Volume 4, Issue 2, February 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Reconfigurable Hardware for ZUC Stream Cipher

Mr.Praneet R Shah*, Prof.N.B.Hulle

G.H.R.I.E.T Wagholi,Pune..

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). The design is coded using VHDL language and for the hardware implementation, a XILINX Virtex-5 FPGA is used. In this paper a reconfigurable implementation of ZUC stream cipher using Carry Look Ahead Adder is presented. This achieved a throughput of 3.2180 Gbps.

Keywords— Long Term Evolution networks security, ZUC, FPGA, 4G, Virtex 5.

I. 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 (LTE-Advanced), 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 128-EEA3 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.

II. 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. 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,……., 2^{31}-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 15-bit 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:

LFSR With Initialisation Mode (u)

{1. $v=2^{15}s15+2^{17}s13+2^{21}s10+2^{20}s4+(1+2^8)s0 \bmod (2^{31}-1)$;
2. $s16=(v+u) \bmod (2^{31}-1)$;
3. If s16=0, then set $s16=2^{31}-1$;
4. (s1, s2,…, s15,s16)▶ (s0, s1,…, s14,s15) }

In the working mode, the LFSR does not receive any input, and works as follows:

LFSR With Work Mode

{1. $s16=2^{15}s15+2^{17}s13+2^{21}s10+2^{20}s4+(1+2^8)s0 \bmod (2^{31}-1)$;
2. If s16=0, then set $s16=2^{31}-1$;
3. (s1, s2, …, s15, s16)_(s0, s1, …, s14,s15) }

The bit-reorganization layer extracts 128-bit from the cells of the LFSR and forms 4 of 32-bit words, where the first three will be used by the nonlinear function F in the bottom layer, and the last word will be involved in producing the keystream. Let s0, s2, s5, s7, s9, s11, s14, s15 be eight cells of LFSR. Then the bit-reorganization forms four 32-bit words X0, X1, X2, X3 from the above cells as follows: X0=s15H || s14L, X1=s11L || s9H, X2=s7L || s5H and X3=s2L || s0H with respect at the rule that siH means the bits 30...15 and siL means the bits 15…0 of si respectively. The nonlinear function F has two 32-bit memory cells R1 and R2. Let the inputs to F be X0, X1 and X2, which come from the outputs of the bit-reorganization. Then function F outputs a 32-bit word W. The detailed process of F is as follows:

F (X0, X1, X2)

{ 1. W = (X0 ⊕R1) + R2;
2. W1 = R1 + X1;
3. W2 = R2 ⊕X2;
4. R1 = S(L1(W1L || W2H));
5. R2 = S(L2(W2L || W1H)); }

where S is a 32 X 32 S-box and L1, L2 are linear transformations.

The 32X32 S-box S is composed of four 8X8 mini Sboxes, i.e., S=(S0, S1, S2, S3), where S0=S2, S1=S3. The definitions of S0 and S1 can be found in the official cipher specifications. L1 and L2 are linear transformations from 32-bit words to 32-bit words.

For the cipher operation firstly the key loading procedure expands the initial key and the initial vector into 16 of 31-bit integers as the initial state of the LFSR and then two stages are executed; initialization stage and working stage. In the first stage, a Key/IV initialization is performed and the cipher is clocked without producing output. The second stage is a working stage in which every clock cycle produces a 32-bit word of output.

