An FPGA Implementation of ZUC Stream Cipher

Mr. Berad S. S.[1], Prof. Rahane S.B.[2]

[1]ME Electronics Engineering
Amrutvahini College of Engineering Sangamner (MS), 422605

[2]Asst.Prof. Electronics Engineering
Amrutvahini College of Engineering Sangamner (MS), 422605

Abstract—In this paper a hardware implementation of ZUC stream cipher is presented. ZUC is a stream cipher that forms the heart to the 3 GPP confidentiality algorithm 128-EEA3 and the 3GPP integrity algorithm 128-EIA3, offer in reliable security services in Long Term Evolution networks (LTE). A detailed hardware implementation is presented in order to reach satisfactory performance results in LTE systems. The design was coded using VHDL language and for the hardware implementation, a XILINX Spartan-3FPGAwasused. Experimental results in terms of performance and hardware resources are presented.

Keywords-Long Term Evolution networks security, Zuc stream cipher ,Hardware implementation ,FPGA.

I.INTRODUCTION

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 [1], 128-EEA1 for confidentiality and 128-EIA1 for integrity have been designed by SAGE/ETSI Security Algorithms Group of Experts [2]. 128-EEA1 and 128-EIA1 are based on SNOW3G stream cipher [3]. Also, the 3rd Generation Partnership Project (3GPP), together with the GSMA association specifies a second set of algorithms, 128-EEA2 and 128-EIA2 [4], which are based on AES block cipher [5- 6]. Finally, 3GPP with GSM association specifies a third set of algorithms for confidentiality and integrity the 128-EEA3and 128-EIA3 respectively [7]. Both ciphers are based on ZUC stream cipher [8]. 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 paper an efficient FPGA implementation of ZUC stream cipher is presented.

The advantages of Virtex-5FPGA [9] 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. Our FPGA hardware implementation covers 385 slices and achieves 2.08 Gbps throughput in maximum frequency operation. Comparisons with other stream ciphers implementations [10-12] are provided. The comparisons prove that the proposed system is efficient in terms of throughput. The paper is structured as follows. In Section 2 the ZUC stream cipher specifications are briefly described. In Section 3, the proposed architecture is examined in detail, while in Section 4 synthesis results and comparisons with previously published designs are given. Finally, Section 5 concludes the paper.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. Compared with block ciphers, stream ciphers are more efficient when implemented in hardware environment, like Field Programmable Gate Array (FPGA). In this paper, we propose three optimized schemes in the FPGA implementation of a novel and recently proposed stream cipher, ZUC, which is a new cryptographic algorithm proposed for inclusion in the '4G' mobile standard called LTE (Long Term Evolution). These three schemes are based on reusing area of Sbox, calculation of CSA tree and pipelined architecture to implement ZUC on FPGA respectively. We also evaluate each optimized scheme in terms of performance and consumed area in Xilinx FPGA device to compare their actual hardware efficiency. According to the evaluation results, the third scheme, namely pipelined

architecture implementation, optimizes hardware implementation of ZUC for the best performance and achieves a throughput of 7.1 Gbps using only 575 slices by speeding up the key stream generating on FPGA. To our knowledge, it is an extremely efficient hardware implementation of ZUC at present. Moreover, it also shows that ZUC is quite flexible to balance different throughput with consumed area. The development of a good random number generator has been a hot topic in cryptology. Feedback shift registers have been introduced to fill this requirement. The shift registers must have a very high degree of nonlinearity. An LFSR consists of shift register and feedback function. Due to the linearity of LFSR, we can determine the LFSR which generates any output sequence using the Berlekamp–Massey algorithm by knowing 2n output bits only.

II.SCHEME OF IMPLEMENTATION

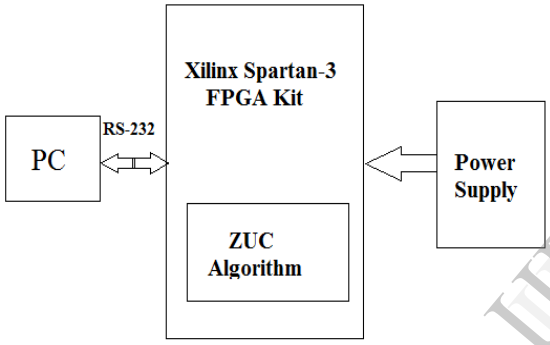

Fig. 1. Block Diagram of System

An FPGA Implementation of ZUC Stream Cipher consist of personal computer, Xilinx Spartan-3 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.

III. ZUC STREAM CIPHER

ZUC is a word-oriented stream cipher [8] that takes a 128-bit Key and a 128-bit Initial Vector (IV) as input, and outputs a key stream of 32-bit words. ZUC has three logical layers. The top layer is a Linear Feedback Shift Register (LFSR) of 16 stages, the middle layer is for bit reorganization (BR), and the bottom layer is a nonlinear function F. 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.

