Embeddable ADC-Based True Random Number Generator for Cryptographic Applications Exploiting Nonlinear Signal Processing and Chaos

Sergio Callegari, *Member, IEEE*, Riccardo Rovatti, *Senior Member, IEEE*, and Gianluca Setti, *Senior Member, IEEE*

***Abstract*—We present a true random number generator which, contrary to other implementations, is not based on the explicit observation of complex micro-cosmic processes but on standard signal processing primitives, freeing the designer from the need for dedicated hardware. The system can be implemented from now ubiquitous analog-to-digital converters building blocks, and is therefore well-suited to embedding. On current technologies, the design permits data rates in the order of a few tens of megabits per second. Furthermore, the absence of predictable, repeatable behaviors increases the system security for cryptographic applications. The design relies on a simple inner model based on chaotic dynamics which, in ideal conditions, can be formally proven to generate perfectly uncorrelated binary sequences. Here, we detail the design and we validate the quality of its output against a couple of test suites standardized by the U.S. National Institute of Standards and Technology, both in the ideal case and assuming implementation errors.**

***Index Terms*—Analog-digital conversion, chaos, cryptography, pipeline converters, statistical approach to dynamical system theory, statistical signal processing.**

I. Introduction

RANDOM number generators (RNGs) represent a critical point in the implementation of many security schemes and can be regarded as fundamental cryptographic primitives. For instance, random numbers are inherent in the computation of algorithms, such as the DSA, in the synthesis of confidential keys for symmetric- and public-key crypto-systems as RSA moduli, and in many communication protocols [1]. The ability of cryptographic techniques to foil pattern analysis is strongly dependent on the unpredictability of the random generators they employ, so that generators suitable for security related applications must meet much stronger requirements than those for other applications [2].

It is generally recognized that ideal (or so-called true) random number generators (TRNGs) can only be approximated. An ideal source must in fact be capable of producing infinitely long sequences made of perfectly *independent* bits, with the property that, when restarted, it does never reproduce a previously delivered sequence (nonrepeatability). Practically implemented RNGs can be classified into two major categories [3], namely *pseudo*-RNGs and *physical*-RNGs. Pseudo-RNGs are deterministic, numeric algorithms that "expand" short seeds into long bit sequences. Conversely, physical-RNGs rely on micro-cosmic processes resulting in macroscopic observables which can be regarded as random noise (examples are quantum noise, emission timing in radioactive decay, thermal noise, shot noise, jitter in free running oscillators, timing of human activities such as keystrokes, etc). Pseudo-RNGs are those more distant from the ideal specifications. Being necessarily based on finite memory algorithms, they exhibit periodic behaviors and generate correlated samples. The same reason also makes them completely repeatable. While the periodicity issue can easily be neglected (as the period can be expanded with regard to the time scales on which a piece of equipment is employed), the possibility of retrieving information about the seed through the observation of output subsequences and their repeatable (and thus predictable) behavior can hardly be desirable in applications such as data security and cryptography [2], [3]. Their substantial advantage is the algorithmic nature which makes them easily embeddable in any digital circuit or system. Physical-RNGs, on the other hand, are the best approximation of TRNGs and, in common perception, are often completely identified with them. Unfortunately, they may require very specialized hardware and/or environmental conditions, which make them expensive to embed. In spite of this liability, security related applications have recently been strongly pushing their development and deployment, so that bigger players in Information Technology (e.g., Intel and Via) are now introducing physical-RNGs in their security platforms [4], [5].

Obviously, it would be desirable to devise RNGs matching the features of physical sources and, at the same time, the design-friendliness of digital sources. For instance, recent proposals have reduced the design cost of TRNGs by exploiting phase-locked-loops (PLLs) and field programmable gate arrays (FPGAs) [6]. Notwithstanding some successful realizations, an aspect of many physical generators that should not be underestimated is their very low data rate, particularly when related to the employed resources. For example, in a recent realization a 90-MHz FPGA with two parallel operating internal PLLs could deliver only $\sim$ 70 random~kbit/s [6]. Even the physical-RNG recently introduced by Intel is limited to 75 kbit/s [4]. This is clearly due to the need of coping with "naturally available" noisy quantities.

Here we introduce a novel methodology for the implementation of TRNGs which, exploiting statistical signal processing

Manuscript received October 30, 2003; revised September 3, 2004. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Stefan Katzenbeisser.

S. Callegari and R. Rovati are with ARCES and DEIS, University of Bologna, Bologna, Italy (e-mail: scallegari@arces.unibo.it; rrovatti@arces.unibo.it).

G. Setti is with ENDIF University of Ferrara, Ferrara, Italy, and also with ARCES, University of Bologna, Bologna, Italy (e-mail: gsetti@ing.unife.it).

Digital Object Identifier 10.1109/TSP.2004.839924

techniques, can close the gap with the throughput and easiness of implementation of pseudo-RNGs. Our proposal stems from the consideration that the micro-cosmic processes employed in physical generators are nowadays often regarded as *chaotic*. In other words, it is conjectured that the underlying models could (at least) in principle be expressed deterministically and that it is their extreme *complexity* and *sensitivity to initial conditions* to make them unpredictable and, to the coarse observer, practically random [7]. Under these premises, rather than exploiting natural phenomena like noise that are hardly controllable and mathematically unmanageable, it would be much better to rely on *artificial* and simpler ones that derive unpredictability from complexity. It is interesting to note that discrete-time chaotic models exist which are now extensively understood [8], so that their suitability for RNGs can be *analytically* and not just *heuristically* proven [9], [10]. Furthermore, we are now able to show that they can be practically realized out of analog-to-digital converters (ADC) parts which are now ubiquitous in mixed-mode systems.

We regard this last feature as remarkable. In fact, it enables our TRNG to gain for free some noteworthy features of the ADC they derive from. Remind that analog-to-digital conversion has historically seen many years of refinement, large volumes and proportionally large investments. A particularly important point is the possibility of embedding TRNGs as parts of larger, mostly digital systems. Consider that digital building blocks can usually be re-used or embedded in other circuits at very low costs. This is because their design can be expressed into hardware description languages and there are CAD/EDA tools making it relatively easy to re-target designs for other technologies. However, the same does not generally apply to analog blocks (and surely not to conventional physical-RNGs). Nonetheless, for ADCs the process of coding designs in re-usable forms is by far more advanced than for any other complex analog subsystem. Hence, our methodology lends itself particularly well to designing reusable and embeddable TRNG cores (similarly to existing ADC cores) and to high levels of design automation. Furthermore, given the controllable, artificial nature of the involved models and the standard speed of the current generation of ADCs, we regard our proposal as capable of a few tens of megabits per second, i.e., to be more than two order of magnitude faster than conventional TRNGs. The methodology is here thoroughly discussed and validated against two standard randomness test suites, one simpler and designed mostly for on-line testing, and the other more stringent and complete [11], [12].

Let us point out that such an increase in the performance and throughput of TRNGs pave the way for their wider exploitation in secure self organized mobile *ad-hoc* networks. These can be distinguished from traditional *ad-hoc* networks, such as conventional wireless LAN, by the absence of any central authority, the very high number of nodes, the large variability of the node density and the high level of cooperation. In an ideal setting, users may enter and exit the network at any time, from anywhere and by means of a terminal that does not rely on the existence of any strong pre-defined agreement or infrastructure. The quest for security in such networks, which is particularly complicated by the absence of a central trusted authority, is surely a hot topic in the communication field [13]. A promising solution proposed in [14] relies on an approach similar to PGP in the sense that users issue certificates for each other based on their personal acquaintances [14]. Clearly, PGP-like algorithms depend on the availability of fast, reliable, and unpredictable RNGs for generating session keys. The performance of the RNGs either in terms of security, throughput or power consumption is thus a key issue for their practical implementation in each mobile terminal.

Introducing complex dynamics, it must be remarked that chaos is now proposed for security related applications at various degrees, up to its usage inside the very cryptographic algorithms. Let us emphasize again that contrarily to these approaches, we take the conservative path of exploiting chaos only for TRNGs. In fact, in this particular task, the mathematical tools necessary for approaching complex dynamics are already mature enough to *formally prove* the effectiveness of the approach. As a result, this work is organized as follows.

- First, we introduce Markov chaotic sources (piece wise affine Markov 1-D maps), i.e., the specific kind of chaotic system that is going to be exploited. In particular, we try to merge an intuitive vision of the tools required to analyze these systems, with more formal concepts.
- Second, we review pipeline ADCs, i.e., the flavor of analog-to-digital converters which appears best suited for the reuse of building blocks in chaos based true random generators.
- Then, we join the two above items illustrating the architecture and design that we propose for true random bit generation.
- Eventually, we evaluate performance. In doing so, we validate a large number of sequences generated following our scheme against two different randomness test suites, both standardized by the U.S. National Institute for Standards and Technology (NIST) [11], [12]. Being that our design comprises ADC parts, in these experiments we take into account the unavoidable nonidealities that may accompany their practical implementation. We statistically describe the potential deviations from nominal behavior and the effects of noise. Then, by extensive Montecarlo simulation, we estimate the percentage of random sources that can be regarded as compliant with the test suite when operating at their maximum rate, i.e., a realistic bound for the production yield.

II. Markov Chaotic Sources

It has now been known for a long time that dynamical systems including nonlinear elements can exhibit nonclassical behaviors including very irregular, aperiodic, noise-like trajectories, and an extreme sensitivity to intial conditions, which can make two identical systems apparently starting at identical conditions end up with totally different outputs [7], [8]. These behaviors are generally called *chaos* and, intuitively, are very appealing for random number generation [15]–[17]. However, notice that, although noise-like, the outputs of a chaotic source are generally unevenly distributed and always correlated (due to the deterministic model responsible of their generation). Hence, it is

not possible to use them directly. Conversely, some processing is required for digitalization, de-correlation and balancing.

