Iran J Sci Technol Trans Electr Eng (2019) 43 (Suppl 1):S259–S268
https://doi.org/10.1007/s40998-018-0122-0

RESEARCH PAPER

New and Efficient Method for Extending Cycle Length of Digital Chaotic Systems

Lahcene Merah[1] · Adda Ali-Pacha[2] · Naima Hadj-Said[2] · Mecheri Belkacem[1]

Received: 29 July 2017 / Accepted: 18 July 2018 / Published online: 30 July 2018

Abstract
Implementing chaotic systems using digital computers with finite arithmetic precision leads to significant degradations on their quality of chaotic dynamics, and the important shortcoming found on digital chaotic systems is their limited cycle length. Notable efforts have been exerted recently to overcome this problem and enhance the quality of digital chaotic generators, and the aim is to generate chaotic sequences with long cycle lengths. Perturbation of chaotic systems orbits is the most efficient technique that has been adopted in this context. In this paper, we propose a new method for perturbing the orbits of chaotic systems. Compared to many proposals, our method does not need an external generator to perturb the chaotic orbit, and it has a self-perturbation mechanism. Evaluation results showed that the proposed method can extremely extend the cycle length of a given chaotic system in which no repeated patterns have been detected even using low arithmetic precision. The results also showed that the perturbed chaotic system has good statistical proprieties in terms of randomness; it passed successfully a set of statistical tests (NIST and Diehard). The whole system has been implemented in FPGA-based hardware, and real-time results are given. Compared with some proposals, the proposed method has provided better results in terms of randomness and hardware performance.

Keywords Chaos · Cryptography · NIST · Diehard · Cycle length · FPGA

1 Introduction

Random numbers play an important role in nowadays digital systems, and they can be found in many digital systems. As digital computers dominate data processing fields today, random number generators implemented in digital computers are known as pseudo-random number generators (PRNGs). The deterministic nature of the process leads to the term pseudo-random. PRNGs algorithms are widely used today thanks to their simplicity of implementation in both software and hardware. They are capable of generating sequences of numbers which appear random-like from many aspects.

Though they are necessarily periodic and their periods are very long, they pass many statistical tests and can be easily implemented with simple software routines (Elsherbeny and Rahal 2012). PRNGs have been widely used in Monte Carlo simulations, test pattern generation, cryptography, and telecommunication systems (Liu and Miao 2015). A good PRNG should have characteristics of: (1) long-period random number sequence; (2) a fit in statistical properties; (3) a high throughput rate; and (4) an unpredictability (Li et al. 2012). Finding all aforementioned characteristics together in one PRNG is a big challenge, due to their fixed linear structure, and most known pseudo-random generators (Linear Feedback Shift Registers, Linear congruential generators, etc.) are not secure enough, especially when used for information security.

✉ Lahcene Merah
l.merah@lagh-univ.dz

Adda Ali-Pacha
a.alipacha@gmail.com

Naima Hadj-Said
naima.hadjsaid@univ-usto.dz

Mecheri Belkacem
b.mecheri@lagh-univ.dz

[1] Department of Electronics, University of Laghouat, Bp 37G route de Ghardaia, Laghouat, Algeria

[2] Department of Electronics, University of Science and Technology of Oran USTO, BP 1505, 31036 El MNaouer Oran, Algeria

Not long ago, many researchers have focused on chaotic signals as a source of randomness because of their attractive proprieties (Oishi and Inoue 1982; Lin and Chua 1993; Bernstein and Lieberman 1991; Kolesov et al. 2001; Stojanovski et al. 2001; Andrecut 1998). A chaotic system by nature is a dynamical system that provides some characteristics such as noise-like behavior, sensitive dependence on initial conditions, unpredictability, aperiodicity and ergodicity. These characteristics are very useful making chaotic systems good alternatives to many conventional PRNGs and cryptosystems.

Unfortunately, chaos dynamics proprieties are more significant when these systems are studied in their original continuous state. The digital implementation of such systems using finite arithmetic precision leads to significant degradations on its dynamical behavior. These degradations are in fact a serious problem that detracts from importance of chaotic systems application.

Therefore, in order to benefit from the chaotic dynamic properties using finite precision computations, the dynamical degradations arising from digitization process should be taken into account and solved as much as possible. Many researchers have been highlighted theses degradation problems with some proposed solutions (Binder and Jensen 1986; Beck and Roepstorff 1987; Palmore and Herring 1990; Fryska and Zohdy 1992; Li et al. 2001a, 2003; Hu et al. 2014; Liu et al. 2014, 2017; Deng et al. 2015a, b).

The digitized chaotic systems are often known as pseudo-chaotic generators (PCGs), like PRNGs, and PCG suitable for cryptography should provide good statistical proprieties and should be unpredictable. Technically, unpredictability can be interpreted as cycle length of the chaotic generator. In other words, due to the computation finite precision; although chaotic generators seem random, they have a finite period that when the generator reaches; then, it will repeat the same orbit. The process of dealing with dynamical degradations of digital chaotic system can be considered as making a periodic system "chaotic", which is known as "chaotification" or "anti-control" of chaos (Hu et al. 2014).

This paper is organized as follows. Section 2 deals with dynamical degradations arisen from digitizing process of chaotic systems. Sect. 3 describes some proposals on the context of overcoming dynamical degradations in digital chaotic signals. Section 4 is devoted to represent our proposed idea for extending cycle length of digital chaotic systems, in which enough details and evaluation of the proposed scheme are presented. The FPGA-based hardware implementation of the proposed scheme is the subject of Sect. 5. Section 6 concludes the paper.

