Table 1,

c. The output of a logical AND function whose input is two specific register bits. Again, the positions of the AND gate inputs are given in Table 1.

Note that the AND operation is equal to multiplication in modulo arithmetic. If we multiply two unknowns, and the register contents are the unknowns that an attacker wants to recover, the resulting equations are no longer linear as they contain products of two unknowns. Thus, the feed forward paths involving the AND operation are crucial for the security of Trivium as they prevent attacks that exploit the linearity of the cipher.

Table 1. Specification of Trivium

	Register length	Feedback bit	Feed forward bit	AND input
A	93	69	66	91; 92
B	84	78	69	82; 83
C	111	87	66	109; 110

3. LCG-TRIVIUM CIRCUIT DESIGN

3.1. Design Circuit of LCG

Figure 2 shows the commonly used block diagram for a LCG operation (seed is ignored). It requires a multiplier, an adder, a comparator and a subtractor blocks. Multiplier is used to multiply previous random value X with a. The result is then added to the increment c. The sum is compared to modulus m. Thus, if $(aX_n + c) \leq m$, then the number is considered a random one. Otherwise, the next operation to undertake is a subtraction of m from the above number, so that the difference $(aX_n + c) - m$ becomes a random number.

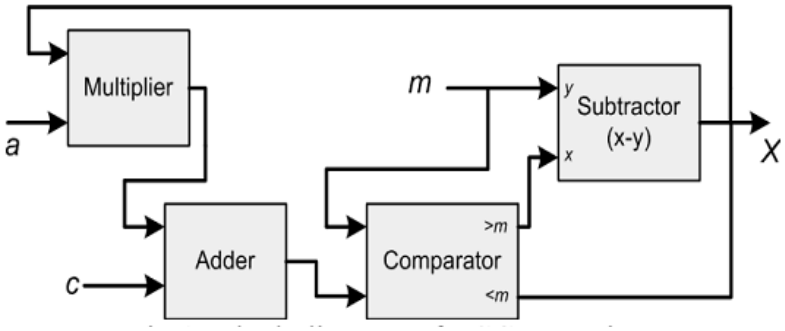

Figure 2. Block diagram of LCG operation

The block diagram of Figure 2 is involving arithmetic operation such as multiplication, addition, subtraction and comparion which make it cumbersome. In order to simplify the process, we suggest to replace the subtractor and comparison block simply by a register as shown in Figure 3.

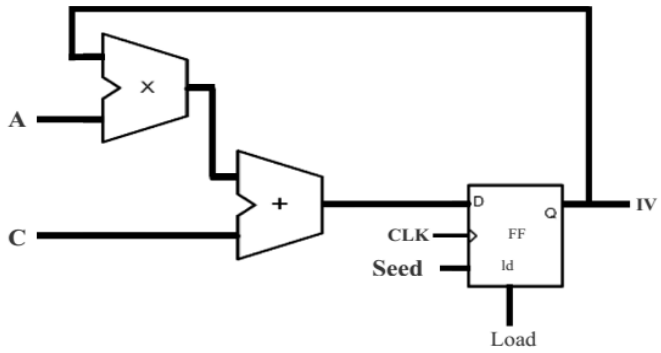

Figure 3. Design circuit of LCG

The design circuit consist of a multiplier, an adder, and a register. Port input *Seed, A,* and *C* are used to pass initial value, multiplier and increment into the circuit. Meanwhile, port *IV* is used to taking out the resulted random numbers.

The functioning process of the design is as follows, initially, signal load has to be HIGH to load the seed value at the output of the register, it also determine the start of the operation. The predefined value (seed) and increment have to be available at the input ports before load goes LOW. After that, each times clock goes HIGH, a random number is produced.

3.2. Design Circuit of Trivium

The design circuit of trivium Figure 4 consists of three registers (A, B, C) of different length, three AND gates seven XOR gate. Port input *IV* , *KEY* , and *7* are used to pass initial value, Initialization Vector, the encrypting key and a non-zero value into the registers. Meanwhile, port S_{out} is used to taking out the key stream.

