This is the author's version of the article that has been presented at IEEE ESSCIRC2006
The editorial version of the paper is available at http://dx.doi.org/10.1109/ESSCIR.2006.307548

# A Fast Chaos-based True Random Number Generator for Cryptographic Applications

Fabio Pareschi*, Gianluca Setti*, and Riccardo Rovatti†
*ENDIF - University of Ferrara, via Saragat, 1 - 44100 Ferrara - ITALY
†DEIS - University of Bologna, viale risorgimento, 2 - 40136 Bologna - ITALY
All authors are also with ARCES - University of Bologna, via Toffano 2/2 - 40125 Bologna - ITALY
fpareschi@ing.unife.it, gsetti@ing.unife.it, rrovatti@arces.unibo.it

***Abstract*— We present the design and the validation by means of state-of-the-art randomness tests of a high-quality true random number generator which internally exploits a pipeline analog-to-digital converter modified to operate as a set of interleaved chaotic maps. Developing the circuit design relying on pipeline A/D technology, which is ubiquity used in all mixed signal systems, allow us to design a fast and very reliable TRNG. A prototype has been implemented in AMS 0.35 $\mu$m 2P3M technology and has a nominal throughput of 40 Mbits per second. The active area occupied by the chip is about 0.52 mm$^2$ and the power consumption is less than 30 mW.**

## I. Introduction

By its very definition, a Random Number Generator (RNG) is a circuit capable of producing infinitely long sequences of perfectly *independent* bits, with the property that, if restarted, it does not reproduce the same sequence (*non repeatability*). RNGs could find several applications in many engineering tasks; for instance they are are widely used in all cryptographic applications, where they are fundamental in the synthesis of confidential keys for symmetric and public-key crypto-systems [1].

Of course, actual implementations of RNGs can only approximate the above definition. They are divided into two main categories, namely *Pseudo-Random* generators, or PRNGs, which are numerical algorithms capable to "expand" short *seeds* into long, irregular, random-like bit sequences [1], [2], and *True Random* generators (TRNGs), which are based on some microscopic processes that can be addressed as *noise*. Of course, this second category is largely used in all security related applications, where the deterministic (and so predictable) nature of the PRNGs makes this first category not suitable for this kind of applications.

Typically TRNGs are based on processes like Johnson thermal noise [3], shot noise, jitter in PLL or free oscillator [4], but can also be based on quantum effects like the radioactive decay. Recently, some RNGs based on photon reflection, see [5]) has been proposed. They can achieve very good results in terms both of quality and speed (which are the two most important figures of merit in a RNG) but they do not represent a solution currently embeddable in silicon integrated technology.

So if we look for a system-on-a-chip solution, or just for a high quality low-cost solution, we must relay on processes which could be easily reproducible in a standard silicon-based embedded technology.

Our solution relays on a simple one dimension chaotic map [6]. The use of discrete-time chaotic maps in the realization of TRNGs has been known since many years and was firstly suggested by Ulam and Von Neumann in 1947 [7]. Recently in [8] it was proposed the use of a pipeline analog-to-digital converters (ADCs) based on 1.5 bit/stage cells [9] to implement a chaotic circuit that has been theoretically proved to generate *independent and identically distributed* - that is *random* – symbols. We report here the implementation and the measure results of a prototype implementing this solution.

The paper is organized as follows. In section II we report some details of the theory grounding our approach. In section III we present the design of the circuit, while in section IV we present results for statistical tests on sequences generated by the prototype.

For the validation we use the test suite provided from the US National Institute for Standard and Technology (NIST) [10].

