# Design and Characteristics of LIZARD Stream Cipher IP Core

**Jinin K Jose[1], Tossy Thomas[2], Nandakumar R[3]**

*[1]M.Tech Student, Department of ECE, GEC Idukki, Kerala, India*
*[2]Assistant Professor, Department of ECE, GEC Idukki, Kerala, India*
*[3]Scientist /Engineer 'D' NIELIT Calicut, Kerala, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

*Abstract- Cryptography is the art of concealing information from the eaves droppers by means of a secret that is only known to the communicating parties. Wireless network security and protection of digital communication can be achieved by using stream ciphers. A stream cipher is a method of bit wise encryption of a text in which a cryptographic key and algorithm are applied to each binary digit in a data stream. This project is to study the light weight stream cipher LIZARD[1]. It uses a 120 bit key, 64 bit initialization vector and has a 121 bit inner state. It offers provable 2/3n security against Time Memory Data Tradeoff attacks and 80 bit security against key recovery attacks. Its a combination of grain like design with Fp(1) mode which results in its hardware efficiency. In this paper design of light weight stream cipher Lizard is discussed and its simulation is obtained on Xilinx ISE using verilog coding.Then it is implemented on Xilinx Spartan 6 FPGA (XC6SLX16-3CSG324).*

***Index Terms***—Cryptography, Light weight stream ciphers, Time Memory Data Trade off attacks, Model Sim.

## 1. INTRODUCTION

Study of techniques or the practice for secure communication in the presence of third parties is known as cryptography or cryptology. Cryptography is all about constructing and analyzing protocols that prevent third parties and the public from reading private messages. Various aspects in information security such as confidentiality, integrity, authentication, and non-repudiation are central to modern cryptography. Various applications of cryptography include electronic commerce, payment cards based on chips, Digital currencies, passwords, and also in military communications.

In a stream cipher especially in a synchronous stream cipher there is a keystream generator block that takes input i.e.a symmetric key and an initialization vector (IV).The key is intended to be a privately known quantity, and the IV is publicly known value. One keystream bit is generated in each running stage. For the encryption and decryption process the same keystream generated is used. The state initialization algorithm which computes the initial state from a given key/IV pair, is efficiently invertible for Trivium[12] and Grain v1[6].The secret key can be immediately revealed if the initial state is known. And even if the state initialization algorithm is not efficiently invertible, variants of Time Memory Data tradeoff attacks often allow for key recovery.

On the other hand LIZARD represents the instantiation of the FP(1)-mode which provides a security level of 2/3n w.r.t. generic TMD tradeoff attacks aiming at key recovery. The design of Lizard is closely related to that of the design of Grain v1, which was selected as the most hardware efficient member of the eSTREAM portfolio and, hence it is considered as a benchmark for new hardware-oriented designs. When compared to Grain v1, the major differences with Lizard are the smaller inner state (121 bit vs. 160 bit), the larger key (120 bit vs. 80 bit) and Lizard introduces the secret key not only once but twice during its state initialization process. All these modifications are a direct consequence of implementing the FP(1)-mode. Time-memory-data (TMD) tradeoff attacks restricts the security level of many previously available stream ciphers like E0[11], A5/1[8], Trivium[12] and Grain[16] to 1/2 n, where n represents the inner state length of the underlying keystream generator. Lizard is a hardware oriented stream cipher for power constrained devices like passively powered RFID tags.Lizard allows to generate up to $2^{18}$keystream bits per key/IV pair, which would be sufficient for many existing communication scenarios like Bluetooth, WLAN or HTTPS.The reminder of paper is organized as follows. Section II briefly explains about the design overview of LIZARD Stream cipher IP Core. In Section III, a detailed view of the experimental setup and the results obtained are given .Finally, the conclusion is drawn in Section IV.

## 2. DESIGN OVERVIEW

### 2.1. Block Diagram

LIZARD stream cipher IP core comprises of 120 bit key, 64 bit Initialization Vector and a plaintext as inputs and the corresponding ciphertext as output.

