*Journal of* ELECTRICAL ENGINEERING, VOL. 57, NO. 4, 2006, 218–225

# CHAOS–BASED TRUE RANDOM NUMBER GENERATOR EMBEDDED IN A MIXED–SIGNAL RECONFIGURABLE HARDWARE

**Miloš Drutarovský — Pavol Galajda** [*]

The paper presents a chaos-based True Random Number Generator (TRNG) implemented in commercially available mixed-signal PSoC reconfigurable devices without any external components. Contrary to the traditionally used sources of randomness (*eg* various "well-behaved" analog noise sources) it uses well-defined deterministic analog circuit that exhibits chaos. A new simple method of mapping the deterministic chaos into the switched capacitor based mixed-signal PSoC devices is proposed. The design is optimized for reduction of influence of circuit non-idealities to the quality of generated random bit stream. The influence of circuit non-idealities is significantly reduced by the proposed XOR corrector and optimized circuit topology. The high quality of generated true random numbers is confirmed by passing standard NIST statistical tests.

K e y w o r d s: cryptography, chaos, Markov chains, PSoC mixed-signal array, statistical tests, NIST test suite

## 1 INTRODUCTION

Random number generators represent basic cryptographic primitives. They are widely used for example as confidential key generators for symmetric key and public-key crypto-systems (*eg* RSA-moduli) and as password sources. In some algorithms (*eg* DSA) or protocols (*eg* zero-knowledge), random numbers are intrinsic to the computation [1]. In all these applications, security depends greatly on the randomness of the source. Because security algorithms and protocols rely on the unpredictability of the random bits they use, True Random Number Generators (TRNGs) play an important role in cryptographic applications.

Classical TRNG uses some random physical phenomenon [2]. Currently the most frequently used phenomenon in embedded TRNGs is a jitter noise of digital clock signals (see *eg* [3], [4]). Embedded TRNGs frequently use some external devices [5] or rather heuristic source of randomness [6]. Although these drawbacks can be overcomed by a design of proper custom circuits, randomness extraction is still a big challenge in the designs based on of the shelf devices as FPGA [7], [8], [9] or general microprocessors [10].

Chaotic circuits represent an efficient alternative to the classical TRNGs. Contrary to traditionally used sources of randomness (*eg* various "well-behaved" analog noise sources like jitter noise, thermal or shoot noise, *etc.*) they use a well-defined analog deterministic circuit that exhibits chaos. Chaotic systems are characterized by a "sensitive dependence on initial conditions", *ie* a small perturbation eventually causes a large change in the state of the system [11]. However, the slightest uncertainty about the initial state (which is unavoidable in all analogue implementations) leads to a very large uncertainty after some time. With such initial uncertainties, the system's behavior can be predicted only for a short time period. Additionally as it has been recently shown, if the state variable is not available to the observer, and the system is well designed, the output of the system cannot be predicted at all [12]. Such implementation is a source of infinite entropy that is absolutely required for good TRNG.

Many random number generators based on analog and deterministic chaotic phenomena have been proposed, see *eg* [13], [14], [15] and references in [12]. Some of them have been simulated only. Others have not been sufficiently optimized for cryptographic applications (they provide certain bias or other deviations). This paper describes a practical implementation of recently proposed deterministic chaos circuit based on Markov map [11], [12], [16]. In contrary with Field Programmable Analog Array (FPAA) implementation [16], proposed chaos based TRNG implementation [17] uses mixed-signal PSoC reconfigurable hardware [18], [19]. It is shown that after suitable modification of originally proposed Markov map it can be easily embedded in the selected PSoC hardware. Although proposed realization provides much lower output speeds than pure FPAA implementation [16], used mixed-signal hardware includes also embedded microcontroller. Such hardware can be in principle used for other cryptographic tasks. Moreover, special attention is devoted to the identification of analog circuit non-idealities. The design is optimized for reduction of their influence to the quality of generated random bit stream. By using NIST statistical tests, it is demonstrated that proposed implementation provide very good quality of generated bit stream.

* Department of Electronics and Multimedia Communications, Technical University of Košice, Park Komenskho 13, 04120 Košice, Slovakia, E-mail: Milos.Drutarovsky, Pavol.Galajda@tuke.sk

ISSN 1335-3632 © 2006 FEI STU

The paper is organized as follows. A brief overview of the theory of Markov chaotic sources is given in Section 2. In Section 3, a new method of mapping chaos based TRNG into PSoC devices is presented. The experimental TRNG hardware used to test the proposed method is described in Section 4. In Section 5, statistical evaluations of internal and output TRNG signals are made. Finally, concluding remarks are presented in Section 6.