Truly, many algorithms exist which can *blindly* perform such tasks, i.e., there are compression methods capable of extracting an almost *independent, identically distributed* (IID) sequence out of a biased or correlated one, without any knowledge of the origin of the latter or of its underlying generation process [18]–[20]. Algorithms of this sort are often used, for instance, to extract random bits from the observation of noise like physical quantities that retain some correlation [4]. Such algorithms are *compressive* in the sense that their output data rate is constrained to be lower than the input one. The ratio among the two should ideally tend to the ratio between the input *entropy* (i.e., amount of information per time unit, measured in bits) and the input bitrate [18]. If algorithms of this sort are employed, chaos-based true random number generation is no different from physical random number generation and suffers the same liabilities, such as the need for complex signal processing, possibly low output rates and, often, no formal guarantee that completely uncorrelated bits are produced in the short term[1].

Fortunately, an explicit exploitation of chaotic dynamics allows to do much better than that. If one knows the model of the chaotic system, then he might look for an *informed* post-processing algorithm to get a true random sequence out of its trajectories. In other words, there is hope that the knowledge of the model can help streamline the postprocessing algorithm, make its implementation lighter and enhance its features. Even better, dealing with artificial chaotic systems, one can think of co-designing the chaotic model and the post-processing algorithm, to get the best possible features under some metric (e.g., the output rate or the easiness of implementation). Clearly, all this requires specific mathematical tools for dealing with the properties of chaotic systems. While such a theory does not yet generally exist, if one sufficiently restricts the range of chaotic systems being considered, many recently introduced tools become available. The underlying idea stems from the realization that chaotic systems enjoy a mixed deterministic/stochastic nature and from the consequent adoption of statistical methodologies for their analysis [10].

Following this introduction, we limit ourselves to a very specific class of nonlinear models, represented by the iteration of so called piece-wise affine Markov (PWAM) maps [8], [10]. These are discrete-time one dimensional (1-D) systems in which the state variable x is updated as

$$x_{n+1} = M(x_n) \tag{1}$$

where $M : [-1, 1] \to [-1, 1]$ is such that an interval partition $\{X_i\}, i = 0, \ldots, p-1$ of $[-1, 1]$ exists so that

1) M is affine on each X_i;
2) either $X_i \cap M(X_j) = \emptyset$ or $X_i \subseteq M(X_j)$ for any i, j (which is equivalent to saying that partition points are mapped into partition points).

If M is properly chosen, the system exhibits chaos[2]. By setting an initial condition x_0 and by iterating the map M, one can then generate a trajectory irregularly wandering in $[-1, 1]$. A typical case is shown in Fig. 1. Plot A shows a chaotic map, where

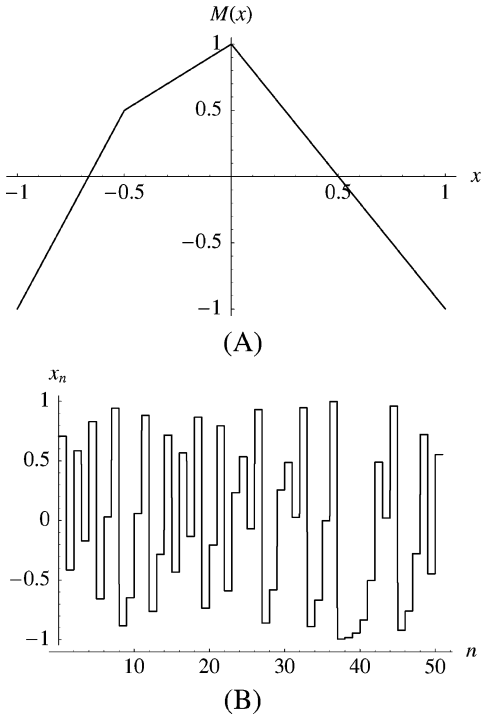


Fig. 1. (A) Example chaotic map. (B) Typical output trajectory.

$$M(x) = \begin{cases} 3x + 2, & \text{for } x < \frac{-1}{2} \\ x + 1, & \text{for } \frac{-1}{2} \leq x < 0 \\ -2x + 1, & \text{elsewhere} \end{cases} \tag{2}$$

and plot B a typical trajectory. Note that the map satisfies properties 1 and 2 for the interval partition composed by the four intervals $X_0 = [-1, -1/2)$, $X_1 = [-1/2, 0)$, $X_2 = [0, 1/2)$ and $X_3 = [1/2, 1]$. We anticipate that if M is chaotic, changing x_0 even by an extremely small quantity leads to a completely different trajectory, which progressively (and exponentially) increases its distance form the former as n increases. Hence, even in the absence of noise, the behavior of the system is intrinsically unpredictable since the limited precision with which the state is known at any time prevents long-term forecasting.

A thorough and engineering-oriented illustration of PWAM chaotic maps can be found in [10]. Here, it is sufficient to observe that the classical tools of system theory—which are generally based on the observation of system trajectories over time—are insufficient to get *typical* behaviors and general system properties, precisely due to the large variety of trajectories that can be generated. To overcome this impasse, the introduction of statistical tools is required. The intuitive idea is to pretend to be observing a huge number of trajectories at the same time, in order to be able to extrapolate common features. Suppose that a *very large* number of initial conditions is generated, following a given probability density function (pdf) ρ_0 in $[-1, 1]$. When M is applied, an equally large number of x_1 are obtained. These will distribute in $[-1, 1]$ according to a certain density ρ_1. If M is known, it can be expected that ρ_1 can be computed from ρ_0. Similarly, one can compute ρ_2 from ρ_1 and so on. In other words, one can introduce a *functional operator*

[1]The properties of many algorithms can only be proven *asymptotically*.

[2]More precisely, one wants M to be *measure-preserving* and *exact* following the definitions in [8].

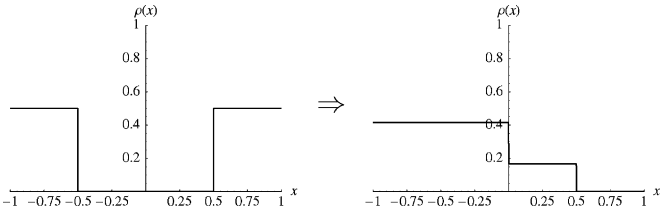


Fig. 2. Example transformation of stepwise probability density functions by the map in Fig. 1.

$\mathbf{P}$ companion to M, called Perron–Frobenius Operator [8], [10]. While M transforms points into points, $\mathbf{P}$ transforms probability densities into probability densities. Interestingly, in spite of M being nonlinear, $\mathbf{P}$ is a linear operator, which makes it far more manageable from a mathematical point of view. It can also be proven that if M is properly chosen, $\mathbf{P}$ is *constrictive*, so that a unique asymptotic density ρ_∞ exists, regardless of ρ_0.[3] The abovementioned properties 1 and 2 assure that in this case ρ_∞ is stepwise on the interval partition $\{X_i\}$ [10].

At this point, it is interesting to observe what happens if, instead of considering any possible initial density ρ_0, one limits himself to initial probability densities which are stepwise in $\{X_i\}$: all subsequent probability densities $\rho_1, \ldots, \rho_\infty$ are then compelled to be stepwise on the interval partition. This is noteworthy. In fact, in this case, rather than considering probability densities, one can consider the finite probability of the state variable x being at any of the intervals X_i. In other words, at each time step n, a vector of p probabilities $P_{n,0}, \ldots, P_{n,p-1}$ can be substituted for ρ_n. Similarly a finite dimension operator $\mathcal{K}$ can be substituted for the functional operator $\mathbf{P}$. Being $\mathcal{K}$ linear and finite-dimensional, it can be represented by a matrix, named the *kneading matrix*. It has been proven that the entries of $\mathcal{K}$ can be obtained from M as (see, e.g., [10] or [21] and references therein)

$$\mathcal{K}_{i,j} = \frac{\mu(X_i \cap M^{-1}(X_j))}{\mu(X_i)} \tag{3}$$

where μ is the common interval (Borel) measure, i.e., if $X_i = [a, b]$, then $\mu(X_i) = b - a$. For instance, the kneading matrix corresponding to the map in Fig. 1(a) is

$$\mathcal{K} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}. \tag{4}$$

As an example, suppose that $P_0 = (1/2, 0, 0, 1/2)$. Then $P_1 = P_0\mathcal{K} = (5/12, 5/12, 1/6, 0)$, as shown in Fig. 2, where we represent the corresponding pdfs.

It is now interesting to try to give a meaning to the individual entries in $\mathcal{K}$. With reference to the usual example map, suppose that it is known only that the initial condition x_0 falls in X_0, and nothing else. This is loosely equivalent to saying that $P_0 = (1, 0, 0, 0)$. What can be said about x_1? This point can obviously fall either in X_0, or in X_1, or in X_2, but not in X_3. Furthermore, from $P_0\mathcal{K}$ one can estimate that x_1 will fall in X_0 with probability 1/3, in X_1 with probability 1/3 and in X_2 with probability 1/3. In other words, the entry $\mathcal{K}_{i,j}$ of $\mathcal{K}$ represents the conditioned probability by which a trajectory starting in X_i falls in X_j at the next step.

