High Performance ASIC Implementation of the SNOW 3G Stream Cipher

Paris Kitsos
Computer Science,
Hellenic Open University and
Dept. of Computer Science and Technology,
University of Peloponnese, Greece
e-mail: pkitsos@ieee.org

George Selimis
VLSI Lab, Department of Electrical & Computers Engineering,
University of Patras, Patras, Greece
e-mail: gselimis@ece.upatras.gr

Odysseas Koufopavlou
VLSI Lab, Department of Electrical & Computers Engineering,
University of Patras, Patras, Greece
e-mail: odysseas@ece.upatras.gr

***Abstract*— In this paper a hardware implementation of SNOW 3G stream cipher is presented. SNOW 3G is a stream cipher that forms the heart of the 3GPP confidentiality algorithm UEA2 and the 3GPP integrity algorithm UIA2, offering reliable security services in Universal Mobile Telecommunication System (UMTS). A detailed hardware implementation is presented in order to reach satisfactory performance results in systems such as mobile devices. The design was coded using VHDL language and for the hardware implementation, an ASIC synthesis approach was used. Experimental results in terms of performance and covered area are presented.**

I. INTRODUCTION

The current radio interface protection algorithms for Universal Mobile Telecommunication System (UMTS) [1], UEA1 for confidentiality, and UIA1 for integrity of signaling messages - were designed by SAGE/ETSI Security Algorithms Group of Experts [2]. No weakness has been discovered in these algorithms, and there is no indication that a weakness is likely to be found. However, if one ever were found, it would be much better to have a replacement. So the 3rd Generation Partnership Project (3GPP), together with the GSM Association, called SAGE wishes to specify a second set of algorithms, UEA2 and UIA2. Apart from the obvious requirements on speed and implementation complexity, the main design criterion for these new algorithms was that they should be fundamentally different in nature from UEA1 and UIA1, for cryptanalytic reasons. SAGE delivered the UEA2 and UIA2 specifications in January 2006 [2]. At the heart of these algorithms is the SNOW 3G [3] stream cipher.

The two basic issues in mobile communications security are Data Confidentiality and Data Integrity. The term of Data Confidentiality is referred to keeping information secret from all but those who are authorized to see it while the term of Data Integrity is referred to ensuring information has not been altered by unauthorized or unknown means. Additionally, in mobile systems the constrains in power consumption and chip covered area are very strict while the performance requirements are essential. Therefore, the existence of supplementary hardware is essential [4], [5] if the designer's goal is to construct efficient systems in limited area resources, like in mobile systems.

In this paper an efficient implementation of the SNOW 3G cipher is presented. The proposed hardware system has been implemented using only the main functionality of the algorithm. It uses the feedback logic in order to support the basic operation scenarios. Also, we improve and accelerate the complex internal operations of the system in order to have a performance efficient system and compatible with the current wireless-cellular communication standards. Our ASIC hardware implementation covers 25016 nand(2:1) equivalent gates and it achieves 7.97 Gbps throughtput in maximum frequency operation. Comparisons with other stream ciphers implementations [6-11] are provided. The comparisons prove that the proposed system outperforms in terms of throughtput efficiency.

This paper is organized as follows: In Section II the SNOW 3G block cipher is described briefly. The proposed architectures and VLSI implementations are presented in detail in Section III. Comparisons with other works and the ASIC synthesis results are shown in Section IV. The paper is concluded in section V.

II. THE SNOW 3G ALGORITHM

SNOW 3G is a word-oriented stream cipher that generates a sequence of 32-bit words under the control of a 128-bit key and a 128-bit initialization variable. First a *key initialization* is performed and the cipher is clocked without producing output. Then the cipher operates in *key-generation* mode and it produces a 32-bit ciphertext / plaintext word output in every clock cycle.

First we denote the functions MUL_α , DIV_α and *MULxPOW* in order to completely define the following equations.

The function MUL_α maps 8 bits to 32 bits. Let *c* be the 8-bit input, then MUL_α is defined as:

$MUL_\alpha(c)$ = (MULxPOW(c, 23, 0xA9) || MULxPOW(c, 245, 0xA9) || MULxPOW(c, 48, 0xA9) || MULxPOW(c, 239, 0xA9)).