## II. Pipeline ADCs and Markov Chaotic Maps

The design of the proposed circuit comes directly from pipeline ADCs technology. The architecture of a standard pipeline ADC, including the proposed modifications, is shown in Fig. 1. The figure depicts a $k$-stages converter; each $i$-th stage computes, usually with a small and fast flash converter, a *coarse* $m$-bit representation $D^{(i)} = \left(d^{(i,0)}, \ldots, d^{(i,m-1)}\right)$ of its input $v^{(i)}$ sampled at the time step $n$, and then calculates (and rescales) an analog *error conversion* $e^{(i)}$ to be passed at the time step $n+1$ to the following stage $(i+1)$-th as its input $v^{(i+1)}$. Note that the last stage, having nothing downhill, is composed only by the coarse converter. A digital logic provides to collect all the intermediate conversions ($k \cdot m$ bits), to elaborate them, and to supply the $l$-bits conversion, with $l \leq k \cdot m$.

The modification we propose is to close the entire pipeline into a loop, discarding the last stage due to the fact that it does not provide any conversion error, and to substitute the digital correction logic with a circuit capable of collecting all data end of extracting random bits from that. Actually, since it is very common to design a pipeline ADC in which all the stages work alternatively at the two phases of the clock, [1] we have the additional restriction to close the loop after an even number of stages.

The relation between the output and the input in a single stage, i.e. the function $v^{(i+1)} = M\left(v^{(i)}\right)$ is always a piece-wise affine function. For example, the $M$ function that can be found in 1.5 bit/stage converter, that is one of the most widely used architecture [9], is depicted in Fig. 2-A, where the input is assumed ranging in $[-1, 1]$.

Intriguing enough, this function can also be used for designing a system belonging to the class known as Piece-wise Affine Markov (PWAM) chaotic maps.

Generally speaking, chaotic maps are 1D discrete-time autonomous systems, whose evolution is described as $x_{n+1} = M(x_n)$ where $M$ is a proper function mapping an interval $I$ into itself. A deep and exhaustive analysis on chaotic maps, as well as a description of PWAM maps, can be found in [6]. Here is enough to recognize that the system modified as in Fig. 1 can be analyzed exactly with the same tools used to analyze chaotic maps. In fact the evolution of a system composed by $k-1$ identical pipeline ADC stages can be described as, assuming all stages has the same input/output relationship $M$:

$$\begin{cases} v_{n+1}^{(0)} = M\left(v_n^{(k-2)}\right) \\ v_{n+1}^{(1)} = M\left(v_n^{(0)}\right) \\ \cdots \\ v_{n+1}^{(k-2)} = M\left(v_n^{(k-3)}\right) \end{cases} \tag{1}$$

[1] This is used to reduce the latency of the whole conversion since every stage provides its output with a delay of half clock cycle.

This is the author's version of the article that has been presented at IEEE ESSCIRC2006
The editorial version of the paper is available at http://dx.doi.org/10.1109/ESSCIR.2006.307548

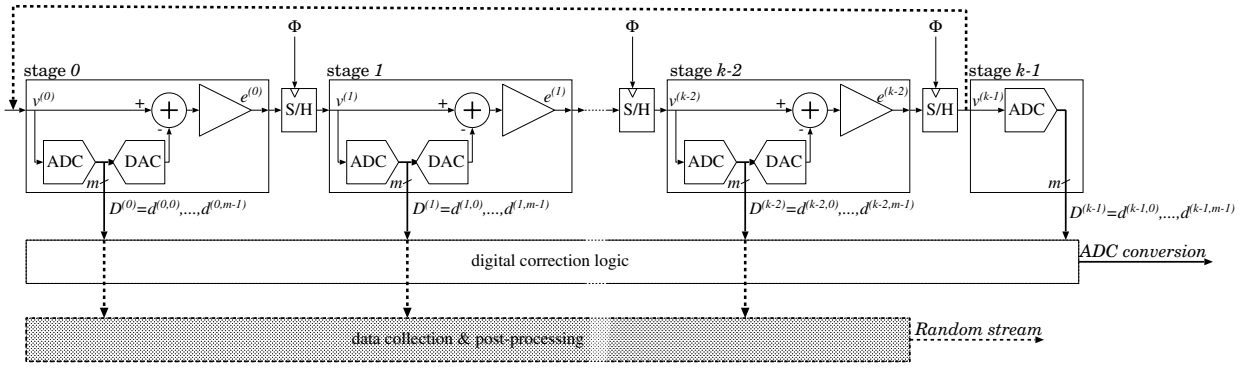


Fig. 1. Basic structure of a pipeline ADC and modifications proposed *(dashed elements)* to obtain a TRNG.

