V.R. Gonzalez-Diaz, F. Pareschi, G. Setti, F. Maloberti: "**A Pseudorandom Number Generator Based on Time-Variant Recursion of Accumulators**"; IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 58, No. 9, September 2011, pp. 580-584.

A Pseudorandom Number Generator Based on Time-Variant Recursion of Accumulators

Victor R. Gonzalez-Diaz, *Member, IEEE*, Fabio Pareschi, *Member, IEEE*, Gianluca Setti, *Fellow, IEEE*, and Franco Maloberti, *Fellow, IEEE*

***Abstract*—This brief presents a pseudorandom number generator that requires very low resources from the hardware design point of view. It is based on a chain of digital accumulators whose coefficients are varied by an auxiliary low-complexity linear feedback shift register. We present a predictability and periodicity analysis of the sequences generated by the proposed architecture to show that the system is a good candidate to be used for applications requiring high-quality pseudorandom sequences in portable devices. The statistical behavior of the proposed solution is also validated by tests from the National Institute of Standards and Technology. The generated pseudorandom sequences pass all tests at both the *level-one* and *level-two* approaches.**

***Index Terms*—Pseudorandom number generator (PRNG), statistical tests, time variant.**

I. Introduction

An EFFECTIVE generation of pseudorandom sequences has positive consequences on many types of application. To name a few, the increasing need of security in cryptography applications makes it necessary to design complex systems that generate deterministic sequences with statistical features as close as possible to a random process [1] and a built-in self-test of digital circuits that uses groups of digital sequences with random characteristics.

Opposed to *true-random* number generators [2] that are based on some intrinsically random natural phenomenon, pseudorandom number generators (PRNGs) [3], [4] are numerical algorithms that, starting from an externally (and possibly randomly) chosen seed, can produce long irregular randomlike sequences, which are nevertheless periodic and fully repeatable. This repeatability property makes them fundamental in many applications such as cryptography.

Independently of the employed methodology, hardware generation of pseudorandom numbers is becoming increasingly difficult due to tight constraints in terms of power and area consumption, which modern devices require, particularly in the field of portable and consumer applications.

Manuscript received December 20, 2010; revised April 28, 2011; accepted June 6, 2011. Date of publication August 15, 2011; date of current version September 14, 2011. This work was supported in part by the Italian National Program FIRB under Grant RBAP06L4S5 and in part by CONACyT Mexico under Grant 131617. This paper was recommended by Associate Editor C.-C. Wang.

V. R. Gonzalez-Diaz is with the Faculty of Electronics Science, Autonomous University of Puebla, Puebla, Mexico.

F. Pareschi and G. Setti are with ENDIF, University of Ferrara, 44100 Ferrara, Italy, and also with ARCES, University of Bologna, 40125 Bologna, Italy.

F. Maloberti is with the MIS Laboratory, Pavia University, 27100 Pavia, Italy.

Digital Object Identifier 10.1109/TCSII.2011.2161165

In a PRNG, the periodicity, the predictability, and, more generally, all its statistical features are important characteristics [5]. For instance, in cryptographic applications, the predictability is important as an attacker may be capable to get information observing the PRNG output, so that the system security is threatened because the seed of the PRNG (which is related to the cryptographic key [3]) can be exposed. The use of cumbersome algorithms (such as the Blum-Blum-Shub [4]) ensures system security, but the cost is extremely high in terms of resources required. Simpler algorithms (such as the Mersenne-Twister [6]) can achieve a very high quality random stream, but they may not be cryptographically secure.

This brief proposes a low-resources architecture that is capable of generating pseudorandom sequences with very good statistical features. Along with the system architecture, which exploits a digital $\Sigma\Delta$ modulator with quantization error mapping function variable within time and is based on a self-recursive structure, we present a basic predictability and periodicity analysis. With this, despite the fact that a formal cryptographic security analysis is beyond the scope of this brief, we show that, notwithstanding the simplicity of the architecture, the proposed PRNG is not easily predictable, and it is a good candidate for embedding in portable applications.

This brief is organized as follows. Section II presents the architecture and the notation used to study the evolution of the system. Section III describes statistical analysis of the PRNG (including predictability and periodicity of the generated sequences) and estimates the lower bound of the computational power required to predict the evolution of the system. Section III-C validates this PRNG with the statistical tests of the National Institute of Standards and Technology (NIST) [7]. The results refer to the level-one testing approach and also to the level-two testing both with the proportion of sequences and the χ^2 approaches. Finally, in Section IV, we discuss some implementation aspects, and in Section V, we draw the conclusion.

