A New Approach to Implement Chaotic Generators Based on Filed Programmable Gate Array (FPGA)

Mohammed A. Aseeri and Mohamed I. Sobhy

Department of Electronics,
The University of Kent at Canterbury
Canterbury, Kent, CT2 7NT, U.K.

Abstract: In this paper, we introduce a new method to implement chaotic generators based on Henon map chaotic system given by the state equations by using Filed Programmable Gate Array (FPGA). The aim of this method is to increase the frequency of the chaotic generators. The new method is based on MATLAB® Software, Xilinx System Generator, Xilinx Alliance tools, Leonardo spectrum or Synplicity Synplify and ModelSim XE PLUSE. The toolbox of the Xilinx System Generator used as toolbox under the MATLAB® Simulink toolbox to convert any MATLAB® Simulink model to the Xilinx System Generator model then to generate the VHDL code for that model. The hardware can be used directly in chaotic communication systems with high frequencies.

1. Introduction

The Xilinx System Generator bridges the gap between conceptual architectural design and the actual implementation in a Xilinx field programmable Gate Array (FPGA). The field programmable Gate Array (FPGA) is type of programmable device. Programmable devices are a class of general-purpose chips that can be configured for a wide variety of applications. They have capability of implementing the logic of not only hundreds but also thousands of discrete devices. The System Generator for Simulink, developed in partnership with The Math Works, Inc enables to develop high-performance DSP systems for Xilinx FPGAs using the popular MATLAB® /Simulink products from The MathWorks, Inc [1]. As a plug-in to the Simulink modeling software, the Xilinx System Generator provides a bit-accurate model of FPGA circuits, and automatically generates a synthesizable Hardware Description Language (VHDL) code and a testbench. This VHDL design can then be synthesized for implementation in Xilinx Virtex®-II, Virtex, and Spartan®-II FPGAs. The Xilinx Blockset enables bit-true and cycle-true modeling, with Xilinx FPGA hardware as the target. It includes parametric blocks for DSP, arithmetic, and logic functions like FFTs, FIR Filters, Multipliers, Memories, and gateway blocks to communicate with the MATLAB® environment, where you also have access to the extensive set of Simulink libraries [2]. But why we used FPGA instead of analogue circuit? The answer is, Analogue chaotic generators have been used for communication systems [3]. Recovery of the information signal depends on how well the receiver is synchronised with the transmitter. This requires that the parameters of both receiver and transmitter be matched to a high degree of accuracy. This requirement is difficult to achieve in analogue systems especially that the values of analogue circuit component are functions of age and temperature. The most obvious solution is to implement the generators using digital hardware. The generators are first represented by a set of non-linear equations and a system-based model is developed to represent these equations directly. The FPGA overcome of that entire problem and in the same time we can get high frequency. Once the VHDL code generated and synthesized then the netlist file will produce, then through the Xilinx Alliance tools the bit file will produce. Once the bit file available then the impact software under the Xilinx Alliance tools will use to download the bit file to target FPGA device. By this way we can control the frequency of the chaotic signal during the FPGA device by using clock, so that the frequency of the chaotic signal depends on the frequency of the clock for the FPGA device. The main thing the output from the FPGA device is digital. To see the analogue output we need Digital to analogue (D/A) converter device. In this case the frequency of the system depends on the clock sampling rate and the sampling rate of the D/A device and the numbers of D/A bits. The presented method is depended on the Xilinx System, which convert the Simulink model from MATLAB® to VHDL code.

2. Implement Henon map chaotic generator USING Xinlinx System Generator

In this paper, we design chaotic generator model using FPGA based on the Henon map chaotic system. The steps of the FPGA design are as follows:

- Develop a system model from the state equations using MATLAB® Simulink Software [4].
- Simulate the model to adjust the required frequency.
- Convert all models by using the Xilinx System Generator blockset.

