An FPGA-Based Implementation of the Pomaranch Stream Cipher

Paris Kitsos
Computer Science
School of Science & Technology
Hellenic Open University (HOU)
Tsamadou 13-15, Patras,
GR-26222, Greece
+30 2610 367535
pkitsos@ieee.org

Odysseas Koufopavlou
VLSI Design Laboratory
Department of Electrical and Computer Engineering
University of Patras
Rio, GR-26500, Greece
+30 2610 997318
odysseas@ece.upatras.gr

ABSTRACT
As the versatility of small, low power devices increases and their use becomes commonplace, a need for secure communications among these devices has arisen. Pomaranch is a recently developed stream cipher with two major advantages: (i) the low hardware complexity, which results in small area and (ii) the good statistical properties. This architecture supports an 80-bit key and 32- to 108-bit IV. FPGA devices were used for the performance demonstration. A maximum throughput equal to 279 Mbps can be achieved, with a clock frequency of 279 MHz.

Keywords
eStream project, stream cipher, Pomaranch, FPGA, hardware implementation.

1. INTRODUCTION
Everyday, technology is innovating the way people interact among each other. In the last years, the telecommunications and generally the wireless networks have revolutionized the way people communicate, and for the first time it gives the customers the feeling of being virtually connected. It is this closeness or convenience that has made current wireless networks so successful. The more we get used to a communication tool, the more we trust it. For good or for bad, this is the reality today, and current technology aims to be even more intimate in the way to interact with people. However, in the last few years some concern has been raised about the security strength of the wireless networks.

Radio-frequency identification (RFID) [1] is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags. An RFID tag is an object that can be attached to or incorporated into a product, animal, or person for the purpose of identification using radio waves. A key question has been the feasibility, security, and privacy of item-level tagging, in which each individual item is given its own RFID tag. Many concerns have raised over the privacy implications of item level tagging.

Also, the simplicity of the algorithms design is major factor that is simple in software implementation but in the hardware implementation might be quite complex. Not only RFID tags however smart cards, Bluetooth and many others are typical examples of products where the amount of memory and power is very limited. The hardware implementations of today's algorithms, such as AES [2] or Triple-DES [3] block ciphers, are inefficient for devices with limited hardware area. So, stream ciphers are used in cases that the low hardware complexity is necessitated.

Figure 1 shows the general diagram of the stream cipher process with stream cipher [4]. The stream cipher usually take two parameters, the secret key, K, and the initialization vector, IV, and produce keystream bits, z_t. In stream encryption each plaintext symbol, P_t, is encrypted by applying a group operation with a keystream symbol, z_t, resulting in a ciphertext symbol c_t. In modern cipher the operation is the simple bitwise XOR.

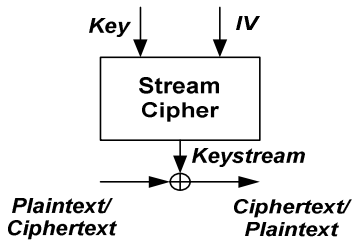


Figure 1. Stream Cipher Diagram.

Decryption takes the subtraction of the keystream symbol from the ciphertext symbol. With the bitwise XOR this is the same operation.

$$m^t = c^t \text{ Å } z^t$$

MOBIMEDIA 2007, August 27-29, Nafpaktos, Greece

DOI 10.4108/ICST.MOBIMEDIA2007.1679

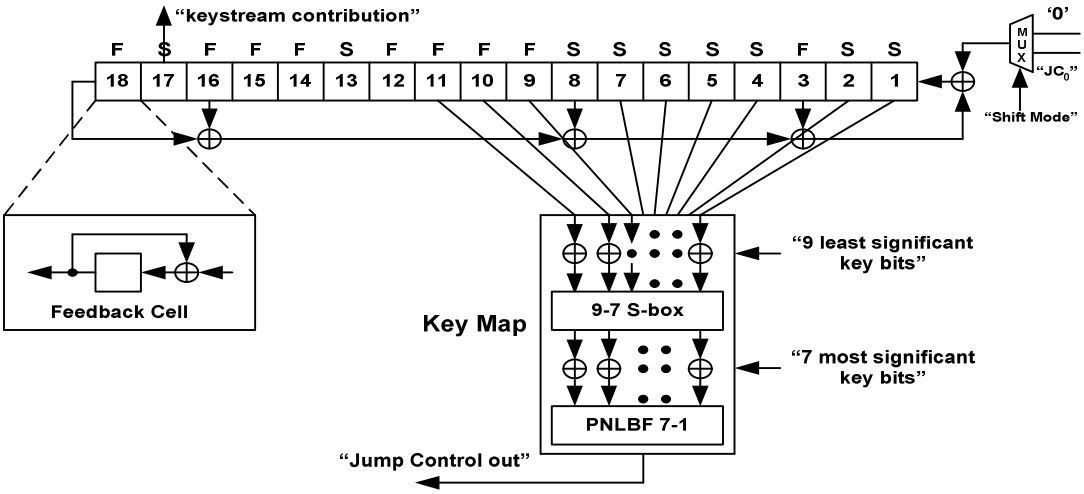
In this paper the implementation on hardware of a new stream cipher called Pomaranch [5] is investigated. This cipher has been submitted and it has been under consideration from the ECRYPT (European Network of Excellence for Cryptology), project [6]. The stream cipher Pomaranch is aimed at area-restricted hardware environments where a key size of 80 bits is required.

