International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Oct-2022)

https://dx.doi.org/10.12785/ijcds/120184

High Throughput Efficient Implementation of Grain v1, Lizard and Plantlet Stream Ciphers for Resource Constraint Devices

Vinod Kumar Gill[1], Ravi Teja Chenna[2], Naveen Kumar Kandula[3] and Dheeraj Kumar Sharma[4]

[1,2,3]*Department of School of VLSI Design Embedded System, National Institute of Technology Kurukshetra, Haryana, India*
[4]*Department of Electronics Communication Engineering, National Institute of Technology Kurukshetra, Haryana, India*

Received 22 Jan. 2021, Revised 15 Jul. 2022, Accepted 23 Jul. 2022, Published 31 Oct. 2022

Abstract: Nowadays, as the number of smart devices is increasing, technologies like Radio Frequency Identification (RFID) and the Internet of Things (IoT) are playing a vital role in communicating among various smart devices which require high security for the data transmission using fewer resources effectively. To implement security with fewer resources available, one of the methods is to use lightweight cryptographic algorithms. Cipher is one of the cryptographic algorithms, which uses the key for encryption and decryption. This paper describes three lightweight symmetric stream ciphers, Grain v1, Lizard, and Plantlet, and their different versions. These versions are implemented on various Field Programmable Gate Array (FPGA) boards such as Kintex7, Virtex7, and Zynq, and their simulations, synthesis, and implementations are performed with the help of Xilinx ISE using Verilog Hardware Description Language (HDL). The area, frequency, throughput, and throughput per area are compared among the different versions of the same cipher and also with existing literature. To generate keystream bits, Grain v1 uses 161 clocks in the basic version, 320 clocks in the serial version, and 11 clocks in the parallel version, Lizard uses 258 clocks in the basic version, 499 clocks in the serial version and 46 clocks in parallel version and Plantlet uses 321 clocks in the basic version and 491 clocks in serial version. The parallel version provides high throughput than the basic and serial versions as the keystream bits generated in each clock cycle are more. Among these versions, the parallel version of Grain v1 and Lizard gave the highest throughput of 9147 and 2412 Mbps along with throughput per area of 50.81 and 9.28 respectively. The parallelism helps in achieving faster data transfer and high operating frequency ranges which is more suitable for the present emerging technologies IOT, blockchains, and Ultra High Frequency Radio Frequency Identification (UHF-RFID).

Keywords: IoT, Resource Constrained Devices, RFID, FPGA, Stream Cipher, Grain v1, Lizard, Plantlet

1. INTRODUCTION

From the past decade, as the digital era came into existence, the internet and communication technologies like the Internet of Things (IoT), Radio Frequency Identification (RFID), etc used to communicate among various smart devices have become an important part of our daily lives. Since the usage of smart devices is increasing, the major problems are storage and security. The technologies like IoTs and RFID have to perform effectively using fewer resources with high security. To implement security with fewer resources available, one of the methods is to use lightweight cryptographic algorithms. Cryptography is the oldest technique used which includes encryption and decryption of data. Encryption is converting the source data to a secured and unintelligible form and decryption is converting the unintelligible form of data to the source data. Ciphers are used in cryptography, which are algorithms used for encryption and decryption using a key. Based on the key, the ciphers can be categorized into two types: symmetric and asymmetric. In asymmetric ciphers, the public key is used at the time of encryption and the secret key is used at the time of decryption whereas in symmetric ciphers, during encryption and decryption process, the same secret key is used.

Symmetric ciphers are mainly divided into two categories: block cipher and stream cipher. Block cipher is a technique that divides the information into a set of fixed length of blocks and then encrypt and decrypt each block. In the stream cipher, the information is bit by bit encrypted and decrypted by using the generated key bitstream. Stream ciphers have an advantage over the block ciphers since the information can be directly encrypted whereas, in the block ciphers, the information has to be stored in memory to divide the information into blocks. Some of the applications of stream ciphers are RFID tags, multi-gigabit communication channels, radio communication, and Global System for Mobile Communications (GSM) communication.

The Grain family (Grain v0, Grain v1, Grain 128a)[1], [2], [3] is one of the portfolios of the stream ciphers, which was accepted in eSTREAM [4]. For the first time, it was submitted by M. Hell, T. Jonasson, and W. Meier [5] in the year 2005. Grain 128AEAD [6] is the latest entry in the Grain family and it is based on Grain 128a

E-mail address: vinod_32018240@nitkkr.ac.in, ravi_32018226@nitkkr.ac.in, naveen_32018222@nitkkr.ac.in, sharma986dheeraj@nitkkr.ac.in

[3] to add extra security. This family consists of a LFSR (linear feedback shift register) and a NFSR (non-linear feedback shift register) which were designed to achieve a high speed at the cost of the additional hardware requirement. The LFSR assures periodicity or steadiness of the output whereas the non-linearity is introduced by the NFSR. The motivation behind that research was to provide security for RFID tags, where memory and power are limited. This design was referred to as Grain v0. Based on this idea, many designs were proposed. One of the designs proposed was Grain v1, which contains 64 bit initialization vector (IV) and 80 bit secret key including a 160 bits of internal state. Grain v1 has a similar design to Grain v0. It uses the same LFSR polynomial function used in Grain v0, but the NFSR polynomial function is different and provides more reliability. For each key/IV pair, a keystream of length 238 is produced in Grain v1 [7].

Hamann, M., Krause, M., Meier, W [8] proposed a Lizard cipher to get attractive features regarding security and efficiency. Specifically, packet mode scenarios are targeted. Lizard is a lightweight stream cipher which is specially designed for limited power resource devices. The design of Lizard is mostly similar to Grain v1. It consists of a 120 bit secret key, 64 bit IV and 121 internal states. It has two NFSRs instead of LFSR, and NFSR when compared to Grain v1 and some parts of the operation are similar to Grain v1. For every key/IV pair 218 keystream bits are obtained [9].

V. Mikhalev, F. Armknecht, and C. Müller proposed a Plantlet [10] cipher from the sprout [11] cipher which stores a non-volatile key. Plantlet is a lightweight cipher and has a larger size of LFSR which overcomes the weaknesses in sprout cipher since, Plantlet has 61 bit LFSR and sprout has 40 bit LFSR. Plantlet also provides more security. Plantlet works on the double layer LFSR and there is no all zero state in the LFSR. Plantlet consists of a 80 bit secret key and 90 bit IV. It has LFSR of size 61 bit and NFSR of size 40 bit and a od 80 counter. It has few modifications in the functions of the sprout cipher but the operation is similar in both the ciphers. One Key/IV pair is used to generate 230 keystream bits [10].

The above mentioned ciphers are implemented using Xilinx ISE Design Suite 14.7 [12]. It is a very useful Electronic Design Tool (EDA) tool which consists of synthesis and implementation. It also generates Register Transfer Level (RTL) schematics, technical schematics and simulation results, which are used to understand the behavior of code on various FPGA boards. It also generates a design summary after synthesis and implementation. RTL schematic is also known as the gate-level schematic which gives an overview consisting of logic gates, adders, etc., based on code and it is independent of the selected device. Technical schematic gives an overview consisting of a number of look up tables (LUT), buffers, flip flops, etc., and it is generated based on the selected devices. Simulation is used to verify the design based on the stimulus applied and monitor for the desired output results.