The function DIV_α maps 8 bits to 32 bits. Let c be the 8-bit input, then DIV_α is defined as:

$DIV_\alpha(c)$ = (MULxPOW(c, 16, 0xA9) || MULxPOW(c, 39, 0xA9) || MULxPOW(c, 6, 0xA9) || MULxPOW(c, 64, 0xA9)).

MULxPOW maps 16 bits and a positive integer *i* to 8 bit.

Let V and c be 8-bit input values, then MULxPOW(V, i, c) is recursively defined:

If *i* is equal to 0, then MULxPOW(V, i, c) = V, else MULxPOW(V, i, c) = MULx(MULxPOW(V, i - 1, c), c).

The symbol "||" denotes concatenation between two bitstreams.

In the *initialization mode* the LFSR receives a 32-bit input word F, which is the output of the FSM. The LFSR has 16 internal stages, each storing a 32-bit word. The main functionality is denoted by the following pseudo code.

Let $s_0 = s_{0,0} \parallel s_{0,1} \parallel s_{0,2} \parallel s_{0,3}$ with $s_{0,0}$ being the most and $s_{0,3}$ being the least significant byte of s_0. Also let $s_{11} = s_{11,0} \parallel s_{11,1} \parallel s_{11,2} \parallel s_{11,3}$ with $s_{11,0}$ being the most and $s_{11,3}$ being the least significant byte of s_{11}.

Compute the intermediate value *v* as:

$v = (s_{0,1} \parallel s_{0,2} \parallel s_{0,3} \parallel 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \parallel s_{11,0} \parallel s_{11,1} \parallel s_{11,2}) \oplus DIV_\alpha(s_{11,3}) \oplus F$,

Finally, set,

$s_0 = s_1$, $s_1 = s_2$, $s_2 = s_3$, $s_3 = s_4$, $s_4 = s_5$,

$s_5 = s_6$, $s_6 = s_7$, $s_7 = s_8$, $s_8 = s_9$, $s_9 = s_{10}$,

$s_{10} = s_{11}$, $s_{11} = s_{12}$, $s_{12} = s_{13}$, $s_{13} = s_{14}$, $s_{14} = s_{15}$,

$s_{15} = v$. The s_i (i=*0, 1, …, 15*) are the internal states of the LFSR.

In the *Keystream Mode* the LFSR does not receive any input. The main functionality is denoted by the below equations. The main functionality is denoted by the following pseudo code.

Compute the intermediate value *v* as

$v = (s_{0,1} \parallel s_{0,2} \parallel s_{0,3} \parallel 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \parallel s_{11,0} \parallel s_{11,1} \parallel s_{11,2}) \oplus DIV_\alpha(s_{11,3})$.

And finally set,

$s_0 = s_1$, $s_1 = s_2$, $s_2 = s_3$, $s_3 = s_4$, $s_4 = s_5$,

$s_5 = s_6$, $s_6 = s_7$, $s_7 = s_8$, $s_8 = s_9$, $s_9 = s_{10}$,

$s_{10} = s_{11}$, $s_{11} = s_{12}$, $s_{12} = s_{13}$, $s_{13} = s_{14}$, $s_{14} = s_{15}$,

$s_{15} = v$.

III. PROPOSED ARCHITECTURE

In Figure 1 the proposed top level hardware architecture of SNOW 3G is presented. The proposed system has as main I/O interfaces a 32-bit plaintext/ciphertext input and a 32-bit ciphertext/plaintext output. In addition it has two input parameters values, a secret key and initialization value IV. The IV value is considered as a four word value IV = (IV_3, IV_2, IV_1, IV_0), where IV_0 is the least significant one. The secret key K also, is considered as a four word value K = (K_3, K_2, K_1, K_0), where K_0 is the least significant one. We have implemented the basic instructions of SNOW 3G algorithm achieving an area efficient system. However, we have increased the performance of some instructions in order to achieve a fair trade-off between system resources and performance. Our proposed hardware system supports both key-initialization and key generation processes. As a set of control logic change the configuration of the proposed hardware system supports both key-initialization and key generation processes.

