Real-time FPGA Implementation of *Lorenz*'s Chaotic Generator for Ciphering Telecommunications

M. S. AZZAZ[1], C. TANOUGAST[2], S. SADOUDI[1] and A. DANDACHE[2]

[1]Laboratoire Systemes de Communications, Ecole Militaire Polytechnique, Alger, Algerie
[2]Laboratoire LICM, Université Paul Verlaine de Metz, Metz Technopole, France

***Abstract* – In this paper, we present a new approach for real-time FPGA implementation of the random key based *Lorenz's* chaotic generator for data stream encryption. We propose a structural hardware architecture designed for a small chip area and high speed performance. This architecture is particularly attractive since it provides a low-cost security telecommunication solution while holding or increasing the encryption throughput rate. We show its feasibility through implementation which is detailed and presented using Virtex Xilinx FPGA. This architecture employs only 1926 slices and allows achieving a random key throughput rate of 124 Mbps by using a low system clock with a frequency of up to 15,5 MHz allowing low power consumption especially for embedded applications.**

I. INTRODUCTION

Chaos based telecommunications have drawn increasing attention since Pecora and Carroll have discovered in 1990 that the chaotic systems can be synchronized [1]. In addition, chaotic signals are derived from nonlinear dynamic systems. They are aperiodic, uncorrelated, broadband, and deterministic and appear random in the time domain [2]. However, many methods based on analogue circuits are used to implement chaotic generators such as switched capacitor or analogue CMOS technology [3, 4]. However, these methods exhibit some practical difficulties since the component values are vary with age, temperature, etc. [5, 6]. Therefore, one must build both the transmitter and the receiver with very high accurate components to ensure information recovery, since the recovery characteristics are very sensitive to parameter mismatch between the transmitter and the receiver. In addition, it is very difficult to deal with the problem of the chaotic synchronization. Hence, analogue implementation is very difficult thought it is possible to overcome this difficulties to some extent.

To overcome this problem, a digital hardware implementation of chaotic generators can be used since the problem of parameter mismatch between the transmitter and the receiver does not exist. Nevertheless, the difficulty of the information recovery depends only on the channel noise sensibility. In this context, advances in VLSI technology have been employed to the manufacturing of reconfigurable logic including FPGA chips and helped their rapid growth in logic capacity, performance and popularity [7]. In a chaotic system implementation case, some previous works have used non optimal VHDL code generation using automatic code generation tools as in [5, 6]. However, the ''high level'' aspect of this method keeps the user far away from realities of the physical implementation. Thus, the result in terms of performance and density of resources used remains out of the designer reach.

In this paper, we propose a new structural hardware architecture for real-time implementation of the *Lorenz*'s chaotic generator using a Virtex-II Xilinx FPGA [8]. Our proposed architecture is based on the *Runge-Kutta* method (RK-4) to resolve the *Lorenz*'s differential equation system [9]. This architecture can be used as hard key generator in a chaotic self synchronizing stream cipher encryption (Fig. 1) [10].

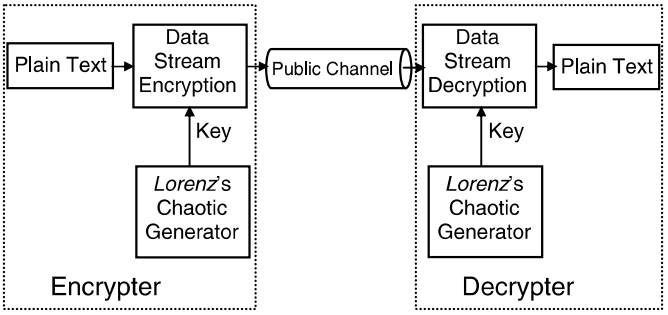

Fig. 1. Illustration of the Chaotic Cryptosystem for communications.

The originality of our approach is that the hardware implementation of the proposed architecture allows a very useful and attractive trade off between high speed, low area cost and data security transmission for an embedded system. Moreover, our proposed architecture can be used for the implementation of others chaotic systems such as *Chua*'s system, *Lü*'s system, *Rössler*'s system, *Chen*'s system, etc. [6, 11, 12].