FPGAs [13] have advantages over ASIC [14] in various domains such as reusable memory and reconfigurable devices and, are used in various applications like Radio Detection and Ranging (RADAR) [15]. Smart Medical [16], Cloud Computing [17] and Data Centre Programs [18] etc. Xilinx ISE Design suite supports many FPGA boards. In this work, the 7 series FPGA boards such as Kintex7 [19], Virtex7 [20] and Zynq [21] are used. Kintex7 [19] is a low cost FPGA with a high end performance built on a 28 nm architecture designed for maximum power efficiency and it consumes 50 less power compared to previous generations of FPGAs. It supports mainstream standards like Peripheral Component Interconnect (PCI) gen3 and 10 gigabit ethernet and it can be used in 3G and 4G wireless, video over IP solutions. Virtex7 [20] is an FPGA board built on 28 nm architecture designed to achieve high I/O bandwidths and high Digital Signal Processing (DSP) performance and it can be used in 10G to 100G networking, portable radars etc. Zynq 7000 [21] consists of a dual core ARM processor equipped with 28 nm Kintex7 [19] based programmable logic and has 6.6 million logic cells. It is used in various embedded applications like multi camera drivers, HDTV, etc.

In the present work, the above described ciphers are implemented in various versions i.e., Grain v1 and Lizard are implemented in basic, serial and parallel versions whereas Plantlet is implemented in basic and serial versions. In the basic version, inputs are taken at a time and output keystream is generated bit by bit. In the serial version, inputs are to be taken in bit by bit form and output keystream is produced in the form of bit by bit and in parallel, the inputs are taken all at a time and output keystream is given n bits at a time. n is 16 and 6 for Grain v1 and Lizard, respectively. To achieve high throughput, the parallel version is considered. All these versions have their advantages and disadvantages. They are used accordingly for different performance motives.

The remaining part of the paper is categorized in the following sections. Section 2 includes the working of the three ciphers. Section 3 describes the simulation results of phases present in different versions of ciphers. The implementation of ciphers performed on various FPGA boards is explained in section 4. The drawn conclusions are outlined in section 5.

2. WORKING OF CIPHERS

This section describes the working of Grain v1, Lizard and Plantlet stream ciphers. The initialization and keystream generation phase are demonstrated. Further, various versions such as basic, serial and parallel are described. It has been outlined in the following subsections.

A. *Grain v1*

Grain v1 [6] is a symmetric stream cipher which takes 80 bits secret key and 64 bits IV as inputs and processes them to generate a keystream. It consists of two 80 bits shift registers i.e., LFSR whose states are represented as $M_1 = m1_l, m1_{l+1}, ...m1_{l+79}$ NFSR whose states are represented as $N_l = n1_l, n1_{l+1}, ...n1_{l+79}$ and a non-linear output function $w(u1) = w(u1_0, u1_1, u1_2, u1_3, u1_4)$ which is a five input variable function.

LFSR is a shift register which has linear feedback. In linear feedback, some values from the shift register are added together to generate a value which is given as feedback input to the shift register. Whereas the NFSR is a shift register which has non-linear feedback. In non-linear feedback some values from the shift register are multiplied and then added together to generate a value which is given as feedback input to the shift register. The non-linear output function takes four inputs from LFSR and one input is taken from NFSR.

Grain v1 works in two phases: initialization phase and keystream generation phase. First in the initialization phase the secret key, IV and 16 padding bits i.e., 16'hFFFF are loaded into the shift registers and for updating the shift registers i.e., the outputs from the feedback function and the output function are added together and then given as feedback to LFSR and NFSR for 160 clock cycles. The feedback function and output functions equations are as mentioned below.

$$m1_{l+80} = m1_{l+62} \oplus m1_{1+51} \oplus m1_{l+38} \oplus m1_{l+23} \oplus m1_{l+13} \oplus m1_{l} \tag{1}$$

$$\begin{aligned} n1_{l+80} = {} & m1_{l} \oplus n1_{l+62} \oplus n1_{l+60} \oplus n1_{l+52} \oplus n1_{l+45} \oplus n1_{l+37} \\ & \oplus n1_{l+33} \oplus n1_{l+28} \oplus n1_{l+21} \oplus n1_{l+14} \oplus n1_{l+9} \oplus n1_{l} \\ & \oplus n1_{l+63}n1_{l+60} \oplus n1_{l+37}n1_{l+33} \oplus n1_{l+15}n1_{l+9} \\ & \oplus n1_{l+60}n1_{l+52}n1_{l+45} \oplus n1_{l+33}n1_{l+28}n1_{l+21} \\ & \oplus n1_{l+63}n1_{l+45}n1_{l+28}n1_{l+9} \\ & \oplus n1_{l+60}n1_{l+52}n1_{l+37}n1_{l+33} \\ & \oplus n1_{l+33}n1_{l+28}n1_{l+21}n1_{l+15}n1_{l+9} \\ & \oplus n1_{l+52}n1_{l+45}n1_{l+37}n1_{l+33}n1_{l+28}n1_{l+21} \\ & \oplus n1_{l+63}n1_{l+60}n1_{l+21}n1_{l+15} \\ & \oplus n1_{l+63}n1_{l+60}n1_{l+52}n1_{l+45}n1_{l+37} \end{aligned} \tag{2}$$

$$\begin{aligned} x_l &= W(M_l, N_l) \\ &= \bigoplus_{j \in A} n1_{l+j} \oplus w(n1_{l+3}, n1_{l+25}, n1_{l+46}, n1_{l+64}, n1_{l+63}) \end{aligned} \tag{3}$$

where A=1,2,4,10,31,43,56

$$\begin{aligned} w(u1) &= w(u1_0, u1_1, u1_2, u1_3, u1_4) \\ &= u1_1 \oplus u1_4 \oplus u1_0u1_3 \oplus u1_2u1_3 \oplus u1_3u1_4 \\ & \quad \oplus u1_0u1_1u1_2 \oplus u1_0u1_2u1_3 \oplus u1_0u1_2u1_4 \\ & \quad \oplus u1_1u1_2u1_4 \oplus u1_2u1_3u1_4 \end{aligned} \tag{4}$$

During the initialization phase, no keystream output is taken from the cipher. Key bit stream generation phase is the second phase. In this phase, the only input to the shift registers is from the feedback functions and the output keystream is generated from the non-linear output function. $m1_{l+80}$ is the feedback function used to update LFSR and $n1_{l+80}$ is the feedback function used to update NFSR. x_l is the output function.

Grain v1 can be implemented in three versions: basic, serial and parallel based on the incoming inputs and as well as the required output bit stream. In the basic version, during the loading stage, the secret key and IV pair (with padding bits) are taken all at a time in a single clock cycle. The output from the feedback function and the output function are produced bit by bit. In the serial version, while loading the secret key, IV and the padding bits, the bits are loaded into shift registers bit by bit. It takes 160 clock cycles to load all the bits. The output is generated in a similar manner as provided in the basic version.

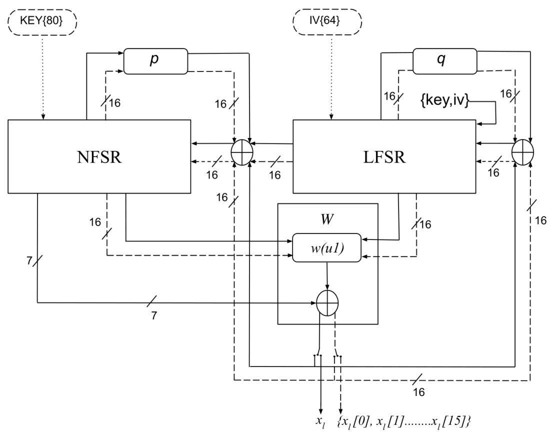

Figure 1. Block Diagram of Grain v1

In the parallel version, inputs are taken in a similar manner as the basic version i.e., in one clock cycle and the output from the feedback function and the keystream output generates 16 bits in one clock cycle since the maximum possible parallelism is 16. In the parallel version the shift registers are updated by 16 bits for every clock cycle. In the initialization phase, it requires 160 clock cycles to update both the shift registers since in the parallel version, authors have taken 16 bits at a time and therefore, only 10 clock cycles are required In Figure 1, p and q represent feedback functions of NFSR and LFSR respectively. While loading the inputs i.e., secret key and IV, dotted lines represent basic and parallel versions and solid lines represent serial versions. In the remaining part of the operation, the solid lines represent basic and serial versions and dashed lines represent parallel versions, the shift registers are updated by 16 bits for every clock cycle. In the initialization phase, it requires 160 clock cycles to update both the shift registers since in the parallel version, authors have taken 16 bits at a time and therefore, only 10 clock cycles are required.