- Run the Simulink model with the Xilinx System Generator blockset and compare the results with the model without using the Xilinx system Generator blockset.
- Generate the system generator model to generate aVHDL code.
- Synthesis the VHDL code by using either Leonardo spectrum or Synplicity Synplify to produce the netlist file, which we need it to produce the bit file during the Xilinx Alliance tools.
- Pass a netlist file through implementation tools " Xilinx Alliance tools " to generate the bitstream file.
- Download the bitstream file on the target FPGA chip using the PC parallel port.

The state equations of the Henon map chaotic generator are given by [5]

$$x_{n+1} = 1 + y_n - ax_n^2$$
$$y_{n+1} = bx_n \tag{1}$$

where a and b are constants and a=-1.4 and b=0.3. The state equations of the Henon map chaotic generator converts to a Simulink model by using the MATLAB® Simulink and the output is controlled by the clock time which is the step size of the simulation. Fig. 1 shows the model of the Henon map chaotic generator.

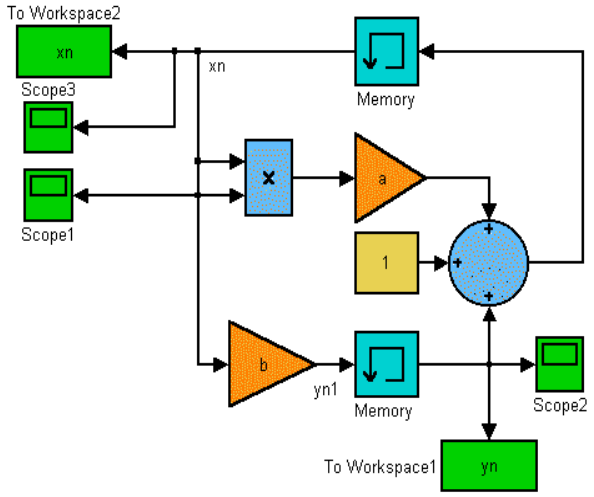

Fig. 1. The Henon map chaotic generator model.

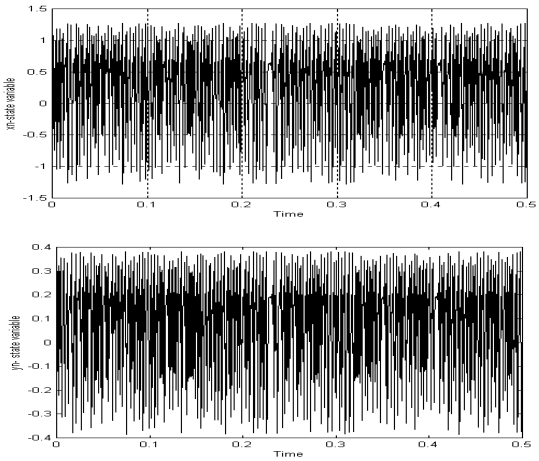

Fig. 2 The Henon map chaotic generator simulation results.

The simulation results of the Henon map chaotic generator using the MATLAB® simulink toolbox are shown in Fig. 2. The xn-*yn* attractor of the simulation results is shown in Fig. 3. The maximum frequency can be achieved by changing the clock of the simulation. Here we choose the clock =0.5x10^{-10} the maximum frequency 1.00GHz as shown in Fig. 4.

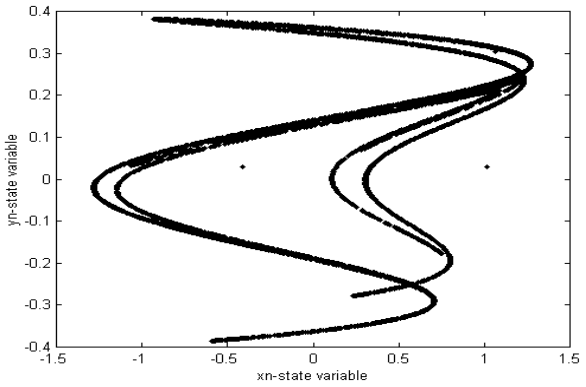

Fig. 3. The Henon map chaotic generator attractor.