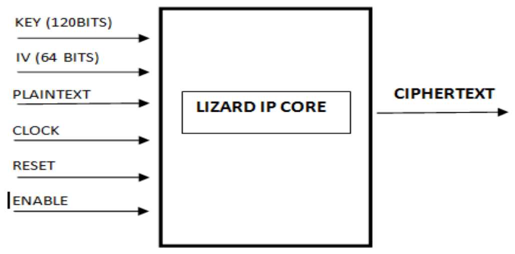


**Fig-1**: Basic Input/output diagram of LIZARD stream cipher

The Input/output diagram gives the black box representation of a design. Figure 1 shows the basic Input/Output diagram of LIZARD stream cipher IP core. The design has 6 inputs in which clock, reset and enable signal is 1 bit wide whereas key is 120 bit wide, IV is 64 bit wide and plaintext is 119 bit wide.

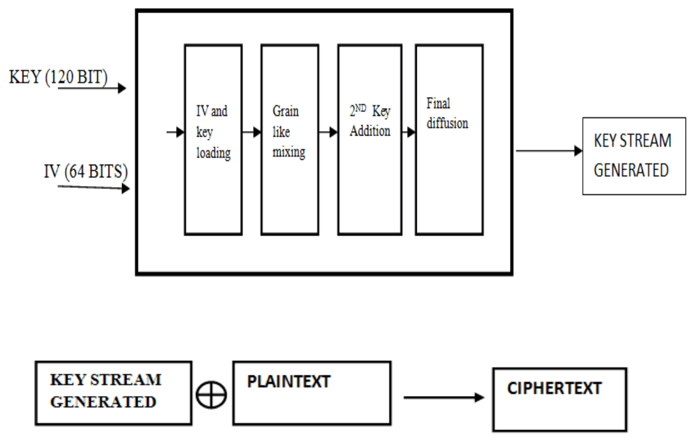


**Fig-2**: Detailed block diagram of LIZARD stream cipher IP core

Firstly the state initialization process takes place in 4 phases. After the completion of the 4th phase, key stream is generated. Then the generated keystream is XORed with the plaintext to obtain the cipher text. This process is called Encryption. Then the ciphertext is again XORed with the keystream to obtain the plaintext. This process is called decryption.

### 2.2. Components in LIZARD Stream cipher

The inner state of Lizard is distributed over two interconnected feedback shift registers (FSRs).Grain uses one linear feedback shift register (LFSR) and one nonlinear feedback shift register (NFSR), which are of the same length. Lizard uses two Non Linear Feedback Shift Registers (NFSRs) of different lengths. The 121-bit inner state of Lizard is distributed over two Non Linear Feedback Shift Registers (NFSRs), NFSR1 and NFSR2, whose contents at time, t=0, 1, 2, 3 ,4, 5, 6, 7, 8............ denotedby $S^t_0$.......... $S^t_{30}$ . and $B^t_0$................. $B^t_{89}$.respectively.

#### 2.21.Non Linear Feedback Shift Register 1(NFSR 1)

NFSR1 in Lizard replaces the LFSR of the Grain family of stream ciphers. It is 31 bit wide and corresponds to the NFSR A10 of the eSTREAM Phase 2 hardware portfolio candidate ACHTERBAHN-128/80[9].It has a guaranteed period of $2^{31}$ -1 and can be specified by the following update relation, defining f1 given by:

$$\begin{aligned} S^{t+1}_{30} = {} & S^t_0 \oplus S^t_2 \oplus S^t_5 \oplus S^t_6 \oplus S^t_{15} \oplus S^t_{17} \oplus S^t_{18} \\ & \oplus S^t_{20} \oplus S^t_{25} \oplus S^t_8 S^t_{18} \oplus S^t_8 S^t_{20} \oplus S^t_{12} S^t_{21} \\ & \oplus S^t_{14} S^t_{19} \oplus S^t_{17} S^t_{21} \oplus S^t_{20} S^t_{22} \oplus S^t_4 S^t_{12} S^t_{22} \\ & \oplus S^t_8 S^t_{18} S^t_{22} \oplus S^t_4 S^t_{19} S^t_{22} \oplus S^t_7 S^t_{20} S^t_{21} \\ & \oplus S^t_8 S^t_{20} S^t_{22} \oplus S^t_{12} S^t_{19} S^t_{22} \oplus S^t_{20} S^t_{21} S^t_{22} \\ & \oplus S^t_4 S^t_7 S^t_{12} S^t_{21} \oplus S^t_4 S^t_7 S^t_{19} S^t_{21} \oplus S^t_4 S^t_{12} S^t_{21} S^t_{22} \\ & \oplus S^t_4 S^t_{19} S^t_{21} S^t_{22} \oplus S^t_7 S^t_8 S^t_{18} S^t_{21} \oplus S^t_7 S^t_8 S^t_{20} S^t_{21} \\ & \oplus S^t_7 S^t_{12} S^t_{19} S^t_{21} \oplus S^t_8 S^t_{18} S^t_{21} S^t_{22} \oplus S^t_8 S^t_{20} S^t_{21} S^t_{22} \\ & \oplus S^t_{12} S^t_{19} S^t_{21} S^t_{22} \end{aligned} \qquad (1)$$

#### 2.22.Non Linear Feedback Shift Register 2(NFSR 2)

NFSR2 is 90 bits wide and uses a modified version of g from Grain-128A[4] as its feedback polynomial.In contrast to NFSR1, the period of 2nd NFSR during keystream generation is unknown because, due to the masking bit from NFSR1 More precisely, f2 of NFSR2 squeezes the taps of g to fit a 90 bit register, resulting in the following update relation given by:-

$$\begin{aligned} B^{t+1}_{89} = {} & S^t_0 \oplus B^t_0 \oplus B^t_{24} \oplus B^t_{49} \oplus B^t_{79} \oplus B^t_{84} \\ & \oplus B^t_3 B^t_{59} \oplus B^t_{60} B^t_{74} \oplus B^t_{10} B^t_{12} \oplus B^t_{15} B^t_{16} \\ & \oplus B^t_{25} B^t_{53} \oplus B^t_{35} B^t_{42} \oplus B^t_{20} B^t_{22} B^t_{23} \\ & \oplus B^t_{55} B^t_{58} \oplus B^t_{62} B^t_{68} B^t_{72} \oplus B^t_{77} B^t_{80} B^t_{81} B^t_{83} \end{aligned} \qquad (2)$$

#### 2.23.Output Function

Output Function is denoted by the letter ‘a’. Output function of LIZARD ’a’ can be written as the direct sum of a linear function with seven monomials, a quadratic function with four monomials, a triangular function with seven monomials, and an another four monomial triangular function, where each tap of NFSR1 and NFSR2 appears at most once in a. Output Function ’a’ is defined through the output bit $z_t$ of Lizard at time t, which is computed as:

$$z_t = L_t \oplus Q_t \oplus T_t \oplus T_t' \qquad (3)$$

where $L_t$ , $Q_t$ , $T_t$ and $T_t'$ can be computed as follows:

$$L_t = B^t_7 \oplus B^t_{11} \oplus B^t_{30} \oplus B^t_{40} \oplus B^t_{45} \oplus B^t_{54} \oplus B^t_{71} \qquad (4)$$

$$Q_t = B^t_4 B^t_{21} \oplus B^t_9 B^t_{52} \oplus B^t_{18} B^t_{37} \oplus B^t_{44} B^t_{76} \qquad (5)$$