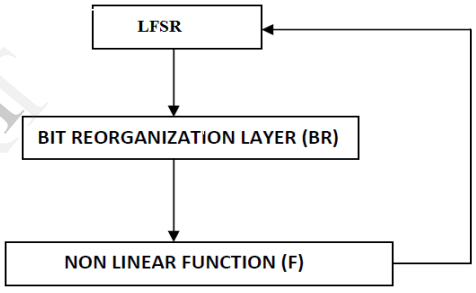

Fig.2: ZUC Stream Cipher using LFSR

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. For a general reference on the subject of LFSRs and related sequence generators, see Klapper and Goreky's book. The most commonly used linear function of single bits is XOR. Thus, an LFSR is most often a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. Applications of LFSRs include generating pseudo-random numbers,

pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common. The mathematics of a cyclic redundancy check, used to provide a quick check against transmission errors, are closely related to those of an LFSR.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 key stream. 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 32×32 S-box S is composed of four 8×8 mini S boxes, i.e., S = (S0, S1, S2, S3), where S0=S2, S1=S3. The definitions of S0and 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 .one technique for removing the linearity is to feed the outputs of several parallel LFSRs into a non-linear Boolean function to form a combination generator. Various properties of such a combining function are critical for ensuring the security of the resultant scheme, for example, in order to avoid correlation attacks.

III ALGORITHM

3.1The linear feedback shift register (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). More specifically, the initialization mode works as follows.

Algorithm 1: LFSRWithInitialisationMode

Input: u

1 begin
2 $v = \{2^{15}S_{15} + 2^{17}S_{13} + 2^{21}S_{10} + 2^{20}S_4 + (1+2^8)S_0\} mod(2^{31}-1);$
3 $S_{16} = v + u \pmod{2^{31}-1};$
4 if $S_{16} = 0$ then
5 then set $S_{16} = 2^{31}-1$
6 $(S_1, S_2, \ldots, S_{15}, S_{16}) \rightarrow (S_0, S_1, \ldots, S_{14}, S_{15});$

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: LFSRWithWorkMode

begin
$S_{16} = \{2^{15}S_{15} + 2^{17}S_{13} + 2^{21}S_{10} + 2^{20}S_4 + (1+2^8)S_0\} mod(2^{31}-1);$
if $S_{16} = 0$ then
then set $S_{16} = 2^{31}-1$
$(S_1, S_2, \ldots, S_{15}, S_{16}) \rightarrow (S_0, S_1, \ldots, S_{14}, S_{15});$

3.2 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. Therefore the bit reorganization stage can mix with the nonlinear function operation together to save clock cycle.

Algorithm 3: Bitreorganization

```
begin
    X0 = S15H || S14L;
    X1 = S11L || S9H;
    X2 = S7L || S5H;
    X3 = S2L || S0H;
```

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*.

3.3 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, X2
1 begin
2     W = (X0 ⊕ R1) ⊞ R2;
3     W1 = R1 ⊞ R2;
4     R1 = S(L1(W1L || W2H));
5     R2 = S(L2(W2L || W1H));
```

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 232 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.

IV. CONCLUSIONS

A ZUC high-speed hardware architecture is described in this paper which has been implemented by means of an FPGA device. Experimental results prove that the ZUC implementation is a flexible solution for LTE applications.

The implementation on FPGA achieves a throughput of 2.08
Gbps at a 65 MHz clock frequency.

REFERENCES

[1] 3G TS 33.401 V 9.3.1 (2010-04) 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3GPP System Architecture Evolution (SAE); Security architecture (Release9).

[2] Specification of the 3GPP Confidentiality and Integrity AlgorithmsUEA2 & UIA2. Document 1: UEA2 and UIA2 Specification ,ETSI/SAGE Specification, Version: 1.1Date: 6th September 2006.

[3] Specification of the 3GPP Confidentiality and Integrity AlgorithmsUEA2 & UIA2. Document 2: SNOW 3G Specification, ETSI/SAGE Specification, Version: 1.1Date: 6th September 2006.

[4] 3GPP System Architecture Evolution (SAE): Security Architecture,3GPP Std. TS 33.401, Rev. 8.2.1, Dec. 2008.

[5] Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards (FIPS) Publication 197, US Nat'l
Inst. Standards and Technology, 2001.

[6] N. Sklavos, X. Zhang, "Wireless Security & Cryptography: Specifications and Implementations", CRC-Press, A Taylor and
Francis Group, ISBN: 084938771X, 2007.

[7] ETSI/SAGE Specification. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-
EEA3 and 128-EIA3 Specification; Version: 1.5; Date: 4th January2011.

[8] ETSI/SAGE Specification. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification; Version: 1.5; Date: 4th January 2011.

[9] XILINX INC. 2006. UG190: Virtex-5 user guide.http://www.xilinx.com/support/documentation/user guides/ug190.pdf.

[10] Z. Liu, L. Zhang, J. Jing and W. Pan, "Efficient Pipelined Stream Cipher ZUC Algorithm in FPGA", The First International Workshop on

ZUC Algorithm, December 2-3, Beijing, China, 2010.
[11] P. Kitsos, G. Selimis, O. Koufopavlou, "High Performance ASIC
Implementation of the SNOW 3G Stream Cipher", IFIP/IEEE VLSISOC
2008 - International Conference on Very Large Scale
Integration (VLSI SOC), Rhodes Island, Greece, October 13-15,
2008.
[12] P. Kitsos and A. N. Skodras, "On the Hardware Implementation of
the MUGI Pseudorandom Number Generator", Fifth International
Symposium on Communications Systems, Networks and Digital
Signal Processing, (CSNDSP'2006), Patras, Greece, 19-21 July,
2006.
[13] International Organization for Standardization, "ISO/IEC 18033-
4:2005: Information Technology – Security Techniques – Encryption
Algorithms – Part 4: Stream ciphers", 2005