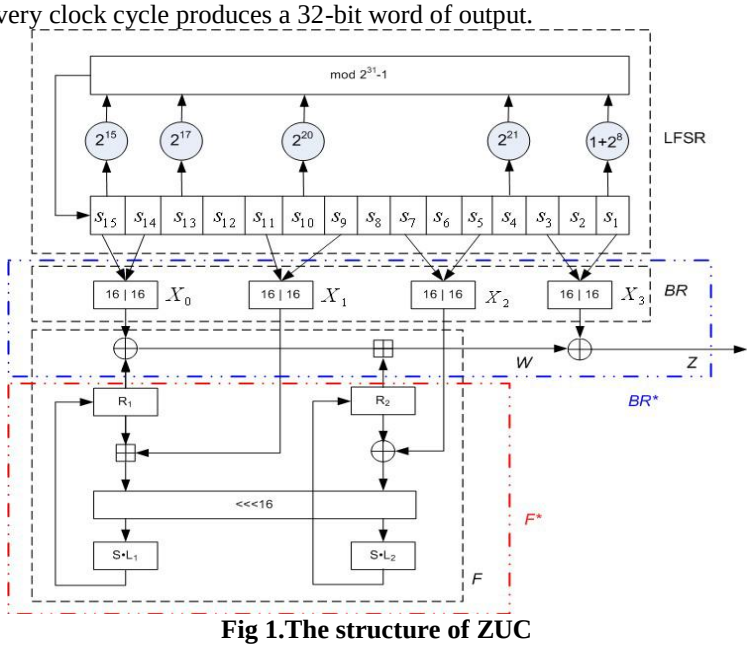

Fig 1.The structure of ZUC

III. 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 (bit-reorganization) 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 $0 \le di \le 15$ are predefined) and to\gether with Key and IV, produce 16 substrings of 31-bit according to the following rule si = ki || di || iv_i ($0 \le i \le 15$). The ki and iv_i 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 S_0,S_1, …, S_{15} 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.

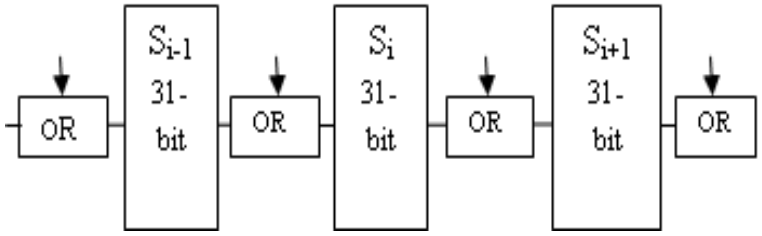

Fig 2. The 31 bit OR gates between the LFSR cells

The BR consists of four simple components that execute concatenations according to the rule described previously. Only wirings are used in hardware.

The function F has two 32-bit registers R1 and R2, two S-boxes, two 32-bit XOR, two32-bit mod 2^{32} adders, two linear transformations L1 and L2 and finally a left cyclical shifter of 16 positions. The transformation L1 performs the operation.

$L_1(X) = X \oplus (X <<<_{32} 2) \oplus (X <<<_{32} 10) \oplus (X <<<_{32} 18) \oplus (X <<<_{32} 24)$ while the L2 performs the operation $L2(X) = X \oplus (X <<<_{32} 8) \oplus (X <<<_{32} 14) \oplus (X <<<_{32} 22) \oplus (X <<<_{32} 30)$.So each transformation uses four 32-bit XOR-gates and four left cyclical shifters.

The Feedback Logic is an arithmetic logic that combines cyclical shifters and additions mod $(2^{31}-1)$. The multiplexer (MUX) changes their configuration according the cipher operation scenario (initialization or working stage). Also, one more adder mod $(2^{31}-1)$ is needed with its result used as first input of the multiplexer. In the Feedback Logic six additions mod $(2^{31}-1)$ are used. The architecture for the two inputs, X, Y, adder mod $(2^{31}-1)$ is depicted in Fig. 3. One adder is used in order to add the values of S0 and 2^8S0, another for the addition of 2^{20}S4 and 2^{21}S10 and another for the addition of 2^{17}S13 with 2^{15}S15. Finally, a three-input adder mod $(2^{31}-1)$ is used to add the three previous sums. For the three-input adder mod $(2^{31}-1)$ two cascaded two-input address mod $(2^{31}-1)$ are used.

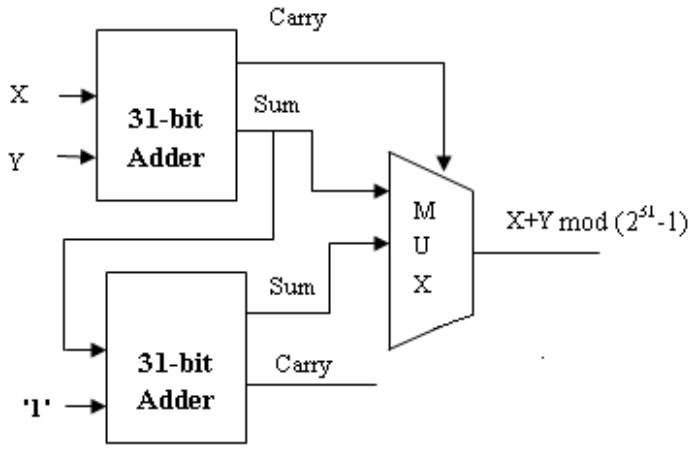

Fig 3. The Architecture of Adder mod $(2^{31}-1)$.