In Figure 1, p and q represent feedback functions of NFSR and LFSR respectively. While loading the inputs i.e., secret key and IV, dotted lines represent basic and parallel versions and solid lines represent serial versions. In the remaining part of the operation, the solid lines represent basic and serial versions and dashed lines represent parallel versions.

B. Lizard

Lizard [8] is a symmetric stream cipher which takes 120 bits secret key and 64 bit IV as inputs and processes them to generate the keystream. The process of generating keystream consists of key register, shift registers, feedback functions and output functions. Two NFSRs of size 90 $(m_0^l, m_1^l, m_2^l, m_3^l, \ldots\ldots m_{88}^l, m_{89}^l)$ and 31 bits $(n_0^l, n_1^l, n_2^l, n_3^l, \ldots\ldots n_{29}^l, n_{30}^l)$ are used in the Lizard. The output function (z_l) is also a nonlinear function which takes values from both shift registers.

In Lizard, the operation basically has two phases: initialization phase and keystream generation phase. The initialization phase is again divided into four phases i.e., key IV loading phase, NFSRs updating phase, second key addition phase and final diffusion phase. In the Key IV loading phase, the secret key is stored in the key register and key values are added with IV, they are further loaded into NFSR2 $(m_0^l, m_1^l, m_2^l, m_3^l, \ldots\ldots m_{88}^l, m_{89}^l)$. The remaining key values are loaded into NFSR1 $(n_0^l, n_1^l, n_2^l, n_3^l, \ldots\ldots n_{29}^l, n_{30}^l)$. The MSB of NFSR1 is considered binary 1 i.e., 1'b1. In the NFSRs updating phase, the outputs from the feedback function and output function (z_l) are added together and given as input to NFSRs. This phase is similar to the initialization phase in Grain v1. Thus, this phase is also termed as a Grain like mixing phase. This phase runs for 128 clock cycles. In the second key adding phase, the key stored in the key register is added with the values present in the NFSRs and stored in the NFSRs. In the final diffusion phase, the NFSRs are updated only with the feedback functions and it runs for 128 clock cycles. The feedback and output functions are modified as given below.

Key IV Loading phase:

$$m_i^0 = \begin{cases} K_i \oplus IV_i & for \;\; i = \{0, 1, \ldots 63\} \\ K_i & for \;\; i = \{64, 65, \ldots 89\} \end{cases} \tag{5}$$

$$n_i^0 = \begin{cases} K_{i+90} & for \;\; i = \{0, 1, \ldots 28\} \\ 1 \oplus K_{119} & for \;\; i = 29 \\ 1 & for \;\; i = 30 \end{cases}$$

Second Key Loading phase:

$$m_i^{129} = K_i \oplus m_i^{128} \quad for \;\; i = \{0, 1, \ldots 89\} \tag{6}$$

$$n_i^{129} = \begin{cases} K_{i+90} \oplus n_i^{128} & for \;\; i = \{0, 1, \ldots 29\} \\ 1 & for \;\; i = 30 \end{cases}$$

NFSRs Updating feedback Functions:

$$m_i^{l+1} = m_{i+1}^l \quad for \;\; i = \{0, 1, \ldots 88\}$$

$$\begin{aligned} m_{89}^{l+1} = {} & n_0^l \oplus m_0^l \oplus m_{24}^l \oplus m_{49}^l \oplus m_{79}^l \oplus m_{84}^l \oplus m_3^l m_{59}^l \\ & \oplus m_{10}^l m_{12}^l \oplus m_{15}^l m_{16}^l \oplus m_{25}^l m_{53}^l \oplus m_{35}^l m_{42}^l \\ & \oplus m_{55}^l m_{58}^l \oplus m_{60}^l m_{74}^l \oplus m_{20}^l m_{22}^l m_{23}^l \\ & \oplus m_{62}^l m_{68}^l m_{72}^l \oplus m_{77}^l m_{80}^l m_{81}^l m_{83}^l \end{aligned} \tag{7}$$

$$n_i^{l+1} = n_{i+1}^l \quad for \;\; i = \{0, 1, \ldots 29\}$$

$$\begin{aligned} n_{30}^{l+1} = {} & n_0^l \oplus n_2^l \oplus n_5^l \oplus n_6^l \oplus n_{15}^l \oplus n_{18}^l \oplus n_{25}^l \oplus n_{14}^l n_{19}^l \\ & \oplus (n_{12}^l (n_{21}^l \oplus (n_{19}^l n_{22}^l (1 \oplus n_{21}^l)))) \oplus (n_{17}^l (1 \oplus n_{21}^l)) \\ & \oplus (n_7^l n_{21}^l (n_{20}^l \oplus (n_8^l n_{18}^l) \oplus (n_8^l n_{20}^l) \oplus (n_{19}^l n_{12}^l))) \\ & \oplus (n_4^l (n_{22}^l \oplus (n_7^l n_{21}^l) \oplus (n_{21}^l n_{22}^l)) \oplus (n_{12}^l n_{19}^l)) \\ & \oplus ((n_8^l n_{18}^l \oplus (n_{20}^l n_8^l) \oplus n_{20}^l)(1 \oplus (n_{22}^l (1 \oplus n_{21}^l)))) \end{aligned}$$

Output Function:

$$z_l = T_l \oplus Q_l \oplus P_l \oplus R_l$$

where

$$\begin{aligned} T_l &= m_7^l \oplus m_{11}^l \oplus m_{30}^l \oplus m_{40}^l \oplus m_{45}^l \oplus m_{54}^l \oplus m_{71}^l \\ Q_l &= m_4^l m_{21}^l \oplus m_9^l m_{52}^l \oplus m_{18}^l m_{37}^l \oplus m_{44}^l m_{76}^l \\ P_l &= m_5^l \oplus m_8^l m_{82}^l \oplus m_{34}^l m_{67}^l m_{73}^l \oplus m_2^l m_{28}^l m_{41}^l m_{65}^l \\ & \oplus m_{13}^l m_{29}^l m_{50}^l m_{64}^l m_{75}^l \oplus m_6^l m_{14}^l m_{26}^l m_{32}^l m_{47}^l m_{61}^l \\ & \oplus m_1^l m_{19}^l m_{27}^l m_{43}^l m_{57}^l m_{66}^l m_{78}^l \\ R_l &= n_{23}^l \oplus n_3^l n_{16}^l \oplus n_9^l n_{13}^l m_{48}^l \oplus n_1^l n_{24}^l m_{38}^l m_{63}^l \end{aligned} \tag{8}$$

In the initialization phase, no keystream output is taken from the cipher. The next phase is the keystream generation phase in which the NFSRs are updated only from the feedback functions. The output from the output function (z_l) is given as the keystream of the Lizard cipher.

