RECONFIGURABLE CRYPTOSYSTEM ARCHITECTURE FOR CIPHER PROCESSING

Babasaheb S. Shelke[1], Dr. Govind. U. Kharat[2]

[1] *ME student, Dept. of E & TC Engg. SPCOE Otur, Pune, Maharashtra, India,*

[2] *Principal, Dept. of E & TC Engg. SPCOE Otur, Pune, Maharashtra, India,*

ABSTRACT

In the world of cryptography, stream ciphers are known as primitives used to ensure privacy over a communication channel. One common way to build a stream cipher is to use a keystream generator to produce a pseudorandom sequence of symbols. ZUC is a stream cipher that forms the heart of the 3GPP confidentiality algorithm 128-EEA3 and the 3GPP integrity algorithm 128-EIA3, offering reliable security services in Long Term Evolution networks (LTE), which is a candidate standard for the 4G network. A detailed hardware implementation is presented in order to reach satisfactory performance results in LTE systems. Stream ciphers are more efficient when implemented in hardware environment, like Field Programmable Gate Array (FPGA). In this paper we are mainly interested in LTE systems and the different architecture and algorithms are used for their implementation. The architectures used in LTE mainly follow the 3GPP confidentiality algorithm 128-EEA3 and the 3GPP integrity algorithm 128EIA3. On above which SNOW3G and ZUC stream ciphers are dependent. Since all are above mainly considered in beyond 3G applications. Here we are implementing ZUC architecture in which the algorithm is designed in VHDL code and implemented on FPGA.

Keyword: - *FPGA, LTE, Reconfigurable Cryptosystem Architecture, Symmetric Ciphers, ZUC.*

1. INTRODUCTION

For encryption purposes there exist, basically, two types of primitives, block and stream ciphers. Block ciphers are classical primitives that have been studied for years. Collected design techniques and cryptanalysis of block ciphers allowed to develop such a standard for encryption as Rijndael (AES). This cipher is widely accepted, and it has strong resistance against various kinds of attacks. On the other hand, although the idea of stream ciphers appeared long ago, the open study and investigation of these primitives began only about 20 years ago. It is widely believed that stream ciphers can be smaller and much faster than block ciphers when implemented. Unfortunately, we still do not have enough knowledge about the design and cryptanalysis of stream ciphers.

Nowadays there are many stream cipher algorithms proposed in both academic and industrial research.In the field of telecommunications, the world is stepping into 4th Generation (4G for short) standard. During the last few years, the 3rd Generation Partnership Project (3GPP) has submitted Long Term Evaluation Advanced (LTEAdvanced), which is the enhancement of the LTE standard, as a candidate for the 4G network.Long Term Evolution (LTE), is the next-generation network beyond 3G that enable fixed to mobile migrations of Internet applications such as Voice over IP (VoIP), video streaming, music downloading, mobile TV and many others. LTE networks will also provide the capacity to support an explosion in demand for connectivity from a new generation of consumer devices tailored to those new mobile applications. The current radio interface protection algorithms for LTE, 128-EEA1 for confidentiality and 128-EIA1 for integrity have been designed by SAGE/ETSI Security Algorithms Group of Experts. 128-EEA1 and 128-EIA1 are based on SNOW3G stream cipher. Also, the 3rd Generation Partnership Project (3GPP), together with the GSM Association specifies a second set of algorithms, 128-EEA2 and 128-EIA2, which are based on AES block cipher. Finally, 3GPP with GSM association specifies a third set of algorithms for confidentiality and integrity the 128EEA3 and 128-EIA3 respectively. Both ciphers are based on ZUC stream cipher.

The most serious reason for these new ciphers is that LTE will be used in many countries worldwide. But Chinese regulation will not allow those algorithms to be used in China, because they were not designed in China. However, ZUC has been designed in China, and thus that it can be used in China. In this project an efficient FPGA implementation of ZUC stream cipher is presented. The advantages of Virtex- 5FPGA are explained using the embedded functions such as Digital Signal Processing (DSP) blocks, with the aim to minimize the registers and Look-Up Tables in the design.