The circuit that executes the Feedback Logic is illustrated in Fig. 4.

$$(S_0+2^8S_0+2^{20}S_4+2^{21}S_{10}+2^{17}S_{13}+2^{15}S_{15}) \bmod (2^{31}-1)$$

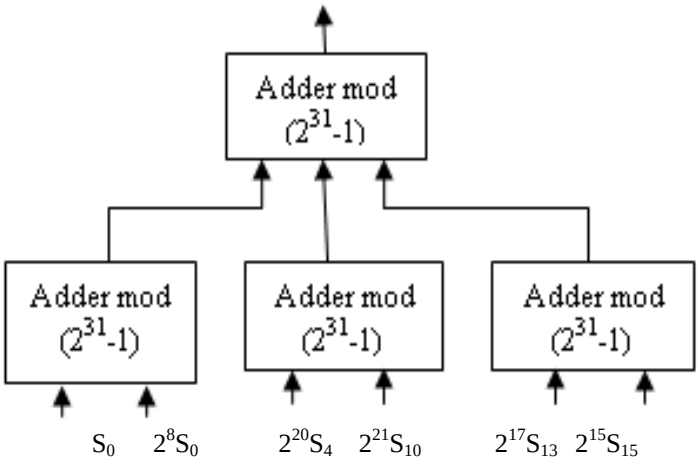

Fig 4. The feedback logic circuit

The operation of the proposed ZUC design (see Fig. 1) 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 (2^{31}-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.

IV. EXPERIMENTAL RESULTS

The system implementation is captured by using VHDL with structural description logic. The VHDL code is simulated and verified by using the test vectors, provided by the 3GPP standard. The implementation has been synthesized with XILINX ISE tool and a Virtex-5 FPGA is used. In previous version of ZUC a pipeline implementation was used. This implementation required more hardware resources but better time performance was achieved compared to this implementation. Three are the basic reasons for this. The first one is the design philosophy of the new version of ZUC. In the new version, the output W of the F function combined with the Feedback Logic by an addition mod (2^{31}-1) rather than by XOR which results in a major increment of the critical path delay. The second reason is that in the ZUC earlier implementation, pipeline registers are used in the Feedback Logic in order to decrease the delay of the critical path. But, this increases the execution of the algorithm latency. The third basic reason is that the initialization stage was implemented in software, which results in a drastically minimizing the hardware resources and the delay of the critical path.

In this paper, the implementation supports all stages of the execution. The initialization stage is executed through the OR-gates between LFSR cells, the adder mod (2^{31}-1) than combines the output of F function (W) with the Feedback Logic and the multiplexer, MUX.

TABLE 1: Comparison of Performance of ZUC

Sr No	Architecture	Frequency	Throughput
1.	**ZUC by Shah and Hulle.**	**100.561 MHz**	**3.218 Gbps**
2.	ZUC by Kitsos, Sklavos and Skodras.	65 MHz	2.08 Gbps
3.	ZUC by Lei Wang, Jiwu Jing, Zongbin Liu, Lingchen Zhang, and Wuqiong Pan.	107 Mhz	3.424 Gbps
4.	ZUC old	222 MHz	7.1 Gbps