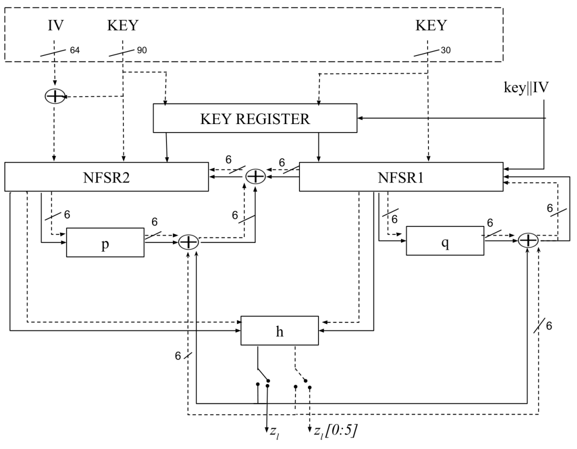

Figure 2. Block Diagram of Lizard

Lizard can be implemented in three versions: basic, serial and parallel based on the inputs and as well as the required output keystream. In the basic version, during the loading phase, the secret key and IV are loaded into the key register, shift registers as mentioned in the equations in a single clock cycle. The outputs from the feedback function and the output function z_l are given bit by bit. In the serial version during the loading phase, the secret key is loaded into the key registers bit by bit and it takes 120 clock cycles to load all the data. The output is given in a similar manner as the basic version. In the parallel version, inputs are taken in a similar manner as the basic version in a single clock cycle. The outputs from the feedback function and the output function (z_l) are given 6 bits in one clock cycle since the maximum parallelism is 6. The shift registers are updated 6 bits for every clock cycle. In the NFSRs updating phase and final diffusion phase, it needs 128 clock cycles to update both the shift registers as in the parallel version, it requires 6 bits at a time. Therefore, only 21 clock cycles are required i.e., 20 clock cycles for updating 120 bits and 1 clock cycle for updating 2 bits.

In Figure 2, p and q represent feedback functions of NFSR2 and NFSR1 respectively. The h represents output function. While loading the inputs i.e., Key and IV, dashed lines represent basic and parallel versions and solid lines represent serial versions. In the remaining part of the operation, the solid lines represent basic and serial versions and dashed lines represent parallel versions.

C. Plantlet

Plantlet [10] is a symmetric stream cipher which takes 80 bit secret key and 90 bit IV as inputs and processes them to generate the keystream. The process of generating keystream consists of key register, counter, key selector, shift registers, feedback functions and output functions. The LFSR and NFSR in Plantlet have a size of 61 bit ($s_{l+0}, s_{l+1}, s_{l+2}, s_{l+3}...s_{l+59}, s_{l+60}$) and 40 bit ($b_{l+0}, b_{l+1}, b_{l+2}, b_{l+3}...b_{l+38}, b_{l+39}$) respectively. The non-linear output function consists of nine input variables in which seven inputs are taken from LFSR and two inputs are taken from NFSR. The 80 bit secret key is stored in the key register and the counter ($c_{l+0}, c_{l+1}, c_{l+2}, c_{l+3}...c_{l+78}, c_{l+79}$) is a MOD-80 counter. The key selector (k_s) is used to select a key bit.

Plantlet works in two phases: initialization phase and keystream generation phase. First, in the initialization phase, the secret key is loaded into the key register, and IV along with the 11 padding bits i.e., 11'h7FD are loaded into the shift registers i.e. the NFSR is loaded with the first 40 bits of IV and next 50 bits of IV are loaded into the first 50 LFSR stages along with 11 padding bits added in remaining LFSR stages. After loading, the 60th bit of LFSR (s_{l+59}) is updated by adding the outputs from the feedback function (P(S_l)) and the output function (Z_l) the 4th MSB from the counter (c_{l+4}) and the key selector value (k_s). This process continues upto 320 clock cycles. The feedback function and output functions equations are as mentioned below.

In the initialization phase:

$$s_{l+59} = z_l \oplus P(S_1)$$

where

$$P(S_l) = s_{l+54} \oplus s_{l+43} \oplus s_{l+34} \oplus s_{l+20} \oplus s_{l+14} \oplus s_{l+0}$$
$$Z_l = S_{l+30} \oplus \sum_{j\in A} b_{l+j} \oplus r(B_l, S_l)$$

where A={1,6,15,17,23,28,34} and

$$r(B_l, S_l) = b_{l+4}s_{l+6} \oplus s_{l+8}s_{l+10} \oplus s_{l+32}s_{l+17} \oplus s_{l+19}s_{l+23} \oplus b_{l+4}s_{l+32}b_{l+38}$$

$$b_{l+39} = Z_l \oplus q(B_l) \oplus k_s \oplus S_{l+0} \oplus c_{l+4} \tag{9}$$

where

$$\begin{aligned} q(B_l) = {} & b_{l+0} \oplus b_{l+13} \oplus b_{l+19} \oplus b_{l+35} \oplus b_{l+39} \oplus b_{l+2}b_{l+25} \\ & \oplus b_{l+3}b_{l+5} \oplus b_{l+7}b_{l+8} \oplus b_{l+14}b_{l+21} \oplus b_{l+16}b_{l+18} \\ & \oplus b_{l+22}b_{l+24} \oplus b_{l+26}b_{l+32} \oplus b_{l+33}b_{l+36}b_{l+37}b_{l+38} \\ & \oplus b_{l+0}b_{l+11}b_{l+12} \oplus b_{l+27}b_{l+30}b_{l+31} \\ k_s = {} & key\ selector\ of\ keyregister\ according\ to\ count \\ & value \\ c_{l+4} = {} & MOD\ 80\ counter\ 4^{th}\ LSB \end{aligned}$$

In keystream generation phase:

$$s_{l+59} = P(S_1)$$

where

$$P(S_l) = s_{l+54} \oplus s_{l+43} \oplus s_{l+34} \oplus s_{l+20} \oplus s_{l+14} \oplus s_{l+0}$$

$$b_{l+39} = q(B_l) \oplus k_s \oplus S_{l+0} \oplus c_{l+4} \tag{10}$$

where

$$\begin{aligned} q(B_l) = {} & b_{l+0} \oplus b_{l+13} \oplus b_{l+19} \oplus b_{l+35} \oplus b_{l+39} \oplus b_{l+2}b_{l+25} \\ & \oplus b_{l+3}b_{l+5} \oplus b_{l+7}b_{l+8} \oplus b_{l+14}b_{l+21} \oplus b_{l+16}b_{l+18} \\ & \oplus b_{l+22}b_{l+24} \oplus b_{l+26}b_{l+32} \oplus b_{l+33}b_{l+36}b_{l+37}b_{l+38} \\ & \oplus b_{l+0}b_{l+11}b_{l+12} \oplus b_{l+27}b_{l+30}b_{l+31} \\ k_s = {} & key\ selector\ of\ keyregister\ according\ to\ count \\ & value \\ c_{l+4} = {} & MOD\ 80\ counter\ 4^{th}\ LSB \end{aligned}$$

During the initialization phase, no output keystream generated from the cipher. After this phase, the keystream generation phase is started. The 61th bit of LFSR (s_{l+60}) is updated excluding the output from the output function and NFSR is also updated excluding the output from the output function. In this phase, the output of the output function (Z_l) is considered as the output of the cipher.

Plantlet can be implemented in two versions: basic and serial. In the basic version, the shift registers and secret key bits are loaded in one clock cycle and the keystream in the keystream generation phase is generated bit by bit i.e., one bit per cycle. In the serial version, the inputs are loaded to the key register and shift registers bit by bit and the remaining operation is the same as in the basic version. Here, 170 clock cycles are needed to load the inputs serially bit by bit to shift registers and key registers. One clock cycle is required for the padding bits. Therefore, 171 clock cycles are required and similar output is generated as in the basic version.