## 2 AN OVERVIEW OF THE THEORY OF MARKOV CHAOTIC SOURCES

An ideal Random Number Generator (RNG) is a discrete memory-less information source that generates equiprobable symbols. We consider an ideal state chain (Fig. 1), which corresponds to the ideal RNG.

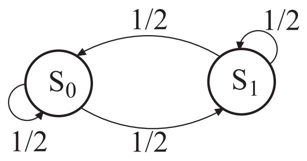


**Fig. 1.** State chain of an ideal RNG

The system is in the state $S_0$ when an output of the RNG has been "0" and in the state $S_1$ when an output has been "1". A RNG output has always the same probability (1/2) to get the system in either state, $S_0$ or $S_1$. It is known that these sources can be build up from Piece-Wise Affine (PWA) Markov chaotic maps [12]. These maps are one dimensional, discrete- time, in which the state variable $x(n)$ is computed as

$$x(n+1) = M[x(n)] \tag{1}$$

where $M : [-1,1] \to [-1,1]$, and $x(0)$ is the initial condition. We focus on the recently proposed piece-wise chaotic map [12], [16]

$$M(x) = (2x+1) \bmod 2 - 1 \tag{2}$$

which is plotted in Fig. 2 together with its Markov partition $X_0 = [-1, -1/2]$, $X_1 = [-1/2, 0]$, $X_2 = [0, 1/2]$ and $X_3 = [1/2, 1]$. The map $M(x)$ can be equivalently expressed as

$$x(n+1) = \begin{cases} 2x(n) - 2 & \text{for } x > 1/2 \\ 2x(n) & \text{for } x < -1/2 < 1/2 \\ 2x(n) + 2 & \text{for } x < -1/2 \end{cases} \tag{3}$$

The main advantage of this chaotic map is its increased robustness in the case of an analogue implementation [12], [16]. The dynamics of equation (1) can be represented by the Markov chain and that the evolution can be studied through a square matrix (often referred to as the kneading matrix ) defined for the map (3) in Fig. 2 [11], [12].

$$K = \begin{pmatrix} 0 & 0 & 1/2 & 1/2 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix}$$

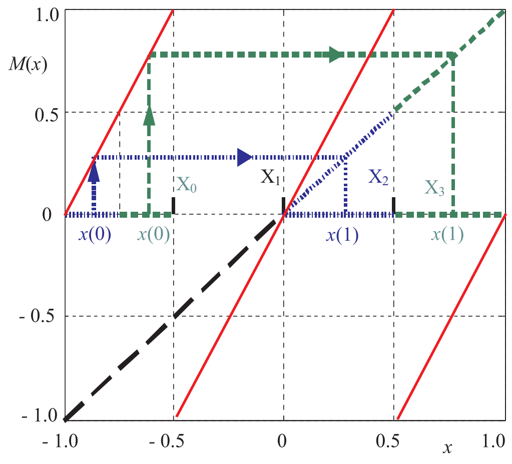


**Fig. 2.** Proposed piece-wise chaotic map based on (3) and its kneading matrix $K$

Each partition set $X_i$, $i = 0,1,2,3$ is associated with a Markov state $x_i$, so that $x(n) \in X_i \Leftrightarrow \mathrm{x(n)} = \mathrm{x}_i$ (*ie*, $\mathrm{x}_i$ is a quantization of $x_i$ ), and each entry $K_{ij}$ is the conditional probability $Pr(x_{n+1} = j | x_n = i)$. The entry in the $i$-th row and $j$-th column of matrix $K$ corresponds to the fraction of $X_i$ that is mapped into $X_j$. From this it easily follows that the sum of the rows entries of any kneading matrix is always equal to 1.

It is interesting to try to give a meaning to the individual entries in matrix $K$. We suppose that it is known only that the initial condition $x(0)$ falls in $X_0$ (as it is shown in Fig. 2). The state variable in the next time step $x(1)$ obviously falls either in $X_2$ with probability 1/2, or in $X_3$ with the same probability (1/2), but not in $X_0$, nor $X_1$.

In other words, the entry $K_{ij}$ of $K$ represents the probability by which a trajectory starting in $X_i$ falls in $X_j$ at the next time step. This iterative process can be interpreted as a Markov state chain with 4 states (Fig. 3). The state 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 state machine changes from a one state to another.