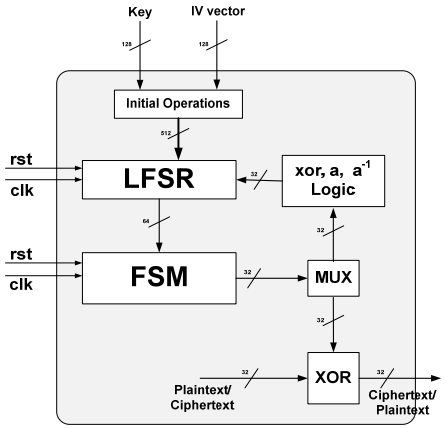

Figure 1: The proposed Top Level architecture of SNOW 3G

The main parts of the proposed architecture of SNOW 3G are the *Initial Operations*, the *Linear Feedback Shift Register (LFSR)*, the *Finite State Machine (FSM)* and the *Feedback Logic Unit*. The LSFR consists of sixteen register stages named s_0, s_1, $s_{2...}$, s_{15} each holding 32 bits. The FSM has three 32-bit registers *R1*, *R2* and *R3*, two S-boxes *S1* and *S2*, two 32-bit *XOR* and two 32-bit *modulo adders*. The initial operation component includes a complicated XOR tree according Table I. The Feedback Logic is an arithmetic logic that combines XOR operations and finite field multiplication and inversion. The multiplexer (MUX) component changes their configuration according the cipher operation scenario.

As we referred in Section II, the SNOW 3G cipher operates in two modes. First the key initialization mode occurs, which is presented in Figure 2. During key initialization process the LFSR and the internal FSM registers fetch their initial values. Firstly, the Key and the IV vectors are been transformed according Table I transformation table, where "1" denotes a 32-bit vector of ones. The initial values stored at LFSR stages through OR gates. Additionally, the R1, R2 and R3 FSM registers are set to zero.

Table I: The initial values of the LFSR

$s_{15} = \mathbf{k_3} \oplus \mathbf{IV_0}$	$s_{14} = \mathbf{k_2}$	$s_{13} = \mathbf{k_1}$	$s_{12} = \mathbf{k_0} \oplus \mathbf{IV_1}$
$s_{11} = \mathbf{k_3} \oplus \mathbf{1}$	$s_{10} = \mathbf{k_2} \oplus \mathbf{1} \oplus \mathbf{IV_2}$	$s_9 = \mathbf{k_1} \oplus \mathbf{1} \oplus \mathbf{IV_3}$	$s_8 = \mathbf{k_0} \oplus \mathbf{1}$
$s_7 = \mathbf{k_3}$	$s_6 = \mathbf{k_2}$	$s_5 = \mathbf{k_1}$	$s_4 = \mathbf{k_0}$
$s_3 = \mathbf{k_3} \oplus \mathbf{1}$	$s_2 = \mathbf{k_2} \oplus \mathbf{1}$	$s_1 = \mathbf{k_1} \oplus \mathbf{1}$	$s_0 = \mathbf{k_0} \oplus \mathbf{1}$

Then a clocking process starts. The FSM produces the output F= ($s_{15} \boxplus$ R1) $\oplus$ R2, where $\boxplus$ symbol denotes the integer addition modulo 2^{32} operation and $\oplus$ symbol denotes the 32-bit bitwise XOR operation. The FSM registers are updated according the following equations: r=R2 $\boxplus$ (R3 $\oplus$ s_5) where R3 =S2(R2), R2=S1(R1) and R1=r, where S1 and S2 are the S-box transformations. The LFSR receives, as input, the 32-bit output (denoted as F in the below equations) word from the FSM, and its s_{15} value is updated according the equation $s_{15}=a^{-1}s_{11} \oplus s2 \oplus s_0 a \oplus F$, where α be a root of the irreducible GF(2^8)[x] polynomial $x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239}$. The

clocking process takes 32 clock cycles in order the initialization process to be completed.