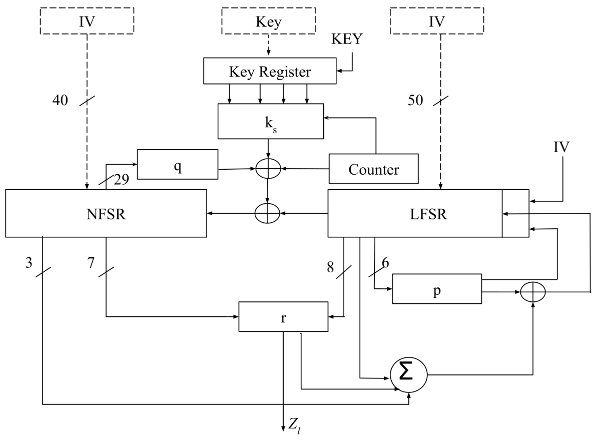

Figure 3. Block Diagram of Plantlet

In Figure 3, p and q represent feedback functions of LFSR and NFSR respectively. The r represents output function. While loading the inputs i.e., secret key and IV, dashed lines represent the basic version and solid lines represent the serial version. In the remaining part of the operation, the solid lines represent basic and serial versions.

3. TIME COMPLEXITY

A. Grain v1

In Grain V1, the number of the addition and multiplication operations for the LFSR update function, NFSR update function, and Output functions are 5 0, 22 29 and 16 13, respectively. Hence, the total number of addition and multiplication operations in the key generation phase for the GrainV1 are 43 42 respectively. Therefore, the time complexity of the Grain V1 is equal to $43t_{add}$ + $42t_{mul}$ = 43 + $42n^2$. For basic and serial versions, the time complexity will be 43 + $42n^2$. Since, 16 unrolling loops are used in parallel version of GrainV1 therefore time complexity is 16(43 + $42n^2$) = 688 + $672n^2$.

B. Lizard

In Lizard, the number of the addition and multiplication operations for the NFSR1 update function, NFSR2 update function, and Output functions are 26 18, 15 14 and 21 31, respectively. Hence, the total number of addition and multiplication operations in the key generation phase for the Lizard are 62 63 respectively. Therefore, the time complexity of the Lizard is equal to $62t_{add}$ + $63t_{mul}$ = 62 + $63n^2$ which is better compared to time complexity of past work [22] i.e, $67t_{add}$ + $95t_{mul}$ = 67 + $95n^2$. For basic and serial versions, the time complexity will be 62 +$63n^2$. Since, 6 unrolling loops are used in parallel version of Lizard therefore time complexity is 6(62 + $63n^2$) = 372 + $378n^2$.

C. Plantlet

In Plantlet, the number of the addition and multiplication operations for the NFSR update function, LFSR update function, and Output functions are 16 14, 5 0 and 12 6, respectively. Hence, the total number of addition and multiplication operations in the key generation phase for the Plantlet are 36 20 respectively. Therefore, the time complexity of the Plantlet is equal to $36t_{add}$ + $20t_{mul}$ = 36 + $20n^2$. For basic and serial versions, the time complexity will be 36 +$20n^2$.

4. SIMULATION RESULTS

A. Grain v1

Grain v1 stream cipher has three versions i.e., basic, serial and parallel and it works in two phases i.e., initialization phase and key bitstream generation phase. Figures 4-9 show the simulation results of the initialization phase and keystream generation phase for (a) Basic version, (b) Serial version and (c) Parallel version of Grain v1.

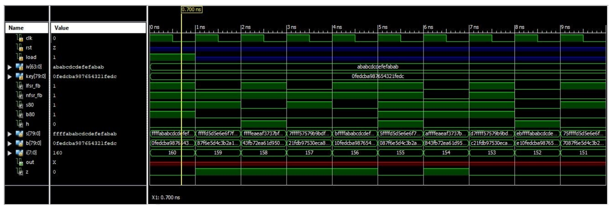

Figure 4. Simulation output waveform of initialization phase for Basic version of Grain v1

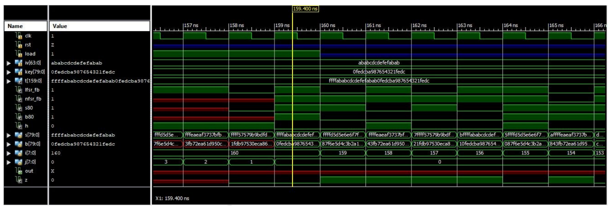

Figure 5. Simulation output waveform of initialization phase for Serial version of Grain v1

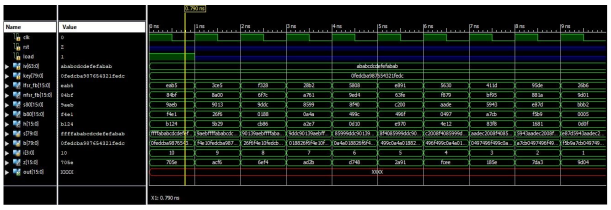

Figure 6. Simulation output waveform of initialization phase for Parallel version of Grain v1

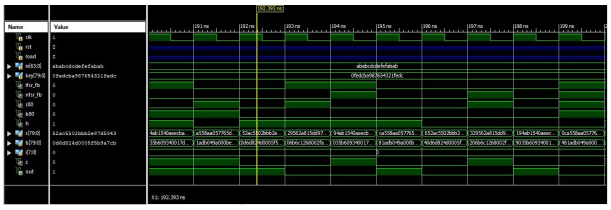

Figure 7. Simulation output waveform of key generation phase for Basic version of Grain v1

The variables present in the simulation results are clk, rst, load, iv, t, key, lfsr˙fb, nfsr˙fb, s80, b80, h, s, b, i, j, out, z. The variable “clk” is used to denote the clock cycle based on which the operations are performed. “rst” is used to reset which clears all the values in the registers

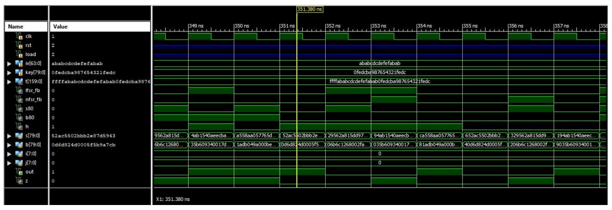
Figure 8. Simulation output waveform of key generation phase for Serial version of Grain v1

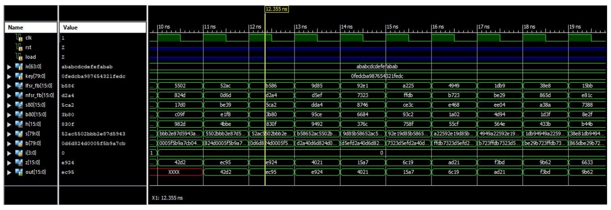
Figure 9. Simulation output waveform of key generation phase for Parallel version of Grain v1

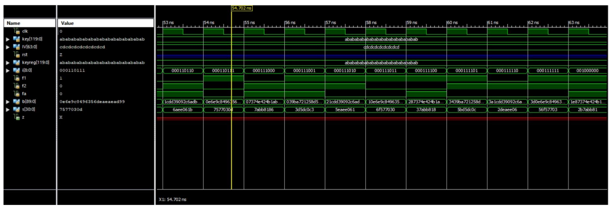
Figure 10. Simulation output waveform of NFSRs updating phase for Basic version of Lizard

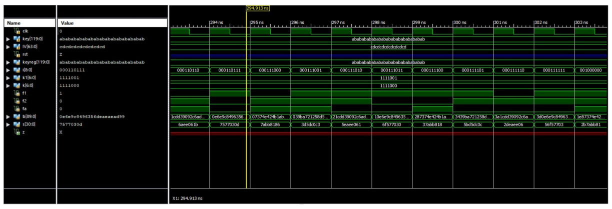
Figure 11. Simulation output waveform of NFSRs updating phase for Serial version of Lizard