2 Dynamical Degradations in Digital Chaotic Systems

The digital implementation of chaotic systems on computers has the same concept of analog-to-digital conversion process in electronics. Depending on the computing arithmetic precision provided by the computer, the implemented chaotic function generates time series with values rounded to the nearest continuous value. The difference between the rounded value and the real one represents the quantification error. Following the analytic idea used in Li et al. (2003), consider a one-dimensional chaotic system $T : X \rightarrow X$, defined on [0, 1]:

$$X(n+1) = F(x(n)) \tag{1}$$

Assume that the fixed-point arithmetic precision is adopted; the chaotic iterations will be confined in the following discrete set:

$$S = [x_n \in [0,1] | x_n = n \times 2^{-L}, n = 0, 1, ., 2^L - 1]$$

where L represents the word size (Largest precision). Hence, any quantity of the digital chaotic system (1) (initial condition, control parameter or generated output) can never take a value out of the specified set S. The digital chaotic system (1) can be represented as follows:

$$\tilde{X}(n+1) = F_L(\tilde{X}(n)) = D_L o F(\tilde{X}n)) \tag{2}$$

where $\tilde{X}(n)$ represents the digital state vector and D_L the quantization function. Three quantization functions are used in almost today's computers: floor function, ceil function and round function.

2.1 Quantization Error in Digital Chaotic Systems

D_L will introduce the quantization error to the digital sequence generated by the system (1); consequently, due to the high sensitivity of the chaotic systems to the initial parameters, the quantification error quickly propagates into the mainstream of the generated sequence and changes the topology of the systems attractor. The pseudo-orbits (which designate the digitized chaotic orbits) represented in finite precision arithmetic can be entirely different from the theoretical ones (Li et al. 2005).

Some computation precisions lead to different and unexpected chaotic and non-chaotic structures. As an example, Fryska and Zohdy (1992) have implemented a 3-D piecewise linear chaotic system using single-precision floating-point arithmetic (32 bits) and extended double-precision floating-point arithmetic (80 bits). The first case gave a two-scroll chaotic attractor that is highly instable. The surprising results were in the second case, even though

the used parameters were kept unchanged, unexpected results were obtained: a periodic attractor (Fig. 1).

The influence of the arithmetic precision (quantization error) on chaotic dynamics has also been studied in depth in Palmore and Herring (1990), where the authors concluded that quantization errors using floating-point arithmetic can propagate into the mainstream of a chaotic system and, within 30–50 steps, destroy all the accuracy of the result.

2.2 Cycle Length in Digital Chaotic Systems

A very important issue related to digitization of chaotic systems is their long-term dynamics. Numerous studies have mentioned the direct relationship between digitized chaotic systems cycle length and the used arithmetic precision. Accordingly, the digital chaotic systems are considered as periodic and its longest orbit can never exceed 2^L. In accordance with Li et al. (2003), a typical orbit of a computerized chaotic system has a limited length n, where $n \ll 2^L$ in most cases . The *orbit length* has two main parts (Fig. 2), *transient length* (l) from x_0 to x_l and *cycle length* (n) from x_l to x_n, and consequently the orbit length $= l + n$. Numeric simulations have found that maximal length of computerized chaotic orbits is $O(2^{\varepsilon L})$, where $0 < \varepsilon < 1$ and $1/\varepsilon >> 1$ is right for many chaotic systems (Li et al. 2003).

3 Purifying Digital Chaotic Systems from Degradations

The agreed issue is that digital chaotic systems are periodic and their cycle lengths may be rather short (despite the existence of long ones). This sensitive issue makes the usefulness of applying chaos in cryptography meaningless.

Thus, to counteract degradations resulting from digitizing continuous chaotic systems, extensive research has been done recently from both theoretical and experimental points of views. The proposed solutions can be classified into 3 main categories, using higher arithmetic precision, cascading multiples chaotic systems and randomly perturbing the digital chaotic orbits. The efficiency evaluation of theses remedies can be found in Li et al. (2005).

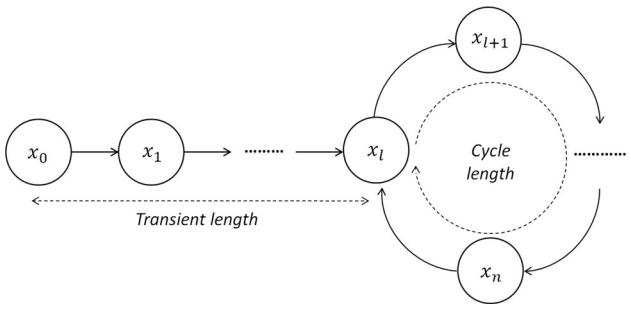

Fig. 2 Digitized chaotic systems typical orbit

3.1 Using Higher Arithmetic Precision

By increasing the computation arithmetic size, the cycle length of the digitized chaotic system can be effectively extended. The average cycle length will be prolonged exponentially as the precision increases (Hu et al. 2014), but, however, the computation arithmetic is still finite and there still exist plenty of short chaotic orbits; consequently, due to the computation power available today, it is unreasonable to trust this idea and confirm its efficiency.

Some weak control parameters lead to a worse distribution and produce chaotic sequences with short cycles; in this case, increasing arithmetic precision has no effect which implies that this idea is still vulnerable.

3.2 Cascading Multiples Chaotic Systems

This idea is based on using multiple chaotic generators and then combines their outputs together for producing the whole chaotic sequence. In Heidari-Bateni and McGillem (1994), the authors have cascaded two chaotic systems in a spread spectrum communication system, one for generating the chaotic sequence and the other for generating the initial

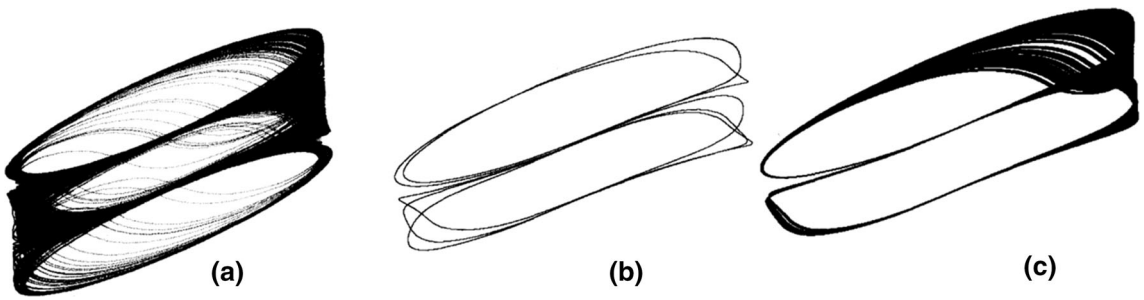

Fig. 1 The chaotic attractor studied in Fryska and Zohdy (1992), **a** implemented using single-precision floating-point arithmetic (32 bits), **b** using extended double-precision floating-point arithmetic (80 bits) and **c** obtained from the exact solution