1.1 BLOCK DIAGRAM

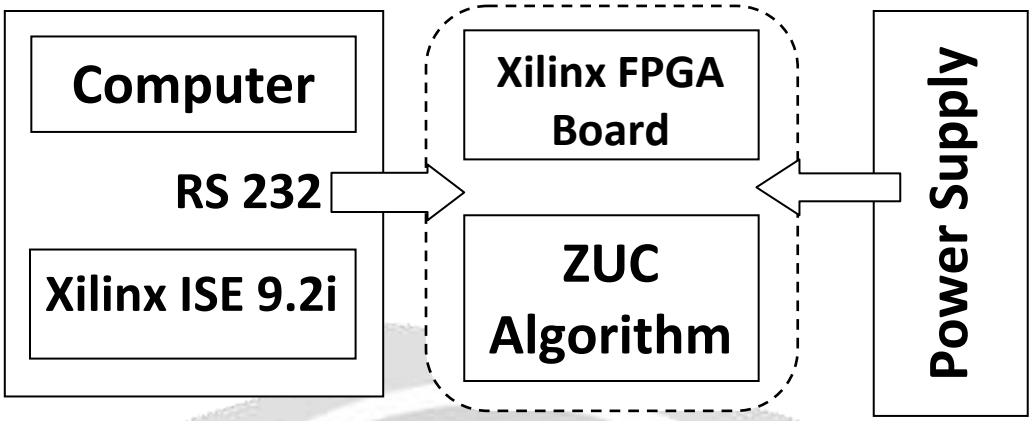


Fig.1 Block Diagram of System

An FPGA Implementation of ZUC Stream Cipher consist of personal computer, Xilinx FPGA kit containing ZUC Algorithm, Power Supply. Data to be Encrypted and encrypted data , ZUC encryption will be sent from PC to Xilinx kit. Xilinx kit will encrypt data using encryption key and encrypted data will again sent back to PC. On PC using Hyperterminal, we can observe data before encryption ZUC encryption key and encrypted data. Xilinx Field Programmable Gate Arrays (FPGAs) are highly flexible, reprogrammable logic devices that leverage advanced CMOS manufacturing technologies, similar to other industry-leading processors and processor peripherals. Like processors and peripherals, Xilinx FPGAs are fully user programmable. For FPGAs, the program is called a configuration bit stream, which defines the FPGA's functionality. The bit stream loads into the FPGA at system power-up or upon demand by the system. The process whereby the defining data is loaded or programmed into the FPGA is called configuration. Configuration is designed to be flexible to accommodate different application needs and, wherever possible, to leverage existing system resources to minimize system costs. Similar to microprocessors. Xilinx FPGAs optionally load or boot themselves automatically from an external nonvolatile memory.

1.2 ZUC STREAM CIPHER

Cipher systems are usually subdivided into block ciphers and stream ciphers. Block ciphers tend to simultaneously encrypt groups of characters, whereas stream ciphers operate on individual characters of a plain text message one at a time. ZUC is a word-oriented stream cipher, which is the core function of 3GPP confidentiality algorithm: 128-EEA3 and the 3GPP integrity algorithm: 128-EIA3.It takes a 128-bit Key and a 128-bit Initial Vector (IV) as input, and outputs a keystream of 32-bit words. The execution of ZUC has two stages: key initialization stage and working stage.