II. Proposed PRNG

Digital accumulators are widely used in digital processors, digital $\Sigma\Delta$ modulators for fractional frequency synthesizers [8], [9], digital-to-analog converters [10], etc. Let us consider the block level model of the modulus M digital accumulator shown in Fig. 1, where X is the (constant) input, p_0 (the accumulator sum) is the state variable of the system, and the two outputs are e_0 (the sum modulus M, which is also called *quantization error*) and y_0 (the carry output, which is set when

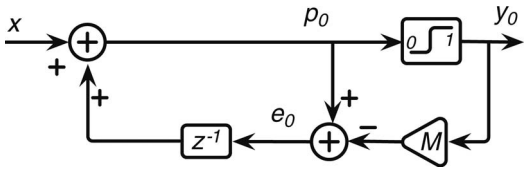

Fig. 1. Block-level model for the modulus M digital accumulator.

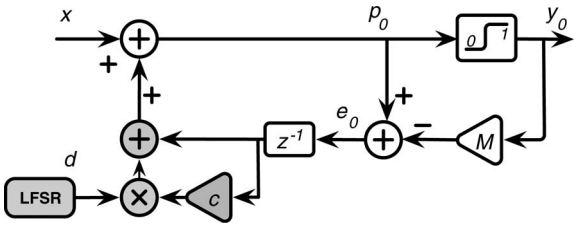

Fig. 2. Block-level model for the modulus M digital accumulator with time-variant coefficient.

the accumulator overloads, i.e., when the sum is bigger than or equal to M). Mathematically, indicating with i the time step, we have

$$\begin{aligned} p_0[i] &= X + e_0[i-1] \\ y_0[i] &= \begin{cases} 0 & p_0[i] < M \\ 1 & p_0[i] \geq M \end{cases} \\ e_0[i] &= p_0[i] - My_0[i] = p_0[i] \pmod{M}. \end{aligned} \tag{1}$$

Under the condition that X and M are relatively prime [11], [12], the quantization error e_0 becomes uniformly distributed on the range of its M possible values. However, in the general case, the system may regrettably generate even very short periodic sequences, which are composed only by a few among all possible values.

Many methods have been proposed to improve the statistical properties of the e_0 sequence for a generic input. We consider here the method first proposed in [9], which consists of varying the accumulator's feedback coefficients with time, as shown in Fig. 2. The quantization error e_0 is scaled by a coefficient $c \ll 1$, multiplied by a binary variable $d \in \{0, 1\}$ generated by a simple congruential PRNG based on a linear feedback shift register (LFSR) [13] and then fed back to the input. The scaling by the c coefficient can be achieved with a digital $\Sigma\Delta$ modulator, i.e., by another accumulator like that in Fig. 1, which takes e_0 as input, and whose output y_1 replaces the signal $c\,e_0$ in the feedback path. The implementation of this architecture on a m bit digital hardware (i.e., assuming $M = 2^m$) is shown in Fig. 3, and it is very simple since only two m-bit adders are required along with a simple congruential auxiliary PRNG. The multiplication between y_1 and d is simply obtained with a one-bit AND gate, and the result is used as the carry input of the first accumulator. The scaling coefficient is $c = 1/M = 2^{-m}$. In this architecture, the e_0 sequence has good statistical properties for all the possible values of X [9].

In this brief, we propose to use a chain of n time-varied accumulators to generate a pseudorandom stream. The proposed topology is a modular system identical to a multistage $\Sigma\Delta$ converter, where the input of the generic stage k is the quantization error e_{k-1} of the previous stage, whereas the carry

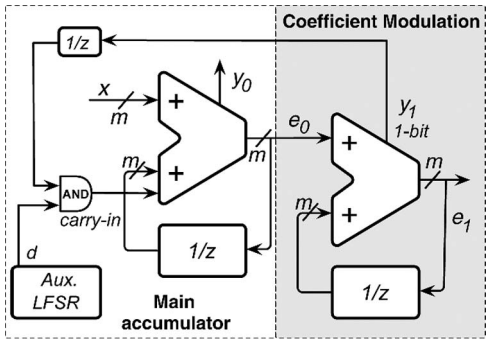

Fig. 3. Implementation of the time-variant coefficient accumulator of Fig. 2 with $M = 2^m$.