The chain in Fig. 3 is not suitable for a direct realization of the ideal RNG since it is sequential (it has a memory). However, it is possible to easily build a rigorously independent binary sequence from $x(n)$ [12]. In fact, it is sufficient to aggregate the Markov states into two macro-states $S_0$ and $S_1$ shown with dotted lines in Fig. 3. Note, that this aggregation is different from that introduced in [12], [16] for the purpose of a better implementation in PSoC devices as it will be described in the

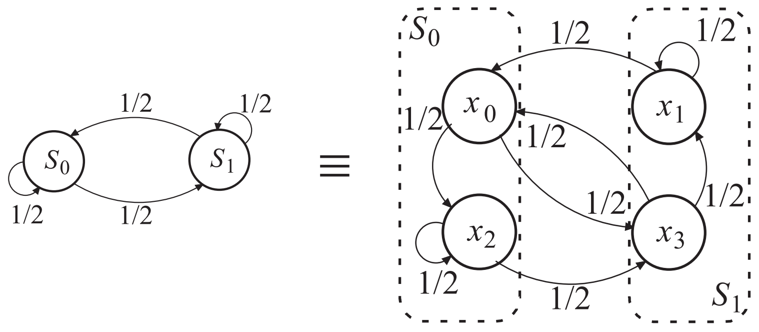


**Fig. 3.** Markov chain of the RNG

next section. It can be easily shown that the resulting diagram is also identical to the state chain of the ideal RNG shown in Fig. 1.

## 3 PSOC BASED TRNG

### 3.1 An Overview of PSoC Architecture

The PSoC device family [18] consists of a mixed-signal array with an on-chip CPU. A PSoC device includes configurable Analog Blocks (AB) [19] and Digital Blocks (DB), as well as programmable interconnections. This versatile architecture allows the user to create customized peripheral configurations to match the requirements of each individual application. Additionally, a fast CPU, flash program memory, SRAM data memory, and configurable IO interfaces are included in a PSoC device. The basic features of the device are as follows:

- Powerful Harvard CPU architecture
- Advanced peripherals (PSoC Blocks):
  4 rail-to-rail continuous analog PSoC blocks
  8 Switched Capacitor (SC) analog blocks
  8 digital PSoC blocks
- Programmable references voltage $V_{ref} = 1.3$ V (set by internal bandgap reference) or $V_{ref} = V_{cc}/2$
- Internal 24 MHz oscillator (allows to build a real single chip application)
- Precision, programmable clocking (provides two phase clocks – $\Phi_1$, $\Phi_2$ for SC)
- Flexible On-Chip Memory
- Programmable pin configurations

The discussed architecture is quite versatile. It allows for many different functions merely by altering the internal circuits switches. The on-chip CPU controls the functionality of the digital and analog blocks and it can dynamically change the parameters (*eg* gain of SC block can be set by specifying $C_A$ and $C_F$ capacitors shown in Fig. 4) and topology of these blocks [18], [19].

### 3.2 TRNG Architecture

As shown in [12], the chaotic map in Fig. 2 can be implemented by a pipeline-ADC with 1.5-bit/stage architecture. Here, we introduce different implementations of the chaotic map (3) by using four SC (4-SC) analog PSoC blocks. The equation (3) can be rewritten according to Tab. 1, where $b(n+1)$ is a binary output of the TRNG at time step $n+1$. The steps given by equations in Tab. 1 can be directly mapped into four PSoC SC blocks shown in Fig. 4. Addition/subtraction of the reference voltage and comparison operations from steps one and two are mapped to SC1 and SC3, respectively. Multiplication by 2.0 from step three is realized in SC3 and SC4. Comparison from step three is implemented in SC3. We have chosen a common SC implementation operating on two-phase clocks ($\Phi_1$ and $\Phi_2$ driven by the on-chip oscillator). During the first phase, block SC1 adds the reference voltage $\pm V_{ref}$ (actual polarity is controlled by on-chip CPU) to the input voltage $V_{in}$ according to

**Table 1.** TRNG chaotic map equations optimized for PSoC SC blocks

| 1st step | 2nd step |
|---|---|
| $x'(n) = \begin{cases} x(n) - V_{ref} \text{ for } x(n) > 0 \\ x(n) + V_{ref} \text{ for } x(n) < 0 \end{cases}$ | $x''(n) = \begin{cases} x'(n) - V_{ref} \text{ for } x'(n) > 0 \\ x'(n) + V_{ref} \text{ for } x'(n) < 0 \end{cases}$ |
| **3rd step** | |
| $x(n+1) = 2x''(n) \text{ for all } n$ | $b(n+1) = \begin{cases} \text{"1" for } x(n+1) > 0 \\ \text{"0" for } x(n+1) < 0 \end{cases}$ |