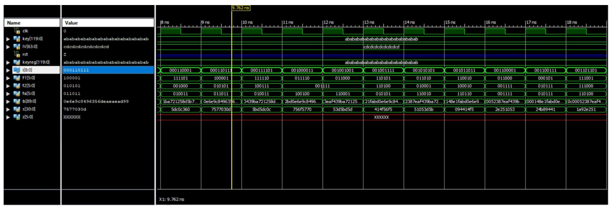
Figure 12. Simulation output waveform of NFSRs updating phase for Parallel version of Lizard

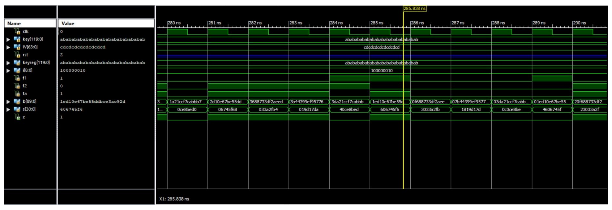
Figure 13. Simulation output waveform of key generation phase for Basic version of Lizard

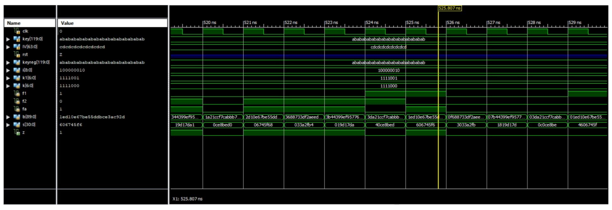
Figure 14. Simulation output waveform of key generation phase for Serial version of Lizard

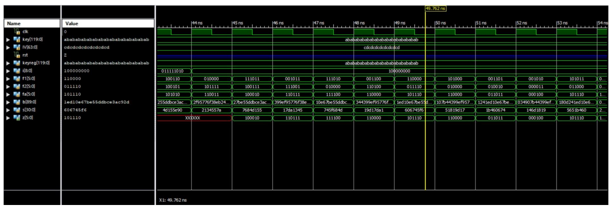
Figure 15. Simulation output waveform of key generation phase for Parallel version of Lizard

as well as signals i.e., when rst=1 at any clock cycle the values will be assigned to zero. “load” is the variable representing loading, i.e., when load=1, the key and IV are loaded into the lfsr and nfsr. The “iv” represents the initialization vector. The “key” is the secret key. The “lfsr˙fb” and “nfsr˙fb” are the outputs of feedback functions. “t” represents the temporary register which stores the secret key and IV and it provides bit by bit to the shift registers in serial. The “h” is the output of the output function and “z” is the value obtained after adding the seven bits from nfsr with h. The “s80” and “b80” are the inputs to the shift registers which are obtained from the outputs of feedback and output functions. The variable “s” represents the LFSR and “b” represents the NFSR. The variable “i” is used for counting helps in completing the initialization phase for 160 clock cycles and “j” is used to count the bits which are given bit by bit to the shift registers from the temporary register. The variable “out” is the output of the cipher which is connected to z at the keystream generation phase. Here, the size of the variables lfsr˙fb, nfsr˙fb, s80, b80, h, z, out is 1 bit in serial and basic versions but 16 bits in the parallel version.

In Grain V1, to load key IV values into shift registers and to update FSRs, the clock cycles for basic, serial and parallel versions required are 1 160, 160 160 and 1 10 respectively.

B. Lizard

As discussed in the previous section, Lizard is a Grain like stream cipher which also has three versions: basic, serial and parallel. There are two phases similar to Grain v1: Initialization phase and keystream generation phase. Only NFSRs updating phase and key generation phase are shown here because final diffusion phase is similar to NFSRs updating phase. Figures 10-15 show the simulation results of NFSRs updating phase and the key generation phase for (a) Basic version, (b) Serial version and (c) Parallel version of the Lizard cipher.

The variables used in the simulation results are clk, rst, key, IV, keyreg, s, b, f1, f2, fa, z, i, k, k1. The variables "clk", "rst", "key" and "IV" represent the same as described in Grain v1 simulations. The registers "s", "b" are the NFSR1 and NFSR2. "f1" and "f2" are the output values which are generated from feedback functions and they are given as input to the shift registers "b" and "s" respectively as explained in section 2.2. "fa" is the output from the output function which is added with "f1" and "f2" in the second phase (NFSRs updating phase). It is also connected to the output "z" in the key generation phase. "z" is the output of the cipher, which is connected to the output of the output function fa in the key generation phase. "i" is the variable used to count the clock cycle which is used to perform the operation. The variables "k" and "k1" are used only in the serial version. The variable "k" is used for loading the key into the key register and "k1" is used to load the values into the shift registers. The variables f1, f2, fa, z are of size 1 bit in basic and serial versions and 6 bits in parallel.

In Lizard, to load key IV values into shift registers, to update FSRs, for second key addition and final diffusion the clock cycles for basic, serial and parallel versions required are 1, 128, 1, 128, 242, 128, 1, 128 and 1, 22, 1, 22 respectively.

C. Plantlet

Plantlet has two versions i.e., basic and serial. It has two phases i.e., initialization phase and keystream generation phase. Figures 16-19 show the simulation results of the initialization phase and keystream generation phase for (a) Basic version (b) Serial version of the Plantlet.

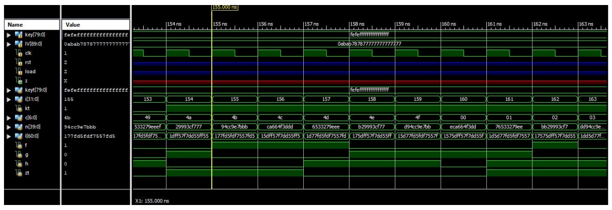

Figure 16. Simulation output waveform of initialization phase for Basic version of Plantlet

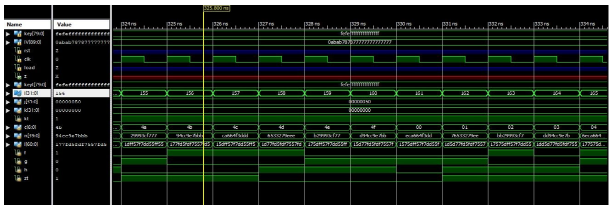

Figure 17. Simulation output waveform of initialization phase for Serial version of Plantlet

The variables used in the simulation results are clk, rst, key, IV, load, z, keyt, i, j, k, kt, c, n, l, f, g, h, zt. The variables "clk", "rst", "key", "IV" and "load" represent the same as described in Grain v1 simulations. The variables "keyt", "n", "l" represents key register, non-linear feedback shift register and linear feedback shift register, respectively. The variable "h" is the intermediate value used in the output function. The variables "f", "g", "zt" are the outputs of the linear feedback function, non-linear feedback function and output function, respectively.

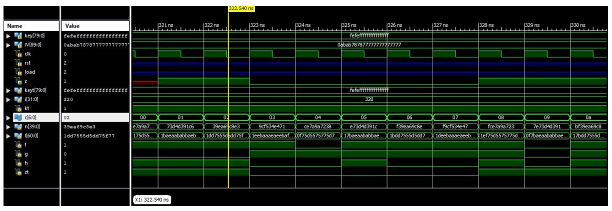

Figure 18. Simulation output waveform of key generation phase for Basic version of Plantlet

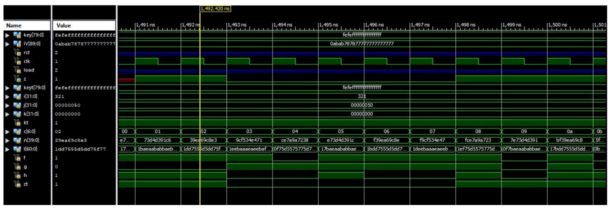

Figure 19. Simulation output waveform of key generation phase for serial version of Plantlet

The variable "i" is used to count the value which helps to perform the operation. The variable "j" is used in the serial version to load the secret key into the key register bit by bit and "k" is used to load the IV into the shift registers. The variable "z" is the output of the cipher which is connected to zt in the key generation phase. The variable "c" is used to define the mod 80 counter and "kt" is the value of the specified secret key bit of the key register based on the counter (c) value i.e., when c=4 then, the 4th bit of the key register is assigned to the kt.

