# *RO*-based *PRNG*: *FPGA* implementation and stochastic analysis

Luciana De Micco, Maximiliano Antonelli
and Hilda A. Larrondo
Physics and Electronic Departments,
School of Engineering, National University of Mar del Plata.
J.B. Justo 4302 Av., 7600 Mar del Plata, Argentina.
CONICET

Eduardo Boemo
Digital Systems Laboratory,
Universidad Autónoma de Madrid,
Campus Univ. de Cantoblanco,
28049 Madrid, Spain.

***Abstract*—This paper deals with the use of Ring Oscillators (*RO*s) as pseudo random number generators (*PRNG*). The design, made for *ALTERA Cyclone* III ©, using low level primitives is explained. Two relevant characteristics of a *PRNG* are considered to validate the design: 1) the equiprobability of all possible outcomes and 2) the statistical independence of consecutive values. In this work these properties are measured via Information Theory Quantifiers. A dual entropy plane is used to represent the time series and easily visualize the results obtained with different configurations. The quality is also compared with other available *PRNG*s by means of the dual entropy plane. Our method constitutes an effective reduction of the complete analysis made with test suites like *DIEHARD* or *NIST*.**

## I. Introduction

The jitter and phase noises present in ring oscillators, are not convenient in several applications of *RO*s, for example in the implementation of *on-chip oscillators* to generate clocks in high-speed circuits[1], [2], [3]. However they are the source of randomness for *RO*-based *PRNG* [4], [5]. Furthermore *RO*s can be implemented in a full-digital circuit like Field Programmable Gate Arrays (*FPGA*s) as they basically are just a string of inverters.

In [4], Sunar et al. presented a *PRNG* using stochastic jitter by combining several *RO*s. They required a post processing of the bit stream, based on resilient functions, to mask imperfections in the entropy source and to increase immunity against changes in environmental conditions. The entropy of the bit stream was used to validate the results in [4].

Wold et al. [5] proposed an enhanced version with better random characteristics and without a post processing. They only added an extra D flip-flop at each ring output. The effectiveness of their proposal was tested by means of test suites available in the open literature [6], [7], [8].

In this paper a detailed description of a very compact hardware implementation of the *RO*-based *PRNG* proposed in [5] is done. In order to validate the randomness of the noise sequences generated, two quantifiers derived from the information theory are used. They define a dual entropy plane $H_{BP}$ vs $H_{hist}$. $H_{hist}$ is a measure of the first characteristic of a *PRNG* pointed in the abstract, the equiprobability among all possible values. $H_{BP}$ is a measure of the second characteristic pointed in the abstract, the independence between consecutive values. This methodology was successful to evaluate randomizing techniques applied to chaos-based *PRNG* [9]. A comparison with other options both physical and algorithmic, proposed in the literature is made showing that, in spite of their simplicity, *RO* are good candidates as *PRNG*.

Organization of the paper is as follows: section II describes the hardware implementation of the *RO*s mapped in *FPGA* Cyclone III. Section III shows how the normalized entropies are determined (to keep this paper short we do not detail already published results); IV presents the results obtained for different configurations of the same *PRNG*s proposed in [5], and the statistical comparison with other utilized *PPRNG*s. Finally we present our conclusions in Sec. V.

## II. Hardware implementation.

The implemented *PRNG*'s consist of several *RO*s with their outputs XORed together and sampled by a *D* flip flop, The flip flop latches the output at a selected frequency (here 100MHz)[5]. The physical implementation is made on *ALTERA*© *Cyclone* III *EP3C120* development kit with a *EP3C120F780C7N FPGA*. The design is made with *Quartus*© II 13.1 software.

### A. Chip Overview.

*FPGA*s consist of a large number of logic array blocks (*LAB*s), with groups of logic elements (*LE*s) for implementing sequential as well as combinatorial circuits. In the *Cyclone* III family architecture each *LAB* contains 16 *LE*s. Basically, each *LE* is a Flip Flop (*FF*) with a four-input look-up table (*LUT*) (see Fig. 1). Each *LUT* can implement any function of four variables. The *FF* and the *LUT* can be used together or independently, [10].