2. ZUC Work Flow

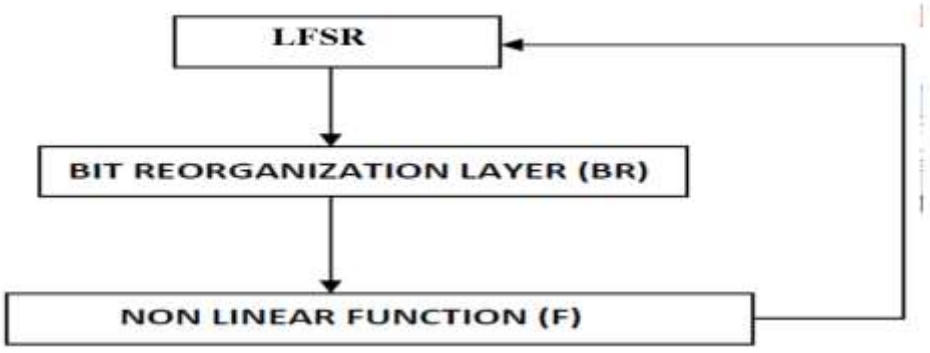


Fig.2 Work flow of ZUC Stream Cipher

In the first stage, a key initialization on LFSR (Linear Feedback Shift Register) is performed. The second stage is a working stage. In this stage LFSR does not receive any input. After working stage, during the key stream generating, with every clock tick, it produces a 32-bit word of output. In the specification, the algorithm is divided into three logical layers: a linear feedback shift register (LFSR) of 16 stages as the first layer, Bit-reorganization (BR) for the middle layer, a nonlinear function F the bottom layer. The LFSR has 16 of 31-bit cells (s0, s1, …, s15) Each register takes values from {0,1,……., 231-1}. In the key loading procedure 128-bit Initial key and 128-bit initial vectorc16 bytes each other: k = k0||k1||……||k15 and IV = IV0||IV1||……||IV15. Then load into the registers of LFSR as follows: si = ki||Di||IVi ($0 \le i \le 15$). Here, Di is a 15bit constant. In the initialization, the LFSR receives a 31-bit input word u, which is obtained by removing the rightmost bit from the 32-bit output W of the nonlinear function F, (u=W>>1). More specifically, the initialization mode works as follows:

2.1 The linear feedback shift registers (LFSR)

The LFSR has 16 of 31-bit cells (s0, s1, … , s15). This LFSR has two stages of operations: the initialization stage and the working stage. In the initialization, the LFSR receives a 31-bit input word u, which is obtained by removing the rightmost bit from the 32-bit output W of the nonlinear function F, (u=W>>1).

Algorithm 1: LFSR with Initialization Mode (u)

1 begin
2 $u = 2^{15}S_{15} + 2^{17}S_{13} + 2^{21}S_{10} + 2^{20}S_4 + (1 + 2^8)S_0 \bmod (2^{31} - 1);$
3 $S_{16} = (v + u) \bmod (2^{31} - 1);$
4 $if\ S_{16} = 0\ then$
5 $then\ set S16 = 2^{31} - 1$
6 $(S1, S2, \ldots, S15, S16) \qquad (S0, S1, \ldots, S14, S15);$

More specifically, the initialization mode works as follows. In the working mode, the LFSR does not receive any input, and it works as Algorithm 2 shown. As Algorithm 2 shown, the LFSR works independently with other part of ZUC. If we can get new *S*16 per clock, the shift registers can shift per clock cycle.

Algorithm 2: LFSR with Work Mode

1 begin
2 $u = 2^{15}S_{15} + 2^{17}S_{13} + 2^{21}S_{10} + 2^{20}S_4 + (1 + 2^8)S_0 \bmod (2^{31} - 1);$
3 $if\ S_{16} = 0\ then$
4 Then set $S_{16} = (2^{31} - 1)$
5 $(S1, S2, \ldots, S15, S16) \qquad (S0, S1, \ldots, S14, S15);$

2.2 The Bit-reorganization

The middle layer of the algorithm is the Bit-reorganization. It extracts 128 bits from the state of the LFSR and forms 4 of 32-bits words which will be used by the nonlinear function F in the bottom layer. Let *S*0*; S*2*; S*5*; S*7*; S*9*; S*11*; S*14*; S*15 be 8 registers of LFSR.