The following of this paper is structured as follows: After a short introduction of Pomaranch stream cipher, the hardware design and architecture approach are presented. In the next section, performance analysis results and comparisons with previous published stream ciphers are given. Finally, section 5 concludes the paper.

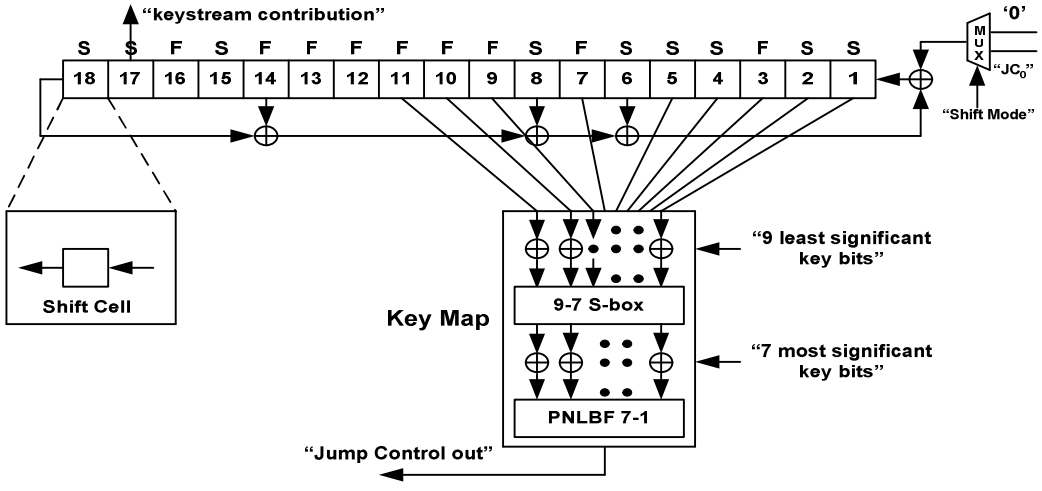
2. POMARANCH STREAM CIPHER

Pomaranch is a stream cipher that follows a classical design of synchronous bit-oriented stream ciphers and consists of a keystream generator producing a secure sequence of bits that is further XORed with the plain text previously converted into bits. The keystream generator of Pomaranch is called Cascade Jump Controlled Sequence Generator (CJCSG) and is primarily intended for hardware implementation. Along with providing an appropriate security level it can be used in a wide range of hardware platforms included those having very limited computing and memory resources.

The CJCSG is a binary one clock pulse cascade clock control sequence generator with a bit stream output that operates in the Initialization Value (IV) accommodation mode. The CJCSG consists of five sections plus the incomplete sixth section that has the Jump Register (JR) only. Hereafter the total number of sections is denoted N, thus N = 6. The sections are numbered from 1 to N and every section having odd number is of type 1 (Figure 2a) and having even number is of type 2 (Figure 2b).



(a)



(b)

Figure 2. Jump Register Section (a) Type 1 (b) Type 2.

Type 1 is used for the odd numbered sections and type 2 for the even numbered sections. The feedback taps of the type 1 jump registers are taken from cells number 3, 8, 16 and 18. The type 2 jump registers have feedback taps at cells 6, 8, 14 and 18. The positions of the F- and S-cells in the type 1 registers are FSFFFSFFFFSSSSSFSS. The type 2 F- and S-cell positions are SSFSFFFFFFSFSSSFSS. Input to the Key Map of the type 1 jump registers is taken from cells 1, 2, 4, 5, 6, 7, 9, 10, 11 while the input to the Key Map of the type 2 jump registers is taken from cells 1, 2, 3, 4, 5, 7, 9, 10, 11. The key-stream contribution is taken from the cell 17 of the jump registers. S-box is defined by the inversion operation in the multiplicative group of $GF(2^9)$ when the finite field is defined by the irreducible polynomial $f(x) = x^9 + x + 1$.

The Key Map implements a key-dependent filter function on the state of the JR and contains a 9-to-7 bit S-box and a balanced nonlinear Boolean function of 7 variables (PNLBF). Finally, the outputs from jump register sections 1 to 5 is replaced with the nonlinear function G which output is XORed to the contribution from section 6.