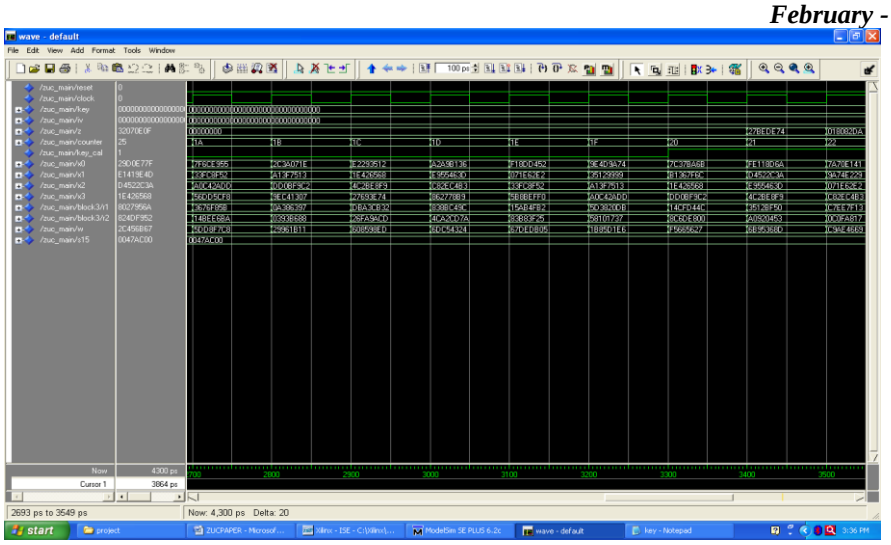

Fig 5: Output window for Key and IV both 0.

The system is implemented with Carry Look Ahead Adder. Cost and processing time of this system is reduced by implementing Carry Look Ahead Adder by using NAND gates, which achieved a throughput equal to 3.2180 Gbps at a 100.561 MHz clock frequency.

IV. 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.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers. The author would like to thanks of research center G.H. Raisoni Institute of Engineering and Technology, University of Pune, Wagholi, Pune, Maharashtra, INDIA.

REFERENCES

1) Paris Kitsos, Nicolas Sklavos and Athanassios N. Skodras "IEEE Standard: "An FPGA Implementation of the ZUC stream cipher".14th Euromicro Conference on Digital System Design, 2011.
2) Thomas Fuhr, Henri Gilbert, Jean-René Reinhard, and Marion Videau," Analysis of the Initial and Modified Versions of the Candidate 3GPP Integrity Algorithm 128-EIA3".
3) Dave Gardner, "Defination of eStream Ciphers and ZUC", May 12, 2011.
4) TANG Ming, CHENG PingPan,QIU ZhenLong, "Differential Power Analysis on ZUC Algorithm".
5) Lei Wang, Jiwu Jing, Zongbin Liu, Lingchen Zhang, and Wuqiong Pan, "Evaluating Optimized Implementations of Stream Cipher ZUC Algorithm on FPGA".
6) Michalis Galanis, Paris Kitsos, Giorgos Kostopoulos, Nicolas Sklavos, and Costas Goutis, " Comparison of the Hardware Implementation of Stream Ciphers", The International Arab Journal of Information Technology, Vol. 2, No. 4, October 2005,267-274.
7) Scott fluhrer,Itsik Mantin and Adi Shamir "Weakness in the key scheduling algorithm for RC4".
8) Sandeep Kumar, Kerstin Lemke, Christof Paar," Some Thoughts about Implementation Properties of Stream Ciphers".
9) Sourav Sen Gupta, Anupam Chattopadhyay, Koushik Sinha,Subhamoy Maitra, Bhabani P. Sinha, Fellow, "High Performance Hardware Implementation for RC4 Stream Cipher",1-15.
10) Jaya Dofe, Manish Patil," Hardware Implementation of Modified RC4 Stream Cipher Using FPGA", IOSR Journal of Engineering (IOSRJEN) ISSN: 2250-3021 Volume 2, Issue 6 (June 2012), PP 1447-1450
11) Allam Mousa and Ahmad Hamad, "Evaluation of the RC4 Algorithm for Data Encryption".
12) P. Kitsos, G. Kostopoulos, N. Sklavos, and 0. Koufopavlou, 'Hardware Implementation of the RC4 Stream Cipher".
13) Vandana Malode, Nagnath Hulle, "Hardware Implementation of RC4 Stream Cipher for Wi-Fi Security".
14) Michalis Galanis, Paris Kitsos, Giorgos Kostopoulos, Nicolas Sklavos, and Costas Goutis, "Comparison of the Hardware Implementation of Stream Ciphers" The International Arab Journal of Information Technology, Vol. 2, No. 4, October 2005.