Then the Bit-reorganization forms 4 of 32-bit words *X*0*;X*1*;X*2*;X*3 from the above registers as Algorithm 3. Compared with software implementation, the concatenation of signal in hardware is only needed to change the signals order, and it nearly doesn't need extra time to do this work.

Algorithm 3: Bit Reorganization

begin
$X0 = S_{15H} || S_{14L};$
$X1 = S_{11L} || S_{9H};$
$X2 = S_{7L} || S_{5H};$
$X3 = S_{2L} \,||\, S_{0H};$

Therefore the bit reorganization stage can mix with the nonlinear function operation together to save clock cycle. In Algorithm 3, *S*15*H* denotes the leftmost 16 bits of integer *S*15, *S*14*L* denotes the rightmost 16 bits of integer *S*14, *S*15*H* //*S*14*L*, denotes the concatenation of strings *S*15*H* and *S*14*L*.

2.3 The nonlinear function F

The nonlinear function F has 2 of 32-bit memory cells *R*1 and *R*2. The input of the nonlinear function is the *X*0*;X*1*;X*2, which are the output of the bit reorganization step, the nonlinear function F outputs a 32-bit word W. The detailed process of F, as shown in Algorithm 4.

Algorithm 4: The Nonlinear function

Input: X0, X1, X1
begin

1. $W = (X0 \oplus R1) + R2;$
2. $W_1 = R1 + X1;$
3. $W_2 = R_2 \oplus X_2;$
4. $R1 = S(L_1(W1L \| W_{2H}));$
5. $R2 = S(L_2(W2L \| W_{1H}));$

In Algorithm 4, the S is a 32×32 S-box, *L*1 and *L*2 are linear transformations, which are defined as equation 1, 2. In the nonlinear function stage, the critical path is *W*1 = *R*1 _ *R*2, _ denotes the modulo 2^{32} addition.

The other operations in nonlinear function step comparing to modulo addition is needed less time to complete. So this nonlinear function and bit reorganization operation can be done in one clock cycle, that is to say, if LFSR can renew every clock cycle, the ZUC can generate 32-bit key per clock cycle. Section 3 will focus on designing a novel pipelined architecture to renew LFSR per clock cycle.