Usually, the logic synthesis software assigns *LE*'s resources without the designer intervention. But in the design of *RO*-based *PRNG*s it is necessary to control the exact location of each individual component for two reasons: 1) to avoid the simplification of the inverters performed by the synthesis tool; 2) to locate each *RO* in the desired place. In *Altera* the use of low-level primitives enables one to control the hardware implementation for each *cone of logic* [11]. Consequently low-level primitives and assignments are employed inside the *HDL* (hardware description language) code employed in our design.

Strings of *RO*s can be programmed on the chip by instantiating the *LUT*s as inverters. In the case of *RO*s it is necessary

**978-1-4799-6848-0/14/$31.00 ©2014 IEEE**

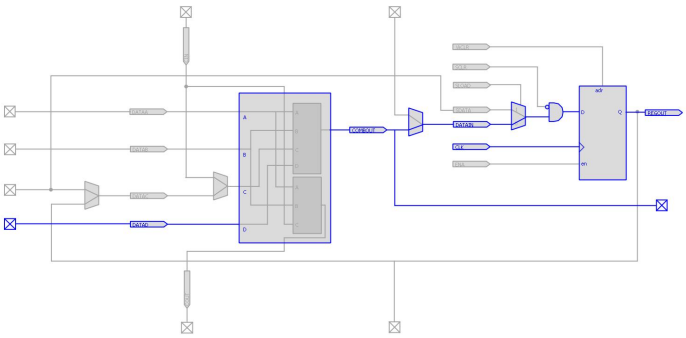

Fig. 1. *LE* implementing an inverter and a Flip Flop, Chip Planner view.

to prevent the *Quartus* II synthesis engine to merge two *NOT* gates in series, by using a primitive called *LCELL*. A *LCELL* always consumes one logic cell and it is not removed from the project during logic synthesis.

These primitives allow one to break up the design into manageable parts. Each cone is as small as a *LCELL* instantiation. To create a *RO*, *LCELL*s are programmed as inverter-buffers. Figs. 2 and 3 show how this primitive is implemented by the Quartus II compiler.

Furthermore, to avoid the synthesis tool to optimize removing the redundant buffers away, the Ignore *LCELL* Buffers must be set in *OFF* in the *More Analysis & Synthesis Settings* dialog box. Also *Remove Redundant Logic Cells* must be set to *OFF*.

In order to place each *RO* at a desired position, it must be assigned to a previously defined *LogicLock region*. In this way the *fitter* will keep all the elements of each ring inside the same region, [12]. The process of mapping all the elements to a particular location on the chip (*LogicLock* region) is achieved by the *Assignment Editor* tool, that also allows one to verify that the placements are actually still there, after the *Synthesis* and *Place & Route* processes.

Fig. 4 shows the 50 *LogicLock* regions used in this paper as they are established in the die. One *RO* is assigned to each region. Regions are spread over the die for a future analysis of location importance. Each region has 16 *LAB*s, to allow us to increase the number of inverters of each ring, an issue to be considered in future work.

3−inverters, employed in a *RO* and the *FF* were all mapped onto a *LE* each, meaning that the block utilization is 4 of 16 *LE*s for any *LAB*.

Fig. 1 displays a single *LE*, there an inverter is implemented in the *LUT* and it can be seen the exact *LUT* input that is used. Also the output *FF* of the ring is mapped there.