The circuit is controlled by two signals ***load*** and *WE*. The functioning process of the design is as follows: initially, signal *load* has to be HIGH (WE = HIGH) to load the *IV* value at the output register A, the key value at the output register B and the value 7 at the output register C. It also determines the start of the operation. The predefined value *IV, KEY* and *7* have to be available at the input ports before *load* goes LOW (WE=HIGH). After that, each time the clock goes HIGH, a key stream is produced. Each time WE goes LOW while load remain LOW, the value *IV* is reloaded at the output of register A.

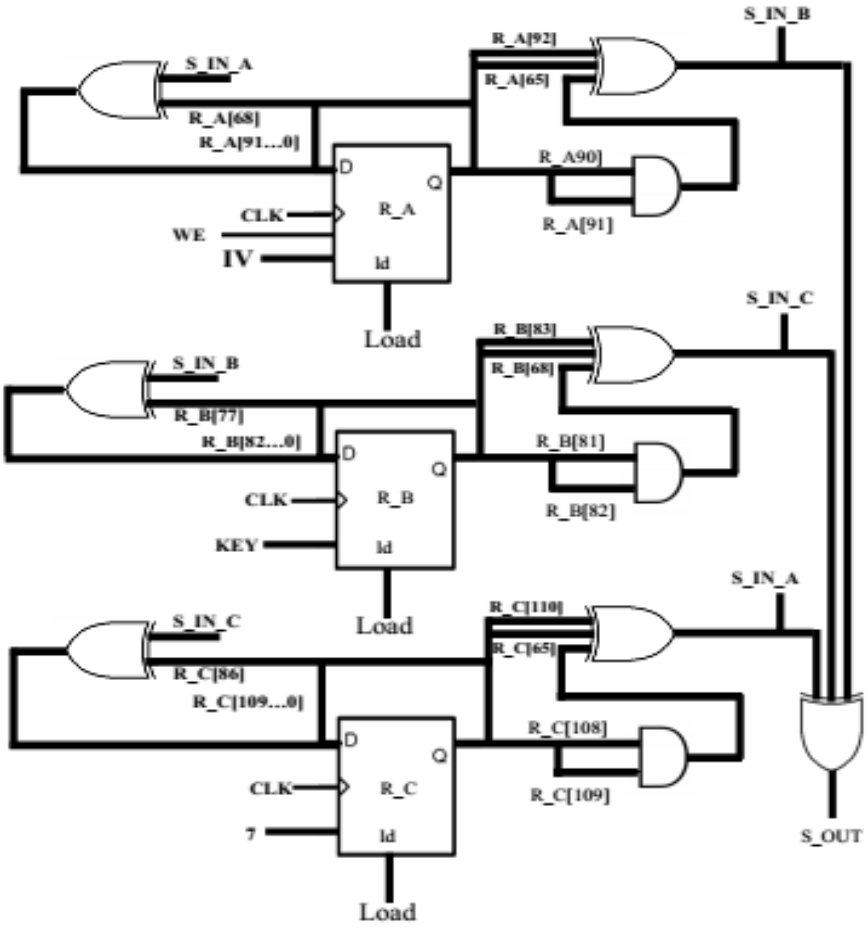

Figure 4. Design circuit of Trivium

3.3. LCG-Trivium Design

Figure 5 shows the diagram of the LCG-Trivium circuit. It is made up of the LCG and Trivium circuit design in this paper, and the CSM (Cipher State Machine). The CSM is designed according to encryption with Trivium.