3. ZUC ARCHITECTURE

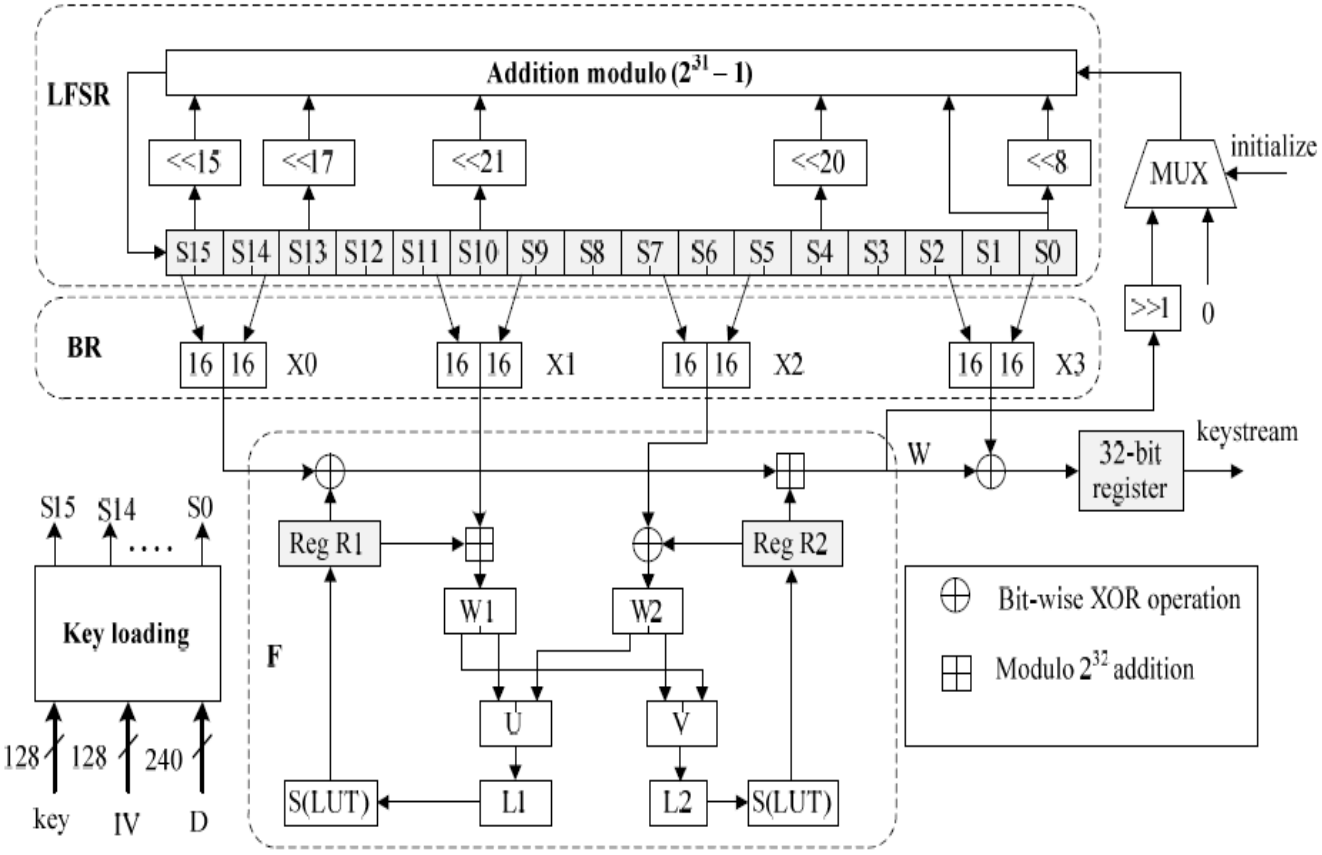


Fig.3 ZUC Architecture

The aim of the work is to ascertain that the ZUC stream cipher can operate on a recent hardware device for efficient use on LTE networks. The hardware implementation of the ZUC stream cipher is illustrated in Fig. 1. The proposed system has as main I/O interfaces a 32-bit plaintext/ciphertext input and a 32-bit ciphertext/plaintext output. As a set of control logic change, the configuration of the proposed hardware system supports all stages of operation. In addition it has two inputs, a 128-bit secret key, Key, and 128-bit initialization value, IV. Our system supports, the

initialization stage, the working stage and the keystream producing stage. The main parts of the proposed architecture of ZUC are the Key Loading, the Linear Feedback Shift Register (LFSR), the BR (bitreorganization) and the nonlinear function F. Finally, a Control Unit (that is not shown in the figure) is responsible for the correct operation of the stream cipher.

The Key Loading part use a 240-bit D constant, D = d0 ||d1 ||…||d15 (where $0 \leq di \leq 15$ are predefined) and to\gether with Key and IV, produce 16 substrings of 31-bit according to the following rule si = ki || di || ivi ($0 \leq i \leq 15$). The ki and ivi are considered the 16 bytes of the Key and IV respectively where k0 and iv0 are the most significant ones. Those substrings are used as initial values of the 31-bit LFSR cells S0,S1, …, S15 respectively. The substrings si are parallel loaded as the LFSR initial values through the OR gates as in Fig. 2. When the values are fetched the OR-gates are forced by zeros.

3.1 The Feedback Logic

It is an arithmetic logic that combines cyclical shifters and additions mod (231-1). The multiplexer (MUX) changes their configuration according the cipher operation scenario (initialization or working stage). Also, one more adder mod (231-1) is needed with its result used as first input of the multiplexer. In the Feedback Logic six additions mod (231-1) are used. The architecture for the two inputs, X, Y, adder mod (231-1) is depicted in Fig. 3. One adder is used in order to add the values of S0 and 28S0, another for the addition of 220S4 and 221S10 and another for the addition of 217S13 with 215S15. Finally, a three-input adder mod (231-1) is used to add the three previous sums. For the three-input adder mod (231-1) two cascaded two-input address mod (231-1) are used.

