A High-Speed Hardware Implementation of the LILI-II Keystream Generator

P. Kitsos[1], N. Sklavos[2], O. Koufopavlou[2]

[1]Digital Systems and Media Computing Laboratory
School of Science & Technology
Hellenic Open University, Patras, Greece
e-mail: pkitsos@ieee.org

[2]Electrical & Computer Engineering Department
University of Patras, Greece

Abstract—**The LILI-II keystream generator is a new LFSR based synchronous stream cipher with a 128-bit key. In this paper a high-speed hardware implementation of the LILI-II generator is presented. The proposed architecture is suitable for application with high-performance and area-restricted requirements. Reprogrammable LFSRs are used for the algorithm implementation. With this technique the usage of the LILI-II algorithm is universalized, and different security levels can be achieved. Three FPGA devices are used for the synthesis purposes. A maximum throughput equal to 366 Mbps can be achieved, with a clock frequency of 366 MHz. Finally, comparisons with other previous published implementations of others stream ciphers are given.**

I. INTRODUCTION

The growing requirements for high-speed, and high-level secure communications, force the system designers to propose hardware implementations of cryptographic algorithms. In addition, wireless communication technology has advanced at a very fast pace during the last years, creating new applications and opportunities. For all these reasons the new encryption algorithms have to operate efficiently in a variety of current and future applications, performing different encryption tasks.

Many attempts have been taken place in order to propose new encryption algorithms and methods. For example the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project [1] had as main scope to put forward a portfolio of strong cryptographic primitives of various types. In addition the ECRYPT NoE Project [2] plans to manage and co-ordinate a multiyear effort to identify new stream ciphers suitable for widespread adoption.

The stream ciphers (keystream generators) are divided in self-synchronizing or synchronous. Many keystream generators either self-synchronizing, for example the Hiji-Bij-Bij [3] or synchronous such as A5 [4], E0 [5], and W7 [6], are fast and can be implemented with minimized hardware resources.

In this paper an efficient hardware implementation of the LILI-II [7] keystream generator is proposed. LILI-II is a specific cipher of the LILI family keystream generators. LILI-II is a corrected and enhanced version of the LILI-128 keystream generator publishing in [1], removing any weakness that the LILI-128 designers found. In general hypothesised attacks on LILI-128 and the request for a re-keying proposal prompted a review of the LILI-128 parameters, to ensure that provable security properties could be maintained while achieving an effective key size of 128-bit.

The following of this paper is structured as follows. After an overview of the keystream generator specification, we present the generator hardware architecture. This is followed by the design underlying, performance and implementation aspects. Finally, concluding remarks and some extensions are made in the final section.

II. LILI-II KEYSTREAM GENERATOR

The LILI-II keystream generator is a simple and fast generator using two binary LFSRs and two functions to generate a pseudorandom binary keystream sequence. Fig.1 illustrates the structure of the LILI-II.

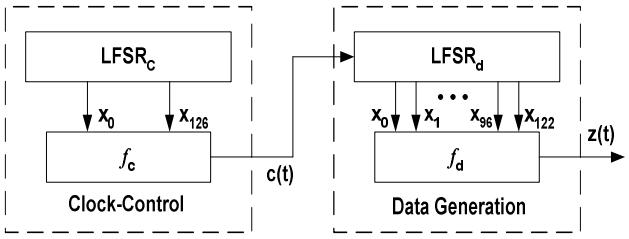

Fig. 1. LILI-II keystream generator

The components of the keystream generator can be grouped into two subsystems, based on the functions they perform: clock control and data generator. The LFSR for the clock-control subsystem is regularly clocked. The output of this subsystem is an integer sequence which controls the clocking of the LFSR within the data-generator subsystem. If it is regularly clocked, the data-generation subsystem is a nonlinearly filtered LFSR. The state of the generator is defined to be the contents of the two LFSRs. The functions f_c and f_d are evaluated on the current state data, and the feedback bits are calculated. Then the LFSRs are clocked and the keystream bit is the output.

The LILI-II generator may be viewed as clock-controlled nonlinear filter generator. Such a system, with clock control, is named stop-and-go generator. However, the use of stop-and-go clocking produced repetition, of the nonlinear filter generator output in the keystream, may

permit attacks. This system is an improvement on that proposal, as stop-and-go clocking is avoided. For LILI-II, $LFSR_d$ is clocked at least once and at most $d=4$ times between the production of consecutive keystream bits.