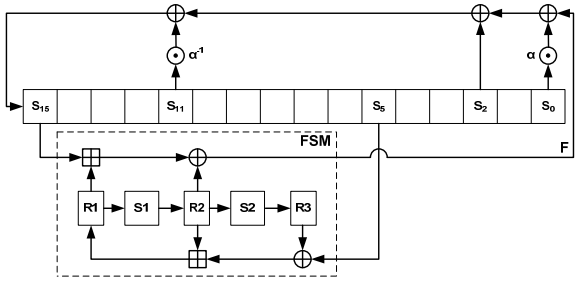

Figure 2: SNOW 3G algorithm during key initialization

In Figure 3 the keystream-generation operation of SNOW 3G is presented. After the key-generation process the system is up to process the data in order to encrypt or decrypt. First, the cipher is clocked once and the output is discarded. Finally, the produced output sequence, called running key, is added bitwise to the plaintext sequence. The result is the ciphertext sequence. In decryption, the same operation is done. In every clock cycle the 32-bit ciphertext word $z_t=F \oplus s_0$ is produced.

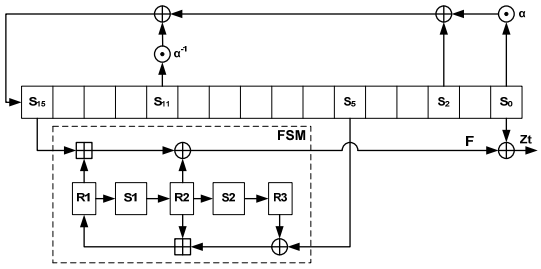

Figure 3: SNOW 3G algorithm during key-generation mode

In Figure 4 the proposed hardware implementation of SNOW 3G is presented in detail. The operation of the proposed design starts with the initial parallel loading of the LFSR initial values. This is the reason of using the OR-gates (Figure 5). After, the initial values is fetched the input vector is forced equal to zeros.

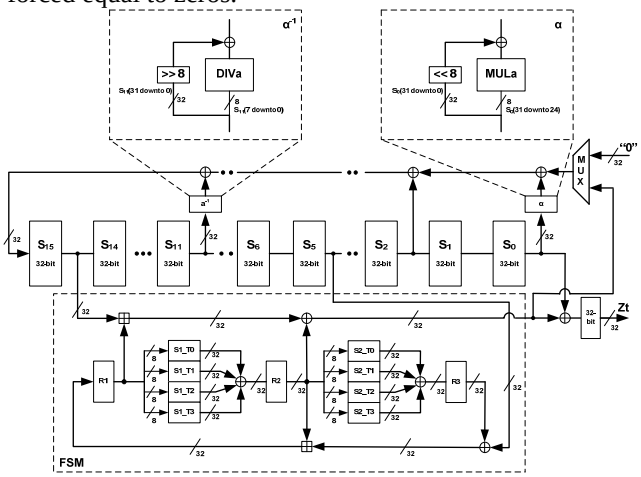

Figure 4: Proposed architecture of the SNOW 3G stream cipher

Two main elements α and α^{-1}, where α be a root of the irreducible GF(2^8) [x] polynomial $x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239}$ are involved in Feedback Logic. The multiplication with α is equal to a byte shift of a 32-bit word to the left and an XOR with the result of MUL_α. Similarly a multiplication with α^{-1} can be implemented as a byte shift of a 32-bit word to the right and an XOR with the result of DIV_α. The functions MUL_α and DIV_α are based in finite field arithmetic and they can be implemented as lookup table from the tables MUL_{alpha} and DIV_{alpha} defined in [3]. The optimization process of such implementation is based exclusively on the synthesis tool and assorted used libraries. Therefore, the power consumption is also depended on the synthesizer libraries. Since the synthesizer can run a highly sophisticated optimization process, the resulting circuit has small critical path delay and low power consumption.