This paper is organized as follows. Section 2 gives a short description and characterization of the *Lorenz*'s chaotic model. Section 3 describes the proposed architecture of *Lorenz*'s chaotic system including the simulation results. The hardware implementation results on Virtex-II Xilinx FPGA technology and the performance evaluation are presented in Section 4. In this section a comparison with other works is also given. Finally, a conclusion and future work are given in Section 5.

II. *LORENZ*'S CHAOTIC SYSTEM

The system of *Lorenz* is a famous example of chaotic system. It is represented by the following nonlinear equation system [13]:

$$\frac{dx}{dt} = \sigma(y - x) \quad (1.a)$$

$$\frac{dy}{dt} = -xz + rx - y \quad (1.b)$$

$$\frac{dz}{dt} = xy - bz \quad (1.c)$$

The solution of this nonlinear equation system depends mainly on the initial conditions specified by the initial values of x= x_0, y = y_0 and z = z_0. A numerical solution of this system with a fourth order *Runge-Kutta* method (RK-4) with the following value coefficients (h = 0.01, coef = 1/6) [9] using *Matlab* simulation tool [14], with *Lorenz*'s parameters values = 10, r = 28 and b = 8/3 and initial conditions (x0 = 0, y0 = 5, z0 = 20), gives the corresponding chaotic signals x, y and z and the two different attractors of the chaotic system shown in Figure 2. The first and the second parts relates to the phase plane (x-y) and the phase plane (y-z), respectively. These results will be useful as references for our implementation results (see Section 4).

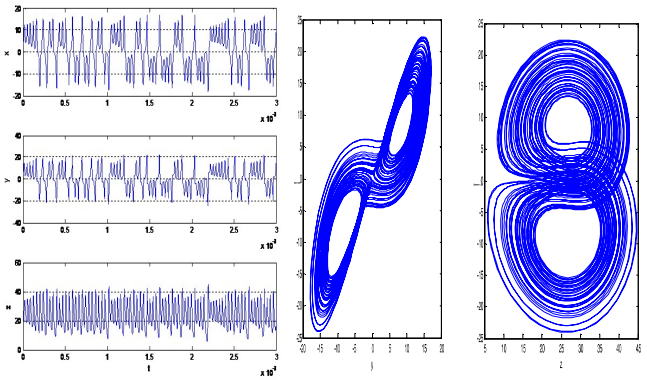

Fig. 2. *Matlab* simulation results of *Lorenz*'s Chaotic Signals and attractors in phase plane: (x-y) and (y-z).

III. PROPOSED ARCHITECTURE AND MODELIZATION

A. RTL Architecture of Lorenz's Chaotic Generator

Our proposed architecture consists of the implementation of the RK-4 method to resolve the *Lorenz*'s differential equations system [9]. An overview of the proposed Register Transfer Level (RTL) architecture for a *Lorenz*'s chaotic generator is given in Figure 5. This random key architecture is based on fixed parameters , r and b and consists of the structural feedback of the three main blocks: *ff1*, *ff2* and *ff3*. These three functional units realize the equations (1.a), (1.b) and (1.c), respectively. These Units are composed simply by an adder, a substractor and a multiplier logic arithmetic operators in accordance with the set RK-4 solution of equation (1). Our data-path processing architecture for *Lorenz*'s random keys generator is depicted in Figure3.

B. Functional Simulation Results

The proposed architecture was simulated for the correct functional operation with test vectors returned by a software implementation. This validation consists to model and describe directly the RK-4 method with the VHSIC Hardware Description Language (VHDL). However, it should be noted that the continuous chaotic signals are real. To get around this problem in the VHDL language, we have adopted the implementation based on a finite solution numbers with a fixed point representation of real data on 32 bits (16Q16). i.e. all data are fixed point format with 16 bits integer and 16 bits fraction. Unlike the approach presented in [5, 6], the fixed-point arithmetic allows a very useful and attractive trade off between high speed, low area cost and data transmission security.

To test the effectiveness of our solution, we have simulated our RTL architecture of the *Lorenz*'s chaotic generator with *ModelSim* simulator tool [15]. The results obtained are presented in Figure 4 where the chaotic signals *x, y* and *z* are represented with 32 bits using the followings RK-4 value coefficients (h = 0.01, coef = 1/6) [9], with *Lorenz*'s parameters values =10, r = 28 and b = 8/3 and (x0 = 0, y0 = 5, z0 = 20) as the initial conditions. It can be seen that the functional hardware simulation results are very similar those of the RK-4 numerical resolution by using *Matlab* simulation tool (see Section 2).