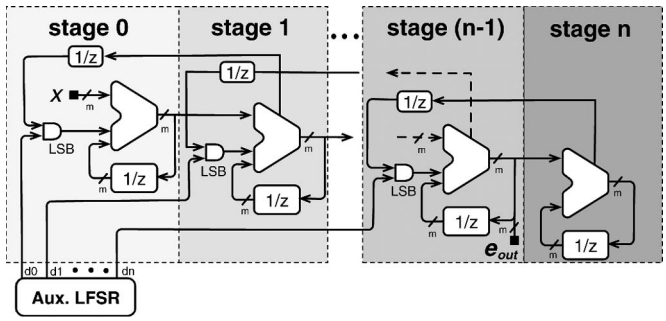

Fig. 4. Architecture of the proposed PRNG.

output y_k is used in the previous stage for the generation of the time variant coefficient, as previously described. Note that an additional stage is required to generate the signal y_n used in the stage $n-1$. For the proposed architecture, whose block diagram is shown in Fig. 4, the evolution is regulated by

$$\begin{aligned} p_k[i] &= \begin{cases} X + e_0[i-1] + y_1[i-1]d_0[i], & k = 0 \\ e_{k-1}[i] + e_k[i-1] + y_{k+1}[i-1]d_k[i], & 0 < k < n \\ e_{n-1}[i] + e_n[i-1], & k = n \end{cases} \\ y_k[i] &= \begin{cases} 0 & p_k[i] < M \\ 1 & p_k[i] \geq M \end{cases} \\ e_k[i] &= p_k[i] - My_k[i] = p_k[i] \pmod{M}. \end{aligned} \tag{2}$$

Note that the presence of an auxiliary PRNG is mandatory.

We will show in the following that the output sequence e_{n-1} has the characteristic of a very good pseudorandom sequence. The seed of this PRNG can be considered as the input signal X, along with the initial states of all the accumulators and the seed of the LFSR auxiliary PRNG. Note that, along with the high quality and the simple and recurrent architecture, another one of the advantages of the proposed architecture is to rely on a digital Sigma-Delta modulator [9], [10] architecture. This means that, considering an environment where resources are very limited, we could design an accumulator chain that can work as a digital $\Sigma\Delta$ converter and also as a high-security PRNG when required.

The system we propose is composed of nine stages (i.e., $n = 8$) with $m = 8$ bits resolution, where e_7 is the system output. The auxiliary PRNG is a maximum-length LFSR composed of eight stages, which is the minimum number of stages to simultaneously generate the $d_0 \ldots d_7$ values required. The choice of these parameters has been made since they are a good tradeoff between system performance and complexity.

III. Proposed PRNG Statistical Features

In this section, we analyze a few aspects related to the statistical properties of the proposed PRNG and show that, despite its simplicity, the statistical features are comparable with those achieved with much more complex architectures. We first propose a brief study on the PRNG predictability and periodicity, where, in both cases, we start from the case of the simple chain of accumulators, i.e., considering $d_k = 0$, $\forall k$. Then, we consider how the system is changed by the introduction of the time-varying coefficients as in (2). Finally, we provide statistical test results for sequences generated by the PRNG. The results were obtained using, at both level one and level two, the SP800-22 test suite provided by the NIST [7].

A. Predictability of the PRNG

Assuming $d_k = 0$, we get a system that is actually linear in modular arithmetic, and its evolution can be easily predicted by means of a limited number of observations. In fact, if we define with

$$i^{(k)} = i(i+1)(i+2)\dots(i+k-1) \tag{3}$$

the Pochhammer symbol for a rising sequential product and we indicate with $\bar{e}_k$ the initial condition of the kth accumulator, i.e., $e_k[0] = \bar{e}_k$, we can unroll the recursive evolution equation (2) and write the evolution of the generic kth stage as

$$e_k[i] = \frac{i^{(k+1)}}{(k+1)!}X + \sum_{l=0}^{k} \frac{i^{(l)}}{l!}\bar{e}_{k-l} \pmod M \tag{4}$$

i.e., we get that the output sequence e_k out of accumulator k is completely determined by the constant input X and the initial conditions of all the accumulators down to the first one, i.e., $\bar{e}_k, \bar{e}_{k-1}, \dots, \bar{e}_0$.