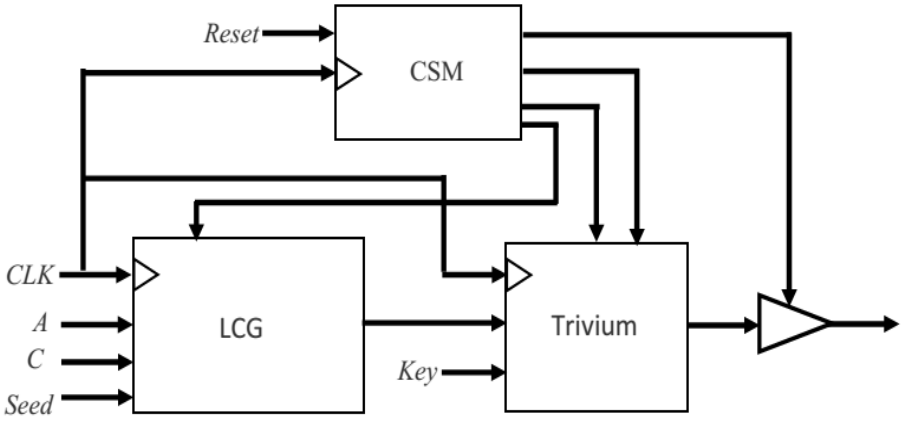

Figure 5. Block diagram circuit of LCG-Trivium

Let's look at the details of running Trivium:

Initialization. Initially, an 80-bit IV is loaded into the 80 leftmost locations of register A, and ann80-bit key is loaded in the 80 leftmost locations of register B. All other register bits are set to zero with the exception of the three rightmost bits of register C, i.e., bits C_{109}, C_{110}, and C_{111} which are set to 1.

Warm-up phase. In the first phase, the cipher clocked 4 x 288 = 1152 times. No cipher output is generated.

Encryption Phase. The bits produced hereafter, i.e., starting with the output bit of cycle 1153, form the key stream. The warm-up phase is needed for randomizing the cipher sufficiently. It makes sure that the key stream depends on both the key and the Initialization Vector. From the details of running Trivium, an ASM chart (Algorithmic State Machine chart) is develop and presented in Figure 6.

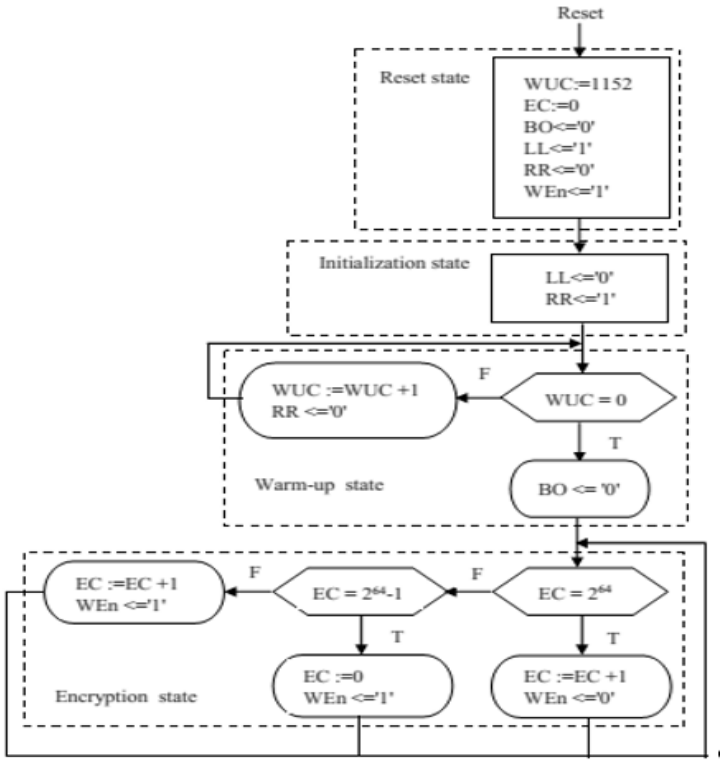

Figure 6. ASM chart of CSM

The ASMD has four states. The Reset state indicates that the circuit is in the pre-initialization, here, the initial value Seed is placed to the output register of LCG and the tristate buffer moved to high impedance, when Reset is asserted. When the next rising edge of clock is asserted, the FSMD moves to the initialization state and the value at the output of LCG and the value of Key are loaded into Trivium as IV and KEY respectively. At the next clock rising edge, the FSMD moves to warm-up state and the cipher is sufficiently randomized for 1152 clock cycles. When the cipher is sufficiently randomized, the FSMD moves to encryption state and the tristate buffer is enable, a key stream is generated for each clock cycle and a new value of IV is generated and loaded in the Trivium after each 2^{64} clock cycles.