[3]Provided that ρ_0 is sufficiently well behaved [10]

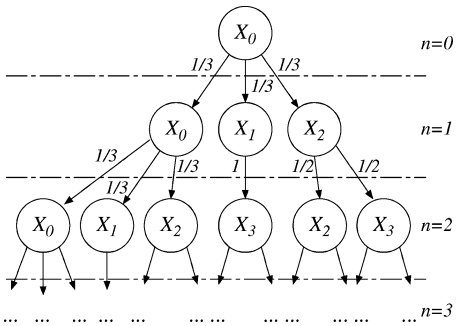


Fig. 3. Example probability tree based on the map in Fig. 1(a).

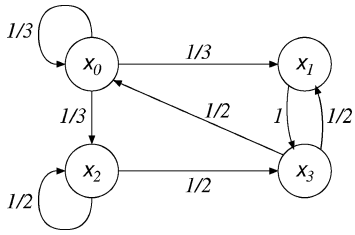


Fig. 4. State chain relative to the map in Fig. 1(a).

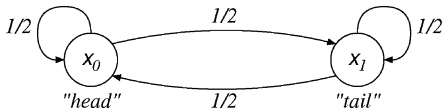


Fig. 5. State chain corresponding to the toss of an unbiased coin.

The process can be iterated. For instance, knowing that x_0 falls in X_0, one could build the tree-graph reported in Fig. 3, by which the probability of finding x_n in any of the partition intervals can easily be obtained. As an example, note that the probability of finding x_2 in X_2 turns out to be $1/3 \cdot 1/3 + 1/3 \cdot 1/2 = 5/18$. It is obviously convenient to compact infinitely long tree-graphs such as that in Fig. 3 into finite graphs containing loops, such as that in Fig. 4. Note that this graph can be interpreted as a *Markov chain* [9], [10], or a transition graph for a *probabilistic state machine*. The machine is in its discrete state x_i when the chaotic system has its continuous state variable x in the partition interval X_i. The weights assigned to the graph arrows represent the probabilities by which the machine travels from a state to another. The idea that a chaotic dynamics could embed a Markov chain was first conjectured by Kalman [22].

At this point, the concept of how one can design a chaotic source *optimized* for the generation of true random bits should start to emerge. One starts from a Markov chain that is *simple* and known to produce IID bits and then goes back through the steps that have just been introduced, obtaining first a kneading matrix from the state chain and then a PWAM chaotic map from the kneading matrix. As an example, consider the Markov chain illustrated in Fig. 5. This corresponds to the toss of an unbiased coin and thus generates random bits: the system is in state x_0 when the last toss outcome has been "head" and in state x_1 when the last outcome has been "tail". A new coin toss always hs the same probability of bringing the system in either state.

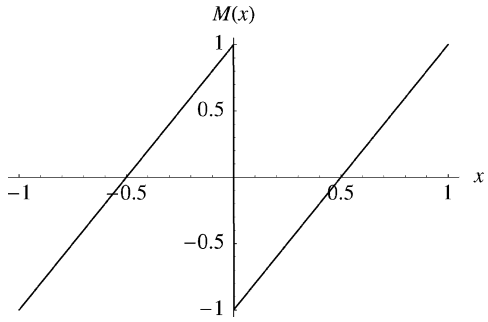


Fig. 6. Bernoulli shift.

The corresponding kneading matrix is 2×2

$$\mathcal{K} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}. \tag{5}$$

A few alternatives exist for designing PWAM maps corresponding to this $\mathcal{K}$. One of them is the well-known Bernoulli shift, where $M(x) = 2x \bmod 2 - 1$, which is pictured in Fig. 6. In this map, the interval partition has only two elements: $X_0 = [-1, 0)$ and $X_1 = [0, 1]$.

Following this approach, the design of a chaos based true-random source in principle requires only implementing the shift. Then, the sole post-processing that needs to be applied to the system trajectories is their binary quantization: a binary output "0" is generated when $x < 0$, and a "1" when $x \geq 0$. We remark that, by construction, this source is such that the observation of past sequences from the output of the quantizer cannot give *any* information about future values.

Since the probability of being in either of the two states is exactly one-half we get that each observation of the above Markov chain gives 1 bit of information [23], i.e., that this is a system able to generate one-bit of entropy per cycle.

To the best of the authors' knowledge, this elegant way of producing true random values from chaotic sources was first introduced by Kalmans conjecture [22]. Unfortunately, the practical application of the methodology can be hindered by implementation problems. In fact, all the PWAM maps that come out of the kneading matrix in (5) are not well suited for electronic realization. Briefly speaking, the main problem is that such maps are unable to maintain the state confined into the interval $[-1, 1]$ in presence of noise or implementation errors. This and other issues are thoroughly discussed in [24]–[26]. Luckily, in the following sections, such issues shall be removed as a side effect of the modifications introduced to the method for the reuse of standard ADC parts.

III. Pipeline ADCs

The duty of an ADC is to map an input variable v_0 defined on an interval I of $\mathbb{R}$ into its best numerical representation B with a finite number l of binary digits $b_0, \ldots, b_{l-1}$. In the following, I shall be normalized to $[-1, 1]$ so that b_0 is a sign bit, and $b_1, \ldots, b_{l-1}$ represent a mantissa.

In a few types of ADCs, notably successive approximation converters and pipeline converters, the mapping is completed in more than one step, exploiting a binary (or other radix) search of the digital value which is closer to the input analog quantity. Particularly, in pipeline converters this is done by performing a series of k coarse conversions. For a thorough description of the operating principles of pipeline ADCs, see [27]. Here, for simplicity, we assume that all the conversion stages are identical: this is common, though not strictly necessary. Let us refer to the structure presented in Fig. 7. Each conversion stage i computes a *coarse* m-bit representation $D_i = (d_{i,0}, \ldots, d_{i,m-1})$, with $m \ll l$ and $m \geq l/k$, of its input quantity and then calculates an analog *conversion error* e_i to be passed to the next stage as its input v_i. Note that the last stage, having nothing downhill, does not need to compute e_{k-1}. Observe that, apart from the first stage, which directly provides a coarse digital version of v_0, all other stages provide coarse representations of the intermediate conversion errors and not of the whole converter input. Hence, the digital outputs of the various stages must be processed by some *digital correction logic* to retrieve the l bits $b_0, \ldots, b_{l-1}$ from the $k \cdot m$ bits $d_{0,0}, \ldots, d_{k-1,m-1}$. In addition, observe that the conversion errors are necessarily *signed quantities* so that all stages in the chain must be capable of converting signed values. Finally, note that the conversion error e_i of the stage i is bounded to be smaller than its input v_i, so that before passing it to the next stage as v_{i+1} it is sensible to rescale it in order to make every v_i span $I = [-1, 1]$ as v_0. This is a necessary condition for having identical stages.

The major advantage of pipeline ADCs is that the various stages can be separated by *sample and hold* (S/H) blocks (the analog equivalent of latches). This permits us to synchronize the flow of information among the stages. A stage is thus free to start operating on the next piece of data as soon as it has calculated its analog conversion error. Just as in a digital pipeline, this permits us to tie the throughput to the inverse of the latency of a *single stage*, with a greater speed. However, this also complicates the digital correction logic, which is now compelled to be *sequential* (i.e., to have memory). In fact, at each instant n, only the stage 0 of the pipeline provides a coarse digital output D_0 relative to v_0 at time n, while the general stage i provides D_i relative to v_0 at time $n - i$. The digital correction logic needs to remember the outputs of the generic stage i up to $k-i$ steps back in the past, in order to have at each instant a full word $d_{i,0}, \ldots, d_{i,m-1}$ relative to v_0 at k steps in the past. In the following, and only when relevant, we remark the pipeline timing by slightly complicating the notation introduced so far and by tagging each quantity with a superscript $^{(n)}$ to indicate that it is relative to time-step n. For instance, $e_i^{(n)}$ is the conversion error of stage i at time n, relative to the conversion of the global ADC input $v_0^{(n-i)}$.

In the simplest incarnation of a pipeline ADC, each stage is a *single bit converter*. Its quantization function is thus

$$q(x) = \begin{cases} \frac{-1}{2}, & \text{if } x < 0 \\ \frac{+1}{2}, & \text{if } x \geq 0 \end{cases} \tag{6}$$

for all stages so that $m = 1$ and $D_i = d_{i,0}$ is the binary representation of $q(v_i)$, equal to 0 if $x < 0$ and to 1 otherwise. Obviously $e_i = v_i - q(v_i)$. With this, if $v_i \in [-1, 1]$, $e_i \in [-1/2, 1/2]$. Hence, to bring v_{i+1} into the whole $[-1, 1]$ interval, all the rescalers need to be gain blocks with gain 2. The operations that the digital correction logic needs to perform to

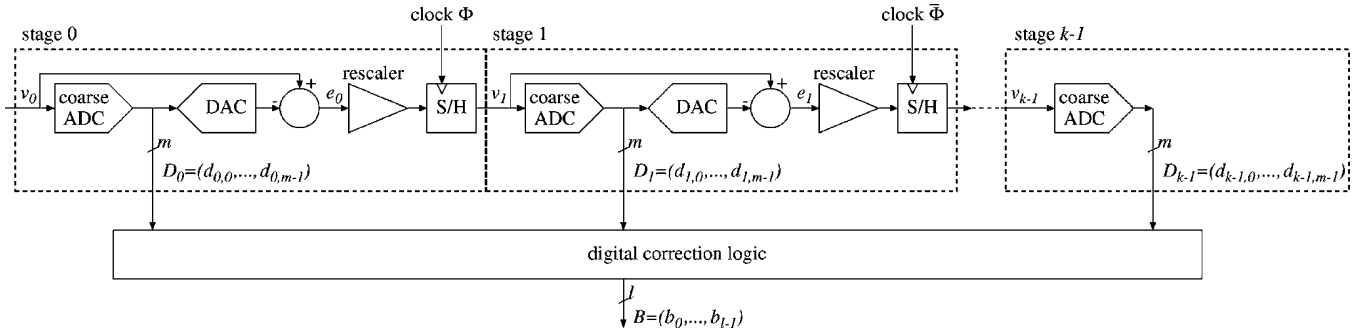


Fig. 7. Basic structure of a pipeline ADC.