$$\begin{aligned} T_t = {} & B^t_5 \oplus B^t_8 B^t_{52} \oplus B^t_{34} B^t_{67} B^t_{81} \oplus B^t_2 B^t_{28} B^t_{41} B^t_{65} \\ & \oplus B^t_{13} B^t_{29} B^t_{50} B^t_{64} B^t_{75} \oplus B^t_6 B^t_{14} B^t_{26} B^t_{32} B^t_{47} B^t_{61} \\ & \oplus B^t_1 B^t_{19} B^t_{27} B^t_{43} B^t_{57} B^t_{66} B^t_{78} \end{aligned} \qquad (6)$$

$$T_t' = S^t_{23} \oplus S^t_3 S^t_{16} \oplus S^t_9 S^t_{13} S^t_{48} \oplus S^t_1 S^t_{24} S^t_{38} S^t_3 \qquad (7)$$

### 2.3. State Initialization

The state initialization process can be divided into 4 phases. Phases 1-3 constitute an instantiation of the FP(1)-mode which provides 80-bit security against generic time-memory-data (TMD) tradeoff attacks aiming at key recovery. Phase 4 is a consequence of the necessity to make sure that NFSR1 is not in the all-zero state after phase 3.

## 2.31.Phase 1: Key and IV Loading

Let K = (K₀)………… (K₁₁₉) denote the 120-bit key and IV = (IV₀)……………..(IV₆₄) the 64-bit public IV. The registers of the keystream generator are initialized as follows:

$$B_j^0 = \begin{cases} K_j \oplus IV_j & ,for\ 0 \le j \le 63 \\ K_j, & for\ 64 \le j \le 89 \end{cases} \quad (8)$$

$$S_j^0 = \begin{cases} K_{(j+90)}, & for\ 0 \le j \le 29 \\ K_{119}, & for\ j = 29 \\ 1, & for\ j = 30 \end{cases} \quad (9)$$

## 2.32. Phase 2: Grain-like Mixing

In this phase the cipher is clocked 128 times without producing the actual output. For t = 0,1,2…………..127, the output $z_t$ is fed back to both of the FSRs as shown in figure 3.

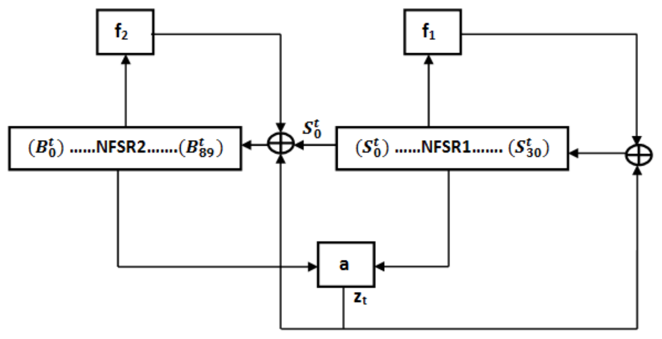

f₂
f₁
$S_0^t$
$(B_0^t)$ ……NFSR2…….$(B_{89}^t)$
$(S_0^t)$ ……NFSR1……. $(S_{30}^t)$
a
$z_t$

**Fig-3**: Lizard in phase 2 of the state initialization

For t = 0,1,2…………..127, the update relations are given as:-

$$B_j^{t+1} = B_{j+1}^t \quad (10)$$
$$B_{89}^{t+1} = z_t \oplus S_{i+1}^t \oplus f_2(B^t) \quad (11)$$
$$S_i^{j+1} = S_{i+1}^j \quad (12)$$
$$S_{30}^{t+1} = z_t \oplus f_1(S^t) \quad (13)$$

## 2.33.Phase 3: Second Key Addition

This phase is similar to phase 1.The same secret key that we used in the first phase is used here. In this phase the 120 bit keystream is xored in a bit wise fashion to the inner state of the keystream generator.