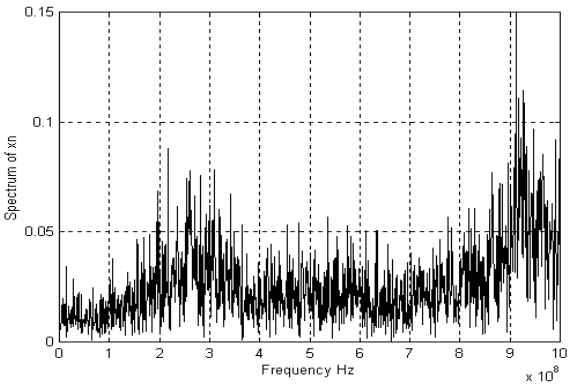

Fig. 4. The Henon map chaotic generator spectrum.

Now, convert the model, which was shown in Fig.1 to Xilinx System Generator Model using the Xilinx Blockset under the MATLAB®.

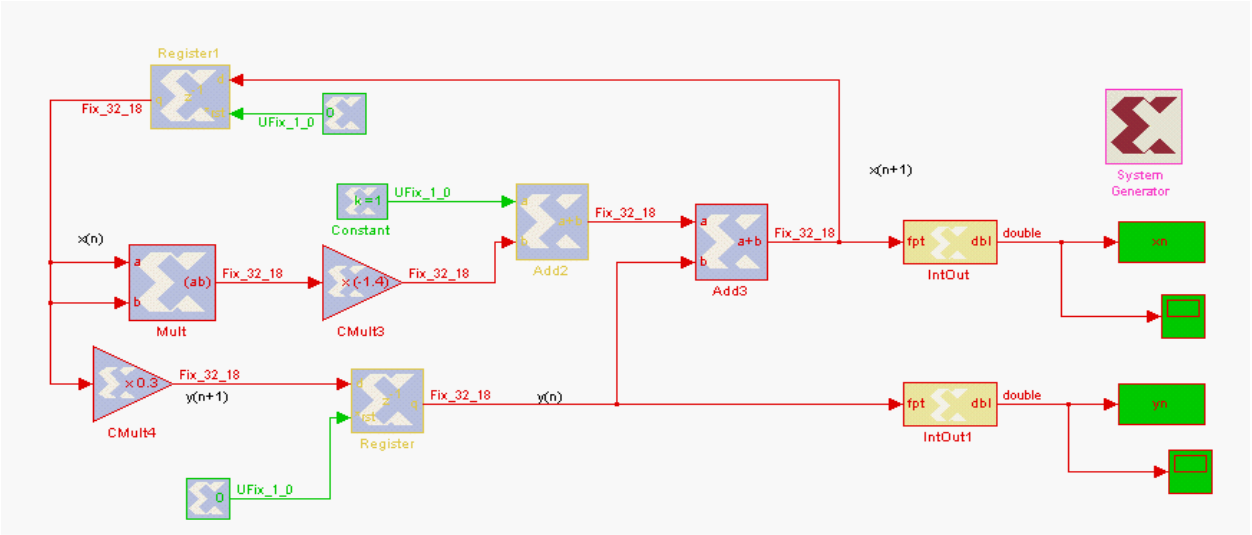

Fig. 5. The Henon map chaotic generator using Xilinx System Generator.

To make synchronization for the Henon map chaotic generator blocks all the number of bits equals 32 and binary of points equal 18 as shown in Fig. 5. The stop time was choose as 10^3 x dt and the fixed step size as dt where dt=10^{-2}. Fig. 6 shows the results of the simulation model of the Henon map chaotic generator. The attractor of the model was shown in Fig. 7.