In Plantlet, to load key IV values into shift registers and to update FSRs, the clock cycles for basic, serial versions required are 1 320, 171 320 respectively.

5. IMPLEMENTATION RESULTS

Various versions of the three stream ciphers are implemented on 7 series FPGA boards supported by Xilinx ISE design suite 14.7. The parameters generated in the design summary such as area, number of LUT, maximum operating frequency, throughput, throughput per area. Results are also being compared with the previous literature work.

Table I shows the implementation results of Grain v1 on Kintex7, Virtex7 and Zynq boards and compared with the results of the Grain v1 on various boards as mentioned in [[22]-[27]]. The implementation results of Lizard on Kintex7, Virtex7 and Zynq boards and their comparison with the results of the Lizard on Spartan7 in [29] are depicted in Table II. Table III refers to the implementation results of Plantlet on Kintex7, Virtex7 and Zynq boards and their comparison with the results of the Plantlet on Spartan7 in [29]. Table IV shows the comparison results between proposed ciphers and some other ciphers implemented in 7 series FPGA boards. For the results mentioned in proposed section of each table are same for Kintex7 (XC7K480T-3FFG1156), Virtex7 (XC7VX690T-3FFG1930) and Zynq (XC7Z020-3CLG484) boards.

The above proposed results are compared with the existing implementations and the basic version of Grain v1 achieved a frequency of 592 MHz and a throughput of 592 Mbps which is greater than that of [[22]-[27]]. The

TABLE I. IMPLEMENTATION RESULTS OF GRAIN V1 STREAM CIPHER ON VARIOUS FPGA BOARDS.

Board	Version	Area	Freq (MHz)	Throughput (Mbps)	Thr/Area	Reference
XC7K480T-3FFG1156	basic	169	592	592	3.50	Proposed
XC7VX690T-3FFG1930	serial	93	413	413	4.44	
XC7Z020-3CLG484	parallel	180	571	9147	50.81	
XC7S50 FGG484-1	basic	62	333	333	5.37	[22]
	serial	26	250	250	8.94	
	parallel	111	250	4000	36.03	
XC3S700A-4FG484	basic	318	177	177	0.56	[23]
XC3S400	serial	-	124	124	-	[24]
XC3S50-5PQ208	serial	44	196	196	4.45	[25]
	parallel	384	130	2080	5.98	
XC3S50-5PQ208	serial	122	193	193	1.58	[26]
	parallel	356	155	2480	6.97	
XC2S15-5	serial	48	105	105	2.18	[27]
	parallel	-	105	105n	-	

TABLE II. IMPLEMENTATION RESULTS OF LIZARD STREAM CIPHER ON VARIOUS FPGA BOARDS.

Board	Version	Area	Freq (MHz)	Throughput (Mbps)	Thr/Area	Reference
XC7K480T-3FFG1156	basic	251	492	492	1.96	Proposed
XC7VX690T-3FFG1930	serial	265	313	313	1.18	
XC7Z020-3CLG484	parallel	260	402	2412	9.28	
XC7S50 FGG484-1	basic	108	277	277	2.56	[22]
	serial	60	100	100	1.67	
	parallel	150	200	1200	4.98	

TABLE III. IMPLEMENTATION RESULTS OF PLANTLET STREAM CIPHER ON VARIOUS FPGA BOARDS.

Board	Version	Area	Freq (MHz)	Throughput (Mbps)	Thr/Area	Reference
XC7K480T-3FFG1156	basic	119	487	487	4.09	Proposed
XC7VX690T-3FFG1930	serial	158	368	368	2.33	
XC7Z020-3CLG484						
XC7S50 FGGA484-1	basic	62	303	303	4.89	[22]
	serial	56	270	270	4.82	

serial version achieved a frequency of 413 MHz and a throughput of 413 Mbps, better than that of results mentioned in [22]. The parallel version achieved a frequency of 571 MHz, throughput of 9147 Mbps and throughput per area of 50.81 which is highest among results provided in [[22]-[27]].

The basic version of Lizard achieved a frequency of 492 MHz and a throughput of 492 Mbps that is greater than the results of [22]. The serial version achieved a frequency of 313 MHz and throughput of 313 Mbps, which is better than results present in [22]. The parallel version achieved a frequency of 402 MHz, throughput of 2412 Mbps and throughput per area of 9.28 which are better when compared to the results of [22].

Plantlet achieved 487 MHz frequency and 487 Mbps throughput in the basic version, which are higher as compared to the previous work [22]. The serial version of the Plantlet also attained frequency of 368 MHz and a throughput of 368 Mbps showing improvements as compared to the previous implementation given in [22].

While implementing the above ciphers on various FPGA boards, though the area obtained may be more in some cases when compared to past work, throughput and throughput per area achieved are improved which are useful to achieve the high-speed requirements of today's emerging technologies like RFID, IoTs needed less resources for their efficient implementation.

The throughput is defined as the rate at which output bits are being generated i.e., the number of output bits generated divided by encryption time which is also defined as number bits per cycle multiplied with the operating frequency.

The maximum throughputs are achieved by implementing parallel version as multiple key stream bits are generated per clock cycle. For example, the frequency

TABLE IV. PERFORMANCE COMPARISONS OF PROPOSED GRAIN V1, LIZARD, AND PLANTLET WITH SOME STREAM CIPHERS ON THE XILINX 7 SERIES PLATFORM.

References	Area(slice)	Freq(MHz)	Throughput (Mbps)	Thr/Area	Version	Device
Proposed Grain v1	169	592	592	3.50	basic	XC7K480T-3FFG1156
	93	413	413	4.44	serial	XC7VX690T-3FFG1930
	180	571	9147	50.81	parallel	XC7Z020-3CLG484
Proposed Lizard	251	492	492	1.96	basic	XC7K480T-3FFG1156
	265	313	313	1.18	serial	XC7VX690T-3FFG1930
	260	402	2412	9.28	parallel	XC7Z020-3CLG484
Proposed Plantlet	119	487	487	4.09	basic	XC7K480T-3FFG1156
	158	368	368	2.33	serial	XC7VX690T-3FFG1930
						XC7Z020-3CLG484
Mickey 2.0 [22]	78	250	250	3.21	basic	XC7S50 FGGA484-1
	107	384	384	3.59	basic	
	51	250	250	4.90	serial	
	70	384	384	5.49	serial	
Expresso [28]	79	261	261	3.3	basic	XC7S100
	73	297	297	4.07	serial	
	68	511	511	7.7	serial	
	113	444	1778	15.73	parallel	

obtained for parallel version of Grain v1 is 571 MHz and 16 bits are generated in each clock cycle, so the Throughput is 16 times Clock frequency i.e. 9147 Mbps.