The update relations for NFSR2 and NFSR1 are given as:-

$$B_j^{129} = B_j^{128} \oplus K_j,\ for\ j \in \{0,\ldots.89\} \quad (14)$$

$$S_i^{129} = \begin{cases} S_i^{128} \oplus K_{i+90} & ,for\ i \in \{0,\ldots 29\} \\ \quad 1, & for\ i = 30 \end{cases} \quad (15)$$

The rightmost bit of the NFSR is set to 1 to avoid all zero states which can make our NFSR ineffective.Phases 1 to 3 constitute an instantiation of the key loading and mixing steps of the FP(1)-mode, which, in our case, yields 80-bit security w.r.t. generic TMD tradeoff attacks aiming at key recovery.

## 2.34. Phase 4: Final Diffusion

Phase 4 is required in order to obtain the necessary diffusion with respect to the bit at the rightmost position of NFSR1, which was set to 1 in phase 3.In this phase the cipher is again clocked 128 times .Output generated is discarded in this phase. LIZARD in phase 4 operation is shown in figure 4.

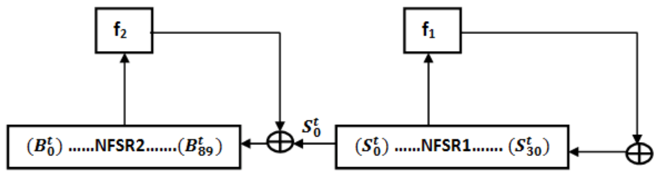

f₂
f₁
$S_0^t$
$(B_0^t)$ ……NFSR2…….$(B_{89}^t)$
$(S_0^t)$ ……NFSR1……. $(S_{30}^t)$

**Fig-4:** Lizard in phase 4 of the state initialization

For t = 129, . . . , 256,the full update relations that will be used for NFSR2 and NFSR1 are given as:-

$$B_j^{t+1} = B_{j+1}^t \quad (16)$$
$$B_{89}^{t+1} = S_0^t \oplus f_2(B^t) \quad (17)$$
$$S_i^{t+1} = S_{i+1}^t \quad (18)$$
$$S_{30}^{t+1} = f_1(S^t) \quad (19)$$

Where,

$$\begin{aligned} f_1(S^t) = {} & S_0^t \oplus S_2^t \oplus S_5^t \oplus S_6^t \oplus S_{15}^t \oplus S_{17}^t \oplus S_{18}^t \\ & \oplus S_{20}^t \oplus S_{25}^t \oplus S_8^t S_{18}^t \oplus S_8^t S_{20}^t \oplus S_{12}^t S_{21}^t \\ & \oplus S_{14}^t S_{19}^t \oplus S_{17}^t S_{21}^t \oplus S_{20}^t S_{22}^t \oplus S_4^t S_{12}^t S_{22}^t \\ & \oplus S_8^t S_{18}^t S_{22}^t \oplus S_4^t S_{19}^t S_{22}^t \oplus S_7^t S_{20}^t S_{21}^t \qquad (20) \\ & \oplus S_8^t S_{20}^t S_{22}^t \oplus S_{12}^t S_{19}^t S_{22}^t \oplus S_{20}^t S_{21}^t S_{22}^t \\ & \oplus S_4^t S_7^t S_{12}^t S_{21}^t \oplus S_4^t S_7^t S_{19}^t S_{21}^t \oplus S_4^t S_{12}^t S_{21}^t S_{22}^t \\ & \oplus S_4^t S_{19}^t S_{21}^t S_{22}^t \oplus S_7^t S_8^t S_{18}^t S_{21}^t \oplus S_7^t S_8^t S_{20}^t S_{21}^t \\ & \oplus S_7^t S_{12}^t S_{19}^t S_{21}^t \oplus S_8^t S_{18}^t S_{21}^t S_{22}^t \oplus S_8^t S_{20}^t S_{21}^t S_{22}^t \\ & \oplus S_{12}^t S_{19}^t S_{21}^t S_{22}^t \end{aligned}$$

$$\begin{aligned} f_2(B^t) = {} & B_0^t \oplus B_{24}^t \oplus B_{49}^t \oplus B_{79}^t \oplus B_{84}^t \\ & \oplus B_3^t B_{59}^t \oplus B_{60}^t B_{74}^t \oplus B_{10}^t B_{12}^t \oplus B_{15}^t B_{16}^t \\ & \oplus B_{25}^t B_{53}^t \oplus B_{35}^t B_{42}^t \oplus B_{20}^t B_{22}^t B_{23}^t \qquad (21) \\ & \oplus B_{55}^t B_{58}^t \oplus B_{62}^t B_{68}^t B_{72}^t \oplus B_{77}^t B_{80}^t B_{81}^t B_{83}^t \end{aligned}$$