compute the number $B^{(n)}$ corresponding to $v_0^{(n-k)}$ consist in calculating

$$B^{(n)} = \begin{cases} +\sum_{i=1}^{k} d_{i,0}^{(n-k+i)} \cdot 2^{-i}, & \text{if } d_{0,0}^{(n-k)} = 1 \\ -\sum_{i=1}^{k} d_{i,0}^{(n-k+i)} \cdot 2^{-i}, & \text{if } d_{0,0}^{(n-k)} = 0. \end{cases} \tag{7}$$

If $B^{(n)}$ is represented in a sign+mantissa binary notation, this merely reduces to putting side by side the bits $d_{i,0}^{(n-k+i)}$ to obtain the vector (b_i) for $i = 0, \ldots, l-1$.

It is now interesting to consider the tie between v_i and v_{i+1}. Intriguingly, in the above scheme, $v_{i+1} = M(v_i)$ where M is the Bernoulli map illustrated in Fig. 6. The possibility of re-using pipeline ADC parts for the realization of random sources is thus getting apparent.

Unfortunately, as already mentioned, it is not advisable to directly apply the Bernoulli shift to the implementation of discrete time chaotic sources. In this regard, an interesting point is that the map in Fig. 6 is not either the best candidate for practical pipeline ADCs. For precision and robustness reasons, ADC designers normally deviate from the scheme of principle illustrated so far and tend to rely on stages which implement some sort of *redundancy*. A typical case is given by the so-called *one bit and a half* stages [27]. In this arrangement, the quantization function employed at each stage is

$$q(x) = \begin{cases} -1, & \text{for } x < \frac{-1}{2} \\ 0, & \text{for } \frac{-1}{2} \le x < \frac{1}{2} \\ 1, & \text{for } x \ge \frac{1}{2}. \end{cases} \tag{8}$$

As before, D_i is a binary representation of $q(v_i)$, with the sole difference that now two bits are required. Being that $q(v_i)$ is generally obtained by confronting v_i with the values $\pm 1/2$ by means of two comparators, it is common to take a thermometric coding for D_i, so that each $d_{i,j}$ is the output of a comparator and

$$D_i = (d_{i,0}, d_{i,1}) = \begin{cases} 00, & \text{for } x < \frac{-1}{2} \\ 01, & \text{for } \frac{-1}{2} \le x < \frac{1}{2} \\ 11, & \text{for } x \ge \frac{1}{2}. \end{cases} \tag{9}$$

Again, $e_i = v_i - q(v_i)$, so that if $v_i \in [-1, 1]$, e_0 lies in $[-1/2, 1/2]$. Hence, just as before, it is necessary to set the rescalers to gain 2 to have all the v_i taking values in the same range as v_0.

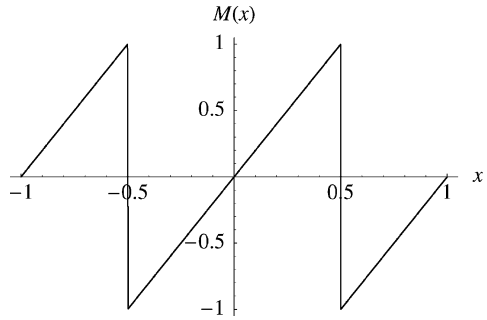


Fig. 8. Map derived from the building blocks of one and a half bit per stage pipeline ADCs.

The particular coding taken for the coarse conversion outputs $D_0, \ldots, D_{k-1}$ clearly complicates the digital correction logic required to compute B. However, this is largely compensated by many other advantages, as described again in [27]. In any case, the digital correction logic is not relevant to this work. We instead focus on the analog stages and on their usage for random number generation. As before, it is essential to concentrate on the tie between v_i and v_{i+1} to this aim. This is represented by a new map M given in Fig. 8.

IV. Application of Pipeline ADC Blocks to True Random Bit Generation

The map in Fig. 8 fulfills the requisites for being PWAM. Furthermore, a dynamical system based on its iteration exhibits chaotic behavior, hence the tools introduced in Section II can be applied. Eventually, note that, as shown in [25], [28], and [29], this map does not share the state confinement problems of the Bernoulli shift and is suitable for the *practical* implementation of chaotic circuits. The interval partition on which M is PWAM is $\{X_i\} = \{[-1, -1/2), [-1/2, 0), [0, 1/2), [1/2, 1]\}$, so that the corresponding kneading matrix is

$$\mathcal{K} = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}. \tag{10}$$

With this, the probabilistic finite state machine associated to the chaotic source is described by the Markov chain in Fig. 9.

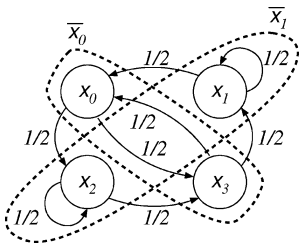


Fig. 9. State chain corresponding to the kneading matrix in (10) and proposed state aggregation.

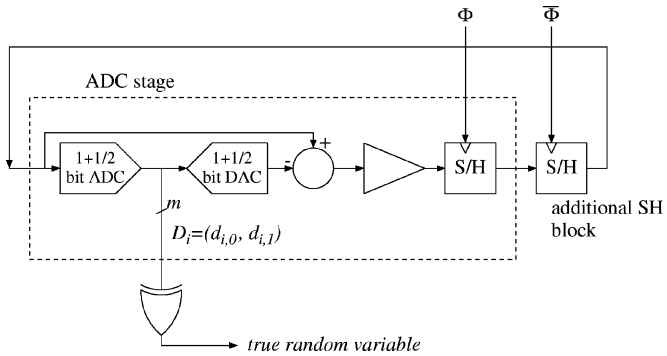


Fig. 10. Basic block diagram of a true random source based on ADC parts.

This chain is clearly not suitable for the direct generation of random symbols.[4] However, its particularly regular structure allows us to *aggregate* the states of the graph surprisingly obtaining another, simpler Markov chain [30]. The aggregation must be done along the dotted lines also shown in Fig. 9, introducing two new macro-states: state $\bar{\mathsf{x}}_0$ corresponds to the system being either in x_0 or in x_3, while state $\bar{\mathsf{x}}_1$ corresponds to the system being either in x_1 or in x_2. The resulting diagram is identical to that in Fig. 5. Hence, it is intuitive that the system can be deployed for random-bit generation with minimal modifications.

Now, let us consider how to do so practically. In principle, it should be sufficient to take a single stage from a one and a half bit per stage ADC and to close the output in a loop onto its input. In fact, this should already realize the computation of $x_{n+1} = M(x_n)$. In practice, this arrangement requires an additional S/H block driven in phase opposition from the one inherent in the stage, to prevent the system from behaving as an algebraic loop when the S/H is in sample mode. In order to evaluate whether the system is in macro-state $\bar{\mathsf{x}}_0$ or $\bar{\mathsf{x}}_1$, one can then deploy the digital outputs of the stage. In fact, the partition intervals on which the kneading matrix is computed are partially coincident with the quantization intervals of the coarse converter. The correspondence is as shown in the table at the bottom of the page. Hence, for determining the macro-state, it is sufficient to take the exclusive or of $d_{i,0}$ and $d_{i,1}$. The complete arrangement is thus as illustrated in Fig. 10.

Having proven that an ideal output statistics can be obtained, it is now also possible to show that a source designed following the proposed guidelines is completely *unrepeatable*. This is discussed here, rather than in the more theoretical Section II, as it is a consequence of *implementation*. The basic idea is that in an analog circuit whose state cannot be known with infinite accuracy and which is necessarily affected by noise, the sensitivity to initial conditions of the underlying chaotic system does not allow for two identical trajectories at different runs. Let us make the hypothesis that the system state is represented by a voltage whose maximum swing is V_{swing} and that its state can be measured with a minimum error that is Gaussian distributed with variance σ_E^2. With this, it is not possible to set the initial condition x_0 with a relative precision better than approximately $\pm 3\sigma_E / V_{\text{swing}}$. Thanks to the map slope, equal to 2, and to the addition of further (Gaussian) processing noise of variance σ_P^2 the standard deviation on the voltage representing x_1 is necessarily

$$\sigma_1 = \sqrt{(2\sigma_E)^2 + \sigma_P^2} \tag{11}$$

and x_1 cannot be determined with a relative precision better than $\pm 3\sigma_1 / V_{\text{swing}}$. Then, the standard deviation on the voltage representing x_2 happens to be

$$\sigma_2 = \sqrt{(4\sigma_E)^2 + (2\sigma_P)^2 + \sigma_P^2} = \sqrt{16\sigma_E^2 + 5\sigma_P^2} \tag{12}$$

and, in general, the variance at step n is given by

$$\sigma_n = \sqrt{4^n \sigma_E^2 + \frac{1}{3}(4^n - 1)\sigma_P^2}. \tag{13}$$

When $6\sigma_n$ becomes comparable with V_{swing}, even knowing x_0 it becomes impossible to estimate x_n or, even worse, the digital outputs of the system. From (13) this happens in approximately $\bar{n}$ steps, where

$$\bar{n} \approx \log_2 \left(\frac{V_{\text{swing}}}{6\sqrt{\sigma_E^2 + \frac{1}{3}\sigma_P^2}} \right). \tag{14}$$

Note that these calculations assume that the processing noise σ_P^2 is not pathologically large. In other words, it is assumed that it cannot prevent the normal operation of the circuit (e.g., by

[4]The state sequences produced by such a machine are not made of independent symbols. For instance, as a counter example, it is easy to see that $\Pr(\mathsf{X}^{(n)} = \mathsf{x}_1 \mid \mathsf{X}^{(n-1)} = \mathsf{x}_0) \neq \Pr(\mathsf{X}^{(n)} = \mathsf{x}_1 \mid \mathsf{X}^{(n-1)} = \mathsf{x}_3)$, i.e., the probability of being at a certain state at time n is *dependent* on the state the system was in at time $n-1$.