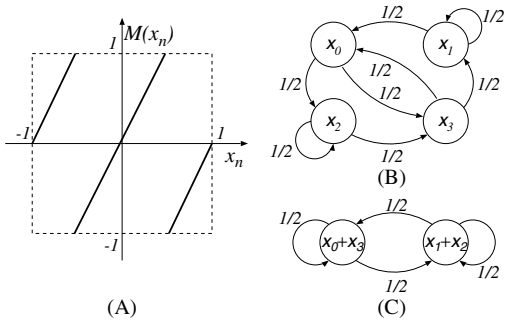


Fig. 2. (A) PWAM map that can be found in 1.5 bit ADC pipeline converter stage; (B) Markov chain associated to the PWAM; and (C) simplified chain.

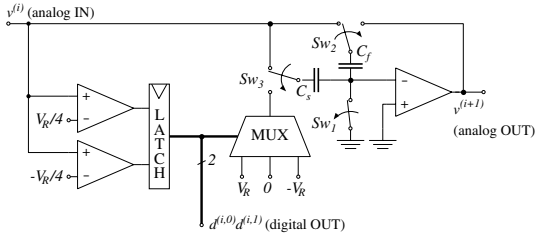


Fig. 3. One and half bit ADC stage implemented in the prototype.

It is easy to notice that the system (1) is perfectly equivalent to a system composed by $k-1$ independent chaotic maps $x_{n+1}^{(i)} = M\left(x_n^{(i)}\right)$. Moreover, since the map associated to the $M$ function is a PWAM map, its evolution can be studied through a Markov chain [6]. If we consider the PWAM map in Fig. 2 the associated Markov chain, obtained considering the transitions of the map state $x_n$ in the four intervals $\{X_0, X_1, X_2, X_3\} = \{[-1, -1/2[\,, [-1/2, 0[\,, [0, 1/2[\,, [1/2, 1]\}$, we get the chain in Fig. 2-B. Intriguing, if we simply *aggregate* the states of the graph two by two, we get the chain of Fig. 2-C, which is exactly the chain describing a random bit generator. Note that implementing a circuit described by the chain of Fig. 2-C means effectively implementing a *Random Bit Generator*.

To recognize in which of the two macro-states of Fig. 2-C the system is, it is enough to look at the output of the coarse ADC. So no additional analog hardware is required in order to implement this Markov chain (i.e. a RNG) from an ADC stage; what we need to get a RNG from the closed pipeline is only to re-elaborate the digital output of all stages, i.e. to substitute the digital correction logic.

In addition we can introduce also a digital *post-processing* stage. It consists of an algorithm (typically, based on a very simple hardware) which elaborates the output bit-stream in order to reduce the residual correlation, if still present. A post-processing is always used in TRNGs to *improve* the quality of the output stream; the most widely used is probably the Von Nuemann post-processing [11], that has the property to remove any biasing in the bit-stream.

## III. Circuit Implementation

We designed and fabricated a prototype of this TRNG in AMS 0.35 $\mu$m 3.3 V CMOS technology. Following the idea described in the previous section, we based the core of the circuit is the 1.5 bit A/D cell whose schematic is shown in Fig. 3 [9]. While a single-ended configuration is shown for simplicity, the actual implementation is fully-differential.

Assuming a two-phase clock, its behavior can be described as follows. In the first phase the capacitors $C_s$ and $C_f$ sample the input voltage, while in the second the output voltage is being computed. This is particularly important, since it allow us to directly connect the input of a stages to the output of the previous one if the two stages work on the two different phases of the clock. This reflects in a very important simplification of the pipeline, since the S/H stages usually required to separate different stages in a pipeline are no more necessary.

The two comparator indicate the interval in which the map state is; to perform the state aggregation necessary to obtain the Markov map of Fig. 2-C it is enough to xor the two bits $d^{(i,0)}$ and $d^{(i,1)}$.