## 2.4. Keystream generation

After the 4 phases of state initialization the 31 bit initial state of NFSR 1 is $(S_0^{257})$……….. $(S_{30}^{257})$ and 90 bit initial state of NFSR2 is $(B_0^{257})$……….. $(B_{89}^{257})$.The first keystream bit that is used for plaintext encryption is $z_{257}$. For t greater than 257,states of two registers and $z_t$ is computed using the equations given above in state initialization section.Figure 5 depicts the structure of Lizard stream cipher during keystream generation. As Lizard is designed to be operated in packet mode, the maximum size of a plaintext packet encrypted under the same key/IV pair is

$2^{18}$ bits and no key/IV pair may be used more than once, i.e., for more than one packet.

Lizard is really a synchronous stream cipher. i.e., the keystream bits $z_{257}$, $z_{258}$..........are generated in a bitwise fashion (and independently of the plaintext/ciphertext) and consequently, the individual plaintext bits $x_i$ can be encrypted and then transmitted as they arrive. The same obviously holds for the decryption of the ciphertext bits $y_i$.

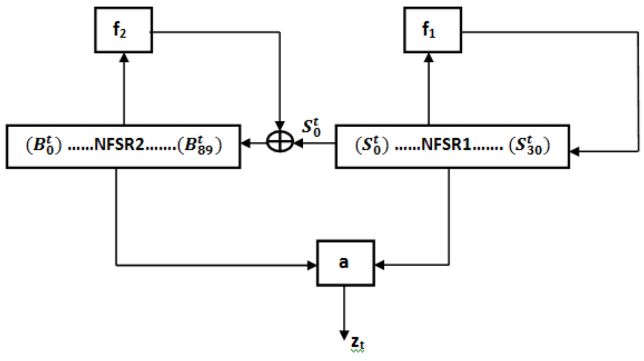

f₂
f₁
(B₀ᵗ) ......NFSR2.......(B₈₉ᵗ)
S₀ᵗ
(S₀ᵗ) ......NFSR1....... (S₃₀ᵗ)
a
zₜ

**Fig-5** :Lizard in keystream generation mode

## 3. EXPERIMENTAL RESULTS

This section includes the experimental setup, simulation results and FPGA implementation of LIZRAD Stream cipher IP core. Before going into more detailed about experimental setup, the algorithm or the design steps can be verified using a numerical example.Numerical Examples are a great way to understand the algorithm of the specified cipher design. For that a random test vector can be selected. Design verification using numerical examples can be done with the help of matlab coding or it can be done manually.

### 3.1 Experimental Setup

The arrangement consists of two linear feedback shift registers whose outputs are given as input to an output function which generates the output key stream in a bitwise manner. Two feedback shift registers contains two feedback functions in which the AND and XOR operations take place. NFSR1 and NFSR2 is represented as S register and B register respectively. Output bit is represented by the letter z. Setup for keystream generation mode is represented in figure in section II (D).