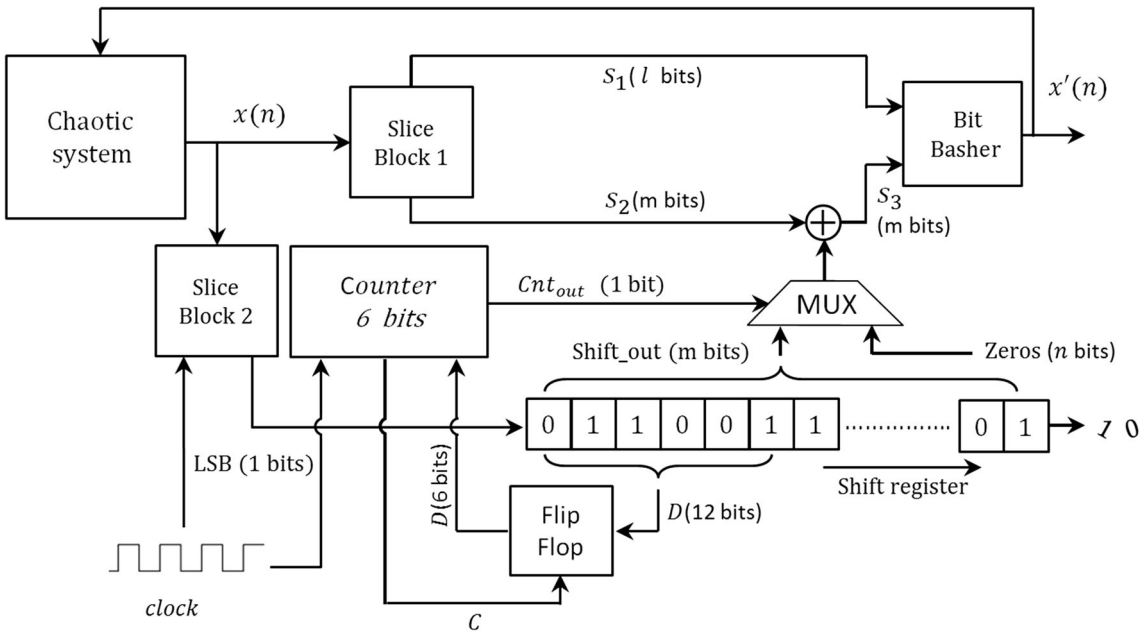

Fig. 3 The proposed scheme for extending cycle length of digital chaotic systems

conditions for the other system. After N iterations for the second chaotic system, the first system generates a new initial condition to initiate the second one.

Although this idea can effectively prolong the period of the generated chaotic sequence; Li et al. (2005) and Binder and Jensen (1986) assumed that the distribution of output sequence of digital chaotic system for the proposed idea is not uniform in discrete phase space even though the input sequence is uniformly distributed.

Another important issue related to the cascading multiple chaotic systems technique is the hardware cost and performance. In fact, what we can benefit from this technique in terms of cycle extension may be lost in the cost of hardware implementation of such designs, and consequently, reduces its performance in general.

3.3 Perturbing the Chaotic System Orbit

The basic idea of this method consists of using either random or pseudo-random sequence to perturb the digital chaotic orbits. This technique is more efficient compared to the other two and was supported and adopted by many theorists and practitioners (Černák 1996; Li et al. 2001b, 2005; Fryska and Zohdy 1992; Hu et al. 2014; Merah et al. 2015; Blank 1994; Hasimoto-Beltrán and Ramírez-Ramírez 2011; Tao et al. 1998; Shu-Bo et al. 2009; Tong 2013). Experiments have shown that such a simple remedy can improve the dynamical properties of computerized (digital) chaos to some extent (Li et al. 2003). Perturbation techniques can be classified into three main categories: perturbing the orbits of a given digital chaotic system, perturbing the control parameters, and perturbing both orbits and control parameters; the first two methods are mostly used.

Tao et al. (1998) had easily deduced that the new cycle length of the perturbed schemes is $T^{'} = \sigma \Delta(2^L - 1)$, where σ is a positive integer, Δ is the perturbation period, and $(2^L - 1) = T$ is the cycle length of the perturbing signal. Assuming perturbing system uses the same finite precision as the digital chaotic system and has a maximum length of 2^L, the lower bound of the system cycle length is $T^{'}_{\min} = \Delta 2^L$ which is greater than the cycle length 2^L.

4 The Proposed Perturbation Mechanism (PPM)

In this section, the proposed idea for perturbing a given chaotic system orbits and extending its cycle length is described. As mentioned previously, the proposed circuit is self-perturbation; that is, our digital chaotic system does not need an external perturbation source; it provides a self-generator for perturbing its orbits.

As shown in Fig. 3, the proposed scheme has 4 main parts: the chaotic system, a counter, a shift register of m memory cells, and slice blocks, each part has its own function as follows:

- *The chaotic system* It has the role of generating the chaotic sequence $x(n)$ which has L bits of length.
- *Slice block 1* This block divides the binary sequence of the chaotic output $x(n)$ into two parts: the sequence S_1

that represents the l upper bits of $x(n)$ and the sequence S_2 that represents the m lower bits of $x(n)$; note that $L = l + m$ where L is the largest computation precision used to implement the chaotic system.