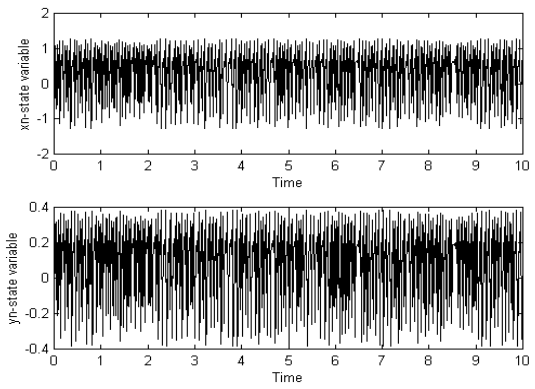

Fig. 6. The Henon map chaotic generator attractor.

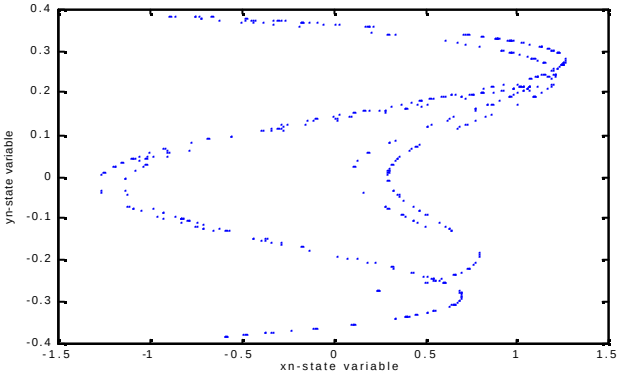

Fig. 7. The Henon map chaotic generator spectrum.

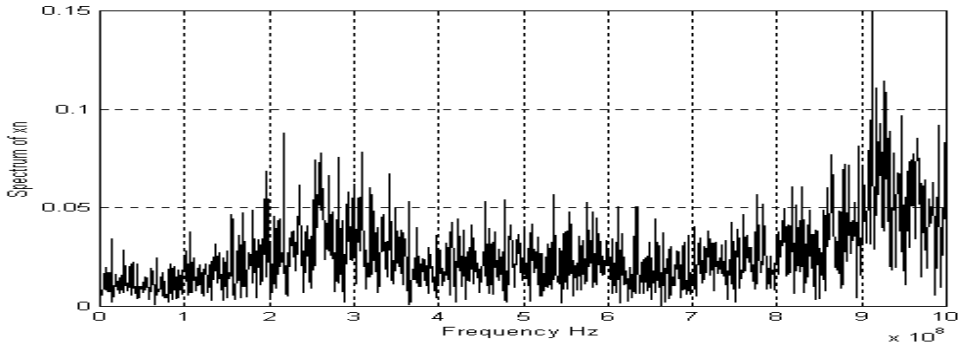

Fig. 8. The Henon map chaotic generator using Xilinx System Generator.

Comparing this result with the result from MATLAB® Simulink including the delay in the system it's the same. Fig. 8 shows the spectrum of Xilinx Henon map Generator, which depends on the values of step size of the model dt. To increase the frequency of the Henon map generator system the value of dt was reducing and adding another CMult block with value 10^{-2}/dt to control the frequency, dt was reducing up to $5x10^{-6}$. Now, System generator block was used to generate the VHDL code for the Lorenz system. The ModelSim SE PLUS Software was used to compiling the VHDL code and simulates it by using two files VCOM and VSIM, which was generated already from the Xilinx System Generator. Fig. 9 shows the simulation results of VHDL Henon map Code using the ModelSim as in the Hardware results. The ModelSim Simulation used to know what should be the results from the Xilinx System Generator in the Hardware comparing with the MATLAB® simulation results. We require to the synthesis tools either Leonardo Spectru m [6] or Synplicity's Synplify [7] to synthesis the code. This will produce netlist file, then must be put it as input through the back end of the Xilinx Alliance tools [8]. This tool will then generate a bitstream file that can then be downloaded on the FPGA by using parallel port cable, which connected to the FPGA device Board.