Every stage produces 1 random bit every clock cycle, half of them in the first stage, half in the opposite phase of the clock. The throughput of the system with $l-1$ stages is so $l-1$ bits per clock cycle.

We designed our prototype including two pipelines. The first one is composed by two stages and is intended for testing the correct behavior of the chaotic map, while the second one is composed by eight stages and its purpose is to work as a random bit generator. They are biased by two different biasing circuits to avoid interferences. The layout of the chip is shown in Fig. 4 while the circuit characteristics are reported in Table I. The circuit has been designed to work with a clock up to 5 MHz; this means that the circuit maximum output data rate is up to 40 Mbit/s for the eight-stages pipeline.

In the prototype no post-processing stages have been included; this because it was intended to test only the analog core. The post processing stages however have been considered only during the test phase, processing off-line all the acquired sequences before testing them.

## IV. Test Results

Even if it is possible to theoretically prove that the proposed architecture is capable of generate independent and identically distributed

This is the author's version of the article that has been presented at IEEE ESSCIRC2006
The editorial version of the paper is available at http://dx.doi.org/10.1109/ESSCIR.2006.307548

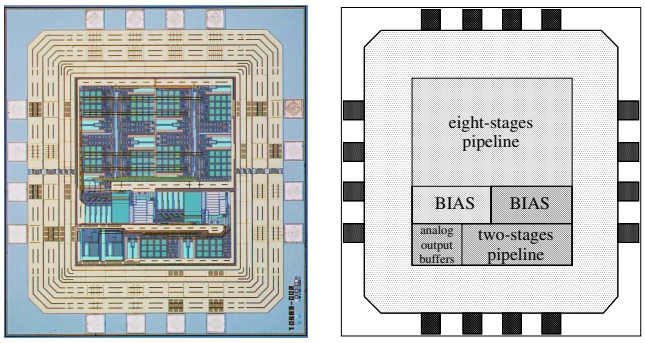


Fig. 4. Die microphotograph and floor plan of the designed ADC-based RNG.

| | |
|---|---|
| **Working frequency**: | 5 MHz |
| **Data throughput**: | 5 Mbit/s per stage |
| *(two-stages pipeline)*: | 10 Mbit/s |
| *(eight-stages pipeline)*: | 40 Mbit/s |
| **Area** (with pads): | 2.400 mm$^2$ |
| | (1480 $\mu$m x 1620 $\mu$m) |
| **Area** (without pads): | 0.752 mm$^2$ |
| *(two-stages pipeline)*: | 0.234 mm$^2$ |
| *(eight-stages pipeline)*: | 0.518 mm$^2$ |
| **Power supply voltage**: | 3.3 V |
| **Power consumption**: | 56 mW |
| *(two-stages pipeline)*: | 27 mW |
| *(eight-stages pipeline)*: | 29 mW |

TABLE I
CIRCUIT CHARACTERISTIC FOR THE DESIGNED ADC-BASED RNG.

symbols, a real implementation could be quite far from the ideal one, due to unavoidable implementation errors. For this reason, it is important to accurately test the real implementation of the circuit.

The most natural idea for checking the ideality of a RNG would be compute the entropy of the generated sequences. According to Shannon [12] the entropy $H$ measured in *bit* of any message is:

$$H = -\sum_{i=0}^{n-1} p_i \log_2 p_i \tag{2}$$

where $p_i$ is the probability of state $i$ out of $n$ possible states. In case of a random bit generator that produces $k$-bits binary sequences, the possible sequences are $n = 2^k$ and every sequence should have the same probability, i.e. $p_i = 2^{-k}$. Thus, for a *perfect* random bit generator, the associated entropy is $k$ bits, i.e. the number of generated bits.

However, this is only a stochastic definition. To get a measurement of the entropy of a system, we have to assume that the system is ergodic, so we can measure the frequency of all $2^k$ sequences and consider it as their probability, and calculate the entropy. The closer the system to the ideal one, the closer the entropy to $k$ bits. Also, the value of $k$ should be as large as possible, since an ideal random number generator has to produce *infinite-length* sequences. It is obvious that this approach is not possible.