There are many factors that determine the frequency of each *RO*, and contributes to the unpredictability of the output:

1) Placement within the *LAB*: different placements between rings could result in timing differences.
2) Connections: even having exactly identical placement of the *LUT*s with respect to each other in a given ring, it is not possible to have exactly the same *routing resource usage* in the connections. A small difference in *routing resource usage* could affect the ring delay.
3) Input selection: the *fitter* will choose which *LUT* input is utilized during the routing stage. But the delay through the *LUT* depends on which of the four inputs is used and consequently the rings could also have different delays.
4) Neighborhood: even if the design locks down all the placement and routing of a section and everything is physically locked, the timing can change by a few picoseconds depending on what is placed and routed around the ring.

| | | |
|---|---|---|
| Total logic elements | $847/119,088$ | $(< 1\%)$ |
| Total combinational functions | $629/119,088$ | $(< 1\%)$ |
| Dedicated logic registers | $617/119,088$ | $(< 1\%)$ |
| Total registers | 617 | |
| Total memory bits | $131,072/3,981,312$ | $(3\%)$ |

TABLE I. COMPILATION REPORT, *RO*-BASED *PRNG* USING 15 *RO*S AND 3 INVERTERS EACH.

In Fig. 5 (RTL view) it is shown a *PRNG* using 3 *RO*s followed by a XOR gate.

Finally, Table I shows the compilation report of the *PRNG* using 15 *RO*s each with 3 inverters.

## III. QUANTIFIERS

Let $X = \{x_i, i = 1, ..., N\}$ be a length $N$ output of a given symbolic source with alphabet $\mathcal{A} = \{a_i, i = 1, ...M\}$. Each element of $X$ is $x_i \in \mathcal{A}$. In the case of *RO* the alphabet is binary consisting of two symbols $\mathcal{A} = \{0, 1\}$. The output is converted into words of $n$ elements. In our case $n = 6$. It means we work with a time series $Y = \{y_i, i = 1, ...K$ with $K = N/6\}$ of natural numbers $y_i \in [0, 2^6 - 1]$.

The obvious *PDF* (probability density function) to characterize $Y$ is the normalized histogram of the $K$ words $Y$; let us call it $PDF_{hist}$. Its normalized Shannon entropy $H_{hist}$ is given by:

$$H_{hist} = \frac{\sum_{i=1}^{K} p_i \ log \ p_i}{logK} \tag{1}$$

We call this *PDF non-causal* because it does not change if we permute the time order of the words, and consequently can not detect any causal connection between consecutive words. The normalized entropy $H_{hist}$ quantifies the equiprobability of the words among all possible values. For a *PRNG* its ideal value is $H_{hist} = 1$.

To quantify statistical independence between consecutive words we use a *causal PDF*, proposed by Bandt & Pompe [13]. This *PDF* is obtained by assigning ordering patterns to overlapped segments, of length $D$, of the time series trajectory. The process is as follows: 1) group $D$ consecutive words $\{y_i, y_{i+1}, ..., y_{i+D}\}$ (let us stress that in our case each $y_i$ is a natural number $\in [0, 63]$). The ordering of the $D$ values inside each group is compared with the order of numbers $\{1, 2, ..., D\}$. There exist $D!$ possible permutations of $D$ elements. Each permutation is called an *ordering pattern* [14] and is labelled with a permutation number $\pi = 1, ..., D!$. The normalized histogram of $\pi_i$'s is called the Bandt &

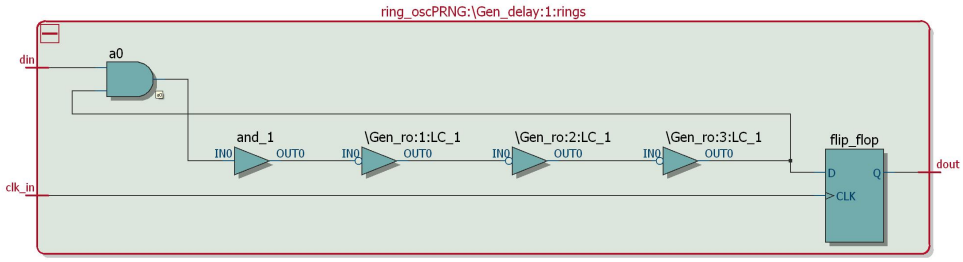


Fig. 2. RTL view one ring with 3 inverters.