3. PROPOSED ARCHITECTURE

The architecture that performs the Pomaranch stream cipher is shown in Figure 3. Consists of 6 JR sections, 5 1-bit 2x1 multiplexers (MUX), 4 1-bit 2x1 XOR gates and finally the H function. H function performs the operation of the nonlinear function G which output is XORed to the contribution from section 6.

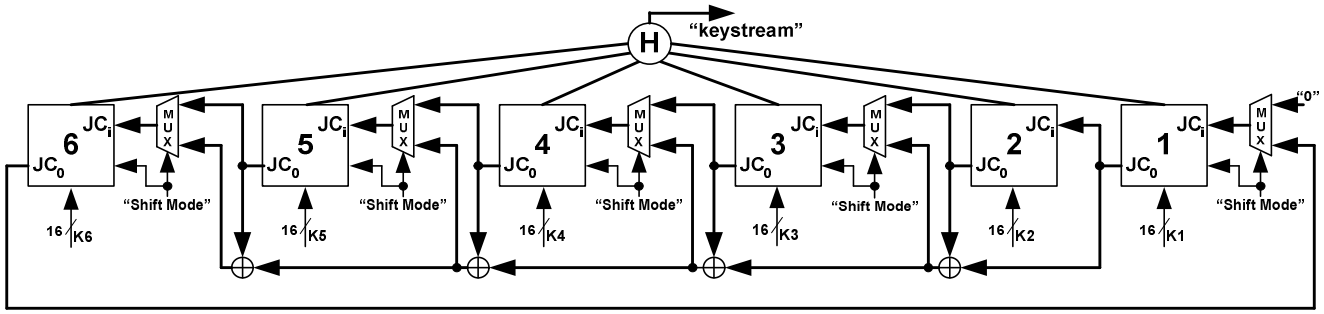


Figure 3. Pomaranch Stream Cipher Architecture.

The multiplexers dictated by the "shift mode" signal and passing to each MUX output the previous JC_0 input during the shift mode operation or the output of the XOR gate during the keystream generation mode of operation.

The operation of the proposed architecture is described in later lines. Firstly the initialization phase is executed. The initialization phase consists of key setup, IV setup and the run-up. During the key setup the state of the jump registers preset with some values according the cipher specifications. Then run the generator for 128 steps in the Shift Mode. Finally, save the 18-bit states of all 6 jump registers (call it the Initialization Vector) for later use during the IV setup. During the Shift Mode the JCo (the Key Map output) of section i (i = 1, . . . ,5) is added to the feedback of the R_{i+1}. The tap from cell 1 in the R_5 is added to the feedback of the R_1 and this closes "the big loop". The configuration of the jump registers does not change in the Shift Mode, they all operate as if the JC bit was constantly zero. The sequence of steps for the IV setup is the following:

1. The IV can have an arbitrary length in the range from 32 to 18N bits. If the IV length is less than 18N then extend the IV to 18N bits by cyclically repeating its bits.
2. XOR the 18N-bit (extended) IV with the Initialization Vector saved after the key setup and load the result into the N jump registers. The 18 most significant bits of the IV modify R_1 (msb of the IV modifies the msb of R_1), the next 18 bits of the IV similarly modify R_2 and so on.
3. Run the generator in the Shift Mode for S = 88 steps.
4. If any of the N registers has the all-zero state then set its least significant bit to 1.

Finally, perform a run-up of 64 steps in the Key-Stream Generation Mode discarding the output bits.
Whether, the initialization phase completed the CJCSG starts generating the key-stream in the Key-Stream Generation Mode.

The JRs are implemented according to the figures 2 and 3. Both S-boxes and PNLBF functions are implemented with Look-Up-Tables (LUTs).

4. PERFORMANCE ANALYSIS RESULTS

The proposed architecture (Figure 3) was captured by using VHDL with structural description logic. The VHDL code was synthesized for Xilinx FPGA devices (Virtex-E, Virtex-II and Virtes-IV respectively) [7]. The synthesis results and performance analysis are shown in Table 1 indicating the number of D Flip-Flops (DFFs), Configurable Logic Blocks (CLBs) slices and Function Generators (FGs).

Table 1. Synthesis Results.

FPGA DEVICE	# CLBs	# FGs	# FFs	Freq. (MHz)	Throughput (Mbps)
V50ECS144	368	735	108	89	89
2V40CS144	363	725	108	111	111
4VFX12SF363	341	682	108	278	278

The throughput is estimated after the initialization phase. The smallest FPGA devices with low hardware resources utilization by the each FPGA family were used.