In order to overcome the impasse, some statistical test has been developed to analyze a sequence of bit and evaluate if it can be considered random. We presents results from NIST SP 800-22 test suite [10] which is, to the best of the authors' knowledge, the most reliable test suite currently available. The code we used for the testing comes directly from NIST website [13] and is the latest version available the moment (version 1.8). We also have considered the FIPS 140.2 test suite [14] from NIST, and Marsaglia's DieHard test suite [15]; results are very similar and they are not reported here.

To test our RNG we have acquired sequences of bits generated at different speed, post-processed them and tested with the suite. We

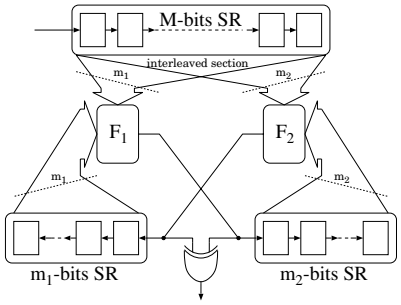


Fig. 5. Block scheme of the finite-state machine used for the non linear shift register post-processing.

have considered three different post-processing techniques:

- **Von Neumann post processing.** This post processing has been introduced by Von Neumann in 1951 [11]. This technique consist of converting the bit pair '01' into the output 0, the bit pair '10' into the output 1, and of discarding bit pairs '00' and '11'. The expected decimation is 1:4.
- **Parity-based post processing.** The output of the system is computed as the parity of non-overlapping string of $n$ bit, resulting in a decimation of 1:$n$. It is also known as Xor post-processing. We have considered a depth equal to 4 to have the same decimation as in the Von Neumann post processing.
- **Non-linear shift register post processing.** This scheme operates at no throughput loss. The block diagram of the overall system can be found in Figure 5 and it is a simplification of the SHA algorithm. For a more detailed discussion about this scheme, see [16].

Here we present results for the sequences obtained by the 8 stages pipeline, working at the nominal speed of 5 MHz, with a throughput of 40 Mbit/s. The length of the analyzed sequences has been set, after the post processing stage, to 1 Mbit.

The NIST suite is composed of 15 tests. Each test has to be interpreted in a statistical way [10], i.e. it analyzes a binary sequence and gives a probability (called P-value) that the sequence has been randomly generated. A P-value equal to 1 means that the sequence has the maximum probability to be generated by an *ideal* generator, a value near to 0 means that it is very unlikely, though still possible, that the sequence is effectively random.

Due to this reason, we have to be very careful when looking at randomness tests results. A single P-value can hardly be considered indicative for a generator; slightly better, and this is the most common approach to test physical RNGs, is to consider all P-values from all different tests in a suite.

In this paper we consider a completely different approach. If considering and testing a single sequences is not enough to make a good characterization of a RNG, we can repeat the test several times on different sequences, and check the overall results. The higher the number of sequences tested, the higher the accuracy of the characterization. So, we acquired 5000 different sequences and we analyzed all of then, comparing the results obtained with the results expected when testing several true random sequences

As a first test (*first level test*, since we consider each sequence stand-alone), we can check the probability that a single sequences has to pass the test, simply computing the ratio of sequences effectively passing the tests. According to NIST, we can say that a test is passed if we get a P-value greater than a chosen significance level $\alpha$, which NIST suggests to take equal to $\alpha = 0.01$. When analyzing a large number of sequences, due to a small but non-zero probability that sequences generated by an ideal random generator have to fail the test, we have an expected ratio of "good" sequences equal to $1 - \alpha = 0.99$. Since we analyze only a finite number of sequences,