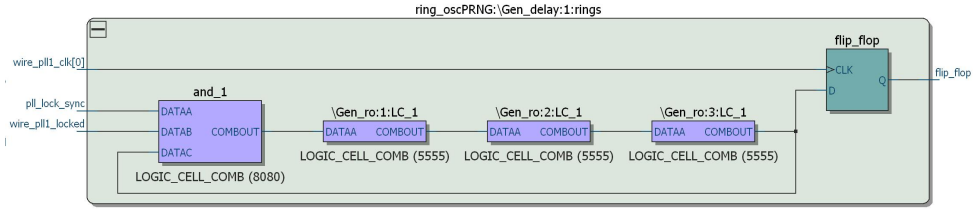


Fig. 3. Technology map viewer (post mapping), one ring with 3 inverters.

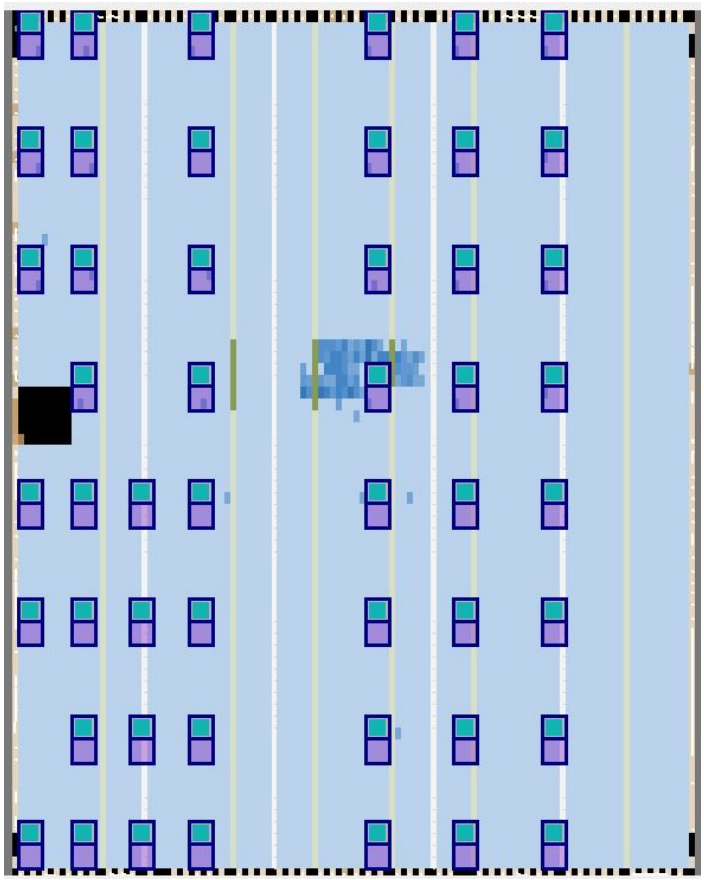

Fig. 4. *Chip Planner* view *LogicLock* regions.

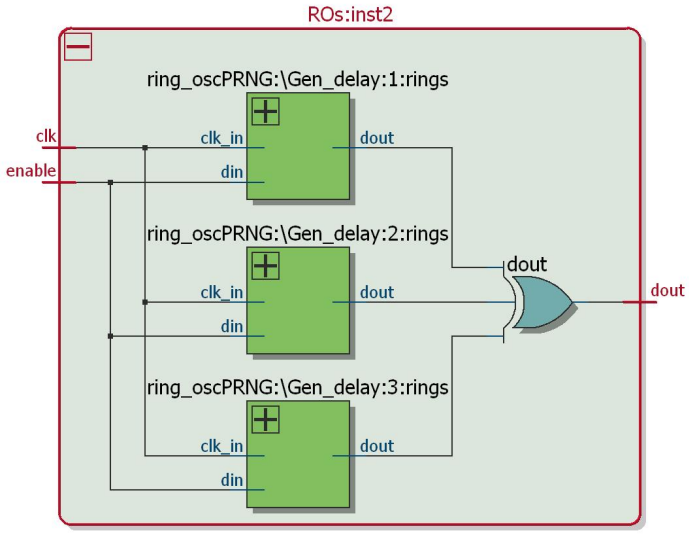


Fig. 5. *RTL* view of *PRNG* with 3 *RO*s.