4. SIMULATION RESULTS

The proposed block diagram of LCG-Trivium shown in Figure 5 is designed in Quartus II. 10.1 and ModelSim-Altera 6.5e. using VHDL-HDL and experimentally verified on FPGA DE2-115 Board. Some important informations of synthesis results are presented.

4.1. RTL Simulation Diagram

Figure 7 depicts the RTL technical schematic for the LCG-Trivium cipher. CSM receives two signals, clock and reset, and generates four signals. Signal BO control the tristate buffer to be sure that not cipher output is generated during the 1152 cycles after initialization. Signal LL is used to load initial values into LCG_TOP. Signal RR is used to load initial values into registers of Trivium. Signal WE enable LCG_TOP to generate a new random value and also used to load into register A of Trivium the new value of IV. LCG_TOP block receives six signals and generates one, which is the Initialization Vector. Trivium block receives five signals and generates one, which is the cipher output bit.

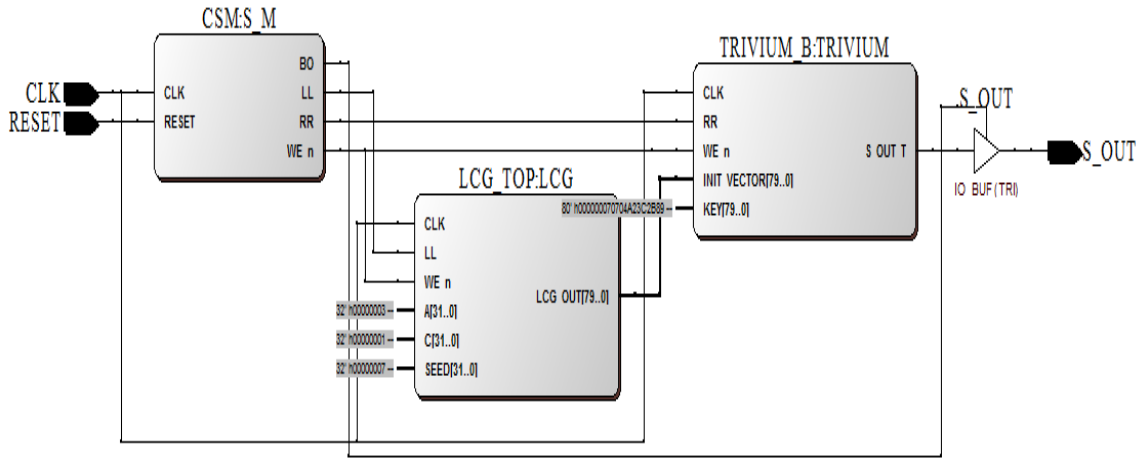

Figure 7. RTL schematic of LCG-Trivium

4.2. Behavior Simulations

Figure 8 to Figure 11 depict the results of behavior simulation of the design LCG-Trivium. Figure 8 shows the behavior of main signals of the design circuit in the Reset and Initialization phases. At the beginning of the Reset phase, only key, seed, increment and multiplier signals have known values while other are unknown, the key stream signal is at high impedance and reset signal is HIGH. When the reset signal goes to LOW, then at the next rising edge of clock signal, the LCG block generates the first random number and that number is the IV signal. With this, the system moves to the Initialization state. RR signal goes LOW, LL signal goes HIGH, BO goes LOW, the key stream signal remain to HIGH impedance and WE goes HIGH then the initial values are loaded into the registers of Trivium at the next rising edge of clock signal and the system moves to next state which is warm-up. Figure 9 shows the behavior of main signals in the warm-up phase. Here the key stream signal remain at HIGH impedance. Figure 10 shows the system in the encryption phase. Cipher output is generated every times clock goes HIGH. A new value of IV is also generated every times WE signal goes LOW. Figure 11 shows the randomizing of the cipher during the warm-up phase.