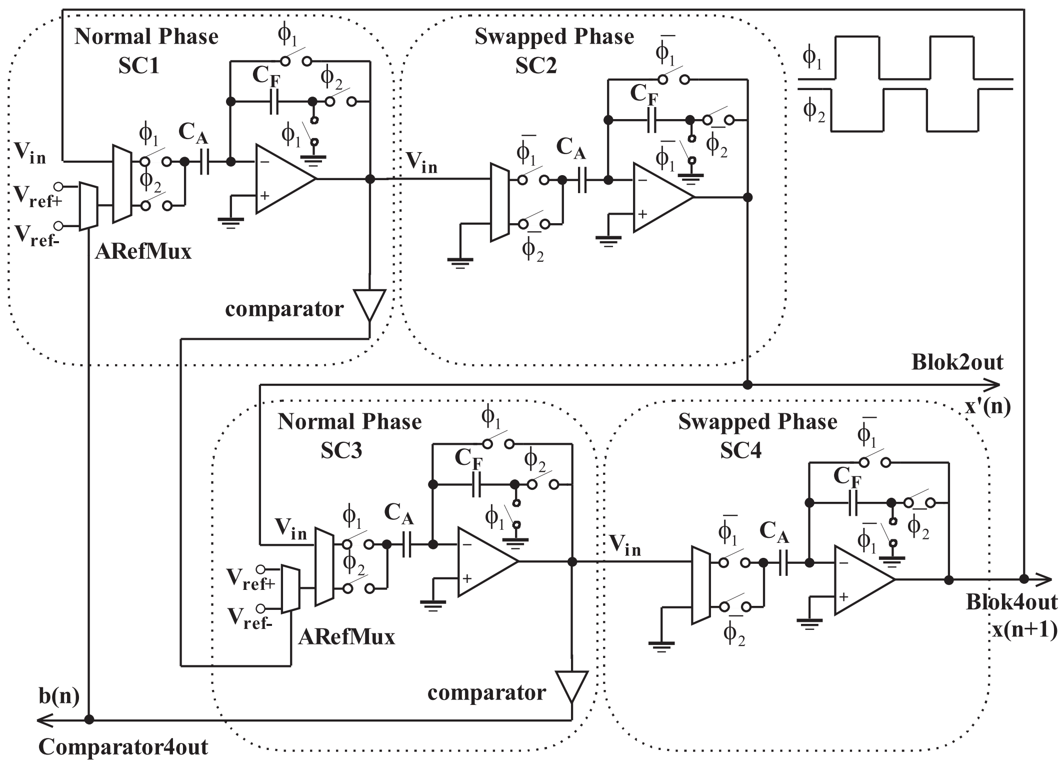


**Fig. 4.** PSoC implementation of the 4-SC blocks TRNG

the polarity of the analogue voltage at the output of the block SC3. If this voltage is negative, the reference voltage $V_{ref+} = +V_{ref} = 1.3$ V, will be added in the next clock phase to the input voltage, otherwise the voltage $V_{ref-} = -V_{ref} = -1.3$ V must be added. During the second clock phase, the block SC1 tests the polarity of the output voltage for the block SC3. The operation of the block SC3 is similar to that of the SC1.

Blocks SC2 and SC4 realize the Sample and Hold function between blocks SC1 and SC3. This is needed for a correct operation of SC blocks connected as a ring oscillator. Additionally, blocks SC3 and SC4 define the gain of the circuit, which has to be as close as possible to the ideal gain equal to 2.0. In our case the gain is limited by discrete values of capacitors $C_F$ and $C_A$ to the value

$$(27 \times 19)/(16 \times 16) = 513/256 \doteq 2.004 \tag{4}$$

It can be shown that the structure in Fig. 4 processes two independent bit streams, $b_1(n)$ and $b_2(n)$ in a pipeline processing structure. The bit streams are read out sequentially from this structure and they are overlapped according to the equations

$$\begin{aligned} b_1(n) &= b(2n), & n &= 0,1,\dots \\ b_2(n) &= b(2n+1), & n &= 0,1,\dots \end{aligned} \tag{5}$$

Properties of the proposed 4-SC TRNG:

- it uses voltage range from AGND$= V_{cc}/2$ to $\pm 2V_{ref}$ ($x(n)$ is actually between and 5V, with a low distortion in power supply boundary voltages),
- there exist two independent bit streams (this fact can be used for effective post-processing, as will be shown in Section 3.5),
- it requires relatively simple control that must be performed by the on-chip CPU (only one analog comparator interrupt is used), the rate of the output TRNG data is limited by the speed of the on-chip CPU and can be up to 60 Kbit/s
- it uses no external devices.