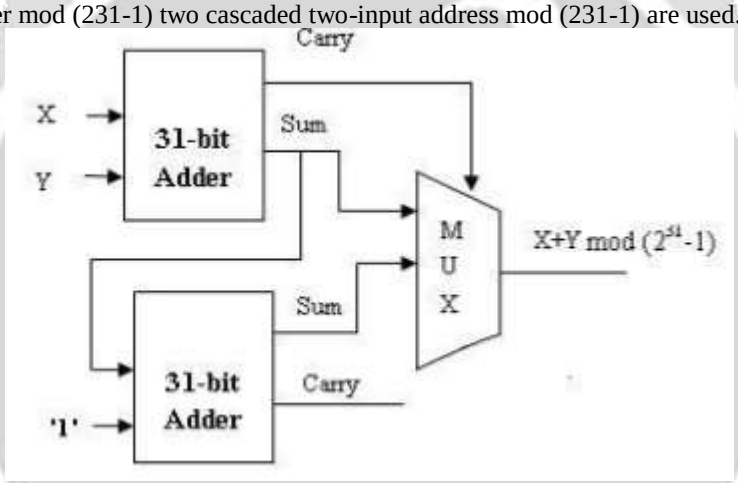


Fig. 4 The Architecture of Adder mod (2^{31}-1).

3.2 Feedback logic Implementation

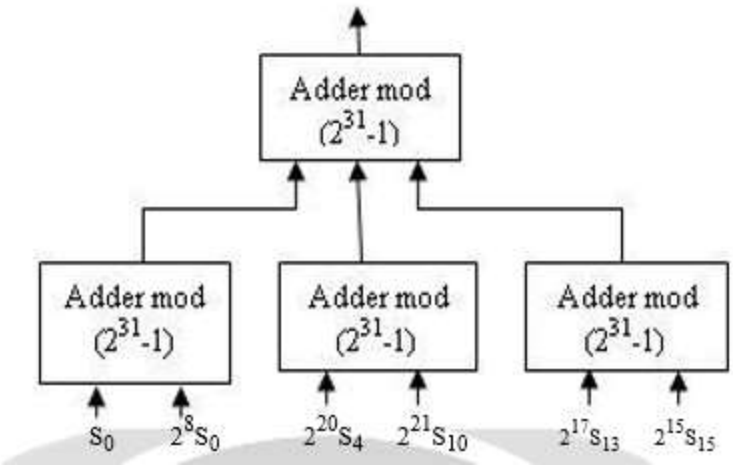
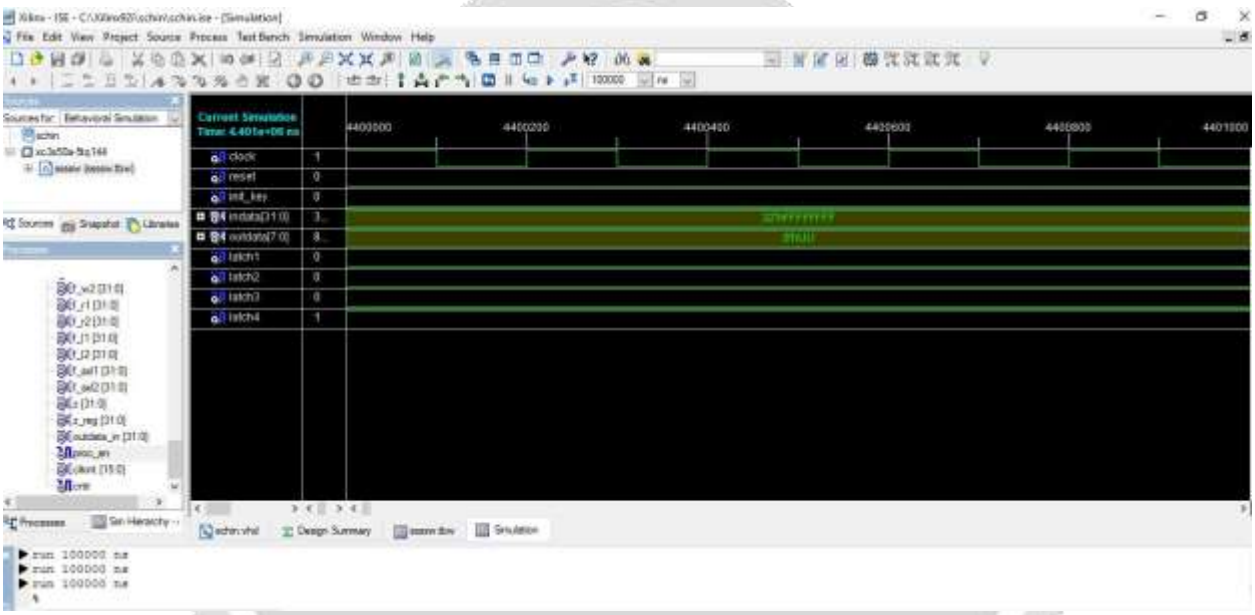