$d_{i,0}, d_{i,1}$	partition interval	state	macro-state
00	X_0	x_0	$\bar{\mathsf{x}}_0$
01	X_1 or X_2	x_1 or x_2	$\bar{\mathsf{x}}_1$
11	X_0	x_3	$\bar{\mathsf{x}}_0$

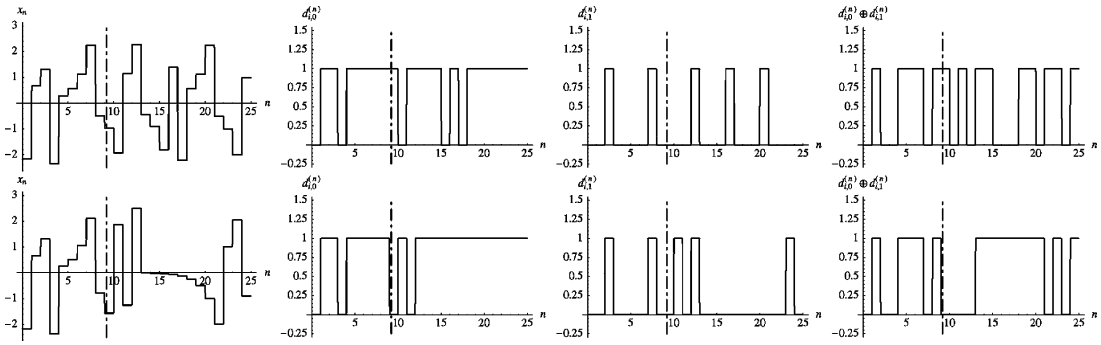


Fig. 11. Simulation of the true random bit generator based on the architecture in Fig. 10. The two rows report two run starting at the same initial condition. From left to right: analog state, first digital output of the ADC stage, second digital output of the ADC stage, XOR of the two (random variable). The dashed-dotted line shows the time-horizon after which the two runs diverge.

making blocks saturate). This is a reasonable assumption since the analog parts we are thinking of employing are designed to work in a high-resolution ADC and thus cause a processing noise that must be orders of magnitude smaller than the maximum signal swing.

Moreover, the value of $\bar{n}$ in (14) is an approximation since it takes into account only the *stretching* caused by the map. If the map noninvertibility is also taken into account, the decay of correlation between trajectories, even if they originate from two nearby initial points, need to be modeled with more sophisticated tools. In any case (see [10] and references therein), the logarithmic link between the initial uncertainty and the number of time steps needed to achieve complete unpredictability maintains its validity.

The previous observations have a twofold consequence.

Note first that, if we try a second run of the same generator, striving to start it at some known initial conditions (possibly eavesdropped in attacking the overall cryptosystem), after approximately $\bar{n}$ steps the random bits they produce become uncorrelated. Both to illustrate this phenomenon and the general operation of the circuit in Figs. 10 and 11 shows the results of a simulation, with all the most important signals. The two rows show two runs started at nominally identical conditions. The voltage swing is set to 5 V and the noise floor to 1 mV. With this, $\bar{n} \approx 9 - 10$. As is evidenced by the plots, after about 9–10 steps (i.e., in a remarkably short time), the binary outputs of the two runs become quite different.

As a second remark, note that this fast approaching to complete unpredictability means that only the bits deriving from the first $\bar{n}$ cycles must be discarded after startup to yield a completely unrepeatable behavior.

It must now be observed that, simple as it looks, the architecture in Fig. 10 is still not the best way of deploying IP originally devised for pipeline ADCs. To justify this statement, it is necessary to detail a bit more this type of converters. As an example, we take the architecture recently proposed by Abo and Gray [27], capable of very low voltage operation (1.5 V) and a remarkable throughput of over 14 Msample/s at 10-bit resolution. The converter is built from nine identical stages, whose schematic diagram is shown in Fig. 12.

As can be seen, in the stage there is no explicit S/H block. *Switched capacitor* operation, which is the usual, optimal choice for this kind of circuits, achieves a similar effect for free. The

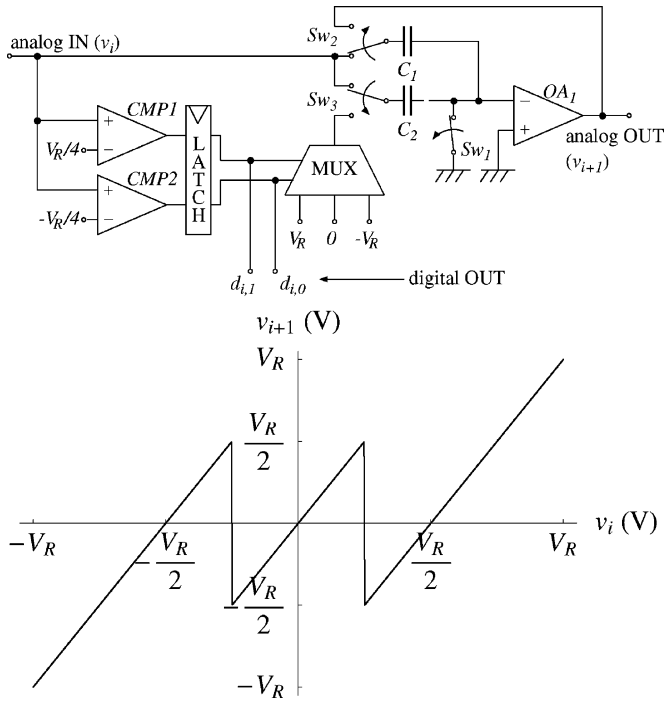


Fig. 12. One and a half bit ADC stage as proposed by Abo and Gray (top). Corresponding analog input to analog output characteristics (bottom).

operation is based on the clocked commutation of the three switches Sw_1, Sw_2 and Sw_3. When the driving clock Φ is high, the latch is transparent, and the switches are in the positions shown in the schematics. Consequently, the two capacitances C_1 and C_2 get charged at the input voltage v_i, while the latch outputs report the thermometric code corresponding to the coarse conversion of v_i. The analog output of the circuit, corresponding to v_{i+1} is null. Since the internal circuit variables are continuously dependent on v_i, this situation corresponds to the stage "sample mode." When the clock line goes low, the digital latch starts keeping its output information fixed, while all the switches change their position. With this, all the circuit variables—comprising the outputs—stop being dependent on v_i, which is equivalent to the circuit going into "hold mode." Thanks to C_1 being equal to C_2, the analog output of the circuit settles to the sum of the voltages on C_1 and C_2 (i.e., $2v_i$) plus an offset provided by the analog mux and dependent on the coarse conversion results. Hence, the two comparators play the

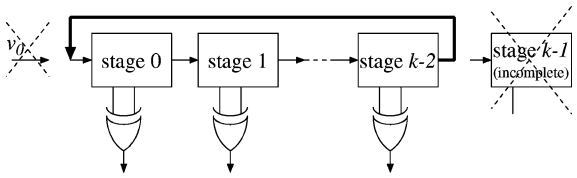


Fig. 13. Modifications required on an ADC pipeline to transform it into a random bit generator.

role of the coarse ADC in the diagram of Fig. 7, the analog mux plays the role of the coarse DAC, and the operational amplifier, together with the surrounding capacitors and switches, has both the role of the rescaler and the S/H. The following stage is obviously driven by clock $\overline{\Phi}$ (and then other Φ and $\bar{\Phi}$ stages are further alternated). Note that the normalized range $[-1, 1]$ considered so far, maps into $[-V_R/2, V_R/2]$ in this circuit.

Given the switched capacitor arrangement, it is surely not impossible, but certainly cumbersome to extrapolate from the stage IP the parts necessary to build a further S/H stage in order to realize the architecture in Fig. 10. Fortunately, this is not necessary, and it is actually possible to escape from this issue while taking the exploitation of ADC parts to a much higher degree than that shown so far. Consider what happens if one, rather than operating on a single pipeline stage, closes the whole analog pipeline of an ADC converter in a feedback loop. This can be done, as illustrated in Fig. 13, by simply discarding the correction logic of the ADC and its final stage (which is not suitable for providing an output). The only constraint is to make sure that the loop comprises an even number of stages, so that the S/H blocks driven by the clock Φ and those driven by $\overline{\Phi}$ are correctly alternated. For simplicity we assume that the ADC number of stages k is odd, so that $k-1$ is even. With this, by discarding the last (incomplete) stage one gets an even number of stages. Clearly, if k is even, one can always choose to discard not just the last stage, but also the last but one to satisfy the constraint.