- *Slice block 2* This block extracts the LSB (low significant bit) from the chaotic output $x(n)$ at each clock cycle.
- *Shift register* It has m bits of size; this block has the role of loading the LSB generated from the slice block 2 and shifts it to the right at each clock cycle. The shifting operation works continuously, when the register cells are full; the register excludes the right bit to accept new bit on its left side (Fig. 3).
- *Counter* It has the most important role in the circuit because it defines the perturbation periods T_p. The counter is controlled by a flip-flop, and this last has an initial integer value N (12 bits) which represents the maximum that the counter can reach. When the counter reaches the maximum N, two signals named Cnt_{out} and C change their states from '0' to '1', for the signal C, when activated; it tells the flip-flop to load new data (D) from the shift register. The signal Cnt_{out} controls a multiplexer that outputs either a sequence of zeros (m bits of length) or the shift register content (of m bits of length bits of length).
- *Bit basher block* It has the role of collecting the signals S_1 and S_3 (perturbed version of the signal S_2) to produce the new perturbed sequence $x'(n)$.

Roughly speaking, we can understand from what have been said above that the chaotic signal $x(n)$ is perturbed during variable periods $T_p = N \times T$ (where T represents the system clock period). During the counting process, S_1 is XORed with zeros which means that it is kept unchanged ($S_3 = S_1$) and obviously $x(n) = x'(n)$. If the counter reaches its maximum, the signal Cnt_{out} will be changed from '0' to '1' and then S_2 is XORed with the shift register content (output of the multiplexer); in this case, $x'(n)$ becomes a perturbed version of $x(n)$.

Table 1 Computed transient length and cycle length of the logistic map output for different initial conditions

The initial condition	Transient length	Cycle length
0.25	56	62
0.1	32	38
0.62	44	62
0.37	102	38
0.73	57	39
0.86	50	38
0.96	45	38

4.1 Experimental Evaluation of the Proposed Perturbation Mechanism

The PPM has been evaluated by applying it on the logistic chaotic map given by:

$$x(n+1) = rx(n)(1 - x(n)) \tag{3}$$

where r is the control parameter. Before applying our PPM, we will try to evaluate the original chaotic system given above by using low arithmetic precision. So, by using a largest arithmetic precision $2^{16}(L = 16)$, and fixing r to 3.9 (chaotic behavior), we have randomly selected a set of initial conditions and for each one we have computed the transient length and cycle length of the logistic map output. Table 1 summarizes the obtained results.

It is clear from the given results that the logistic map provides very short cycle lengths using the adopted arithmetic precision. In this case, the chaotic properties of the logistic map are meaningless. The system is no longer chaotic, and it is purely periodic. The auto-correlation function can also help to detect periodicities on the logistic map output. The result of the auto-correlation function performed on the logistic map is given in Fig. 4 in which a strong correlation is observed on the output sequence.

The PPM is now applied to the original system of the logistic chaotic map, by fixing $m = 15$ bits and $l = L - m = 1$. The output of the new logistic map is shown in Fig. 5. Figure 6 shows the auto-correlation function result performed on the modified logistic chaotic map for more than 70 million samples. It is clear that the modified logistic map provides good independence between samples compared to the original one.

It should be noted that the available computing configuration did not allow us to process more than 100 millions samples (i5-2500K CPU with 8Gb of RAM). Thus, we

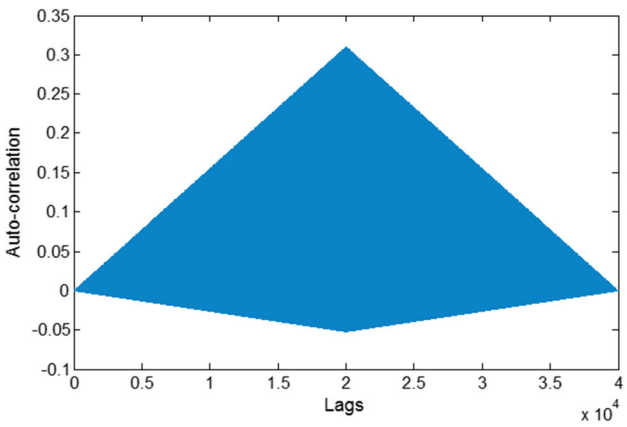

Fig. 4 The auto-correlation function of the chaotic sequence generated from the original digital chaotic system of the logistic map

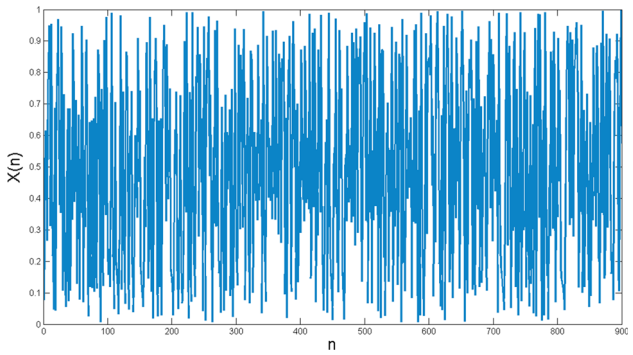

Fig. 5 The output sequence of the modified digital logistic chaotic map with largest precision $L = 2^{16}$

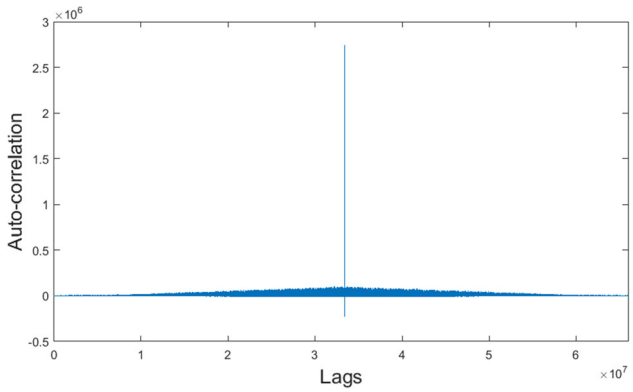

Fig. 6 Auto-correlation function result performed on the modified digital logistic chaotic map with the largest precision $L = 2^{16}$ and for more than 70 million samples

have confirmed that no periodic orbits have been detected for a sequence of 100 millions samples and less.