Fig. 5 The feedback logic circuit

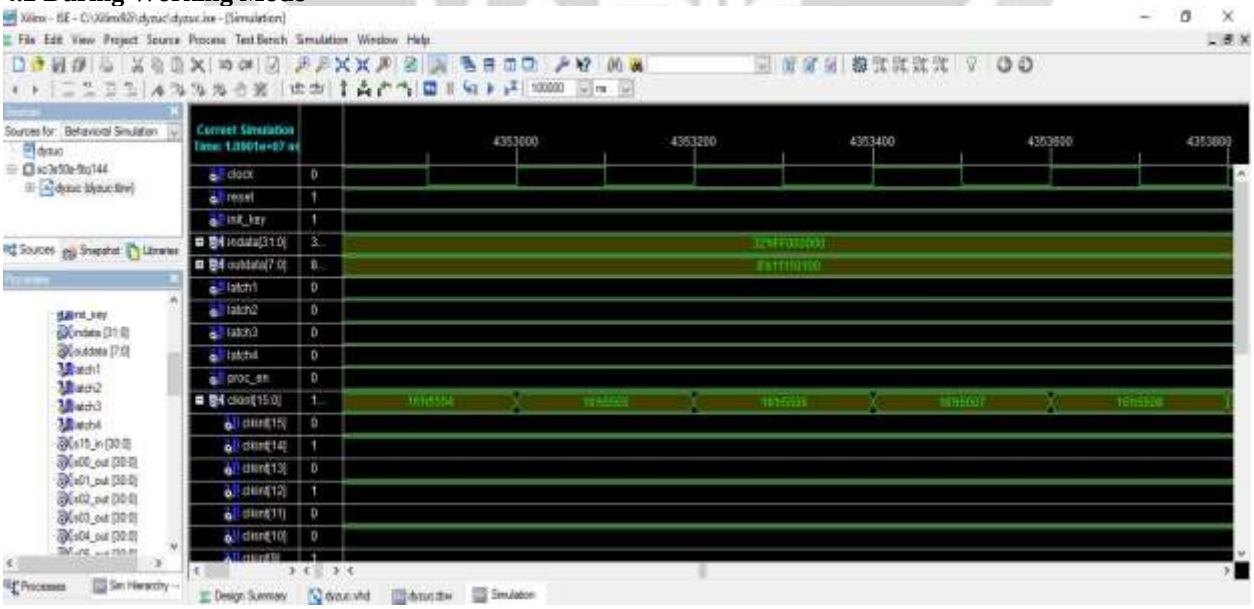
The operation of the proposed ZUC design (see Fig. 2) starts with the initial parallel loading of the LFSR initial values. Also, the values of R1 and R2 registers are set equal to zero. Then, during the initialization stage, the LFSR receives a 31-bit word as input through the multiplexer MUX (input 1 of the multiplexer is selected). This input is produced by the addition mod (231-1).between the 31-bit output of the function F called W (the rightmost bit of the output W is removed, W>>1) and the output of the feedback logic. During this operation the cipher is clocked without producing output. For this reason a 32-bit output register is located at the output of the cipher that holds the produced data. In addition, during the working mode, the LFSR does not receive any new input and input 2 of the multiplexer is selected. The cipher is executed once, and the output W is discarded. After that, the cipher produces a 32-bit keystream, Z, each clock cycle. The keystream produced by bit-by-bit XOR between the W and X3 word that is output of BR layer. In this stage of operation the 32-bit output register latches its input to the output.