The key stream generated is then XORed with the message or the plaintext to form a ciphertext, which is called encryption and then the ciphertext is again XORed with the key stream to obtain the plaintext back i.e. decryption.The simulation is done using verilog language in Xilinx ISE. Xilinx ISE (Integrated Synthesis Environment) is a software tool manufactured by Xilinx for analysis and synthesis of High level Description Language designs

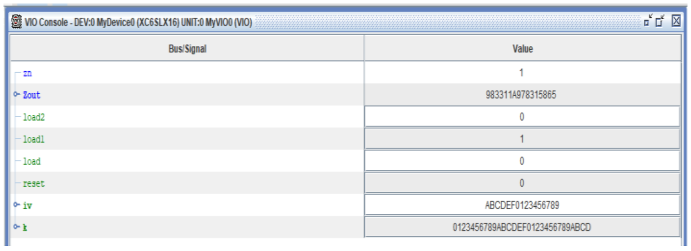
### 3.2. Simulation Results

Simulation results are obtained for the LIZARD Stream Cipher IP core for the test vectors
Key = 0*0123456789ABCDEF0123456789ABCD,
IV = ABCDEF0123456789 and
Plaintext = 51084CE73A5CA2EC87D7BABC297542
The outputs obtained ie,Keystream generated = 983311A97831586548209DAFBF26FC93
Cipher text = C93B5D4E426DFA89CFF727139653BE

Intermediate outputs obtained are:-
```
Phase1:- B register = 1010 1010 1110 1110 1010 1010
                      0110 0110 1010 1010 1110 1110
                      1010 1010  0110 0110 0000 0001
                      0010 0011 0100 0101 01
S Register = 1001111000100110101011110011001
Phase2:- B register = 0011 1011 0010 0001 0010 1110
                      1010 1000 0110 0001 0100 0101
                      0101 0010 0010 0001 0100 1011
                      0110 0001 0110 1101 11
S Register = 0001101101010000000110111101111
Phase3:- B register = 0011 1010 0000 0010 0110 1011
                      1100 1111 1110 1000 1110 1110
                      1001 1111 1100 1110 0100 1010
                      0100 0010 0010 1000 10
S Register = 1001001110001110101000101101101
Phase4:- B register = 0001 01101000 0011 1001 0001
                      1000 1001 1011 1111 0000 1011
                      1011 0110 1010 0101 0011 0101
                      0000 1010 1100 1110 01
S Register = 0111011010100111101001111010011
```

### 3.3. Hardware Test Results

Hardware implementation of LIZARD stream cipher IP core is done using FPGA Spartan 6 (XC6SLX16-3CSG324).After configuring our device, we can analyze our FPGA design using Chip Scope Pro software. It mainly consists of 3 cores, VIO(Virtual Input/Output), ILA(Integrated Logic Analyzer) and ICON(Integrated Controller).ICON core facilitates the communication between the embedded ILA and VIO cores and the computer running the ChipScopePro Analyzer software. The ILA core can be defined as a customizable logic analyzer core that can be used to surveil the internal signals in our designs. Because the ILA core is synchronous to the design being observed, all design clock constraints applied to our design are also applied to the components inside the ILA core. The virtual input/output core is a customizable core that can both surveil and drive internal FPGA signals in real time. Unlike the ILA and IBA cores, the VIO core does not require on-chip RAM.


VIO Console - DEV:0 MyDevice0 (XC6SLX16) UNIT:0 MyVIO0 (VIO)

| Bus/Signal | Value |
|---|---|
| zn | 1 |
| Zout | 983311A978315865 |
| load2 | 0 |
| load1 | 1 |
| load | 0 |
| reset | 0 |
| iv | ABCDEF0123456789 |
| k | 0123456789ABCDEF0123456789ABCD |

**Fig-6** : VIO console of Key stream generation.

## 3.4. Synthesis Results

Here mainly two synthesis results are taken into account. Resource Utilization Summary and Power Utilization Summary. The number of slice registers utilized was 1164, number of LUT flip-flop pair used was 1445 and number of bonded IOBs was found to be 65.The XPower Analyzer (XPA) of Xilinx ISE is used to get a detailed view of the power distribution for our design, broken down into individual device elements. XPA does an analysis on real design data. The power tools can be used for power optimization as well. The total power consumption is obtained as 0.021W.