## 3.3 Simulation of a possible TRNG deviation

A deviation of the SC gain from the optimal value equal to 2.0 can cause an increased autocorrelation of the generated data, as it is shown in Fig. 5 for the map (3) with the nominal gain 513/256. Moreover, additional gain deviations can be caused by deviations of internal capacitors (according to [18], SC components have 0.1% tolerances). Autocorrelation can be computed by standard equations [20]:

$$\begin{aligned} &\operatorname{corr}(b(k)) = \operatorname{corr}(b(n), b(n-k)) = \\ &= \frac{E\left[\{b(n) - E\left[b(n)\right]\}\{b)n-k) - E\left[b(n-k)\right]\}\right]}{\sqrt{\operatorname{var}(b(n))\operatorname{var}(b(n-k))}} \end{aligned} \tag{6}$$

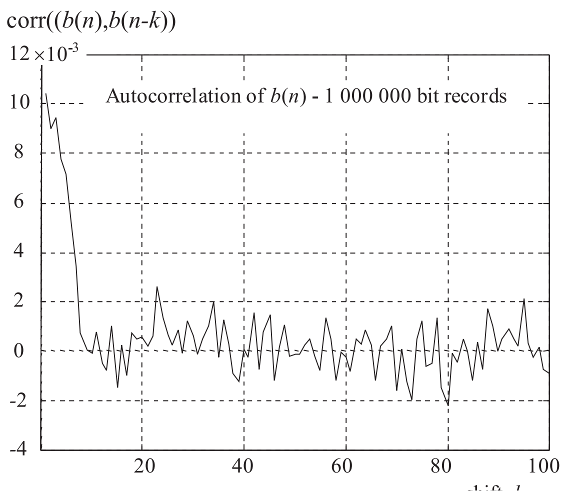


**Fig. 5.** Simulation of autocorrelation of generated binary data for the map (3) with the gain 513/256. The values were evaluated for $L = 1\,000\,000$ bits. Estimated mean value of the generated binary data is $E[b] \approx 0.4993$ and maximal value of the autocorrelation is $\rho = \max\{\text{corr}\,(k),\ k = 1, 2 \dots, 100\} \approx 0.011$

where variation in the case of stationary (and ergodic) random bits can be computed as

$$\text{var}\,(b(n)) = \text{var}\,(b) = E\left[\{b - E[b]\}^2\right] = E[b]\{1 - E[b]\} \tag{7}$$

and mean value $E[b] = Pr(b = 1)$ can be estimated by long run time average. Correlations given by (6) are always between -1 and 1. When two variables are statistically independent, the correlation of (6) is zero. Since we can estimate (6) and (7) only for data records with the limited length $L$, we can expect small deviation from zero. It can be shown (see [1] page 182 combined with [20]) that for random variables with $E[b] \approx 1/2$ (that is guaranteed by the map symmetry) and time averages with sufficiently large length $L$, the test value

$$T(k) = \text{corr}\,(k)\sqrt(L) \tag{8}$$

approximately follows an $N(0.1)$ distribution. This property will be used in the following sections for testing TRNG deviations.

Although the deviation is relatively small, it can cause failures of many statistical tests used in cryptography and therefore an additional corrector must correct it.

## 3.4 Analysis of XOR corrector performance

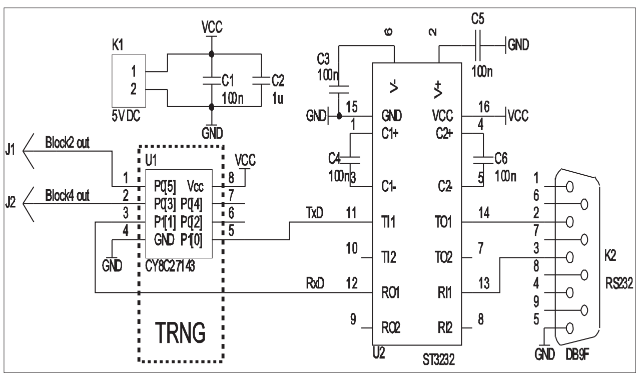
The exclusive or (XOR) operation is commonly used to reduce the bias from the bits generated by hardware random number generators [20]. Typically, the uncorrected bits generated by a TRNG will have mean value different from the ideal value of 1/2. A XOR corrector can reduce a bias (difference from the value 1/2) by XOR-ing two or more independent bits followed by decimation.