The result is a model

$$\begin{cases} v_1^{(n+1)} = M_0(v_{k-1}^{(n)}) \\ v_2^{(n+1)} = M_1(v_1^{(n)}) \\ \cdots \\ v_{k-1}^{(n+1)} \equiv v_0^{(n+1)} = M_{k-2}(v_{k-2}^{(n)}) \end{cases} \tag{15}$$

where $M_0 \equiv M_1 \equiv \cdots \equiv M_{k-2} \equiv M$. By applying the substitution $v_i^{(n)} \rightarrow x_{i-n \bmod k-1,n}$ the model is transformed into

$$\begin{cases} x_{0,n+1} = M_{n \bmod k-1}(x_{0,n}) \\ x_{1,n+1} = M_{1+n \bmod k-1}(x_{1,n}) \\ \cdots \\ x_{k-2,n+1} = M_{k-2+n \bmod k-1}(x_{k-2,n}) \end{cases} \tag{16}$$

from which it should be evident that, as long as all the Ms are equivalent, one has $k-1$ decoupled, time-invariant, first-order systems operating in parallel. The model is *interleaved* in the sense that voltage v_i represents the state variable x_i at time 0, x_{i-1} at time 1, and so on. Alternatively, one can say that the chaotic system that uses the stage i at time n, uses the stage $i+1 \bmod k-1$ at time $n+1$. At each time step, for each of these systems, we can get a random bit per cycle by simply taking the XOR of the digital output variables of the relevant stage. Considerations such as those made above about sensitivity to initial conditions and noise assure that even starting at nominally identical conditions the individual chaotic systems quickly produce uncorrelated outputs. Thus, at each clock half-cycle one can read a whole random word from the XOR array. Apart from the simplification in the reuse of the ADC IP, parallelizing the random generator is hence also very beneficial on throughput. It is worth trying to quantify the performance that can be expected. Starting from the design by Abo and Gray, it is possible to reuse eight of the nine stages of the converter. Each stage can operate at speeds up to $14.3 \cdot 10^6$ conversions per second. Hence, the proposed architecture should be ideally capable of up to $14.3 \cdot 10^6$ random Byte/s (i.e., 109 Mbit/s [5]). The output comes out *in parallel* organized in bytes which get updated every half clock cycle. This eases the interfacing with conventional buses.

It must be remarked that the numbers given in the above paragraph must be taken with care. The possibility of obtaining true-random numbers was in fact proven under the hypothesis that a map *identical* to that in Fig. 8 could be made available by an electronic circuit. In practice, as it is better discussed in the following section, one has always to deal with parts with deviate a little from their nominal models. It is evident that if this happens the model deviations cannot help resulting in the introduction of slight correlations in the output streams. In this case, depending on the amount of residual correlation that can be tolerated, *moderate* degrees of blind post-processing need to be reintroduced to eliminate the redundancy in the output streams. Fortunately, one can expect that the quality of the bits produced by the chaotic source can remain quite high even in presence of small implementation errors. If this is true (as it will be shown shortly), the post processing can be notably lightweight. Consequently, one is in a position to trade some throughput for implementation simplicity. As an example, the following section considers a relatively rough decorrelation algorithm based on parity computation taking bits in groups of four. This results in a fourfold rate reduction. Even with this, the final data rate can remain well in excess of a few tens random megabit per second.

Throughput is clearly not the unique performance index. Note that the overall effort to reuse the design by Abo and Gray is limited to the movement of *a single wire* as far as the analog part is concerned. For what concerns the digital part, the design is in fact made smaller given that a single XOR array (or a couple of levels of XOR gates if one considers also the parity-based post-processing) is substituted for the correction logic of the converter. Also power consumption can be expected to reduce with regard to the converter used as a starting point.

As a final remark, note that the architecture can easily be modified to deal with other kinds of pipeline ADCs (e.g., those based on two and a half bits per stage or other subranging structures [31]).

Note also that in a previous paper [29], we have introduced a method for producing random bits from pipeline ADCs by reusing not just their stages but their digital correction logic as well. With this, at the cost of some inefficiency, the level of reuse of ADC IP can be taken to an even higher level.

[5]Assuming 1 Mb $= 1024.1024$ b.

V. Simulation Results and Yield Estimation

As discussed in the previous sections, if the proposed generator could be implemented with ideal analog components, its ability to produce a stream of true random bits would rely on a sound mathematical proof based on the statistical approach to chaotic systems.

Regrettably, any real-world implementation of the $k-1$ stages generating the system entropy is compelled to be nonideal and the quality of the output streams needs to be validated against standard statistical tests to assess the validity of the approach.

A. Modeling of Nonidealities

From circuit analysis and extensive SPICE simulation one gets that the nonideal behavior of each stage of the ADC can be generally modeled with a piecewise-affine function, i.e., by saying that the ith stage actually implements the map

$$M_i(x) = \nu + \begin{cases} (2+\delta g_1^i)x + 2 + \delta o_1^i, & \text{for } x < -\frac{1}{2} + \delta t_1^i \\ (2+\delta g_2^i)x + \delta o_2^i, & \text{for } -\frac{1}{2} + \delta t_1^i \le x < \frac{1}{2} + \delta t_2^i \\ (2+\delta g_3^i)x - 2 + \delta o_3^i & \text{for } x \ge \frac{1}{2} + \delta t_2^i \end{cases} \tag{17}$$

where the error term ν is due to thermal noise, δg_j^i are the deviations from the ideal gains, δo_j^i are the deviations from the ideal offsets, and δt_j^i are the deviations from the ideal thresholds of comparators.

Note that the contribution of the thermal noise changes at each evaluation of the output can be considered a Gaussian random variable with zero-mean and temperature-dependent variance. Other deviations are constant for all output evaluations but vary from one stage to the other and from one system to another.

To model the distribution of gains, offsets and threshold deviations, consider that inaccuracies due to circuit fabrication (such as those affecting MOST transconductivity and threshold voltages, or capacity of integrated capacitors) can be effectively thought as Gaussian errors whose variance decreases with the area of the device [32]. Such errors are usually very small for high-precision circuits like ADCs. Hence, their effect on macroscopic quantities characterizing the circuit (like slopes and offsets) are well approximated by linear combinations of small-variance independent Gaussian random variable, i.e., by further Gaussian random variables. For this reason we can assume that, over the whole production, all deviations are characterized by variances $\sigma_g^2, \sigma_o^2, \sigma_t^2$. The parameter space is then limited by assuming that $\sigma_g^2 = \sigma_o^2 = \sigma_t^2 = \sigma^2$.

Now that a statistical error model of the system stages is set up, the question is how to employ it. Ideally, one would like to have a mathematical toolbox capable of taking as inputs the statistical descriptions of the error parameters in (17) and of delivering the average and worst-case statistics of the resulting chaotic bitstreams. Unfortunately, the definition of formal statistical error propagation models is extremely hard, probably infeasible in general terms, and surely out of the scope of this work. The best that can be done is to assume that the overall system is robust to nonidealities under some criteria and to *validate such individual hypotheses* by stochastic simulation. Here, we take into account two properties: the ability of the system to retain chaotic behavior and the the ability to produce bitstreams that can pass all the randomness tests published by NIST.

TABLE I
Variance of the Gain/Offset/Threshold Deviations and Corresponding Yield of the Circuits Used as ADC

σ^2	yield ad an ADC
0.003	0.991
0.005	0.796
0.01	0.103

In order to run simulations, it is first necessary to estimate a reasonable level for σ^2. To this aim we preliminary perform another set of Montecarlo simulations. For various tentative values of σ^2, we generate 5000 instances of a $k = 9$-stage ADC (of which 8 stages will be finally adopted). Each instance is simulated while converting an analog quantity spanning the whole admissible range. The ratio between the number of ADCs whose conversion error is not greater than 1 LSB and the number of tested instances is a *yield* of the system used as an ADC. Such an estimated yield is shown in Table I for some values of σ^2.

As expected, the ADC yield critically depends on the implementation quality of the analog blocks. Note that deviations with variances well below 0.01 must be achievable to ensure the functionality as an ADC of a reasonable percentage of the produced circuits. Hence, we can assume $\sigma^2 = 0.01$ to test the proposed generator in worst-case implementation conditions.

As a final remark, note that the presence of noise and nonidealities implies the need for a small post-processing logic. If the bitstream exiting from the raw generator (made of $d_{i,0}^{(n)} \oplus d_{i,1}^{(n)}$) is $b_0, b_1, b_2, \ldots$, we suggest actually emitting the parity bit of groups made of four subsequent bits, i.e., the stream $b'_0, b'_1, b'_2, \ldots$ with $b'_i = b_{4i} \oplus b_{4i+1} \oplus b_{4i+2} \oplus b_{4i+3}$. Again, this is a very cautious choice: lower levels of post-processing would probably suffice in common environments, but we prefer assuring a safety margin.

B. Randomness Test

As already mentioned, to assess the quality of the produced bitstream we resort to two sets of tests, which here deserve some description.

The first is derived form the recommendations contained in FIPS 140.2 [11], a NIST promoted standard for security in digital systems. FIPS tests examine chunks of 20 000 bits counting the occurrences of specific combinations of bits. The final values of the counters are matched against pre-defined thresholds to assess sequence randomness.

Depending on the kind of occurrences counted, the FIPS 140.2 tests are called *Monobit*, *Poker*, *Runs*, and *Long Runs*.

This first test-suit is introduced by NIST having in mind black box online testing of random sources and is consequently relatively lightweight.