Pompe *PDF*, $PDF_{BP}$. The normalized Shannon entropy of this $PDF_{BP}$ is $H_{BP}$ where the subscript $BP$ means "Bandt and Pompe".

In case two values of $y_i$ inside the same group are identical, it is considered that the first one is lower than the last one in order to obtain an unique result. For *PRNG*s this procedure does not produce significant changes in $PDF_{BP}$.

The Bandt & Pompe procedure has the advantages of being: 1) simpler and fast to calculate than block entropies, 2) robust in presence of noise, and 3) invariant to lineal monotonous transformations. It is applicable not only to *PRNG*s but also to any weak stationary process (it means for $k = D$, the probability that $x_t < x_{t+k}$ does not depend on the particular $t$ [13]). The causality property of $PDF_{BP}$ makes the quantifiers based on this *PDF* to discriminate between deterministic and stochastic systems [15].

Bandt and Pompe suggested $3 \leq D \leq 7$. $D = 6$ is adopted in this work.

A full discussion about the convenience of using different quantifiers to measure a given *PDF*is out of the scope of this work. Nevertheless reliable bibliographic sources do exist [16], [17], [18], [9], [19], [20], [21].

In this paper we adopt plane $H_{BP}$ vs $H_{hist}$ [9] to represent each *PRNG*. A higher value in any of the entropies, $H_{BP}$ and $H_{hist}$, implies an increase in the uniformity of the involved *PDF*'s. The point $(1, 1)$ represents the ideal point for a *PRNG* with uniform histogram and uniform distribution of ordering patterns.

## IV. Results

The *Embedded Logic Analyzer* tool is utilized for collecting the random sequences generated. It constitutes a *system-level debugging tool*, provided by *Altera* [22], that captures and storages the real-time signal behavior and allows one to observe interactions between hardware and software in system designs. After acquiring the data and save them into a *SignalTap* II file, they can be analyzed or viewed as a waveform. With this procedure nor extra jitter neither distortion are introduced in the measured signal from the data acquisition chain.

In the case of *RO* based *PRNG* data files with 917504 bits each were generated for each *RO* based *PRNG*. We consider sets of $N_{RO}$ rings, each with 3 inverters; $N_{RO} = 2$, 3, 4, 5, 6, 7, 15, 25 and 50.

Data from *SignalTap* were processed using *Matlab*©. Binary data were grouped in 6-bits words without superposition, so files with 152917 data each were generated. Quantifiers described in section III were calculated for all generated files.

We also evaluated other known noises generators to compare their quality with that of the *RO*-based *PRNG*. The noises analyzed are:

- Mersenne Twister pseudo-random number generator, [23].
- Two algorithms employed for generate random data by Matlab (Multiplicative Congruential method) [24] and Excel [25].
- Two *physical noises*: radioactive decay noise [26] and atmospheric noise [27]. Data files for these noises are available from the referred *websites*.
- Two chaotic map $M^1$ and their iterated versions $M^2$ to $M^8$ [9] for the logistic map (*LOGISTIC*) and the *three way Bernoulli map* (*TWBM*).

Fig. 6 shows the results in the dual entropy plane $H_{BP}$ vs $H_{hist}$ for all these noises. It can be seen that the *physical noises*, the algorithmic *Mersenne Twister*, and the *PRNG*s used in *Matlab* ©(*rand* function) and *Excel*© (*RAND* function), have the maximum value for $H_{BP}$, indicating that all the ordering patterns appear almost the same number of times. However these five noises present very different behavior with respect the $H_{hist}$ quantifier. The *radioactive decay* is the worst, with a value of $H_{hist}$ about 0.5 indicating that this sequence does not exhibit all possible values in the same proportion. In Fig. 6 the numbers next to each marker for the chaotic sequences, indicate the number of iteration. The iterated maps have higher $H_{BP}$ because of their mixing property [9].