The bias of XOR corrected sequence is further decreased and can be computed as (Eq. (3) in [20])

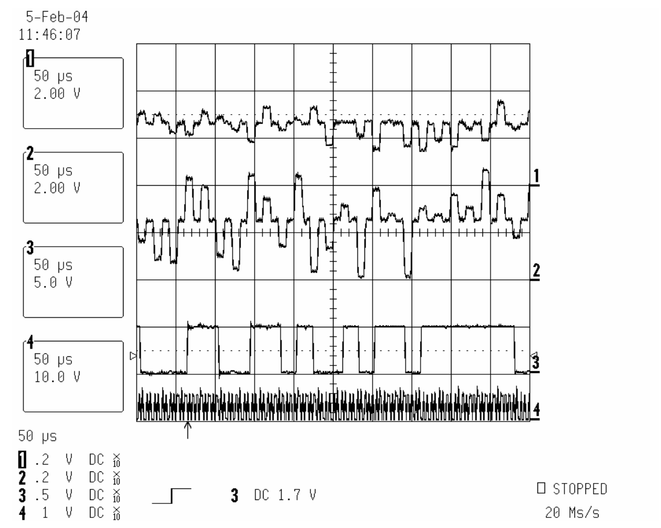
$$E\left[b_1 \oplus b_2\right] = \frac{1}{2} - 2\big(E[b_1] - \frac{1}{2}\big)\big(E[b_2] - \frac{1}{2}\big) \tag{9}$$

$2\big(E[b_1]-\frac{1}{2}\big)\big(E[b_2]-\frac{1}{2}\big)$ being the bias, therefore the XOR correction will always improve bias properties of a TRNG. In our case, this possibility is enabled by the existence of two independent data streams $b_1(n)$ and $b_2(n)$ that can be read out sequentially according to (5).

However, while the biases of individual bits are relatively small ($E[b] \approx 1/2$), they have significant autocorrelation. The question is: can the influence of the autocorrelation be decreased by a simple XOR corrector using independent data streams $b_1(n)$ and $b_2(n)$? This case was

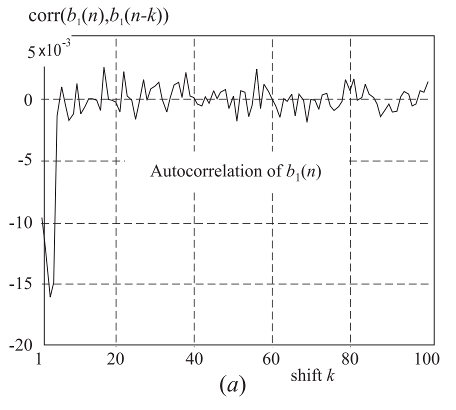


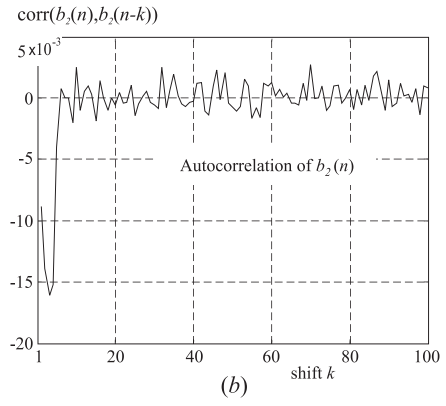
(a)



(b)

**Fig. 6.** a) Schematic diagram of the TRNG hardware with the 8-Pin PSoC device and output waveforms of the 4-SC TRNG implementation. Outputs $J1$, $J2$ are for the testing purposes only. b) Waveforms 1, 2, 3, and 4 represent $x'(n)$ (Block2out), $x(n + 1)$ (Block4out), $b(n)$ (Comparator2out), and SC clock (Clkout), respectively.





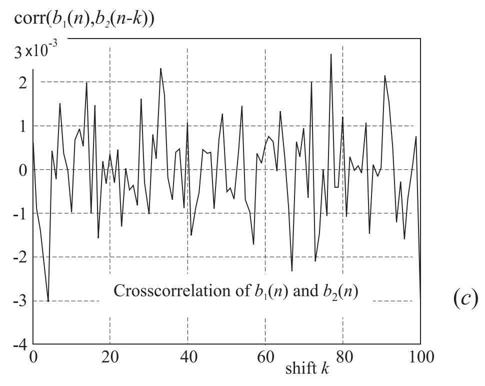


**Fig. 7.** Results of testing of internal 1 000 000 bit data records for 4-SC TRNG @ 5 V: a) autocorrelation of $b_1(n)$, b) autocorrelation of $b_2(n)$ and c) cross-correlation of $b_1(n)$ and $b_2(n)$ for different shift values $k$