This is the author's version of the article that has been presented at IEEE ESSCIRC2006
The editorial version of the paper is available at http://dx.doi.org/10.1109/ESSCIR.2006.307548

| | *Ratio of sequences with P-values greater than alpha* $= 0.01$ | | | $\chi^2$ *test on the uniformly distribution of P-values* | | |
|---|---|---|---|---|---|---|
| SP800-22 test | NEUM | XOR4 | NLSR | NEUM | XOR4 | NLSR |
| Frequency | **0.000000** | 0.991000 | 0.988200 | **0.000000** | 0.650296 | 0.138401 |
| Block Frequency | **0.434000** | 0.990400 | 0.987600 | **0.000000** | 0.131124 | 0.661143 |
| Cumulative Sums | **0.000000** | 0.990600 | 0.989000 | **0.000000** | 0.741332 | 0.063258 |
| Runs | **0.000000** | 0.990000 | 0.991600 | **0.000000** | 0.849924 | 0.139410 |
| Longest Run of Ones | **0.000000** | 0.988400 | 0.990000 | **0.000000** | 0.774201 | 0.950563 |
| Matrix Rank | 0.989600 | 0.988400 | 0.991600 | 0.273925 | 0.135018 | 0.700255 |
| Spectral (DFT) | **0.984400** | 0.987000 | 0.988800 | **0.000024** | 0.435680 | 0.041253 |
| NOT Matching | **0.000000** | 0.990600 | 0.990800 | **0.000000** | 0.931229 | 0.715335 |
| OT Matching | **0.000000** | 0.989200 | 0.987000 | **0.000000** | 0.131895 | 0.067738 |
| Universal | **0.082200** | 0.985800 | 0.985800 | **0.000000** | 0.106533 | 0.202878 |
| Average Entropy | **0.000000** | 0.989200 | 0.987200 | **0.000000** | 0.036178 | 0.087978 |
| Random Excursion | N/A | 0.989780 | 0.990542 | N/A | 0.299634 | 0.986286 |
| Random Exc. Variant | N/A | 0.993612 | 0.993380 | N/A | 0.616292 | 0.216608 |
| Serial | **0.985400** | 0.989800 | 0.987200 | 0.042983 | 0.321094 | 0.297790 |
| Linear Complexity | 0.989000 | 0.986200 | 0.988600 | 0.643672 | 0.480524 | 0.994013 |

TABLE II

RESULTS OF RANDOMNESS TEST FOR THE EIGHT-STAGES PIPELINE OF THE CHAOTIC-RNG RUNNING AT 40 MBIT/S.

we may expect some deviation from this number. If we adopt the three-$\sigma$ criterion, we say that the number of passing chunks is compatible with the randomness of the source if it lies in the range $1-\alpha \pm 3\sqrt{\alpha\left(1-\alpha\right)/T}$, where T is the number of tested strings. Since in our case $T = 5000$, the acceptable range is the interval $[0.9857, 0.9942]$.

Here we present also a *second level test*, involving the results of the tests on all sequences together. Since the P-value for a RNG is a random variable that has to be uniformly distributed in $[0, 1]$, we can check if the P-values obtained from the tests can be considered as coming from an uniform distribution. We performed a $\chi^2$ test over 16 bins; the result of this test is another probability value that can be considered as a *second level* P-value. As in the first level test, we can chose another significance level $\alpha'$, and consider the test passed if this second level P-value is greater than $\alpha'$. We can consider, as in the previous case, $\alpha' = 0.01$.

Results are shown in table II. In bold are indicated the tests whose results are not in the expected range.

As can we see, a part from the Von Neumann post processing, which ah shown to be inappropriate for this generator, with all other post processing techniques results are in the expected range for all tests in the suite, i.e. the generator in combination with these post processing can be effectively considered as a *true* RNG. The characterization we have presented here is not based just on the test of a single sequence, but based on the tests of thousands of sequences, providing results that are high reliable.