Let us assume a PRNG based on an n stages architecture and indicate with $\boldsymbol{X}_n$ its seed, i.e., the vector $\boldsymbol{X}_n = (X, \bar{e}_0, \dots, \bar{e}_{n-1})'$. Predicting the evolution of the PRNG means finding $\boldsymbol{X}_n$ by observing e_{n-1}, i.e., the output of the system.

By collecting $n+1$ observations $e_{n-1}[0], e_{n-1}[1], \dots, e_{n-1}[n]$, we can write the system

$$\begin{pmatrix} e_{n-1}[0] \\ e_{n-1}[1] \\ \dots \\ e_{n-1}[n] \end{pmatrix} = \boldsymbol{A}_n \boldsymbol{X}_n \pmod M \tag{5}$$

where $\boldsymbol{A}_n$ is an $(n+1)\times(n+1)$ matrix with integer coefficients, which depends only on n. Equation (5) is a system of $n+1$ *linear* equations in $n+1$ unknowns with integer coefficients in modular arithmetic. The solution of this system is generally not trivial [14], [15] and not strictly related with the solution of the associated system in $\mathbb{R}^{n+1}$ [i.e., system (5) considered in $\mathbb{R}^{n+1}$, instead of in modulo M arithmetic], which is $\overline{\boldsymbol{X}}_n = \boldsymbol{A}_n^{-1}(e_{n-1}[0], \dots, e_{n-1}[n])'$. However, the aforementioned system has the interesting property that $\boldsymbol{A}_n^{-1}$ is an integer matrix[1] i.e., $\overline{\boldsymbol{X}}_n$ is also integer. In this case, the solution $\boldsymbol{X}_n$ of the modular problem is simply the congruential vector of $\overline{\boldsymbol{X}}_n$. In conclusion, using the simple chain of accumulators as PRNG makes the system easily predicable.

When considering the system in Fig. 4, we are changing the true nature of the system, which is not linear anymore, since the terms y_k are a nonlinear functions of the p_k. Despite the fact that the aim of this section is not to define an algorithm for determining the seed of our PRNG from observations, we can compute, given the solution of (5), a lower bound for the computational power required for predicting the evolution the system of Fig. 4.

Let us unroll the recursive equation (2), i.e.,

$$e_k[i] = \frac{i^{(k+1)}}{(k+1)!}X + \sum_{l=0}^{k} \frac{i^{(l)}}{l!}\bar{e}_{k-l} + \sum_{l=0}^{k}\sum_{s=1}^{i} \frac{s^{(l)}}{l!} y_{k-l+1}[i-s]d_{k-l}[i-s+1] \pmod M \tag{6}$$

and let us collect, as in the previous case, the $n+1$ observations $e_{n-1}[0], e_{n-1}[1], \dots, e_{n-1}[n]$. This leads to the system

$$\begin{pmatrix} e_{n-1}[0] \\ e_{n-1}[1] \\ \dots \\ e_{n-1}[n] \end{pmatrix} = \boldsymbol{A}_n \boldsymbol{X}_n + \boldsymbol{B}_n \boldsymbol{Y}_n \pmod M \tag{7}$$

where $\boldsymbol{B}_n$ is an $(n+1)\times n^2$ coefficients matrix and where

$$\boldsymbol{Y}_n = \begin{pmatrix} y_1[0]d_0[1] \\ \vdots \\ y_n[0]d_{n-1}[1] \\ y_1[1]d_0[2] \\ \vdots \\ y_n[1]d_{n-1}[2] \\ \vdots \\ y_n[n-1]d_{n-1}[n] \end{pmatrix} \tag{8}$$

is an n^2 length vector, which is a nonlinear function of $\boldsymbol{X}_n$. Due to this nonlinearity, system (7) may not have a single solution.

One easy way to solve the impasse of finding the seed $\boldsymbol{X}_n$ is to assume to know *a priori* all the $y_{k+1}[i-1]d_k[i]$ products, i.e., to assume that $\boldsymbol{Y}_n$ is a constant vector. In this way, system (7) can be solved exactly as system (5). Note that, since we have 2^{n^2} different $\boldsymbol{Y}_n$'s, we also have up to 2^{n^2} different solutions $\boldsymbol{X}_n$. The actual seed of the system has to be chosen among the $\boldsymbol{X}_n$ that are *coherent* with the assumed $\boldsymbol{Y}_n$.