## 4. Conclusion

FPGA implementation of LIZARD stream cipher IP core is presented in this paper. Behavioral description of the design is written in verilog HDL and simulated using XILINX ISE Isim simulator. Then the design is successfully implemented in XILINX Spartan 6 FPGA (XC6SLX16-3CSG324).Design verification is performed with the help of Chip Scope tool of Xilinx. Power utilization is analyzed using Xilinx Power Analyzer (XPA). LIZARD stream cipher provides 80 bit security against Time Memory Data Tradeoff attacks aiming at key recovery. LIZARD stream cipher saves on chip area and consumes 16 percent less power than the already existing highly efficient GRAIN V1.

## Acknowledgment

This work is supported by National Institute of Electronics and Information Technology (NIELIT), Calicut.

## References

[1] **Hamann M, Krause M, and MeierW,** "LIZARD - A lightweight stream cipher for power constrained devices",inIACR transactions on symmetric cryptology, 45-79, 2017.

[2] **Armknecht, Frederik, and Vasily Mikhalev,** "On lightweight stream ciphers with shorter internal states.", International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2015.

[3] **Kaur, Gagandeep, and Jaswinder Singh.,** "Review and Comparative Analysis of Stream Cipher for LTE Technology.", International Journal of Advanced Research in Computer Science 8.5 , 2017.

[4] **Gren, Martin, et al,** Grain-128a: a new version of Grain-128 with optional authentication",International journal of wireless and mobile computing 5.1,48-59, 2011.

[5] **Hell, Martin, et al,** "A stream Cipher proposal: Grain-128", Information theory, 2010 IEEE international symposium on. IEEE, 2010.

[6] **Zhang, Bin,** "Near collision attack on the grain v1 stream cipher", International workshop on fast software encryption. Springer, Berlin, Heidelberg, 2010.

[7] **Hell, Martin, Thomas Johansson, and Willi meier,** "Grain: a Stream cipher for Constrained Environments", International journal of wireless and mobile computing 2.1,86- 93,2006.

[8] **Biryukov, Alex, Adi Shamir, and DavidWagner.,** "Real Time Cryptanalysis of A5/1 on a PC.", International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg,2000.

[9] **Naya-Plasencia, Maria,** "Cryptanalysis of achterbahn-128/80.", International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg,(2007).

[10] **Babbage, Steve, and Matthew Dodd,** "The stream cipher MICKEY 2.0.", ECRYPT Stream Cipher (2006).

[11] **Shaked, Yaniv, and Avishai Wool,** "Cryptanalysis of the bluetooth E0 cipher.", International Conference on Information Security. Springer, Berlin, Heidelberg, 2006.

[12] **De Canniere, Christophe, and Bart Preneel.,** "Trivium.", New Stream Cipher Designs Springer, Berlin, Heidelberg, 244-266, 2008.

[13] **Hamann, Matthias, and Matthias Krause.,** "On stream ciphers with provable beyond the birthday-bound security against time-memory-data tradeoff attacks.", Cryptography andCommunications 10.5 : 959-1012.,2018.

[14] **Duval, Sebastien, Virginie Lallemand, and Yann Rotella. ,** "Cryptanalysis of the FLIP family of stream ciphers", Annual International Cryptology Conference. Springer, Berlin, Heidelberg,2016.

[15] **Armknecht, Frederik, and Vasily Mikhalev.,**"On lightweight stream ciphers with shorter internal states.",International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2015.

[16] **Wu, Hongjun,** "The stream cipher HC-128.", New stream cipher designs. Springer Berlin Heidelberg,39-47.,2008.

[17] **David, Mathieu, Damith C. Ranasinghe, and Torben Larsen.,** "A2U2: a stream cipher for printed electronics RFID tags.",2011 IEEE International Conference on RFID. IEEE,2011.

[18] **Kitsos, Paris, et al.,**"FPGA-based performance analysis of stream ciphers ZUC, Snow3g,Grain V1, Mickey V2, Trivium and E0.", Microprocessors and Microsystems 37.2 : 235-245.,2013.