analyzed in [20] for correlation of independent autocorrelated pairs (see section 3.4 in [20]). It was shown that

$$\operatorname{corr}\big(b_1(n)\oplus b_2(n),\, b_1(n-k)\oplus b_2(n-k)\big) \approx \rho^2 + 8\rho\big(E[b_1]-\frac{1}{2}\big)\big(E[b_2]-\frac{1}{2}\big) \tag{10}$$

where $\rho$ is the correlation of autocorrelated pairs that we can (conservatively) approximate it by equation

$$\rho \le \max\{\operatorname{corr}\big(b_1(k)\big), \operatorname{corr}\big(b_2(k)\big),\ k=1,2,\dots,100\} \tag{11}$$

From our simulated data we can expect a reduction of the maximal autocorrelation coefficient $\rho$ to the level (supposing that $E[b_1] = E[b_2] \approx 1/2$ and $\rho \approx 0.011$ as given in Fig. 5)

$$\operatorname{corr}\big(b_1(n)\oplus b_2(n), b_1(n-k)\oplus b_2(n-k)\big)\approx 0.0001 \tag{12}$$

Such a level of the autocorrelation should allow to pass standard statistical tests used in cryptography also for relatively strict test conditions as will be shown in Section 5.

## 4 EXPERIMENTAL HARDWARE IMPLEMENTATION

The experimental TRNG bit streams were acquired from a PSoC based TRNG implementation depicted in Fig. 6. In the Fig. 6, particular testing signals acquired from oscilloscope are shown, as well. These signals are connected to the output pins just for testing purposes. During normal TRNG operations their output is disabled. Presented results have been obtained using 8-pin CY8C27143 PSoC device.

## 5 STATISTICAL EVALUATION OF TRNG DATA

If we know the design of the TRNG we can tailor some statistical tests to be appropriate for this design. From previous analysis we know, that there exist certain autocorrelation of internal TRNG signals. The first tests that we propose for our TRNG are standard cross and autocorrelation tests of internal signals. For complex statistical evaluation of proposed TRNG we use NIST statistical test suite [21].

### 5.1 Correlation and autocorrelation tests

#### 5.1.1 Tests of internal uncorrected TRNG data streams

Two interleaved internal data streams, $b_1(n)$ and $b_2(n)$ have been aquired from testing hardware shown in Fig. 6 for 4-SC @ 5V configuration and 60 Kbit/s output rate. Autocorrelation and cross-correlation values for sequences $b_1(n)$ and $b_2(n)$ computed according (6) are

**Table 2.** NIST test results (uniformity of P-values and proportion of passing sequences) for 1- Gbit 4-SC @ 5V record that passed all tests. Testing used 1000 1-Mbit subsequences and significance level $\alpha = 0.01$

| C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | $P-$value | Proportion | Statistical test |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 109 | 88 | 97 | 114 | 98 | 90 | 88 | 95 | 118 | 103 | 0.322135 | 0.9880 | Frequency |
| 111 | 124 | 105 | 84 | 93 | 95 | 100 | 99 | 87 | 102 | 0.199045 | 0.9860 | Block.–Freq. |
| 98 | 115 | 97 | 94 | 107 | 102 | 98 | 99 | 85 | 105 | 0.757790 | 0.9910 | Cusum |
| 118 | 104 | 103 | 109 | 93 | 101 | 75 | 99 | 104 | 94 | 0.238035 | 0.9930 | Runs |
| 79 | 106 | 90 | 106 | 116 | 108 | 82 | 102 | 108 | 103 | 0.147815 | 0.9920 | Long–Run |
| 104 | 102 | 101 | 102 | 94 | 90 | 92 | 113 | 94 | 108 | 0.839507 | 0.9920 | Rank |
| 99 | 98 | 103 | 107 | 108 | 96 | 91 | 109 | 91 | 98 | 0.917870 | 0.9880 | FFT |
| 97 | 120 | 101 | 90 | 114 | 102 | 85 | 100 | 92 | 99 | 0.350485 | 0.9820 | Periodic–Template |
| 88 | 121 | 85 | 102 | 87 | 92 | 116 | 124 | 100 | 85 | 0.012474 | 0.9870 | Universal |
| 112 | 95 | 112 | 98 | 85 | 83 | 122 | 101 | 101 | 91 | 0.123038 | 0.9840 | Apen |
| 107 | 96 | 98 | 80 | 118 | 111 | 90 | 103 | 105 | 92 | 0.267573 | 0.9880 | Serial |
| 102 | 103 | 93 | 96 | 99 | 113 | 98 | 99 | 106 | 91 | 0.930026 | 0.9890 | Linear–Complexity |

shown in Fig. 7. These results follow closely the simulation results given in Section 3. They confirm the fact that the gain errors are really the most important source of non-idealities of the proposed TRNG. An efficiency of the proposed XOR corrector is tested by the autocorrelation test applied on TRNG output data.