A. Clock Control Subsystem

The LILI-II clock-control subsystem uses a pseudorandom binary sequence produced by a regularly clocked LFSR, the *LFSRc*, with length equal to 128 and a function, fc, operating of the contents of $m=2$ stages of *LFSRc*, to produce a pseudorandom integer sequence, *c*. The *LFSRc* feedback polynomial is chosen to be primitive and is described by the following equation:

$$x^{128}+x^{126}+x^{125}+x^{124}+x^{123}+x^{122}+x^{119}+x^{117}+x^{115}+x^{111}+x^{108}+x^{106}+x^{105}+x^{104}+x^{103}+x^{102}+x^{96}+x^{94}+x^{90}+x^{87}+x^{82}+x^{81}+x^{80}+x^{79}+x^{77}+x^{74}+x^{73}+x^{72}+x^{71}+x^{70}+x^{67}+x^{66}+x^{65}+x^{61}+x^{60}+x^{58}+x^{57}+x^{56}+x^{55}+x^{53}+x^{52}+x^{51}+x^{50}+x^{49}+x^{47}+x^{44}+x^{43}+x^{40}+x^{39}+x^{36}+x^{35}+x^{30}+x^{29}+x^{25}+x^{23}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{11}+x^{9}+x^{8}+x^{7}+x^{6}+x+1$$

The initial state of $LFSR_c$ can be any state except the all zero state. At time instant *t*, the contents of stage 0 and 126 of $LFSR_c$, are the input to the function f_c and the output of f_c is an integer $c(t)$, such that $c(t) \in \{1,2,3,4\}$. The function f_c is given by the following equation:

$$f_c(x_0, x_{126}) = 2x_0 + x_{126} + 1 \quad (1)$$

B. Data Generation Subsystem

The data-generation subsystem of LILI-II uses the integer sequence *c* produced by the clock-control subsystem to control the clocking of a binary LFSR, $LFSR_d$, of length $L_d=127$. The contents of a fixed set of $n=12$ stages of $LFSR_d$ are input to a specially constructed Boolean function, f_d. The truth table for this function is given by the algorithm specifications [7]. The binary output of f_d is the keystream bit $z(t)$. When $z(t)$ is produced, the two LFSRs are clocked again and the process is repeated, to form the complete keystream. The $LFSR_d$ feedback polynomial is chosen to be primitive and is described by the following equation:

The initial state of $LFSR_d$ can be any state except the all zero state. The 12 inputs to f_d are taken from the $LFSR_d$ positions (0, 1, 3, 7, 12, 20, 30, 44, 65, 80, 96, 122).

$$x^{127}+x^{121}+x^{120}+x^{114}+x^{107}+x^{106}+x^{103}+x^{101}+x^{97}+x^{96}+x^{94}+x^{92}+x^{89}+x^{87}+x^{84}+x^{83}+x^{81}+x^{76}+x^{75}+x^{74}+x^{72}+x^{69}+x^{68}+x^{65}+x^{64}+x^{62}+x^{59}+x^{57}+x^{56}+x^{54}+x^{52}+x^{50}+x^{48}+x^{46}+x^{45}+x^{43}+x^{40}+x^{39}+x^{37}+x^{36}+x^{35}+x^{30}+x^{29}+x^{28}+x^{27}+x^{25}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{14}+x^{10}+x^{8}+x^{7}+x^{6}+x^{4}+x^{3}+x^{2}+x+1$$

III. Proposed LILI-II System Architecture

The architecture that performs the LILI-II keystream generator is shown in Fig. 2. This architecture consists of the clock-control subsystem and the data generation subsystem following the specifications demand.

The operation of the LILI-II is described as follows: At the beginning (initialization process) the 128-bit key, *K*, and a public known 128-bit initialization vector, *U*, are combined to form the initial values of the two LFSRs. If the initialization vector has length less than 128-bit multiple copies of the vector will be concatenated, repeated and truncated as required, to form a 128-bit, vector. The initialization process uses the LILI-II structure itself twice. The starting state of $LFSR_c$ is obtaining by XORing the two 128-bit binary strings *K* and *U* while the starting stage of $LFSR_d$ is obtaining by deleting the first bit of *K* and the last bit of *U* and XORing the two resulting 127-bit binary strings. Then the generator is run to produce an output string of 255-bit length. For the second application of the generator, the first 128-bit of the previous output string are feedback and used to form the initial state of $LFSR_c$. The remaining 127-bit are feedback too and used to form the initial state of $LFSR_d$. The generator processes again to produce another one string of 255-bit length. This string is used to configure the generator initial state, when the keystream bit production is begun. Same with previously, the first 128-bit are used for the $LFSR_c$ initial state while the remaining 127-bit are used for the $LFSR_d$ initial state.