Note that refining this choice up to a single coherent $\boldsymbol{X}_n$ may require some additional observations. Note also that this approach gives only partial information on $\bar{e}_n$ and on the auxiliary PRNG internal state.

The lower bound complexity for this approach is the same as solving (and checking for coherence) 2^{n^2} linear systems. Therefore, neglecting the auxiliary PRNG, a brute-force attack

[1]This property was checked using Mathematica for n up to 150.

[16] on the system would try $2^{(n+1)m}$ possible $\boldsymbol{X}_n$ seeds. When the number of stages n is comparable with the number of bits m in each accumulator, the prediction of the system from the observed values has almost the same complexity as a brute-force attack.

B. Periodicity of the Proposed PRNG

For the sake of simplicity, let us start, as in the previous case, by making some considerations on the basic chain of accumulators. The periodicity of the system is defined as the smallest time step $I>0$ for which $e_k[i+I]=e_k[i]$ or, equivalently, $e_k[I]=\bar{e}_k$, $\forall k$. By using (4), we can compute I by solving the system

$$\begin{cases} IX = 0 \pmod M \\ \frac{I^{(2)}}{2!}X + I\bar{e}_0 = 0 \pmod M \\ \dots \\ \frac{I^{(n)}}{n!}X + \sum_{l=1}^{n-1}\frac{I^{(l)}}{l!}\bar{e}_{n-l-1} = 0 \pmod M \end{cases} \quad (9)$$

which is a nonlinear system in I.

Let us neglect the obvious case where $X=\bar{e}_k=0$, which generates a constant output. It is known in the literature [12] that, if M is a prime number, then the system periodicity is $I=M$. Otherwise, I depends on all system parameters, i.e., X, M, and n, and also on all $\bar{e}_k$. In the worst case, i.e., when we get the shortest period, I is the smallest common divisor between M and X, i.e., conditions exist for which $I=2$.

Without entering into the mathematical details, when the time-varying coefficients are introduced in the feedback path as in Fig. 4, the periodicity of the system has to be computed by solving the system $e_k[I]=\bar{e}_k$, $\forall k$, where $e_k[I]$ are computed through (6). Furthermore, we have also to ensure that $y_k[I]=y_k[0]$ and that the state of the auxiliary PRNG at time step I matches the initial one.

It is easy to see that computing the exact value of I is at least as difficult as to predict the evolution of the system. However, due to the condition on the auxiliary PRNG, we know that I has to be *an integer multiple of the periodicity of the auxiliary PRNG*, i.e., we have a lower bound for I, which is *independent* on the seed. Therefore, the periodic behavior can be improved with the auxiliary LFSR.

Note, however, that this periodicity lower bound is usually a strong underestimation of the actual period. As an example, with the proposed parameters ($n=8$, $m=8$, and an eight-stage LFSR), we have a lower bound equal to $2^8-1=255$, but we were not able to observe any periodic behavior in many simulations with several millions time steps.

C. Statistical Tests Results

In this section, we propose some statistical test results for the proposed architecture with $n=8$, $m=8$, and an 8-bit LSFR. We have tested many generated pseudorandom streams with the suite SP800-22 [7], which is a collection of tests developed by the NIST and is the suite most commonly used in the evaluation of RNG and PRNG for cryptographic applications. We used this suite since, in recent years, it has been recognized as the standard *de facto* for random generators testing; furthermore due to uniformity of the tests in the suite, it is possible to apply the so-called *level-two* (or *second level*) testing approach, which has been shown to be much more selective in exposing weak generators [17].