In the case of the *RO*-based *PRNG* sequences, numbers next to each square indicate the quantity of *RO*s employed in that *PRNG* (let us stress that the number of inverters is fixed to 3). The dual entropy plane shows that an increase in the number of *RO*s improves both $H_{BP}$ and $H_{hist}$.

Fig. 7 is a zoom of Fig. 6 around the ideal point $(1, 1)$.

There, it is shown the evolution of the *RO*-based *PRNG* sequences when the quantity of *RO*s increases from 5 to 50 (numbers next to each square). It can be seen that as the number of rings increases, data increase their mixture and also the histogram tends to be more uniform. So both properties are improved. Here a threshold in the number of rings can be determined, as the points saturate at about $(0.997, 1)$, so this is the best *PRNG* possible. Further, using more than 15 *RO*s presents no improvement. As it was previously said, $H_{hist}$ quantifier detects the histogram variation of the sequence, and

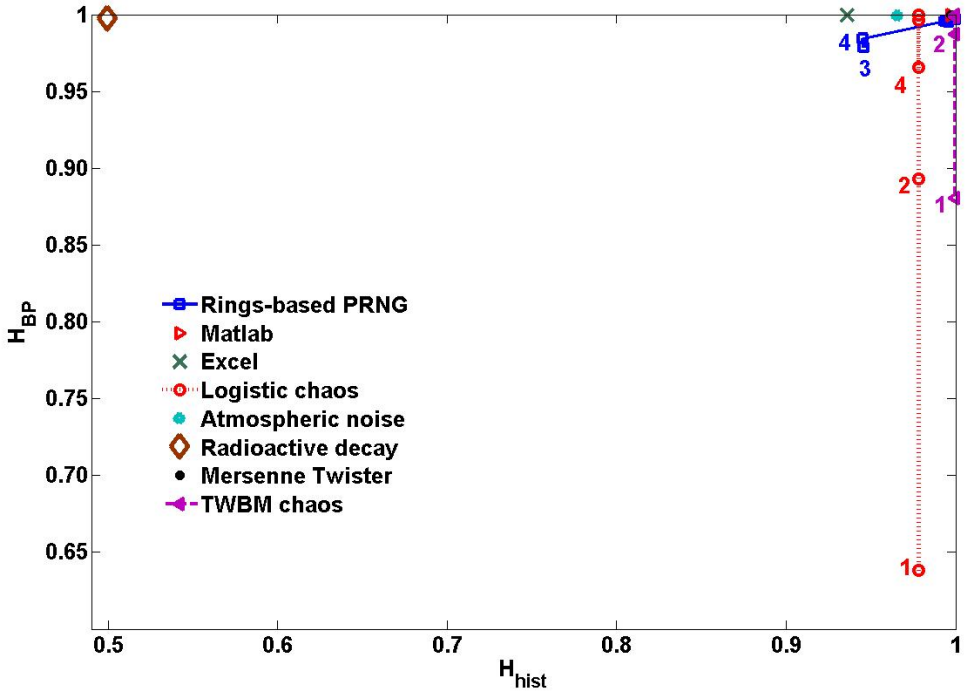


Fig. 6. $H_{BP}$ vs $H_{hist}$ plane for several noises, numbers next to each square indicate the quantity of *RO*s used in the *RO* based *PRNG*. Numbers next to each point in the chaotic sequences labeled *Logistic* and *TWBM* indicate the number of iteration of the chaotic map (see the text for details).