4.1.1 Evaluation of the PPM by Detecting Eventual Cycles

In this case, a simple test is performed on the modified logistic map, and it consists of detecting repeated samples in the chaotic sequence and try to see whether there is a repeated orbits or not. If there are no similar orbits and even there are repeated samples, we can say that the PPM works well.

Experimentally, we tried to detect equal samples and then define their kth positions. We plotted chaotic subsequences starting from the defined kth positions. Thus, experimental results given in Fig. 7 showed (for 106 samples) the detected repeated sample $x_k = 0.94372558550$ (for example) and the k^th positions where x_k is repeated; $k = 263$, $k = 247{,}041$, $k = 498{,}476$, $k = 533{,}784$, $k = 610{,}577$.

It is clear from Fig. 7 that even though the detected samples are identical, the perturbation circuit forces the system to diverge to another different orbit.

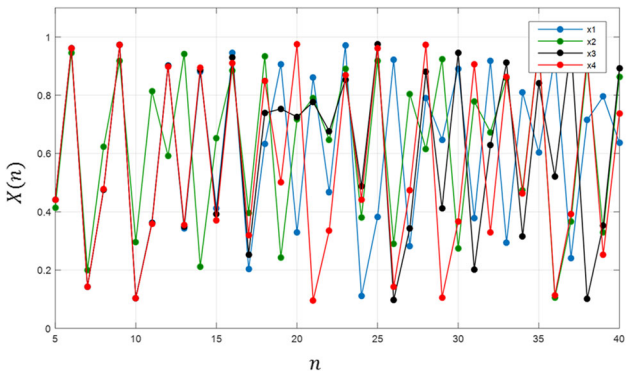

Fig. 7 Superposition of chaotic sub-sequences from the defined kth positions, x_1 from $k = 263$, x_2 from $k = 247{,}041$, x_3 from $k = 498{,}476$ and x_4 from $k = 610{,}577$

4.1.2 Evaluation of the PPM in Terms of Uniformity

Another important criterion that should be provided by good PRNGs is uniformity; this means that numbers generated from a given PRNG should be uniformly distributed. We have evaluated uniformity of the modified logistic map using our PPM and compared the results with the original logistic map. Figures 8 and 9 present the results of the histogram for both logistic maps.

It is clear from the obtained results that the modified logistic map using the PPM provides good distribution and its output is uniformly distributed compared with the original logistic map.

4.1.3 Evaluation of the PPM Using the Approximate Entropy Test

The modified chaotic map has been also evaluated using the approximate entropy test (*ApEn*). This test has been proposed first by Pincus (1991) for the purpose of detecting similar patterns in time series, and time series containing many repetitive patterns has relatively small *ApEn*; a

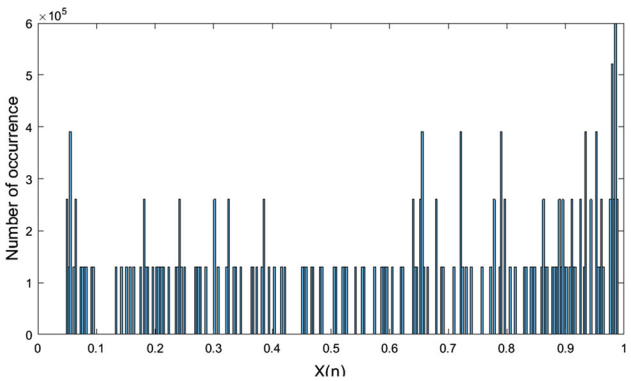

Fig. 8 Histogram of the output of the original logistic map

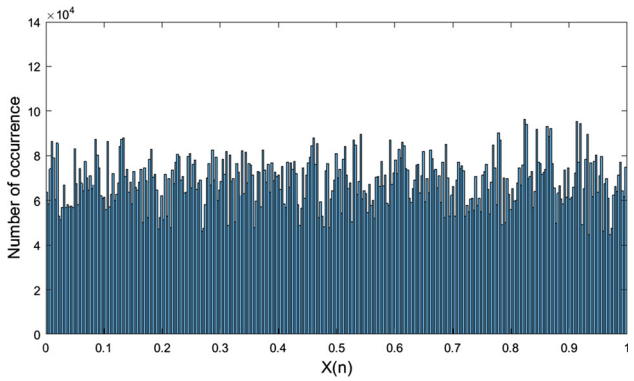

Fig. 9 Histogram of the output of the modified logistic map using the PPM

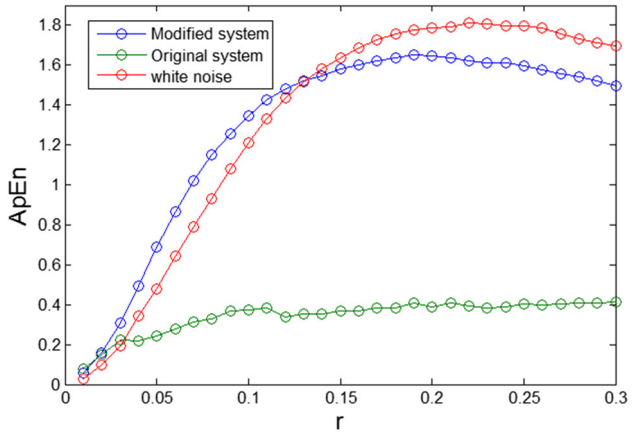

Fig. 10 The approximate entropy of the sequence of the modified logistic map compared to the original one and white noise

complicated process has higher *ApEn* and therefore is less predictable.