For cryptanalysis and security purposes, the keystream bits are not been available to the users during the initialization process. So, a D flip-flop (D_ff) is located at the generator output that does not latch its input bits during the initialization process.

When the initialization process is completed, the output D flip-flop latches the generated keystream bits, that are XORed with the plaintext/ciphertext.

As the Fig. 2 shows the clock-control subsystem is comprised by the $LFSR_c$, the function f_c and the *Clock Pulses* components. The data generation subsystem is comprised by four $LFSR_d$s, four AND logical gates, the function f_d, six pipeline registers and twelve 4x1 multiplexers (MUXs).

The f_c function is implemented by a 3-bit *Ripple Carry Adder* (*RCA*). This adder is used for low area/performance trade-off.

The *Clock Pulses* component is used in order to control, through the AND gates, the $LFSR_d$s. So, if the data generation subsystem requires clocking four times, the *Clock Pulses* component forces the AND4 with "*1*" state, the rest AND gates with "*0*" state, and the $LFSR_d(4)$ is clocked four times. Similarly, if the data generation subsystem requires clocking three times the *Clock Pulses* component forces the AND3 with "1" state, the rest AND gates with "0" state, and the $LFSR_d(3)$ is clocked three times. The same process is followed when the data generation subsystem requires clocking two or one time.

In the proposed hardware architecture the same clock is selected for both subsystems. Someone could claim that the more efficient way in order to implement this algorithm is by using two clocks. The first one to be used for the clock-control subsystem, while the second to be applied for data generation subsystem, with a frequency multiplier. Although this method is inefficient for high speed applications, due to the small margin in a clock time interval.