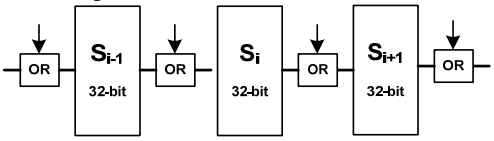

Figure 5: The OR-gates between the LFSR states

Two main operations of the SNOW 3G cipher are S-boxes 1 and 2. The S-boxes of SNOW 3G cipher are the main sources of no-linearity of the cryptographic system. The S-box S1 is based on the round function of RIJNDAEL, while the S-box S2 based in the S_Q. The S_Q is constructed using the Dickson polynomial $g_{49}(x) = x \oplus x^9 \oplus x^{13} \oplus x^{15} \oplus x^{33} \oplus x^{41} \oplus x^{45} \oplus x^{47} \oplus x^{49}$. For an 8-bit input x in GF(2^8) defined by the polynomial $x^8 + x^6 + x^5 + x^3 + 1$ the 8-bit output of S_Q corresponds to $g_{49}(x) \oplus 0x25$.

In the case of S1 S-box, consider an 32-bit input word $w = w_0 \| w_1 \| w_2 \| w_3$ where w_0 is the most and w_3 the least significant byte. In order to compute $S_1(w)$ we use 4 table lookups from the tables S1_T0, S1_T1, S1_T2 and S1_T3 which defined [3]. Each of these tables maps 8 bits to 32 bits. Then $S_1(w)$ is computed according to $S_1(w) = S1\_T0(w_3) \oplus S1\_T1(w_2) \oplus S1\_T2(w_1) \oplus S1\_T3(w_0)$. For the S2 S-box is considering a 32-bit input word $w = w_0 \| w_1 \| w_2 \| w_3$ where w_0 is the most and w_3 the least significant byte. In order to compute $S_2(w)$ we use 4 table lookups from the tables S2_T0, S2_T1, S2_T2 and S2_T3 also defined in [3]. Each of these tables maps 8 bits to 32 bits. Then $S_2(w)$ is computed according to $S_2(w) = S2\_T0(w_3) \oplus S2\_T1(w_2) \oplus S2\_T2(w_1) \oplus S2\_T3(w_0)$.

Another critical operation in SNOW 3G cipher is the 32-bit modulo addition. Two adders are found in the FSM component. In order to achieve satisfactory throughput results we implement these two functions using carry look-ahead adders. In Figure 6 the implementation of the carry look ahead adder is presented. This adder produces the result instantly. With the view to implement a performance efficient system, 32-bit modulo adders are implemented under high covered area architectures. If a multi-cycle adder architecture was used then 32-bit extra clock cycles would be inserted. This fact has as consequence a dramatic decrease of the overall performance. Additionally, we achieve to omit any additional control logic, that a multi-cycle adder architecture requires.

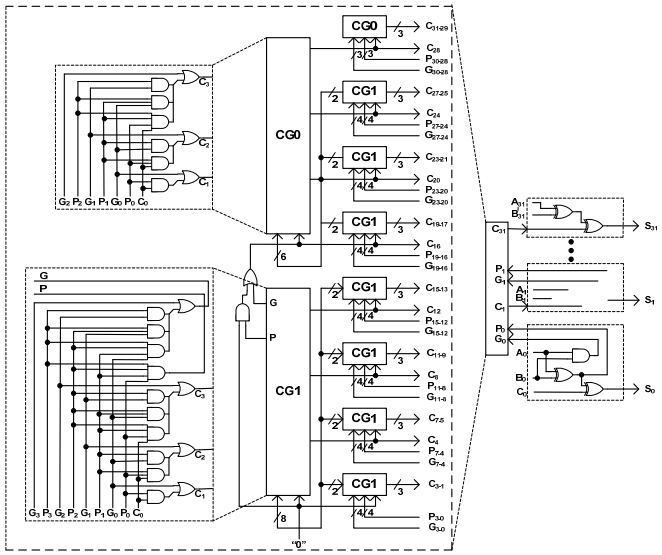

Figure 6: The carry look ahead adder architecture

The multiplexer select the key initialization operation or the keystream-generation of SNOW 3G. A 32-bit register is located at the cipher output that does not latch its input bits during the initialization process.