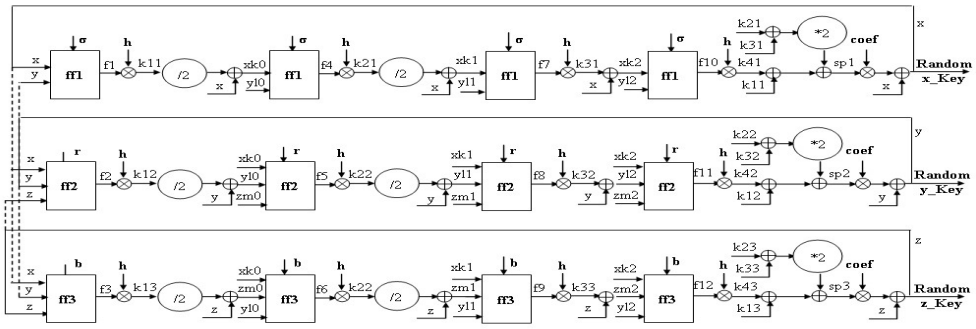

Fig. 3. *Lorenz*'s chaotic generator architecture.

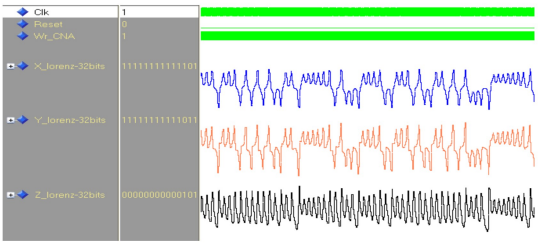

Fig. 4. *ModelSim* simulation results of *Lorenz*'s chaotic generator.

IV. Real Time Hardware Implementation of *Lorenz*'s Chaotic System

An overview of the architecture for a digital hardware implementation of the proposed *Lorenz*'s chaotic system using Virtex-II Xilinx FPGA [8] is depicted in Figure 5.

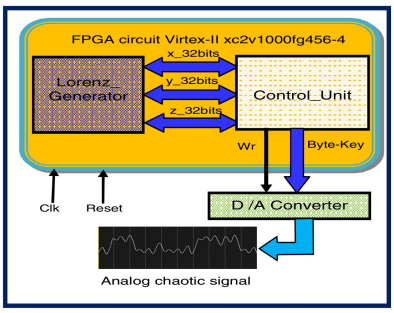

Fig. 5. Digital Hardware architecture of *Lorenz*'s chaotic system.

The architecture system consists of two main modules: *Control_Unit* and *Lorenz_Generator* sub-modules. The *Control_Unit* sub-module is a Moore state machine which manages and schedules the different operations and functions of our proposed chaotic system. *Lorenz_Generator* sub-module generates the random Keys using the RK-4 method as described in Section 3, which implements *Lorenz*'s nonlinear equation system defined par the set of equation (1). Once the chaotic signals (x, y and z) with 32 bit wordlenght are obtained, they are converted to analogue format as a sequence of 8 bits using a Digital to Analog converter (DAC) [16] and this process is repeated so that real-time chaotic signals are obtained at the output of the DAC for visualization on an oscilloscope.

A. FPGA Implementation and Results

Unlike the previous works using VHDL automatic code generation tools [5, 6], our RTL description of the proposed architecture has been implemented on Xilinx Virtex-II FPGA (XC2V1000) [8] using a VHDL structural description. ISE 7.2i of Xilinx software tool [17] have been used for this implementation thus allowing to evaluate the hardware resource requirements and the associated real time constraints. In order to minimize the area logic resources of the FPGA, which are due mainly to the RK-4 solution method to solve the *Lorenz*'s system, an optimization has been carried out. More precisely, we have replaced or approximated most of the multiplication and/or division operations by the simple left or right shifts operations. This has led us to minimize the number of the embedded multipliers in the FPGA implementation. The synthesis results after place and route and performance analysis of our implementation are shown in Table 1. This table specifies the hardware resources in terms of the *Slice* or the *Slice Flip-Flops* numbers and the speed performance. As can be seen, the hardware implementation exhibits good performances in terms of the throughput and resource costs required.