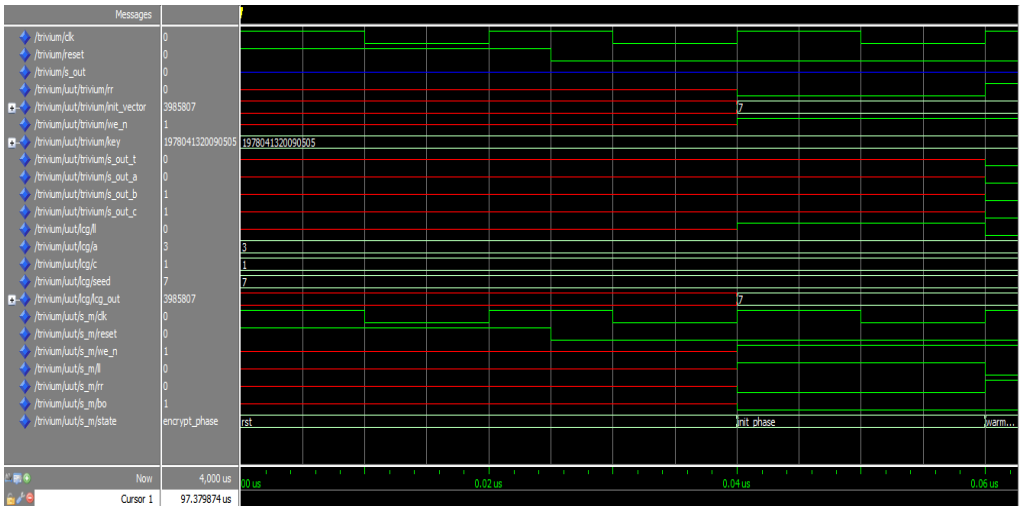

Figure 8. Simulation result of LCG-Trivium in reset and initialization phase

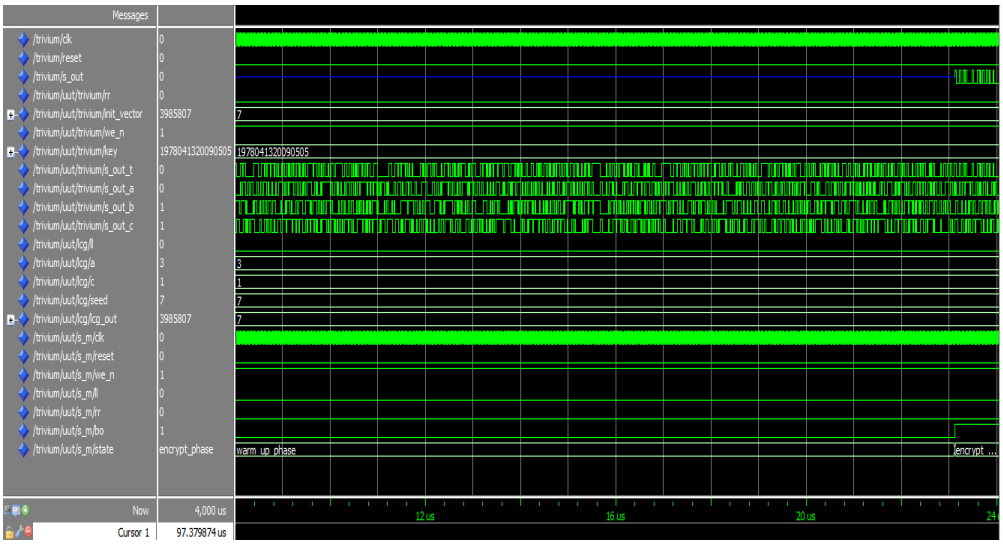

Figure 9. Simulation result of LCG-Trivium in warm-up phase

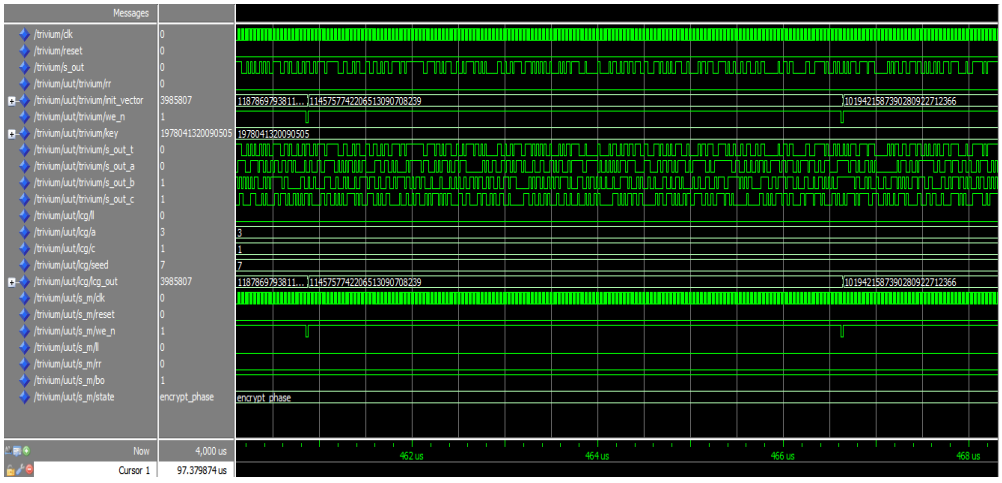

Figure 10. Simulation result of LCG-Trivium in encryption phase

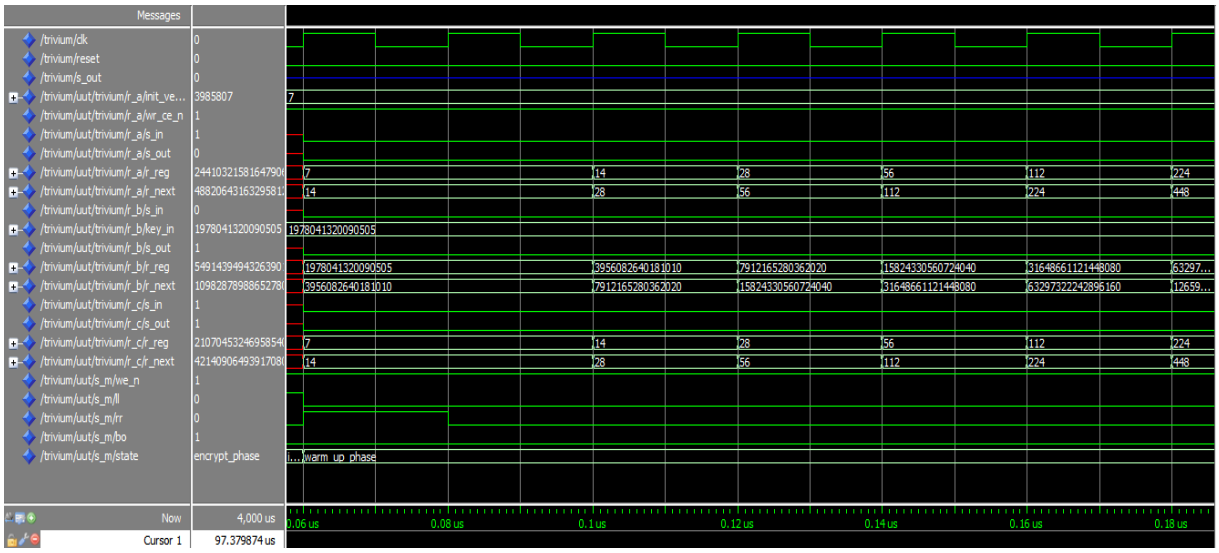

Figure 11. Simulation result of Trivium register's in warm-up phase

4.3. Synthesis Results

Some important data after synthesis step of the proposed LCG-Trivium design circuit into cyclone IV-E EP4CE115F29C7N chip are summarized as shown in Table 2. From the Flow Summary report, it can be seen that the LCG-Trivium circuit Figure 5 required 563 logic elements, 491 combinational functions, 301 dedicated logic registers, 95 memory bits and 9 embedded multiplier 9-bit elements. From the report paths, one path is found with the longest delay of 13.420 ns.

Table 2. Flow Summary Report

Flow Summary Report		
Total logic element	563/114,480	< 1%
Total combinational functions	491/114,480	< 1%
Dedicated logic registers	301/114,480	< 1%
Total pins	3/529	< 1%
Total memory bits	95/3,981,312	< 1%
Embedded Multiplier 9-bit elements	9/532	< 2%
Total PLLs	0/4	< 0%

5. CONCLUSION