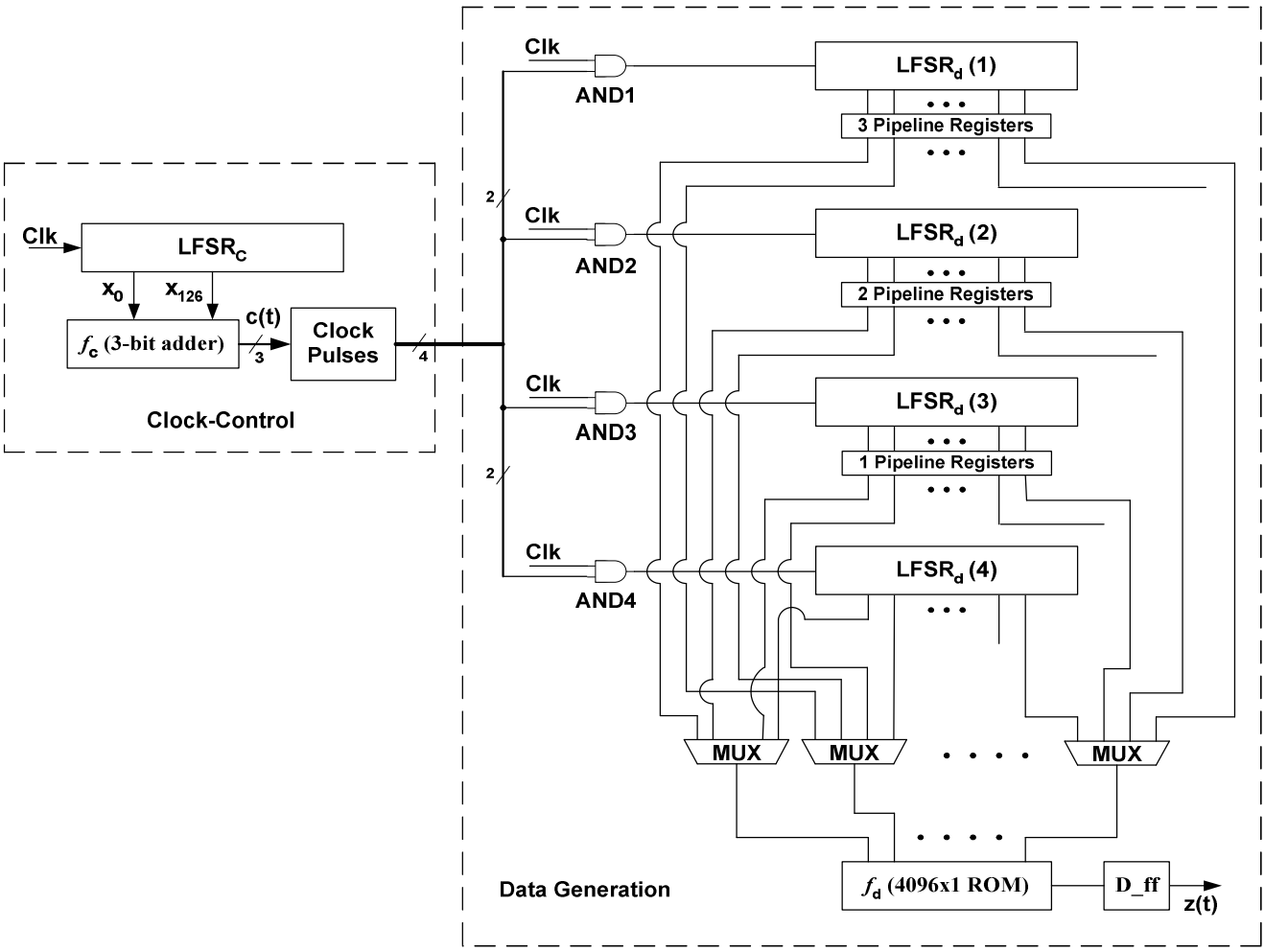

Fig. 2. LILI-II keystream generator architecture

Pipeline registers are located in the $LFSR_d$(i), $1 \le i \le 3$ outputs, in order to equalise the data delays between of them. The multiplexers (MUXs) are used in order to combine the appropriate $LFSR_d$*s* positions, that are used for the inputs to f_d function. The f_d function is implemented by ROM with 4096 per 1-bit elements.

For the LFSRs implementation dedicated LFSRs may be used, depended to the feedback primitive polynomials. However in the proposed architecture the LFSRs are implemented with reprogrammable logic. Fig. 3 depicts the proposed bit-sliced LFSR that the algorithm uses.

With this LFSR architecture we universalize the usage of the LILI-II algorithm and the feedback polynomials of $LFSR_c$ and $LFSR_d$ could be changed. This option is useful in applications with different security levels, which this could be achieved due to the selection of different feedback polynomials.

Each slice *i*, consists of one subfield multiplier (AND gate), one subfield adder (XOR gate), and two one-bit registers ($P(i)$ and $D(i)$). The $D(i)$ register with feedback path comprise the LFSR. Each coefficient $p(i)$ of the feedback polynomial $P(x)$ is stored in $P(i)$ register.

The non-zero coefficients $p(i)$ configure the LFSR, through the AND gates of the feedback path. In case of $LFSR_c$ the feedback polynomial degree is equal to 128 and 128 $D(i)$ registers are used, while in case of $LFSR_d$ the feedback polynomial degree is equal to 127 and 127 $D(i)$ registers are used.

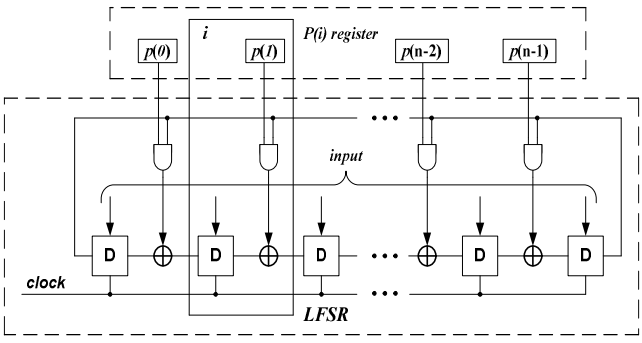

Fig. 3. The proposed bit-sliced reprogrammable LFSR

IV. Hardware Implementation Results

The proposed architecture was captured by using VHDL with structural description logic. The design was synthesised using FPGA devices from XILINX [8]. The synthesis results for the FPGA device are shown in Table I.

TABLE I
HARDWARE IMPLEMENTATION RESULTS

Device	VIRTEX 2V1000FF896 [8]		
Resources	**Used**	**Avail.**	**Utilization**
I/Os	391	432	90 %
Function Generations	938	10240	9.1 %
CLB Slices	469	5120	9.1 %
Dffs or Latches	693	11536	6 %
Blocks RAM	1	40	2.5 %

The embedded block RAM that used by the algorithm is equal to 4096x1-bit RAM. In the proposed architecture reconfigurable LFSRs are used. For this reason the architecture have large number of I/Os. If dedicated LFSRs, depended to the feedback primitive polynomials, are used, the FPGA devices, with minimized hardware resources utilization of each device family, will be used. For example if dedicated LFSRs were used the 2V40CS144 VIRTEX-II FPGA would be used instead of the 2V1000FF896 of the same device family.