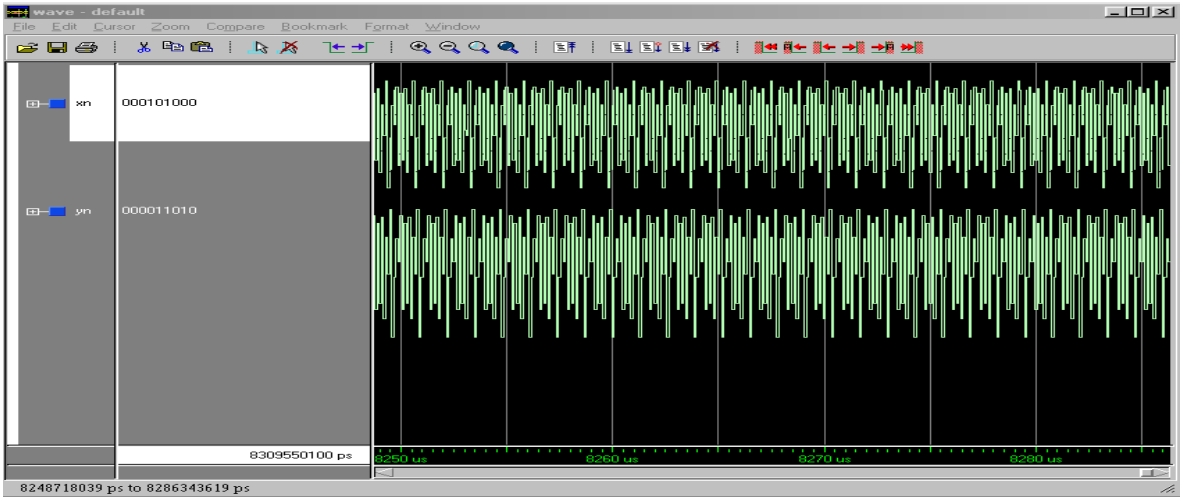

Fig. 9. The Henon map chaotic generator results by ModelSim.

3. Conclusion

A new method to design chaotic generator models in real time is introduced which is capable of implementing the chaotic systems that are given by state equations in real time using Filed Programmable Gate Array (FPGA) system. The method is implemented by MATLAB®, SIMULINK, Xilinx System Generator, Xilinx Alliance tools, Leonardo spectrum or Synplicity Synplify and ModelSim XE PLUSE. A clock time dt (the simulation step size) to control the frequency band is used in this type of chaotic generator. The method is useful to implement the chaotic generators at high frequencies depends on the clock of the simulation of the model itself. Both continuous and discrete chaotic generators can be implemented even if the system cannot be represented by a physical electronic circuit. Modification of any system is a simple change in the block diagram or the parameter values within the block.

4. References

[1] P. Marchand, *Graphics and GUIs with MATLAB*: CRC Press, 1999.
[2] The MathWorks and Xilinx Plans web page: http://www.mathworks.com/company/pressroom/index.shtml
[3] T. Matsumoto, "Chaos in electronic circuits," *Proc. of the IEEE*, vol. 75, No. 8, pp. 1033- 1046, Aug. 1987.
[4] M.I. Sobhy, M. A. Aseeri and A. Shehata," Real time implementation of chaotic models using digital hardware", AMREM 2002, HPEM 13, June 2002.
[5] M. Storace, M. Parodi and D. Robatto, "A hysteresis-based chaotic circuit: Dynamics and applications, " *Int. J. of Circuit. Theory appl*., pp. 527-542, 1999.
[6] The Leonardo Spectrum web page: http://www.mentor.com/synthesis/leonardospectrum/
[7] The Synplicity's Synplify web page: http://www.synplicity.com/
[8] Xilinx Alliance tools web page: http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=ISE+Alliance.