6. Conclusions and Future Work

The present work demonstrates the implementation of various versions of Grain v1, Lizard and Plantlet stream ciphers on FPGA boards using the Xilinx ISE suite. The parallel versions of Grain v1 and Lizard consume less clock cycles to generate the keystream than the serial version and basic version. The maximum parallelism in Grain v1 is 16 since the highest term used in the feedback functions are $m1_{l+64}, n1_{l+63}$ Therefore, 16 values from the shift register can be used at a time without updating the shift registers. Similarly, in Lizard since the highest terms used in the feedback function are m^l_{84}, n^l_{25} Thus, 6 values from shift registers can be used at a time without updating the shift registers. There is no parallel version in Plantlet since the highest terms used are s_{l+54}, b_{l+38} Therefore, no values can be used from registers without updating the LFSR.

The versions are implemented in various FPGA boards of different families like Kintex7, Virtex7 and Zynq. The parallel version of Grain v1 is getting highest throughput of 9147 Mbps as well highest throughput per area 50.81 followed by the parallel version of Lizard having 2412 Mbps of throughput and throughput per area of 9.27, which can be useful to achieve good performance in various resource constrained IoT devices[30].

As per requirements of area, throughput, and throughput-per-area for resource constraint devices [31], the different versions of these stream ciphers can be used respectively. Further, performance metrics of various stream ciphers can be enhanced by using their different versions.

References

[1] M. U. Bokhari, S. Alam, and S. H. Hasan, "A detailed analysis of grain family of stream ciphers," *Int. J. Comput. Netw. Inf. Secur*, vol. 6, no. 6, 2014.

[2] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: Grain-128," in *2006 IEEE International Symposium on Information Theory*. IEEE, 2006, pp. 1614–1618.

[3] M. Gren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: a new version of grain-128 with optional authentication," *International Journal of Wireless and Mobile Computing*, vol. 5, no. 1, pp. 48–59, 2011.

[4] estream, ecrypt stream cipher project. [Online]. Available: http://www.ecrypt.eu.org/stream

[5] M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments." estream, ecrypt stream cipher," Tech. Rep., 2005.

[6] M. Hell, T. Johansson, W. Meier, J. Sönnerup, and H. Yoshida, "An aead variant of the grain stream cipher," in *International Conference on Codes, Cryptology, and Information Security*. Springer, 2019, pp. 55–71.

[7] H. Zhang and X. Wang, "Cryptanalysis of stream cipher grain family," *Cryptology ePrint Archive*, 2009.

[8] M. Hamann, M. Krause, and W. Meier, "Lizard-a lightweight stream cipher for power-constrained devices," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 1, pp. 45–79, 2017.

[9] S. Banik and T. Isobe, "Some cryptanalytic results on lizard," *Cryptology ePrint Archive*, 2017.

[10] V. Mikhalev, F. Armknecht, and C. Müller, "On ciphers that continuously access the non-volatile key," *IACR Transactions on Symmetric Cryptology*, pp. 52–79, 2016.

[11] F. Armknecht and V. Mikhalev, "On lightweight stream ciphers with shorter internal states," in *International Workshop on Fast Software Encryption*. Springer, 2015, pp. 451–470.

[12] Xilinx, ise design suite. [Online]. Available: https://www.xilinx.com/products/design-tools/ise-design-suite.html

[13] W. Wolf, *FPGA-based system design*. Pearson Education India, 2004.

[14] M. J. S. Smith, *Application-specific integrated circuits*. Addison-Wesley Reading, MA, 1997, vol. 7.

[15] Xilinx, adaptable rfsocs, soc, and fpgas enable radar/isr algorithms with the lowest cost and power digital radar/ew. [Online]. Available: https://www.xilinx.com/applications/aerospace-and-defense/digital-radar-ew.html

[16] Xilinx, smart solutions for medical imaging, diagnostics and clinical equipment. [Online]. Available: https://www.xilinx.com/applications/medical.html

[17] Xilinx,cloud computing. [Online]. Available: https://www.xilinx.com/applications/megatrends/cloud-computing.html

[18] Xilinx, adaptable acceleration for the modern data center. [Online]. Available: https://www.xilinx.com/applications/data-center.html

[19] Xilinx 7 series fpgas, kintex7. [Online]. Available: https://www.xilinx.com/products/silicondevices/fpga/kintex-7.html

[20] Xilinx 7 series fpgas, virtex7. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html

[21] Xilinx 7 series fpgas, zynq 7000. [Online]. Available: https://www.xilinx.com/products/silicondevices/soc/zynq-7000.html

[22] B. Li, M. Liu, and D. Lin, "Fpga implementations of grain v1, mickey 2.0, trivium, lizard and plantlet," *Microprocessors and Microsystems*, vol. 78, p. 103210, 2020.

[23] P. Kitsos, N. Sklavos, G. Provelengios, and A. N. Skodras, "Fpga-based performance analysis of stream ciphers zuc, snow3g, grain v1, mickey v2, trivium and e0," *Microprocessors and Microsystems*, vol. 37, no. 2, pp. 235–245, 2013.

[24] A. Jafarpour, A. Mahdlo, A. Akbari, and K. Kianfar, "Grain and trivium ciphers implementation algorithm in fpga chip and avr micro controller," in *2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*. IEEE, 2011, pp. 656–658.

[25] D. Hwang, M. Chaney, S. Karanam, N. Ton, and K. Gaj, "Comparison of fpga-targeted hardware implementations of estream stream cipher candidates," *The State of the Art of Stream Ciphers*, pp. 151–162, 2008.

[26] K. Gaj, G. Southern, and R. Bachimanchi, "Comparison of hardware performance of selected phase ii estream candidates," in *State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report*, vol. 26, 2007, p. 2007.

[27] T. Good, W. Chelton, and M. Benaissa, "Review of stream cipher candidates from a low resource hardware perspective," *SASC 2006 Stream Ciphers Revisited*, vol. 125, 2006.

[28] G. Kumisbek, N. N. Anandakumar, and M. Hashmi, "Fpga implementations of espresso stream cipher," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, pp. 1–6.

[29] Xilinx 7 series fpgas, spartan7. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/spartan-7.html

[30] Itu-t x.1362 simple encryption procedure for internet of things (iot) environments. [Online]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1362-201703-I!!PDF-E&type=items

[31] Y. Watanabe, H. Yamamoto, and H. Yoshida, "Extending felics for automotive pkes systems. in 2019 fse rump session proceedings," 2018.

Vinod Kumar Gill received his B. Tech (Bachelor of Technology) degree in Electronics Communication Engineering from Jaipur Engineering College and Research Centre, Jaipur Rajasthan, India. He is currently pursuing M. Tech (Master of Technology) in VLSI Design from the department of School of VLSI Design Embedded System, National Institute of Technology, Kurukshetra, Haryana, India.

Ravi Teja Chenna received his B. Tech (Bachelor of Technology) degree in Electronics Communication Engineering from Jawaharlal Nehru Technology University Anantapur College of Engineering, Anantapur, Andhra Pradesh, India. He is currently pursuing M. Tech (Master of Technology) in VLSI Design from the department of School of VLSI Design and Embedded System, National Institute of Technology, Kurukshetra, Haryana, India.

Naveen Kumar Kandula received his B. Tech (Bachelor of Technology) degree in Electronics Communication Engineering from Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. He is currently pursuing M. Tech (Master of Technology) in VLSI Design in the department of School of VLSI Design and Embedded System, National Institute of Technology, Kurukshetra, Haryana, India.

Dheeraj Kumar Sharma is working as an Assistant Professor in the Department of Electronics and Communication Engineering, National Institute of Technology Kurukshetra, India. He received his B. Tech degree in Electronics Engineering from Aligarh Muslim University, M. Tech. degree in Electronics and Communication Engineering from Indian Institute of Technology, Roorkee and Ph.D. degree from National Institute of Technology, Kurukshetra. His Research interests include Cryptology, Stream Ciphers, FPGA Implementation, Hardware Implementation.