The other figure of merit in a RNG is the throughput. Even applying an Xor-4 post-processing, which has a decimation ratio of 1:4, the throughput is still 10 Mbit/s. This is a considerably high speed, especially when compared the the other generators available. An high-end quantic RNG like the one described in [5], which is one of the fastest available, achieves a throughput of 4 Mbit/s with a much higher power consumption. If we instead compare our performance with other silicon-based architecture, we get very different results. For example the Intel TRNG [3], which is base on Johnson Thermal noise and on Von Neumann post processing, achieves only an average speed of 75 Kbit/sec. VIA declares [4] that its generator can achieve a data-rate from 4 to 9 Mbits/sec, but with a much smaller entropy rate. However, no results from testing is presented in both cases.

## V. CONCLUSION

By exploiting the statistical approach to the study of non-linear dynamics, we are able to recognize that the architecture used for common pipeline ADCs can be reused for designing a chaotic circuit very appealing for the generation of random numbers.

Following this approach we have designed a prototype of a True-Random Number Generator in 0.35 $\mu$m CMOS technology capable of generating up to 40 Mbit/sec. We tested our prototype with the most advanced tests for randomness available considering few very simple post processing stages.

The results of the tests, as well as the quality of the testing, that has not involved a single sequence but thousands of them, indicate that the proposed circuit can be considered a high quality TRNG, suitable for the most advanced security-related applications.

## REFERENCES

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone *Handbook of Applied Cryptography* CRC Press, 1996.
[2] D. E. Eastlake, S. D. Crocker, and J. I. Shiller, "RFC 1750: Randomness recommendation for security" in *Internet Society Request for Comments*, Internet Engineering Task Force, 1994
[3] B. Jun and P. Kocher, "The Intel Random Number Generator", Crypt. Reasearch, Inc. Apr. 1999. Available at URL: http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf
[4] "Evaluation of VIA C3 Nehemiah Random number generator", February 2003. Available at URL: www.cryptography.com/resources/whitepapers/VIA_rng.pdf
[5] idQuantique, "Random Numbers Generation using Quantum Physics" White paper, 2004. Available at URL: http://www.idquantique.com/products/files/quantis-whitepaper.pdf
[6] G. Setti, G. Mazzini, R. Rovatti, and S. Callegari, "Statistical modeling of discrete time chaotic processes: Basic finite dimensional tools and applications", in *Proceedings of IEEE*, vol. 90, no. 5, May 2002.
[7] S.M. Ulam and J. Von Neumann, "On combination of stochastic and deterministic process", Bull. Amer. Math. Soc 53, 1947.
[8] S. Callegari, R. Rovatti, and G. Setti, "Embeddable ADC-Based True Random Number Generator for Cryptographic Applications Exploiting Nonlinear Signal Processing and Chaos", in "IEEE Transaction on Signal Processing", vol. 53, no. 2, pp. 793-805, Feb. 2005.
[9] T. B. Cho, and P. R. Gray, "A 10 b, 20 Msample/s, 35mW Pipeline A/D Converter", in *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 166-172, Mar. 1995.
[10] "A statistical test suite for random and pseudorandom number generators for cryptographic applications", Special publication 800-22, National Institute for Standards and Technology (NIST), May 2001.
[11] J. Von Neumann, "Various Technique Used in Connection with Random Digits", in *Applied Math Series*, notes by G. E. Forsythe, National Bureau of Standards, 12, pp. 36-38, 1951.
[12] C. E. Shannon, "A Mathematical Theory of Communication", The Bell system technical journal, vol. 27, pp.379-423, July 1948.
[13] National Institute of Standard and Technology, "Random Number Generation and Testing", available at URL http://csrc.nist.gov/rng/
[14] "Security Requirements for Cryptographic Modules", Federal Information Processing Standard (FIPS) publication 140-2, National Institute for Standards and Technology (NIST), May 2001.
[15] G. Marsaglia, The Marsaglia Random Number CD–ROM including the DieHard Battery of test of randomness available at URL http://stat.fsu.edu/pub/diehard/
[16] S. Poli, S. Callegari, R. Rovatti, G. Setti, "Post-Processing of data generated by a chaotic pipelined ADC for the robust generation of perfectly random bitstreams", in *Proceedings of ISCAS*, vol. IV, pp. 585-588, Vancouver, May 2004.