Performance comparisons between the proposed system and previous published architectures are shown in Table II. According to our knowledge, only one other implementation of the LILI-II keystream generator has been previously published. So, comparisons with this implementation [9] and others single-bit synchronous stream ciphers (keystream generators) [10-13] are given in order to have a fair and detailed comparison of the proposed system.

TABLE II
HARDWARE PERFORMANCE COMPARISONS

Stream Cipher	FPGA Device	F (MHz)	Throughput (Mbps)
LILI-II [9]	2V6000FF1152	243	243
A5/1 [10]	2V250FG25	188.3	188.3
E0 [11]	2V250FG25	189	189
Edon80 [12]	2V250FG25	220.75	220.75
WG [13]	ASIC	1000	125
Proposed	V400BG560	158.5	158.5
Proposed	V400EBG560	230	230
Proposed	2V1000FF896	366	366

Different FPGA devices are used for the proposed system integration. The first implementation, in VIRTEX V400BG560 FPGA, achieves a throughput equal to 158.5 Mbps at 158.5 MHz clock. The second one, in VIRTEX-E V400EBG560, achieves a throughput equal to 230 Mbps at 230 MHz clock frequency. Finally the third, in VIRTEX-II 2V1000FF896, achieves a throughput up to 366 Mbps at 366 MHz clock.

In [9] a hardware implementation of the same cipher (LILI-II) is presented. This implementation uses one *LFSRd* with 127 (4:1) multiplexers (MUXs) and aims to a design that requires only a small of hardware resources. However the proposed implementation achieves a considerable higher throughput at the cost of more required hardware resources. In [10], a hardware implementation of the well-known A5/1 cipher is presented, which is used in GSM mobile phones. In [11], the E0 algorithm that Bluetooth system used is presented. In [12] and [13] the hardware implementations of the two new stream ciphers, the Edon80 and WG, are shown. These ciphers have been shown in the latest stream cipher workshop [2].

As the above table illustrates the proposed system implementations outperform all the compared hardware integrations of the other stream ciphers.

V. CONCLUSIONS

In this paper, an efficient hardware architecture of the new keystream generator named LILI-II is presented. It is suitable for application with high-performance and area-restricted requirements. The reprogrammable LFSRs that are used allow the flexible algorithm usage, with alternative security levels. The proposed system implementation achieves a throughput up to 366 Mbps, with a clock frequency of 366 MHz. Finally, the proposed architecture is superior for hardware integration, compared with other well known stream ciphers hardware architectures.

REFERENCES

[1] "NESSIE - New European Schemes for Signatures, Integrity, and Encryption", https://www.cosic.esat.kuleuven.ac.be/nessie/

[2] 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/

[3] P.Sarkar,"Hiji-Bij-Bij:A New Stream Cipher with a Self-Synchronizing Mode of Operation", Progress in Cryptology – Indocrypt 2003,LNCS 2904 T.Johansson and S.Maitra, eds.,Springer-Verlag,2003,pp.36-51.

[4] M. Briceno, I. Goldberg, and D. Wagner, A Pedagogical Implementation of A5/1, http://www.scard.org, May 1999.

[5] Bluetooth Specification, version 1.1, Available at www.bluetooth.org/spec/

[6] Thomas, D. Anthony, T. Berson and G. Gong, "The W7 Stream Cipher Algorithm", Internet Draft, April 2002.

[7] A. J. Clark, E. P. Dawson, J. Fuller, J. Dj. Golic, H. Lee, W. Millan, S. Moon and L. R. Simpson, "The LILI-II Keystream Generator", 7th Australasian Conference on Information Security and Privacy (ACISP 2002), vol. 2384 of Lecture Notes in Computer Science, pages 25-39. Springer-Verlag, July 2002.

[8] Xilinx Inc., Virtex FPGAs, www.xilinx.com, 2005.

[9] Philippe Leglise, Francois-Xavier Standaert, Gael Rouvroy, Jean-Jacques Quisquater, "Efficient Implementation of Recent Stream Ciphers on Reconfigurable Hardware Devices", In Proc. of the 26th Symposium on Information Theory in the Benelux, May 19th-May 20th, 2005, Brussels, Belgium.

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

[11] 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.

[12] D. Gligoroski, S. Markovski, L. Kocarev and M. Gusev, "*Edon80* - Hardware Synchronous Stream Cipher", Symmetric Key Encryption Workshop (SKEW), Scandinavian Congress Center, Aarhus, Denmark, 26-27 May 2005.

[13] Yassir Nawaz and Guang Gong, "The WG Stream Cipher", Symmetric Key Encryption Workshop (SKEW), Scandinavian Congress Center, Aarhus, Denmark, 26-27 May 2005.