Performance comparisons between the proposed system and previous published architectures are shown in Table 2. No other implementation of the Pomaranch stream cipher has been previously published. So, comparisons with others synchronous stream ciphers [8-13] are given in order to have a fair and detailed comparison of the proposed system.

Table 2: Hardware Performance Comparisons.

Stream Cipher	FPGA Device	F (MHz)	Throughput (Mbps)
A5/1 [8]	2V250FG25	188.3	188.3
E0 [9]	2V250FG25	189	189
Grain [10]	XC3S50PQ208	193	193
Grain [11]	XC2V6000-4FF1152	181	181
Trivium [10]	XC3S400FG320	201	201
Trivium [11]	XC2V6000-4FF1152	207	207
Phelix [10]	XC3S200Ft256	46	1472
Phelix [11]	XC2V6000-4FF1152	62.5	1000
MICKEY [11]	XC2V6000-4FF1152	200	200
MICKEY [12]	XCV50ECS14	170	170
Edon80 [13]	XC4VLX15	286	3.58
Proposed	V50ECS144	89	89
Proposed	2V40CS144	112	112
Proposed	4VFX12SF363	279	279

In [8], a hardware implementation of the well-known A5/1 cipher is presented, which is used in GSM mobile phones while in [9], the E0 algorithm that Bluetooth system used is presented. In [10] and in [11] the hardware implementations of a new stream cipher, Trivium, Phelix and MICKEY are shown respectively. These implementations have been shown in the latest stream cipher workshop [6]. In [12] and in [13] two hardware implementations of the MICKEY and Edon80 are presented. These implementations have been submitted in eStream project.

As the above table illustrates the proposed cipher implementation achieves competitive frequency and many times better time performance compared with the others. All in all the cipher achieves a low level of FPGAs utilization, complimentary hardware efficiency and its synthesis results proves that is suitable for area restricted hardware implementations.

5. CONCLUSION

An efficient hardware implementation of the new stream cipher named Pomaranch was presented in this paper. This cipher has been submitted and has been under consideration from the eStream project. Two are the major advantages of the cipher: (i) the low hardware complexity, which results in small area and (ii) the good statistical properties. The synthesis results prove that the Pomaranch cipher is suitable for FPGA implementation.

6. REFERENCES

[1] Simson Garfinkel and Beth Rosenberg, *RFID: Applications, Security, and Privacy*. Addison-Wesley Professional, 2005.

[2] Advanced Encryption Standard, (AES) 2003 http://www.nist.gov/aes

[3] Data Encryption Standard, Federal Information Processing Standard (FIPS) 46, National Bureau of Standards, 1977.

[4] B. Schneier,, *Applied Cryptography, Protocols, Algorithms, and Source Code in C.* John Wiley & Sons 1994.

[5] Cees Jansen, Tor Helleseth and Alexander Kolosha, *Cascade Jump Controlled Sequence Generator and Pomaranch Stream Cipher*. eSTREAM, ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/pomaranchp3.html

[6] eStream, ENCRYPT - European Network of Excellence in Cryptology, "Call for Stream Cipher Primitives", Scandinavian Congress Center, Aarhus, Denmark, 26-27 May 2005, http://www.ecrypt.eu.org/stream/

[7] Xilinx, San Jose, California, USA, Virtex, Field Programmable Gate Arrays, 2005, www.xilinx.com.

[8] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C. E. Goutis, *Comparison of the Hardware Implementation of Stream Ciphers*, The International Arab Journal of Information Technology (IAJIT), Colleges of Computer and Information Society, 2005.

[9] P. Kitsos, N. Sklavos, K. Papadomanolakis and O. Koufopavlou, *Hardware Implementation of Bluetooth Security*, IEEE Pervasive Computing, vol. 2, no.1, pp. 21-29, January-March 2003.

[10] Kris Gaj, Gabriel Southern and Ramakrishna Bachimanchi, *Comparison of hardware performance of selected Phase II eSTREAM candidates*, THE STATE OF THE ART OF STREAM CIPHERS-SASC 2007, Ruhr University Bochum, Germany January 31 - February 1, 2007.

[11] Philippe Bulens, Kassem Kalach, Francois-Xavier Standaert and Jean -Jacques Quisquater, *FPGA Implementations of eSTREAM Phase-2 Focus Candidates with Hardware Profile*, THE STATE OF THE ART OF STREAM CIPHERS-SASC 2007, Ruhr University Bochum, Germany January 31 - February 1, 2007.

[12] Paris Kitsos, *On the Hardware Implementation of the MICKEY-128 Stream Cipher*, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/059, 2006, http://www.ecrypt.eu.org/stream.

[13] Markus Kasper, Sandeep Kumar, Kerstin Lemke-Rust, and Christof Paar, *A Compact Implementation of Edon80*, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/057, 2006, http://www.ecrypt.eu.org/stream.