The algorithm of the *ApEn* works as follows: for a given time series $\{u(i), i = 1, 2, ., N\}$, reconstruct this series as:

$$X(i) = u(i), u(i+1), \ldots, u(i+m-1), i = 1, 2, \ldots n,$$
$$n = N - m + 1.$$

Now, the m-dimensional sequence $X(i)$ is used to construct $C_i^m(r)$ = (the number of $X(j)$ such that $d[X(i), X(j)] \leqslant (r)/(N-m-1)$, where m (= 2 in our case) is an integer

Table 2 Approximate entropy comparison between the modified logistic map and other proposal

References	N	m	r	$ApAn(\sim)$
Liu and Miao (2015)	10^4	2	0.3	1.25
Our system	10^4	2	0.3	1.45

Table 3 NIST statistical tests results performed on the modified logistic map output

Statistical test	Before p_value	After p_value
Frequency	0.57957	0.36176
Block frequency ($m = 128$)	0.05723	0.25206
Cusum-forward	0.81028	0.59774
Cusum-reverse	0.51390	0.96232
Runs	0.12025	0.59414
Long runs of ones	**0.0000**	0.05641
Rank	0.16526	0.94066
Spectral DFT	**0.0000**	0.89051
Non-overlapping templates ($m = 9$)	**0.0000**	0.86899
Overlapping templates ($m = 9$)	**0.0000**	0.64935
Universal	0.09494	0.36005
Approximate entropy ($m = 10$)	**0.0000**	0.96615
Random excursions ($x = +1$)	**0.0000**	0.52115
Random excursions variant ($x = -1$)	**0.0000**	0.14709
Linear complexity ($M = 500$)	0.87168	0.48665
Serial ($m = 16, \nabla\Psi_m^2$)	**0.0000**	0.26919

which represents the length of compared run of data, r (range from 0.1 to 0.3) specifies a filtering threshold, and d represents the distance between the vectors $X(i)$ and $X(j)$: $d[X(i), X(j)] = \max|u(i+j) - u(j+k)|, k = 0, 1, \ldots m-1$. Now we define:

$$\phi^m(r) = (N-m-1)^{-1} \sum_{i=1}^{N-m+1} \log(C_i^m(r))$$

Finally, the *ApEn* is given by:

$$ApEn = \phi^m(r) - \phi^{m+1}(r) \quad (4)$$

The results of the *ApAn* test performed on the modified digital chaotic system, the original system, and white noise sequence (as reference) are shown in Fig. 10. It is clear that the modified system has an *ApAn* much greater than the original system. This result proves the randomness quality of the modified system; we can say that the *ApAn* for the modified system is relatively equal to the *ApAn* of white noise. Table 2 contains a comparison between *ApAn* obtained for our modified system and other proposal (Liu and Miao 2015).

4.1.4 Statistical Proprities of Chaotic Sequence of the Modified Chaotic Map

This section is attended to evaluate our modified logistic map in terms of statistical proprieties of its output, for this

Table 4 Diehard statistical tests results performed on the modified logistic map output

Statistical test	Before p_value	After p_value
Birthday spacings	0.99976	0.02926
Overlapping 5-permutation	**1.0000**	0.99412
Binary rank (31 × 31)	0.99988	0.78610
Binary rank (32 × 32)	0.91109	0.97758
Binary rank (6 × 8)	**1.0000**	0.78514
Bitstream	**1.0000**	0.61735
OPSO	**1.0000**	**1.0000**
OQSO	**1.0000**	**1.0000**
DNA	**1.0000**	**1.0000**
Stream count the 1's	**1.0000**	0.01782
Byte count the 1's	**1.0000**	0.40970
Parking lot	0.10628	0.75554
Minimum distance	**1.0000**	0.13202
3D spheres	0.59229	0.79825
Squeeze	**1.0000**	0.25648
Overlapping sums	0.74016	0.53161
Runs up	0.83121	0.86695
Runs down	0.90629	0.18144
Craps	**0.0000**	0.95711

purpose; two well-known statistical tests batteries are used: NIST and Diehard, and for each test a p_value (significant level) is computed and should greater than 0.01 to say that the test is passed successfully. The results (for $X(0) = 0.55$) of the statistical tests performed both on the original logistic map and the modified one are given in Tables 3 and 4, the underlined results mean that the test is fail.

It is clear from the obtained results that the modified logistic map provides good statistical proprieties of

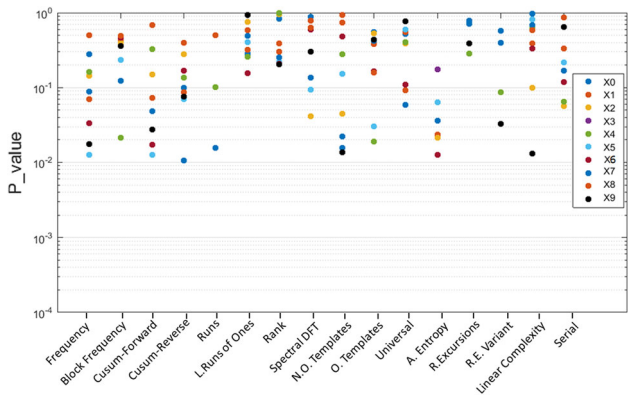

Fig. 11 NIST statistical tests results performed on the modified logistic map output for different seeds

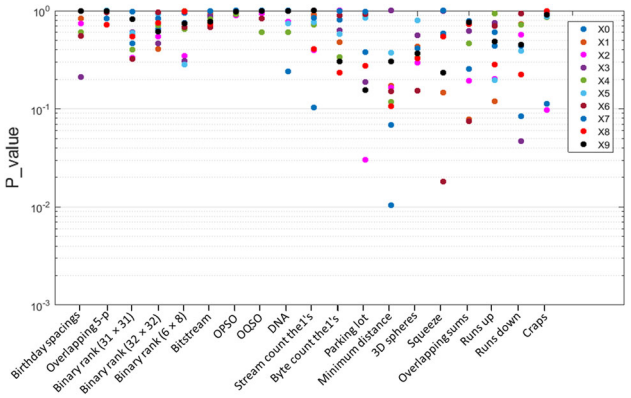

Fig. 12 Diehard statistical tests results performed on the modified logistic map output for different seeds

randomness compared to the original one, and more than 90% of the performed tests are passed successfully.