TABLE I
Implementation results with a Virtex II FPGA for the *Lorenz*'s chaotic System (XC2V1000)

Device utilization summary - 2v1000fg456-4 FPGA			
Number of Slices:	1926	out of	5120
Number of Slice Flip Flops:	791	out of	10240
Number of 4 input LUTs:	2718	out of	10240
Number of bonded IOBs:	11	out of	324
Number of MULT18X18s:	40	out of	40
Number of GCLKs:	1	out of	16
Maximum Frequency:		(15.598MHz)	

The results thus demonstrate that a real time *Lorenz*'s chaotic generator can be efficiency implemented with FPGA technology. If compared with other related works [5, 6], it can stated that an attractive trade off between high speed and low logic resources has been achieved. Indeed, our implementation on a Xilinx Virtex II device uses only 1926 *CLB-Slices*, 40 multipliers and no block RAMs.

In order to evaluate the behaviour of the proposed system, it is necessary to use some evaluation metrics. The metrics used for the evaluation results for this system are the throughput rate and the time latency. The *throughput* rate is defined as the number of bits key in a unit of time for a stream encrypted (or decrypted). More precisely, the number of bits key per unit time duration of the clock period.

$$Throughput = q * f_{clk} \tag{2}$$

where f_{clk} is the operating clock frequency and q is the bit representation of one random key. In our case and from the performance results (table I) we have achieved a maximal throughput of 124 Mbps. This throughput rate is computed after the initialization phase. *Latency* is defined as the time necessary to generate a single random key after the start of the generator. Latency is calculated as:

$$Latency = \frac{nu\_stage\_pipeline}{f_{clk}} \tag{3}$$

where *nu_stage_pipeline* corresponds to the number of the pipeline stages in the design. Our optimized implementation of the *Lorenz*'s chaotic system requires 6 clock cycles to generate one random key. In our case we have obtain a time latency of 388 ns.

B. Real Time Measurement Results

The x, y and z real-time chaotic signal results of the *Lorenz*'s generator, obtained by a direct implementation after optimization, are given in figures (6.a), (6.b) and (6.c), respectively. These Snapshots are given by a Tektronix digital oscilloscope [18]. One can compare these results and those obtained using *Matlab* (figure 2) and *ModelSim* (figure 4) simulation tools to ascertain whether these results are similar. The measured real-time attractors (x-z) and (x-y) are presented in figures (6.d) and (6.e), respectively. These results clearly confirm that the implemented chaotic system work well in the chaotic mode. This clearly validate our implementation method.

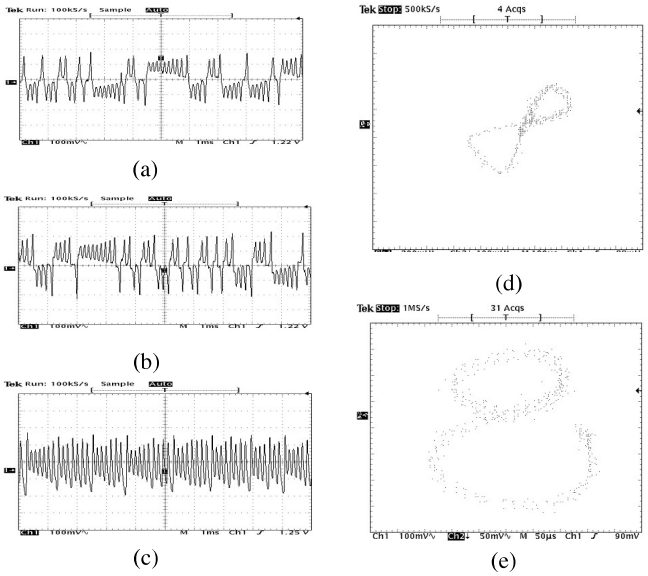

Fig. 6. Real-time results of *Lorenz*'s chaotic generator: (a) x chaotic signal, (b) y chaotic signal and (c) z chaotic signal, (d) (x-y) attractor, (e) (y-z) attractor.