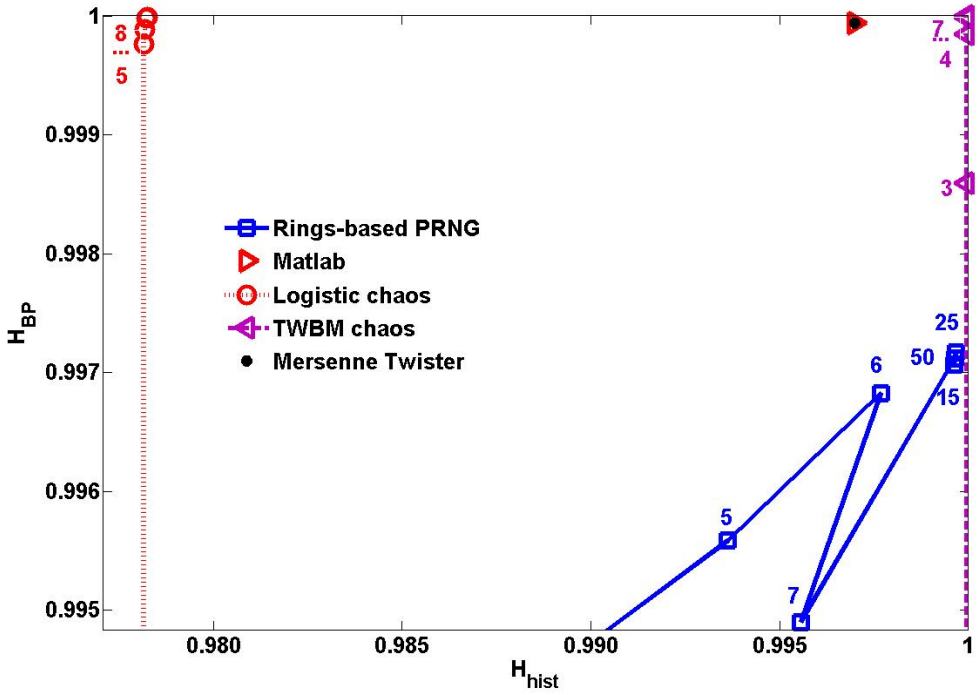


Fig. 7. Zoom of Fig. 6 around the ideal point $(1, 1)$ of the $H_{BP}$ vs $H_{hist}$ plane. Numbers next to each square indicate the number of *RO*s used in that rings-based *PRNG*. Numbers next to each point in the chaotic sequences indicate the number of iteration of the chaotic map.

the $H_{BP}$ quantifier reflects the improvement in the mixing of data. Finally, Mersenne Twister and Matlab sequences present identical value, ideal $H_{BP}$, and a high value of $H_{hist}$ nonetheless the histogram is not perfectly uniform (values are not equiprobable).

## V. Conclusions

*RO*-based *PRNG* implemented here has demonstrated to satisfactorily meet the statistical properties desired by a *PRNG*. They are comparable of other used *PRNG*s and in some cases they are better. They employs few resources of the device and they are simply to implement in a digital platform.

It was demonstrated that for these architectures of *PRNG* the quantity of *RO*s establishes *PRNG*'s statistical properties. It was seen that for 15 *RO*s both output's statistical properties, histogram and mixing, were almost ideal, making unnecessary the increase of the number of rings.

The dual entropy plane proposed here has demonstrated to satisfactorily discern between the *PRNG*'s two main desired properties, the equiprobability among all possible values and the statistical independence between consecutive values. Thus, it allows to clearly see what needs to be improved in a given sequence.

## Acknowledgment

This work was partially supported by the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina (PIP 112-201101-00840), and UNMDP Argentina.

## References

[1] A. Hajimiri, S. Limotyrakis, and T. H. Lee, "Jitter and phase noise in ring oscillators," *IEEE Journal of Solid-state Circuits*, 1999.

[2] M. K. Mandal and B. C. Sarkar, "Ring oscillators: Characteristics and applications," *Indian Journal of Pure & Applied Physics*, vol. 48, pp. 136–145, feb 2010.

[3] N. Gupta, "Article: Voltage-controlled ring oscillator for low phase noise application," *International Journal of Computer Applications*, vol. 14, no. 5, pp. 23–27, January 2011, published by Foundation of Computer Science.

[4] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 109–119, 2007.