FPGA implementation of LCG-Trivium has been designed and implemented on cyclone IV-E EP4CE115F29C7N chip successfully. The proposed circuit is a combination of LCG and Trivium to realize a stream cipher for cryptographic applications. A state machine has designed to add to the system in the objective to coordinate the functioning of the two block. LCG circuit generates random numbers after each 2^{64} cycles of clock and it is load into Trivium, while Trivium generates key stream for each clock cycle in the encryption phase as the ModelSim simulation depicts. With this, the cipher complexity is improved. The quartus simulation shows that, the area utilization is optimum. Based on the longest delay path, the designed circuit can run at a maximum frequency of 75 MHz.

ACKNOWLEDGEMENTS

Tchahou Tchendjeu thanks Dr Emmanuel FOUOTSA for allowing him to attend the 2016 African Mathematical School (AMS) at The University of Bamenda and the anonymous refrees for useful comments for the improvement of this work.

REFERENCES

[1] N. T. Courtois, J. Meier, "Algebraic Attacks on Stream Cipher with Linear Feedback," Advances in Cryptology-ASIACRYPT. Lecture Note in Computer Science, Springer, Vol. 2656, pp. 345-359, 2003.

[2] H. Raddum, "Cryptanalytic Results on Trivium," eTREAM, ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/papersdir/2006/039.pdf.
[3] N. T. Courtois, J. Pieprzyk, "Cryptanalysis of Block Cipher with Overdefined System of Equations," Advances in Cryptology-ASIACRYPT. Lecture Note in Computer Science, Springer, Vol. 5479, pp. 278299, 2002.
[4] C. Berbain, et al., "Algebraic and Correlation Attacks against Linearly Filtered Non Linear Feedback Shift Registers," Selected Area in Cryptography. Lecture Note in Computer Science, Springer, Vol. 5381, pp. 184-198, 2008.
[5] D. H. Lehmer, "Random Number Generation on the BRL High Speed Computing Machines," Math. Rev, Vol 15, pp. 559-560, 1954.
[6] N. Harald, "Random number generation and quasi-monte carlo methods. Society for Industrial and Applied Mathematics", Philadelphia.
[7] C. Dutang, W. Diethelm, "A note on random number generation," 2009
[8] Wolfram Mathematica "RANDOM NUMBER GENERATION," copyright Tutorial Collection 2008.
[9] S. K. Park, K. W. Miller, "Random number generators: good ones are hard to _nd," Association for Computing Machinery, Vol. 31, pp. 1192-2001, 1988.
[10] D. E. Knuth, "The Art of Computer Programming," seminumerical algorithms. Massachusetts: Addison-Wesley 2002.