V. Conclusion and Future Work

This paper proposes a new structural hardware architecture of a random key generator based on a *Lorenz*'s chaotic system for data stream encryption systems. This proposed random key generator architecture based on the *Runge-Kutta* method (RK-4) is particularly attractive since it provides low-cost security communication solutions for embedded systems. We have validated our approach by showing that real-time *Lorenz*'s chaotic signals obtained with our architecture are similar to the software simulation counterparts. Moreover, our experimental results using Xilinx *Virtex* technology have demonstrated that our approach can lead to designs with small logic area, satisfactory throughput rates and low latency for embedded applications. Indeed, an implementation on a Xilinx Virtex II technology requires only 1926 CLB-Slices and no block RAMs. Our FPGA implementation achieves a throughput rate of 124 Mbps at a clock frequency of 15.5 MHz with a low latency time of 388 ns. Ongoing work seeks to further improve the performance of the architecture targeting a full implementation in a pipelined way. However, in this case we will increase the latency time. Finally, our new approach is very simple, exhibits attractive, good performances and can be used for the implementation of others chaotic systems such as Chen's system, *Chua*'s system, *Lü*'s system, *Rössler*'s system, etc.. Therefore, this work will permit to use in choice these chaotic generators in secure digital chaotic communication systems.

References

[1] T. L. Carroll and L. M. Pecora, "Synchronization in chaotic systems'', *Phy. Rev. Let.* Vol, 64, N°8, Feb. 1990, pp.821-824.

[2] A. abel and W. Schwartz, "Chaos Communications-Principles, Schemes and Systems analysis'', *Proc. of the IEEE Inst. for Fundamentals of Electr. Eng. & Electron., Dresden Univ. of Technol*, 90. 2002. pp. 691-710.

[3] T. Matsumoto, "Chaos in electronic circuits'', *IEEE Inst. of Elec. and Elecs Eng.*, vol.75, N°.8, Aug.1987, pp. 1033-1046.

[4] C.Y. Cha and S.G. Lee, "Complementary Colpitts Oscillator in CMOS Technology", *IEEE Tran. on microwave theory and techniques*, Vol. 53, No. 3, March 2005, pp. 881- 887.

[5] M.A. Aseeri, M.I. Sobhi and P. Lee, "Lorenz Chaotic Model Using Field Programmable Gate Array (FPGA)'', *Midwest Symposium on Circuit and Systems*, 2002, pp. 686-699.

[6] M. I. Sobhy, M. A. Aseeri and A. E. R. Shehata., "Real Time Implementation Of Continuous (Chua And Lorenz) Chaotic Generator Models Using Digital Hardware''. *Proc. of the Third International Symposium on Communication Systems Networks and Digital Processing,* 1999, pp.38-41.

[7] Celoxica, "RC200 platform deform developer's kit'', *Celoxica*, 2005.

[8] Xilinx, "VirtexII complete Datasheet", *Xilinx*, 2007.

[9] H. William Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, "Numerical Recipes in C, The Art of Scientific Computing'', *Cambridge University Press,* 1992.

[10] C. Tanougast, S. Weber, G. Millerioux, A. Bouridane and J. Daafouz "VLSI architecture and FPGA implementation of a hybrid message embedded self-synchronizing stream cipher" *4th IEEE Int. Symp. on Elec. Design, Test and Applications*, 2008, pp. 386-389.

[11] P. Kvarda, ''Investigating the Rössler attractor using Lorenz plot and Lyapunov exponents'', *Radioengineering*. Vol.11. N°.3 September 2002, pp 22-23.

[12] H.H Chen, J.S. Chiang, Y. L. Lin, C.I. Lee, ''Chaos synchronization of general Lorenz, Lü, and Chen systems'', *Hsiuping Journal,* Vol. 15, 2007, pp.159-166.

[13] K. M. Cuomo, A. V. Oppenheim, and Steven H. Strogatz, "Synchronization of Lorenz-Based Chaotic Circuits with Applications to Communications'', *IEEE transactions on circuits and systems-11: analog and digital signal processing*, vol. 40, N° 10, October 1993, pp. 626 - 633.

[14] Mathworks, Matlab Software, Version 7.3, *Mathworks*, 2006.

[15] Mentor Graphics, « Modelsim SE User's Manuel, Sofware", Version 6. 4 , *Mentor Graphics*, 2008.

[16] Analog Devices, "LC2MOS Complete, 8-Bit Analog I/0 Systems, AD7569/AD7669'', *Analog Devices*, 1996.

[17] Xilinx, "Integrated Software Environment (ISE)", Version 7.2, *Xilinx*, 2005.

[18] Tektronix, "Digital Real-Time™ Oscilloscopes TDS340A, TDS360 et TDS380", *Tektronix*, 2006.