The second set of tests comes from the NIST special publication 800-22 [12], related to the NIST Random Number Generator Testing Project.

This suite is composed of 15 tests, called *Frequency*, *Block Frequency*, *Runs*, *Longest Runs*, *Matrix Rank*, *Spectral*, *Non-overlapping Template (NOT) Matching*, *Overlapping Template (OT) Matching*, *Universal*, *Lempel-Ziv*, *Linear Complexity*, *Serial*, *Approximate Entropy*, *Cumulative Sums*, *Random Excursions*.

They cover a much wider spectrum of randomness implication than FIPS 140.2 and are open to a more sophisticated hierarchical interpretation and are meant for design-stage testing.

Each test produces a so-called P-value. The P-value is a real number in [0,1] estimating the probability that a finite realization of an ideally random binary process deviates from the ideal statistics more than the given string. Obviously, the higher the P-value the more random the string. We will say that the test is passed if the P-value is greater than $\alpha' = 0.01$.

Since the computation of the test is based on the string at hand, the P-value is itself a random variable, and theory says that if we test many finite strings from a ideally random bit generators, the P-values will distribute uniformly in [0,1].

Hence, on the average, the ratio between the number of bit strings passing the test and the number of tested strings is $1 - \alpha' = 0.99$. Since the size of the sample is finite, we may expect some deviation from this proportion. The adoption of a three-σ criterion is suggested in 800-22. Hence, we say that the number of passing sequences is compatible with the randomness of the source if it lies somewhere in the range $1 - \alpha' \pm 3\sqrt{\alpha'(1-\alpha')/T}$, where T is the number of tested strings.

Finally, measuring the deviation of the empirical distribution of the P-values obtained repeating the same test on many strings from the flat distribution yields a meta-indicator of the randomness of the source. The use of a χ^2 test to measure deviation from uniformity actually yielding a further U-value in [0,1] estimating the probability that the empirical distribution of the P-values actually comes from a uniform distribution in [0,1] is suggested in 800–22. According to the same document, an U-value not less than 10^{-4} is compatible with a perfectly random source.

C. Testing Procedure and Results

A Monte Carlo loop generates instances of a $k = 9$-stage ADC with $\sigma^2 = 0.01$. Each instance is simulated while converting an analog quantity spanning the whole admissible range and is discarded if the conversion error is greater than 1 LSB.

For all ADCs passing this functionality test (approximately 10% of all generated instances as results from Table I), we reuse the first $k-1 = 8$ analog blocks to generate 5×10^6 bits. These bits are tested following both FIPS140.2 and 800-22. We stop when $T = 1000$ different random number generators have been tested each producing its 5×10^6 output bits.

Table II reports the yield with respect to the 4 tests in FIPS 140.2, i.e., the ratio between the number of 20 000-bit chunks that passed that test and the total number of chunks tested ($1000 \times 5 \times 10^6/20\,000 = 250\,000$). Note that practically all tested strings passed the tests and that even an ideal RNG could not result in a 100% pass in this kind of tests.

The same 1000 strings (each from a different generator) of 5×10^6 bits were fed into the 15 tests in 800-22, The second column of Table III reports the yield, i.e., the ratio between the number of generators that passed each test and the number of tested generators.

TABLE II
ESTIMATED YIELD FOR THE 4 TEST OF FIPS 140.2

FIPS 140.2 test	yield
Monobit	0.999876
Poker	0.999936
Runs	0.999772
Long Runs	0.999708

TABLE III
ESTIMATED YIELD FOR THE 15 TESTS IN NIST 800-22. ACCEPTABLE YIELD MEASURE FOR THE 1000 BIT STRINGS TESTED SHOULD LIE IN [0.981,0.999]. U-VALUES ABOVE 10^{-4} ARE CONSIDERED ACCEPTABLE

SP800-22 test	yield	U-value
Frequency	0.991	0.185
Block Frequency	0.992	0.631
Runs	0.996	0.612
Longest Runs	0.987	0.377
Matrix Rank	0.989	0.585
Spectral	0.994	0.00974
NOT Matching	0.989	0.504
OT Matching	0.988	0.492
Universal	0.983	0.627
Lempel-Ziv	0.989	0.000306
Linear Complexity	0.990	0.596
Serial	0.995	0.369
Approximate Entropy	0.991	0.518
Cumulative Sums	0.991	0.516
Random Excursion	0.994	0.968

Note that yield is always extremely high. Moreover, it always falls within the three-σ range $1 - \alpha' \pm 3\sqrt{\alpha'(1-\alpha')/T}$ that, for $\alpha' = 0.01$ and $T = 1000$, is [0.981,0.999].

The third column of Table III reports the U-values corresponding to the distribution of the P-values generated by the 1000 instances of each test. It is suggested in that values above 10^{-4} are compatible with perfectly random sources.

The interpretation of the results is as follows: given the nature of our tests, its outcomes need to be interpreted statistically. Hence, what we have is that

- from our stochastic simulation, even in presence of non-idealities, our RNGs conserve chaotic behavior with a probability indistinguishable from 1;
- from our stochastic simulation, even in presence of non-idealities, our RNGs conserve the property of producing bitstreams passing the NIST tests with a probability indistinguishable from 1.

As a last remark, note that in the tests presented so far we have discarded the RNGs derived from ADCs whose stages would lead to conversion errors larger than 1 LSB. Further tests have shown that our RNG can produce bitstreams compliant with the NIST test-suite even when originating from *bad* ADCs (i.e., ADCs whose stage lead to a conversion error slightly larger than 1 LSB), provided that the LSB is sufficiently small (i.e., that the originating ADC does not have a pathologically low resolution). This is a further proof of the robustness of the proposed approach.

D. Exploitation of On-Line Tests to Identify Transient or Permanent Misbehavior in the Stages of the RNG

In designing RNGs, tests are useful not only to assess the ability of the RNGs to produce sufficiently random data in normal situations, but also to set up on-line procedures capable to trigger alerts when abnormal conditions occur. In the previous paragraphs we have shown that under normal deviations in the system parameters (i.e., when the circuit behaves almost-ideally) the system conserves the property of producing bit sequences that can be considered random under the NIST criteria.

Here, we consider what happens in case of catastrophic misoperation of some RNG stage. By catastrophic it is meant "taking the stage operation far out of specification." Stage failures of this sort may happen due to failing components/electronic-devices or to disconnection of some wiring. Common failure scenarios might include

1) full failure in the stage comparators (output stuck high/low);
2) failure in the switches of the switched capacitor blocks;
3) failure in the clocking of a stage;
4) failure in the high-gain amplifiers used within the switched capacitor circuit blocks (output stuck at higher/lower saturation rail or very large gain loss);
5) very large deviations in the capacitance values of the switched capacitor building blocks.

The question here is whether the RNG structure is such that these kind of failures can be easily recognized or are likely to pass unnoticed, with the RNG simply starting to deliver low quality random numbers.

Luckily, in all the failure scenarios illustrated above, the TRNG that we propose exhibits either a full loss of its chaotic behavior of anyway a loss of its ability to pass the FIPS tests. Hence, our system can effectively use the FIPS tests as online tests.

The previous results deserves some commenting. In fact, it is a direct consequence of the fact that the proposed RNG is built in such a way that even a local failure in one of its stages tend to be *globally exposed*. This is due to the interleaved nature of the system, where each state variable is processed in turn by all the system stages. Such property does naturally prevent hard-to-catch situations where a stage stops producing random bits, but the others continue doing so. If one stage experiences a catastrophic failure, the whole system tends to go down, making it easy to detect the problem, without the need to introduce specific tests.

VI. Conclusions

By exploiting the statistical approach to the study of nonlinear chaotic systems, we are able to recognize that the structure of a common pipelined ADC, if ideally implemented, can be reused to obtain a generator of bits whose true random behavior can be guaranteed by mathematical proof.

Obviously, the possibility of implementing the system from ADC parts which are now ubiquitous makes the design suitable for embedding.

Nonideality in the implementation of the analog part of the ADC leads to a loss of quality in the produced bit string that can be easily compensated for by means of a moderate post processing that reduces the generator bit rate.

By extensive simulation, we show that even a rough post-processing based on XOR is able to give a true random bit generator satisfying commonly accepted NIST standards and still running at not less than 10 Mbit/s, thus exceeding by far the bit rates commonly found in literature and commercial products.

References

[1] J. A. Menezes, P. C. Oorschot, and S. A. Vandtone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC International, 1997.

[2] D. E. Eastlake, S. D. Crocker, and J. I. Shiller, "RFC 1750: Randomness recommendations for security," in *Internet Society Request for Comments*: Internet Engineering Task Force, 1994.

[3] W. Schindler and W. Killmann, "Evaluation criteria for true (physical) random number generators used in cryptographic applications," in *Cryptographic Hardware and Embedded Systems – CHES 2002*, B. S. J. Kaliski, C. K. Koc, and C. Paar, Eds. Heidelberg: Springer-Verlag, 2003, vol. 2523, Lecture Notes in Computer Science, pp. 431–449.

[4] B. Jun and P. Kocher. (1999) "The Intel Random Number Generator," Tech. Rep. Cryptography Research Inc.. [Online]. Available: http://www.cryptography.com/resources/whitepapers

[5] (2003) "Evaluation Summary: VIA C3 Nehemiah Random Number Generator," Tech. Rep.. Cryptography Research Inc.. [Online]. Available: http://www.cryptography.com/resources/whitepapers

[6] V. Fischer and M. Drutarovsky, "True random number generator embedded in reconfigurable hardware," in *Cryptographic Hardware and Embedded Systems – CHES 2002*, B. S. J. Kaliski, C. K. Koc, and C. Paar, Eds. Heidelberg, Germany: Springer-Verlag, 2003, vol. 2523, Lecture Notes in Computer Science, pp. 415–430.