IV. EXPERIMENTAL RESULTS AND COMPARISONS

The proposed implementation has been synthesized with Synopsys synthesis tool Design Compiler and the target technology library is 0.13 μm with 1.2V core voltage. Our implementation covers 25016 nand(2:1) equivalent gates and its critical path is 4.03 nsec. Then the maximum achieved frequency of our system is 249 Mhz. The high value of the achieved throughput is mainly derived from the short critical path of 32-bit modulo adder, which is the basic factor of the whole critical path. The high value of achieved throughput is not realistic for wireless communications but it allows a reduction of frequency operation at least one order in order to reduce the system power dissipation.

The throughtput is estimated after the initialization phase. Performance comparisons between the proposed system and previous published architectures are shown in Table II. We found only one implementation of the SNOW 3G stream cipher. So, comparison not only with this implementation [6] is given, however comparisons with others similar ciphers/generators [7]-[11] are given in order to have a fair and detailed scene of the proposed system.

Table II: Experimental Results

Implementation	Frequency (MHz)	Throughput (Mbps)
Proposed SNOW 3G	249	7968
SNOW 3G [6]	100	2500
MUGI[7]	110	7000
SNOW 2.0 (ROM)[8]	141	4,512
MICKEY 128[8]	166	166
SNOW 2.0 [9]	-	5,659
LILI-II [10]	158.5-366	158.5-366
SNOW 1.0 [11]	66.5	2128

The SNOW 3G implementation in [6] achieves a throughtput equal to 2500 Mbps at a 100 MHz clock frequency and cover up to 15K ASIC gates. The proposed one outperformed this implementation. However is less compact. Also, the proposed implementation outperforms the MUGI implementation in [7]. The MICKEY 128 [8] has been proposed in eStream [12] workshop. LILI-II [10] also uses a 128-bit key. In [8] and [9] two hardware implementations of the SNOW 2.0 cipher are proposed. Finally, a hardware implementation of the initial SNOW 1.0 cipher is presented in [11]. As the above table illustrates the proposed system implementation outperform all the compared hardware integrations of the other stream ciphers.

V. CONCLUSIONS

In this paper, an efficient ASIC architecture of the SNOW 3G stream cipher is presented. The target technology library is 0.13 μm with 1.2V core voltage and achieves a throughtput up to 7.97 Gbps at a maximum clock frequency to 249 MHz. The proposed architecture is superior for hardware integration, compared with other well known stream ciphers hardware architectures. Experimental results prove that the SNOW 3G is a very good solution not only for 3G mobile devices however also for applications with high speed demands.

REFERENCES

[1] 3GPP TS 33.102 V 4.2.0, Technical Specification Group Service and System Aspects, 3G Security Arcitecture, September 2001.

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

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

[4] J.M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits (2nd Edition), Prentice Hall, 2003.

[5] C. Piguet, Low-Power Electronics Design, CRC, 2004.

[6] Elliptic Semicondutor Inc. “CLP-41 SNOW 3G Cipher Core”, on line available at http://www.ellipticsemi.com/products-clp-41.php.

[7] P. Kitsos, A. N. Skodras, “On the Hardware Im[plementation of the MUGI Pseudorandom Number Generator”, In Proc. Of the Fifht International Symposium Communications Systems, Networks and Digital Signal Processing (CSNDSP 2006, Patras, Greece, 19-21 July, 2006.

[8] P. Kitsos, "Hardware Implementations for the ISO/IEC 18033-4:2005 Standard for Stream Ciphers" International Journal of Signal Processing (IJSP), Vol. 3, Number 1, pp: 66-73, 2006.

[9] P. Leglise, F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, “Efficient implementation of recent stream ciphers on reconfigurable hardware devices”, In Proc. of 26th Symposium on Information Theory in the

[10] P. Kitsos, N. Sklavos, O. Koufopavlou, “A High-Speed Hardware Implementation of the LILI-II Keystream Generator”, In Proc. Of the Fifht International Symposium Communications Systems, Networks and Digital Signal Processing (CSNDSP 2006), Patras, Greece, 19-21 July, 2006.

[11] K. Alexander, R. Karri, I. Minkin, K. Wu, P. Mishra, X. Li, “Towards 10-100 Gbps Cryptographic Architectures”, in proc. of CATT/WICAT Annual Research Review, 2003.

[12] eSTREAM, the ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/