TABLE I
RESULTS OF THE RANDOMNESS NIST TESTS FOR THE PROPOSED PRNG. THE FIRST COLUMN IS FOR THE STANDARD APPROACH; THE SECOND ONE IS FOR A LEVEL-TWO APPROACH BASED ON THE PROPORTION OF SEQUENCES PASSING A STANDARD TEST; THE THIRD COLUMN IS FOR A LEVEL-TWO APPROACH BASED ON A CHI-SQUARE TEST. ALL RESULTS ARE IN THE EXPECTED RANGE

SP800-22 test	standard	99%	χ^2
Frequency	0.6020	0.9860	0.3285
Block Frequency	0.1834	0.9840	0.1750
Cumulative Sums	0.4549	0.9830	0.3309
Runs	0.8694	0.9920	0.7237
Longest Run of 1s	0.5901	0.9850	0.7090
Matrix Rank	0.5091	0.9920	0.2482
Spectral (DFT)	0.5157	0.9920	0.9374
NOT Matching	0.9042	0.9820	0.2382
OT Matching	0.5207	0.9830	0.6454
Universal	0.3416	0.9900	0.7749
Approx. Entropy	0.2258	0.9870	0.0499
Random Excursion	0.2099	0.9860	0.3419
Random Exc. Var.	0.1245	0.9890	0.4291
Serial	0.3916	0.9890	0.4286
Linear Complexity	0.1081	0.9910	0.2363

SP800-22 test results are shown in Table I, where we have adopted all three approaches proposed by NIST. The first column is the result of a standard test, where we have generated a single sequence and processed it with all the tests in the suite. The test result is a p-value, which is a number in [0, 1], which should be larger than a *level of significance* α for considering a test passed. The value suggest by NIST is $\alpha=0.01$. According to the table results, $p>\alpha$ for all tests, which can be considered passed.

The second and third columns are the result for level-two approaches. In the second column, we have generated 1000 different sequences from different initial conditions, and we have checked the proportion of sequences, where $p>\alpha$. This number should lie in the confidence interval 0.99 ± 0.0094. For all tests, the ratio of sequences passing the test is in the confidence interval. In the third column, we have taken the same 1000 sequences as above, and we have tested the uniformity of their p-values with a chi-square goodness-of-fit test. The results is a level-two p-value p_T, which has to be larger than a significance level α_T. For all tests, $p_T>0.01$, i.e., this level-two test is also passed.

Note that, when increasing the complexity of the PRNG by considering a higher number of stages, the statistical tests are passed with similar results. Similarly, increasing the complexity of the auxiliary LFSR (note that, in this example, we have considered an eight-stage LFSR, which is the minimum length for which we can change the required coefficients since $n=8$), we obtain similar results. This means that the auxiliary PRNG does not influence the statistical properties of the random-generated stream.

Since we have also shown that the auxiliary PRNG has little influence on the predictability of the system while having a very strong influence on its periodicity, we suggest choosing the auxiliary PRNG with the only aim of maximizing the periodicity of the generated sequences. For this reason, the best choice is a maximum-length LFSR.

TABLE II
COMPARISON BETWEEN THE NUMBER OF RESOURCES OCCUPIED BY THE PROPOSED GENERATOR AND SOME LOW-RESOURCE IMPLEMENTATIONS OF THE MERSENNE-TWISTER PRNG

Design	**This paper**	[18]	[19]	[20]
Architecture	**xc4vfx12 (Virtex-4)**	xc4vfx100 (Virtex-4)	xc2000e (Virtex-E)	N/A
Slices (LTUs)	**57 (91)**	128 (213)	330 (539)	429 (N/A)
RAM	**No**	4 Blocks	2 Blocks	Yes (unspecified)
Speed (MSa/s)	**90.98**	26.13	24.16	38.41

IV. HARDWARE IMPLEMENTATION

The proposed architecture, which is composed of nine accumulators with 8 bits of resolution and an auxiliary 8-bit LFSR, has been synthesized with automatic layout generation tools. In the synthesis obtained by Cadence Silicon Ensemble in a CMOS 0.35-μm process, a total of 474 digital cells are required with an area of $280\ \mu\text{m} \times 280\ \mu\text{m}$. This is a very small amount of digital cells for high-quality PRNG implementation. We have also implemented the proposed architecture on a Virtex-4 field-programmable gate array (FPGA) using Xilinx ISE Web Pack 13.1. The number of slices required is only 57, with no additional random-access memory (RAM) requirements. These resources have been compared in Table II with some implementations of the *Mersenne-Twister* generator on common FPGA platforms [18]–[20], which require 128–420 FPGA slices, depending on the area/timing optimization, and some additional RAM blocks. The proposed architecture has clear advantages in terms of resources required and speed.