To affirm the good statistical properties of the modified logistic map, we have evaluated its output for different (randomly chosen) seeds (initial conditions) using both NIST and Diehard statistical tests. Figures 11 and 12 represent the obtained results for $X_0 = 0$, $X_1 = 0.1$, $X_2 = 1.2$, $X_3 = 0.25$, $X_4 = 0.30$, $X_5 = 0.43$, $X_6 = 0.58$, $X_7 = 0.67$, $X_8 = 0.78$, $X_9 = 0.85$ and $X_{10} = 0.95$.

The obtained results showed that more than 87.5% of the NIST statistical tests have been passed successfully ($p\_value > 0.01$) and more than 89.47% of Diehard statistical tests have been passed successfully ($0.01 \leqslant p\_value < 1$). George Marsaglia (Diehard statistical tests developer) has pointed out that we should not be surprised with occasional p_values near 0 or 1, such as 0.0012 or 0.9983. When a bit stream really FAILS BIG, we will get p_values of 0 or 1 to six or more places. By all means, do not, as a statistician might, think that a $p\_value < 0.025$ or $p\_value > 0.975$ means that the RNG has "failed the test at the 0.05 level". Such p_value happens among the hundreds that Diehard produces, even with good RNGs.

Thus, from all given results, we can say that the modified logistic map using PPM provides good statistical properties.

5 FPGA-Based Implementation of the Modified Logistic Chaotic Map Using the PPM

This section is devoted to present the results of the FPGA-based implementation of our system in terms of performance and hardware resource consumption. In fact, we have implemented two chaotic systems of the logistic map:

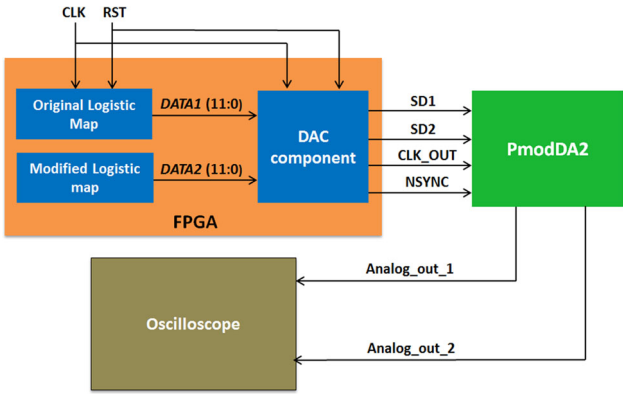

Fig. 13 Block diagram of real-time analog outputs of FPGA-based chaotic logistic maps [original logistic map and the modified one using the PPM (orange)]

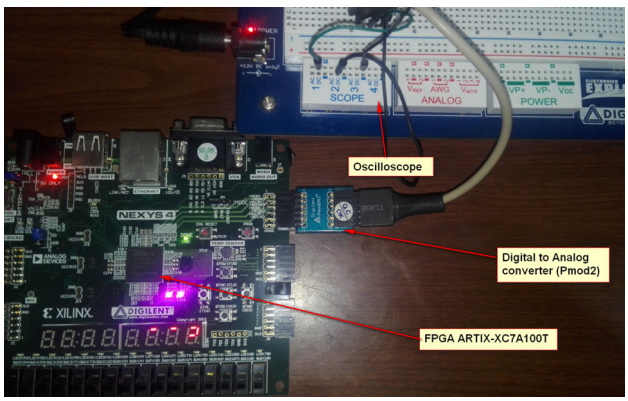

Fig. 14 Components used for real-time implementation of the design

the original logistic map and the modified one only for comparison purpose in real-time.

The system was designed using Xilinx System Generator tool (XSG), after some steps needed for implementation, the system was implemented on Xilinx Artix XC7A100T FPGA device (embedded on Digilent Nexys 4 board). In order to visualize the real-time FPGA output, we have used two digital-to-analog converters of National Semiconductor DAC121S101 of 12 bits of resolution (embedded on the Digilent PmodDA2 board). It is worth noting that in addition to the main VHDL code of the chaotic systems, another VHDL component is added to drive signals between the chaotic generator and the PmodDA2 board.

The DAC component (Fig. 13) receives the parallel data of 12 bits of length (DATA1 from the original logistic map and DATA2 from the modified logistic map), and a START command tells the component to begin the conversion process. The DAC component then converts the data signals from parallel to serial (SD1, SD2) and sends them to the PmodDA2. Another commands, CLK_OUT

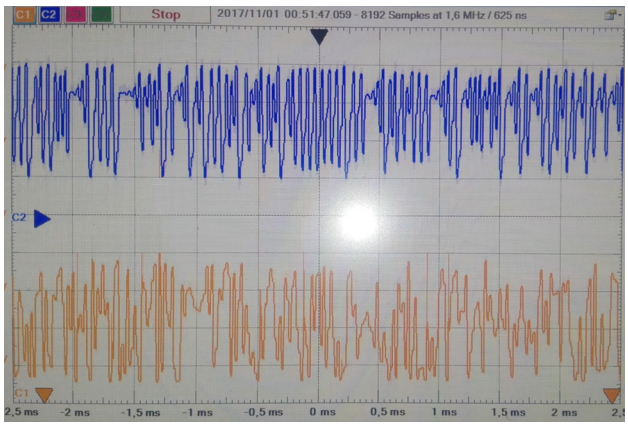

Fig. 15 Real-time outputs of FPGA-based implementation of both original logistic map (blue) and modified logistic map using the PPM (orange)

Table 5 Timing and resource consumption reports

Resources	Liu and Miao (2015)	Using PPM
Slice register	383	30
Slice LUTs	439	108
DSP48E1s	2	1
Frequency	801.66 MHz	1306.33 MHz

and NSYNS are also sent to the Pmod and represent, respectively, the clock signal needed for driving the DACs (25 MHz) and the signal used to latch the data inside the PmodDA2 after the data have been shifted out.

Figure 14 represents the hardware components used for the real-time implementation and visualization of both original logistic map and the modified logistic map using the PPM. Real-time outputs of the implemented design shown on oscilloscope are presented in Fig. 15.