4. RESULTS

4.1 During Initialization Mode



4.2 During Working Mode



5. CONCLUSION

In this system, FPGA device is used for implementation of reconfigurable ZUC hardware architecture. It uses Carry Look Ahead Adder which is the highest speed adder as compared to other. The implementation on FPGA achieved a throughput of 3.2180 Gbps. The system will be implemented for data security.

6. ACKNOWLEDGEMENT

The authors wish to thank the anonymous reviewers. The author would like to thanks of SGMSPM's Sharadchandra pawar College of Engineering, Otur, Savitribai Phule Pune University, Pune, Maharashtra, INDIA.

7. REFERENCES

1) "A Flexible and Energy-Efficient Reconfigurable Architecture for Symmetric Cipher Processing" by Bo Wang, Leibo Liu, Institute of Microelectronics Tsinghua University Beijing, China 978-1-4799-8391-9/15 in 2015 IEEE.
2) "A Very High Security Cryptosystem Architecture for Video Application" by Toan Nguyen Van, Thuan Huynh Huu Faculty of Electronics and Telecommunications HCMC University of Science Ho Chi Minh City, Vietnam in 2014 IEEE.
3) "Evaluating the Optimized Implementations of SNOW3G and ZUC on FPGA" by Lingchen Zhang , Luning Xia , Zongbin Liu, Jiwu Jing published in Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on 25-27 June 2013, Page(s): 436 – 442 ,Print ISBN: 9781-4673-2172-3.
4) "Power analysis and optimization of the ZUC stream cipher for LTE-Advanced mobile terminals" by Traboulsi, S.,Pohl, N., Hausner, J., Bilgic, A. published in Circuits and Systems (LASCAS), 2012 IEEE Third Latin American Symposium on Feb. 29 2012-March 2 2012, Page(s):1 - 4 , Print ISBN: 978-1-4673-1207-3.
5) "Analytical evaluation of the stream cipher ZUC" by Orhanou G.,El Hajji S.,Lakbabi A.,Bentaleb Y. published in Multimedia Computing and Systems (ICMCS), 2012 International conference on date of conference: 10-12 May 2012 Page(s): 927 - 930 Print ISBN:978-1-4673-1518-0.
6) "An FPGA Implementation of the ZUC Stream Cipher" by Kitsos, P., Patras, Greece, Sklavos, N., Skodras, A.N. published in Digital System Design (DSD), 2011 14th Euromicro Conference on Date of Conference:Aug. 31 2011-Sept. 2 2011, Page(s): 814 – 817, Print ISBN:978-1-4577-1048-3.
7) "Evaluating Optimized Implementations of Stream Cipher ZUC Algorithm on FPGA" by Lei Wang, Jiwu Jing, Zongbin Liu, Lingchen Zhang, Wuqiong Pan published in Springer Berlin Heidelberg 13th International Conference, ICICS 2011, Beijing, China, November 23-26, 2011.
8) "Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3" Document 1: 128EEA3 and 128-EIA3 Specification; Version: 1.5, 2011.
9) "Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3" Document 2: ZUC Specification; Version: 1.5, 2011.
10) "Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3" Document 3: Implementor's Test Data", Version 1.1, Jan. 2011.