[7] R. Devaney, *An introduction to Chaotic Dynamical Systems*, 2nd ed. Reading, MA: Addison-Wesley, 1989.

[8] A. Lasota and M. C. Mackey, *Chaos, Fractals and Noise. Stochastic Aspects of Dynamics*, 2nd ed. Heidelberg, Germany: Springer-Verlag, 1995.

[9] T. Kohda and A. Tsundeda *et al.*, *Information Sources Using Chaotic Dynamics*, M. P. Kennedy *et al.*, Eds. Boca Raton, FL: CRC International, 2000, Chaotic Electronics in Telecommunications, ch. 4.

[10] G. Setti, G. Mazzini, R. Rovatti, and S. Callegari, "Statistical modeling of discrete time chaotic processes: Basic finite dimensional tools and applications," *Proc. IEEE*, vol. 90, no. 5, pp. 662–690, May 2002.

[11] *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards FIPS 140-2, May 2001.

[12] "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," National Institute for Standards and Technology, Special publication 800-22, 2001.

[13] L. Buttyan and J. Hubaux, "Report on a working session on security in wireless ad hoc networks," in *Proc. MobiHoc'02*, Lausanne, Switzerland, 2002.

[14] J. Hubaux, L. Buttyan, and S. Capkun, "The quest for security in mobile ad hoc networks," in *Proc. MobiHoc'01*, Long Beach, CA, 2001.

[15] M. Delgado-Restituto, F. Medeiro, and A. Rodríguez-Vázquez, "Nonlinear, switched current CMOS IC for random signal generation," *Electron. Lett.*, no. 25, pp. 2190–2191, 1993.

[16] T. Kuunsela, "Random number generation using a chaotic circuit," *J. Nonlin. Sci.*, vol. 3, no. 4, pp. 445–458, 1993.

[17] G. M. Bernstein and M. A. Lieberman, "Secure random number generation using chaotic circuit," *IEEE Trans. Circuits Syst.*, vol. 37, no. 9, pp. 1157–1164, Sep. 1990.

[18] L. Trevisan and S. Vadhan, "Extracting randomness from samplable distributions," in *Proc. IEEE Symp. Fundations of Computer Science*, Nov. 2000, pp. 32–42.

[19] M. Blum, "Independent unbiased coin flips from a correlated biased source—A finite state Markov chain," *Combinatorica*, vol. 6, no. 2, pp. 97–108, 1986.

[20] J. Von Neumann, "Various techniques used in connection with random digits," *National Bureau of Standards, Applied Mathematics Series*, vol. 12, pp. 36–38, 1951.

[21] M. Dellnitz, G. Froyland, and S. Sertl, "On the isolated spectrum of the Perron-Frobenius operator," *Nonlinearity*, vol. 13, no. 4, pp. 1171–1188, 2000.

[22] R. E. Kalman, "Nonlinear aspects of sampled-data control systems," in *Proc. Symp. Nonlinear Circuit Analysis*, vol. VI, 1956.
[23] B. P. Kitchens, *Symbolic Dynamics*. New York: Springer, 1998.
[24] S. Callegari and R. Rovatti, "Analog chaotic maps with sample-and-hold errors," *IEICE Trans. Fundament.*, vol. E82A, no. 9, pp. 1754–1761, Sep. 1999.
[25] S. Callegari, R. Rovatti, and G. Setti *et al.*, *Robustness of Chaos in Analog Implementations*, M. P. Kennedy *et al.*, Eds. Boca Raton, FL: CRC International, 2000, Chaotic Electronics in Telecommunications, ch. 12, pp. 397–442.
[26] M. Delgado-Restituto and A. Rodriguez-Vazquez, "Integrated chaos generators," *Proceedings of the IEEE*, vol. 90, pp. 747–767, May 2002.
[27] A. M. Abo and P. R. Gray, "A 1.5-V, 10-bit, 14.3-MS/s CMOS pipeline analog-to-digital converter," *IEEE J. Solid-State Circuits*, vol. 34, no. 5, pp. 599–606, May 1999.
[28] M. D. Restituto and A. Rodríguez-Vásquez, "Piecewise affine Markov maps for chaos generation in chaotic communication," in *Proc. ECCTD'99*, vol. 2, Stresa, Italy, Aug. 1999, pp. 453–458.
[29] S. Callegari, R. Rovatti, and G. Setti, "Efficient chaos-based secret key generation method for secure communications," in *Proc. NOLTA*, Xian, China, Oct. 2002.
[30] D. A. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge, MA: Cambridge Univ. Press, 1995.
[31] S. M. Yoo, T. H. Oh, J. W. Moon, S. H. Lee, and U. K. Moon, "A 2.5 V 10 b 120 MSample/s CMOS pipelined ADC with high SFDR," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 2002, pp. 441–444.
[32] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1440, Oct. 1989.
[33] *Chaotic Electronics in Telecommunications*, M. P. Kennedy, R. Rovatti, and G. Setti, Eds., CRC Internationals, Boca Raton, FL, 2000.

Sergio Callegari (M'00) was born on May 26, 1970. He received the Dr.Eng. (Hons.) degree in electronic engineering and the Ph.D. degree in electronic engineering and computer science from the University of Bologna, Bologna, Italy, in 1996 and 2000, respectively, working on the study of nonlinear circuits and chaotic systems.

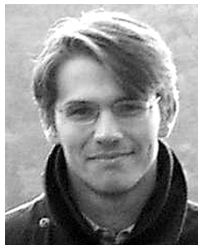
In 1996, he was a Visiting Student at King's College London, London, U.K. He is currently a Researcher at the II School of Engineering, University of Bologna, where he is a Lecturer in digital electronics and sensors. At the same university, he has also joined the recently instituted Advanced Research Center on Electronic Systems for Information and Communication Technologies. His current research interests include sensor design, nonlinear signal processing, internally nonlinear, externally linear networks, and chaotic maps. He has authored or coauthored more than 35 papers in international conferences, journals, and scientific books.

Dr. Callegari was co-recipient of the IEEE CAS Darlington Award in 2004.

Riccardo Rovatti (M'97–SM'03) was born on January 14, 1969. He received the Dr. (Hons.) degree in electronic engineering and the Ph.D. degree in electronic engineering and computer science from the University of Bologna, Bologna, Italy, in 1992 and 1996, respectively.

Since 1997, he has been a Lecturer of Digital Electronics at the University of Bologna. In 2000, he became an Assistant Professor, and in 2001, an Associate Professor of analog electronics. His research interests include fuzzy theory foundations, learning and CAD algorithms for fuzzy and neural systems, statistical pattern recognition, function approximation, nonlinear system theory, and identification as well as theory and applications of chaotic systems. He has authored or co-authored more than 160 international scientific publications. He is one of the editors of the book *Chaotic Electronics in Telecommunications* (Boca Raton, FL: CRC, 2000).

Dr. Rovatti was a Guest Editor of the PROCEEDINGS OF THE IEEE Special Issue on Applications of Nonlinear Dynamics to Electronic and Information Engineering. In 2004, he was a co-recipient of the IEEE CAS Society Darlington Award.



Gianluca Setti (S'89–M'91–SM'03) received the Dr.Eng. (Hons.) degree in electronic engineering and the Ph.D. degree in electronic engineering and computer science from the University of Bologna, Bologna, Italy, in 1992 and in 1997, respectively, for his contribution to the study of neural networks and chaotic systems.

From 1994 to 1995, he was with the Laboratory of Nonlinear Systems (LANOS), Swiss Federal Institute of Technology, Lausanne, as a Visiting Researcher. Since 1997, he has been with the School of Engineering, University of Ferrara, Ferrara, Italy, where he is currently an Associate Professor of Circuit Theory and Analog Electronics. He is co-editor of the book Chaotic Electronics in Telecommunications (Boca Raton, FL: CRC Press, 2000). His research interests include nonlinear circuit theory, recurrent neural networks, and design and implementation of chaotic circuits and systems, as well as their applications to electronics and signal processing.

Dr. Setti served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–I: FUNDAMENTAL THEORY AND APPLICATIONS in the area of Nonlinear Circuits and Systems (1999–2002) and then Chaos and Bifurcations (2002–2004). He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–II: ANALOG AND DIGITAL SIGNAL PROCESSING and as the Deputy-Editor-in-Chief for the *Circuits and Systems Magazine*, both since 2004. In 2001, was the Chair of the Technical Committee of the Circuits and Systems (CAS) Society. He was the Technical Program Co-Chairman of NDES2000 (Catania, Italy) and the Track Chair for Nonlinear Circuits and Systems at ISCAS2004 (Vancouver, BC, Canada) and is currently the Special Session Co-Chair for ISCAS2005 (Kobe, Japan). He was a Guest Editors of the PROCEEDINGSOF THE IEEE Special Issue on the Applications of Nonlinear Dynamics to Electronic and Information Engineering in May 2002. He received the 1998 Caianiello Prize for the best Italian Ph.D. thesis on neural networks and was co-recipient of the 2004 IEEE CAS Society Darlington Award (Best Paper Award for the IEEE TRANSACTIONS ON Circuits and SYSTEMS). Since 2004, he has been a Distinguished Lecturer of the IEEE CAS Society and since, January 2005, has been serving as a Member of the Society's Board of Governors.