The results of the achieved performance and consumed resources are given in Table 5 with the largest precision of $L = 2^{16}$. The obtained results have been compared with other modified logistic map; in fact, we have also implemented the proposed method in Liu and Miao (2015).

It is clear from the achieved results that our system provides the higher performance with fewer resources compared to the modified logistic map proposed in Liu and Miao (2015).

6 Conclusion

New, simple, and efficient method for extending cycle length of digital chaotic systems is presented in this paper. Our proposed idea does not need any external generator of

perturbing the orbit of the chaotic system; it contains a self-mechanism for ensuring perturbation, and this directly reflected on the performance of the design. The cycle length of the new modified chaotic system using our PPM has been extremely increased.

The results obtained from the evaluation of our proposed method prove its efficiency; the new chaotic sequence provides good uniform distribution and the higher approximate entropy compared to other proposals with good results in terms of auto-correlation function. (No repeated patterns are detected.) The statistical tests performed on the proposed system showed clearly the good randomness of its output compared with the original system.

The comparison between our proposed method with other proposals in terms of randomness, approximate entropy, and FPGA hardware implementation performance confirms its efficiency.

References

Andrecut M (1998) Logistic map as a random number generator. Int J Mod Phys B 12(09):921–930

Beck C, Roepstorff G (1987) Effects of phase space discretization on the long-time behavior of dynamical systems. Phys D Nonlinear Phenom 25(1–3):173–180

Bernstein GM, Lieberman MA (1991) Method and apparatus for generating secure random numbers using chaos. US Patent 5,007,087

Binder PM, Jensen RV (1986) Simulating chaotic behavior with finite-state machines. Phys Rev A 34(5):4460

Blank M (1994) Pathologies generated by round-off in dynamical systems. Phys D Nonlinear Phenom 78(1):93–114

Černák J (1996) Digital generators of chaos. Phys Lett A 214(3–4):151–160

Deng Y, Hu H, Xiong N, Xiong W, Liu L (2015a) A general hybrid model for chaos robust synchronization and degradation reduction. Inf Sci 305:146–164

Deng Y, Hu H, Xiong W, Xiong NN, Liu L (2015b) Analysis and design of digital chaotic systems with desirable performance via feedback control. IEEE Trans Syst Man Cybern Syst 45(8):1187–1200

Elsherbeny MN, Rahal M (2012) Pseudo-random number generator using deterministic chaotic system. Int J Sci Technol Res 1(9):95–97

Fryska ST, Zohdy MA (1992) Computer dynamics and shadowing of chaotic orbits. Phys Lett A 166(5–6):340–346

Hasimoto-Beltrán R, Ramírez-Ramírez R (2011) Cycle detection for secure chaos-based encryption. Commun Nonlinear Sci Numer Simul 16(8):3203–3211

Heidari-Bateni G, McGillem CD (1994) A chaotic direct-sequence spread-spectrum communication system. IEEE Trans Commun 42(234):1524–1527

Hu H, Deng Y, Liu L (2014) Counteracting the dynamical degradation of digital chaos via hybrid control. Commun Nonlinear Sci Numer Simul 19(6):1970–1984

Kolesov V, Belyaev R, Voronov G (2001) A digital random-number generator based on the chaotic signal algorithm. J Commun Technol Electron 46(11):1258–1263

Li S, Li Q, Li W, Mou X, Cai Y (2001a) Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudo-random coding. In: IMA international conference on cryptography and coding. Springer, pp 205–221

Li S, Mou X, Cai Y (2001b) Improving security of a chaotic encryption approach. Phys Lett A 290(3):127–133

Li S, Mou X, Cai Y, Ji Z, Zhang J (2003) On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision. Comput Phys Commun 153(1):52–58

Li S, Chen G, Mou X (2005) On the dynamical degradation of digital piecewise linear chaotic maps. Int J Bifurc Chaos 15(10):3119–3151

Li C-Y, Chen Y-H, Chang T-Y, Deng L-Y, To K (2012) Period extension and randomness enhancement using high-throughput reseeding-mixing prng. IEEE Trans Very Large Scale Integr (VLSI) Syst 20(2):385–389

Lin T, Chua L (1993) A new class of pseudo-random number generator based on chaos in digital filters. Int J Circuit Theory Appl 21(5):473–480

Liu L, Miao S (2015) A universal method for improving the dynamical degradation of a digital chaotic system. Phys Scr 90(8):085205

Liu L, Hu H, Deng Y (2014) An analogue-digital mixed method for solving the dynamical degradation of digital chaotic systems. IMA J Math Control Inf 32(4):703–716

Liu Y, Luo Y, Song S, Cao L, Liu J, Harkin J (2017) Counteracting dynamical degradation of digital chaotic chebyshev map via perturbation. Int J Bifurc Chaos 27(03):1750033

Merah L, Ali-Pacha A, Hadj-Said N (2015) Real-time cryptosystem based on synchronized chaotic systems. Nonlinear Dyn 82(1–2):877–890

Oishi S, Inoue H (1982) Pseudo-random number generators and chaos. IEICE Trans (1976–1990) 65(9):534–541

Palmore J, Herring C (1990) Computer arithmetic, chaos and fractals. Phys D Nonlinear Phenom 42(1–3):99–110

Pincus SM (1991) Approximate entropy as a measure of system complexity. Proc Natl Acad Sci 88(6):2297–2301

Shu-Bo L, Jing S, Zheng-Quan X, Jin-Shuo L (2009) Digital chaotic sequence generator based on coupled chaotic systems. Chin Phys B 18(12):5219

Stojanovski T, Pihl J, Kocarev L (2001) Chaos-based random number generators. Part II: practical realization. IEEE Trans Circuits Syst I Fundam Theory Appl 48(3):382–385

Tao S, Ruli W, Yixun Y (1998) Perturbance-based algorithm to expand cycle length of chaotic key stream. Electron Lett 34(9):873–874

Tong X-J (2013) Design of an image encryption scheme based on a multiple chaotic map. Commun Nonlinear Sci Numer Simul 18(7):1725–1733