Note that the lack of a RAM requirement is a twofold advantage since the RAM is usually the bottleneck for both the FPGA speed and resource allocation. The implementation of the Mersenne-Twister without the RAM blocks [19] will costs 5815 slices; this payload in terms of area would result in a significant reduction in the FPGA clock maximum speed.

V. CONCLUSION

We have proposed a new low-complexity PRNG based on a chain of digital accumulators with feedback coefficients. The feedback coefficients are changed within time with the help of a low-complexity LFSR. With this system, the congruent relationships that rule the evolution of a chain of accumulators have been transformed in a nonlinear mapping that increments the periodicity of the output sequence. Moreover, predicting the evolution of the system in general requires almost the same computational power of a brute-force attack. Sequences obtained from the proposed architecture have similar statistical properties of PRNGs used even for cryptographic applications; it passes all NIST statistical tests at the level two, both with the proportion of sequences and the χ^2 approaches.

REFERENCES

[1] R. M. Davis, "The data encryption standard in perspective," *IEEE Commun. Mag.*, vol. 16, no. 6, pp. 5–9, Nov. 1978.

[2] F. Pareschi, G. Setti, and R. Rovatti, "A fast chaos-based true random number generator for cryptographic applications," in *Proc. 26th Eur. Solid-State Circuit Conf.*, Montreux, Switzerland, Sep. 2006, pp. 130–133.

[3] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1996.

[4] L. Blum and M. Blum, "A comparison of two pseudo random number generators," in *Proc. Crypto*, 1982, pp. 61–78.

[5] D. B. Thomas, L. Howes, and W. Luk, "A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, 2009, pp. 63–72.

[6] M. Matsumoto, "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.

[7] National Institute of Standards and Technology (NIST), Special Publication 800-22 A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, May 2001.

[8] T. A. D. Riley, M. A. Copeland, and T. A. Kwasniewski, "Delta-sigma modulation in fractional-N frequency synthesis," *IEEE J. Solid-State Circuits*, vol. 28, no. 5, pp. 553–559, May 1993.

[9] F. Maloberti, E. Bonizzoni, and A. Surano, "Time variant digital sigma-delta modulator for fractional-N frequency synthesizers," in *Proc. IEEE Int. Symp. RFIT*, Singapore, Dec. 2009, pp. 111–114.

[10] S. R. Norsworthy, R. Schreier, and G. C. Temes, *Delta-Sigma Data Converters*. New York: IEEE Press, 1997.

[11] M. J. Borkowski and J. Kostamovaara, "On randomization of digital delta-sigma modulators with DC inputs," in *Proc. IEEE ISCAS*, May 2006, pp. 3770–3773.

[12] K. Hosseini and M. P. Kennedy, "Architectures for maximum sequence length digital delta-sigma modulators," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 11, pp. 1104–1108, Nov. 2008.

[13] D. E. Knuth, *The Art of Computer Programming*. Reading, MA: Addison-Wesley Professional, 1997.

[14] K. Levasseur, "The solution of linear equations with integer coefficients using modular arithmetic and Cramer's rule," *Math. Educ. Res.*, vol. 4, no. 1, pp. 14–27, Nov. 1995.

[15] J. Lipson, *Elements of Algebra and Algebraic Computing*. Reading, MA: Addison-Wesley, 1981.

[16] S. Kwok and E. Lam, "Effective uses of FPGAs for Brute-Force Attack on RC4 Ciphers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 8, pp. 1096–1100, Aug. 2008.

[17] F. Pareschi, R. Rovatti, and G. Setti, "Second-level NIST randomness tests for improving test reliability," in *Proc. IEEE ISCAS*, May 2007, pp. 1437–1440.

[18] T. Xiang and K. Benkrid, "Mersenne Twister random number generation on FPGA, CPU and GPU," in *Proc. NASA/ESA Conf. AHS*, Aug. 2009, pp. 460–464.

[19] S. Chandrasekaran and A. Amira, "High performance FPGA implementation of the Mersenne Twister," in *Proc. IEEE Int. Symp. Electron. Des., Test Appl. (DELTA)*, Jan. 2008, pp. 482–485.

[20] V. Sriram and D. Kearney, "An area time efficient field programmable Mersenne Twister uniform random number generator," in *Proc. Int. Conf. Eng. Reconfigurable Syst.*, Aug. 2006, pp. 244–246.