[5] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in fpga based on oscillator rings," *Int. J. Reconfig. Comput.*, vol. 2009, pp. 4:1–4:8, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/501672

[6] NIST, "Guidelines for evaluating and expressing the uncertainty of nist measurement results; appendix d," Tech. Rep., 2000. [Online]. Available: http://physics.nist.gov/Pubs/guidelines/appd.1.html

[7] G. Marsaglia, "The marsaglia random number cdrom including the diehard battery of tests of randomness," *http://www.stat.fsu.edu/pub/diehard/*, 1995.

[8] NIST, *Analysis of repeatability*, 2000. [Online]. Available: http://www.itl.nist.gov/div898/handbook/mpc/section4/mpc441.htm

[9] L. De Micco, C. M. González, H. A. Larrondo, M. T. Martin, A. Plastino, and O. A. Rosso, "Randomizing nonlinear maps via symbolic dynamics," *Physica A*, vol. 387, pp. 3373–3383, 2008.

[10] ALTERA, "ip-basesuite.html," *http://www.altera.com/products/ip/design/basesuite/ip-basesuite.html.*, 2008.

[11] Altera, "Designing with lowlevel primitives," http://www.altera.com/literature/ug/ug_low_level.pdf, 2007.

[12] ——, "Creating and manipulating logiclock regions," http://quartushelp.altera.com/current/.

[13] C. Bandt and B. Pompe, "Permutation entropy: a natural complexity measure for time series," *Phys. Rev. Lett.*, vol. 88, pp. 174 102–1, 2002.

[14] J. M. Amigó, L. Kocarev, and J. Szczepanski, "Order patterns and chaos," *Physics Letters A*, vol. 355, pp. 27–31, 2006.

[15] O. A. Rosso, L. Zunino, D. G. Pérez, A. Figliola, H. A. Larrondo, M. Garavaglia, M. M. T., and A. Plastino, "Extracting features of gaussian selfsimilar stochastic processes via the bandt & pompe approach," *Phys. Rev. E*, vol. 76, no. 6, p. 061114, 2007.

[16] R. Wackerbauer, A. Witt, H. Atmanspacher, J. Kurths, and H. Scheingraber, "A comparative classification of complexity measures," *Chaos, Solitons, Fractals*, vol. 4, pp. 133–173, 1994.

[17] R. López-Ruiz, H. L. Mancini, and X. Calbet, "A statistical measure of complexity," *Phys. Lett. A*, vol. 209, pp. 321–326, 1995.

[18] O. A. Rosso, H. A. Larrondo, M. T. Martin, A. Plastino, and M. A. Fuentes, "Distinguishing noise from chaos," *Physs. Rev. Lett., pp154102-154106*, vol. 99, 2007.

[19] O. A. Rosso, L. De Micco, H. A. Larrondo, M. T. Martin, and A. Plastino, "Generalized statistical complexity measure: a new tool for dynamical systems," *International Journal of Bifurcation and Chaos, Vol. , No. 3 (2010)* ., vol. 20, no. 3, p. 775785, 2010.

[20] M. T. Martín and A. Plastino, "Generalized statistical complexity measures: Geometrical and analytical properties," *Physica A*, vol. 369, pp. 439–462, 2006.

[21] O. A. Rosso, L. C. Carpi, P. M. Saco, M. Gómez Ravetti, A. Plastino, and H. A. Larrondo, "Causality and the entropycomplexity plane: Robustness and missing ordinal patterns," *Physica A*, vol. 391, p. 4255, 2012.

[22] ALTERA, *Quartus II Handbook Version 9.1*, 2009.

[23] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, Jan. 1998. [Online]. Available: http://doi.acm.org/10.1145/272991.272995

[24] mathworks. Matlab. [Online]. Available: http://www.mathworks.com/

[25] A. I. McLeod, "Remark AS R58: A Remark on Algorithm AS 183. An Efficient and Portable Pseudo-Random Number Generator," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 34, no. 2, 1985. [Online]. Available: http://dx.doi.org/10.2307/2347378

[26] J. Walker, "HotBits: Genuine random numbers, generated by radioactive decay," *online at www. fourmilab. ch/hotbits*, 2001.

[27] M. Haahr. Random.org. [Online]. Available: http://www.random.org/integers