#### 5.1.2 Autocorrelation test of output TRNG data streams

In order to check the possible deviation of autocorrelation values of the TRNG output $b'(n) = b_1(n) \oplus b_2(n)$ we have tested validity of $N(0,1)$ assumption for value $T(k)$ given by (8). For an ideal TRNG all points of the normalized test statistic $T(k)$ should fall into interval with a 99% probability [1, page 177]. Applying this function for averages based on $L = 10\,000\,000$ bit records we have found no particular deviation from $N(0,1)$ assumption. This confirms the fact that the autocorrelation was suppressed to the level lower than $\approx \pm 1\sqrt{10\,000\,000} \approx \pm 0.0003$.

### 5.2 NIST statistical tests

It seems that the NIST statistical test suite is the most comprehensive tool publicly available. It must be used very carefully, as it has been recently shown that it contains some fundamental errors in FFT and Lempel-Ziv tests [22]. These changes were adopted also in the latest NIST suite release (v.1.8, March 15, 2005) [23] that was used in our simulation. A good TRNG should pass all kinds of NIST tests.

Our NIST statistical tests were performed on 1-Gbit of continuous TRNG output records and followed testing strategy, general recommendations and result interpretation described in [21]. We have used a set of $m = 1000$ 1-Mbit sequences produced by the 4-SC generator and we have evaluated the set of P-values (some typical values for 4-SC @ 5V TRNG records are shown in Tab. 2) at a significance level $\alpha = 0.01$. The count of acceptable sequences was within the expected confidence intervals [21] for all performed tests and $P$-values were uniformly distributed over (0,1) interval as expected for ideal RNG. There were no deviations detectable with the NIST test package for given setting and quality of 4-SC TRNGs can be considered as a very good source of random data.

## 6 CONCLUSIONS

In this paper we have described and evaluated a chaos-based method of true random numbers generation embedded in a mixed signal reconfigurable hardware. The circuit topology and the proposed XOR corrector were optimized in order to suppress the circuit non-idealities. The randomness of the sequence of numbers has been extensively tested and no differences from the ideal TRNG have been detected for up to 1-Gbit testing records using NIST testing strategy. The proposed TRNG provides good quality random data at up to 60 Kbit/s data rates. It can be easily extended to custom circuits that can provide significantly higher output data rates.

**Acknowledgments**

This work has been done in the frame of the Slovak scientific projects VEGA 1/1057/04 and VEGA 1/2180/05.

## References

[1] MENEZES, J. A.—OORSCHOT, P. C.—VANSTONE, S. A.: Handbook of Applied Cryptography, CRC Press, New York, 1997, available at: http://www.cacr.math.uwaterloo.ca/hac/.

[2] EASTLAKE, D.—CROCKER, S.—SCHILLER, J. : Randomness Recommendations for Security, Request for Comments 1750, December 1994, www.ietf.org/rfc/rfc1750.txt.

[3] JUN, B.—KOCHER, P. : The Intel Random Number Generator, Cryptography Research Inc., white paper prepared for Intel Corp., April 1999, available at: http://www.cryptography.com /resources/whitepapers/IntelRNG.pdf.

[4] BUCCI, M. — GERMANI, L. — LUZZI, R. — TRIFILETTI, A. — VARANOUOVO, M. : A High-Speed Oscillator-Based Truly Random Number Source for Cryptographic Applications on a Smart Card IC, IEEE Transactions on Computers **52** No. 4, April 2003, 403-409.

[5] TSOI, K. H.—LEUNG, K. H.—LEONG, P. H. : Compact FPGA-based True and Pseudo Random Number Generators, Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), California USA, 2003, 51-61.

[6] EPSTEIN, M. — HARS, L. — KRASINSKI, R. — ROSNER, M. — ZHENG, H. : Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts, In C.D. Walter, C. K. Koc, Ch. Paar (Eds.): CHES 2003, LNCS 2779, Springer, Berlin, 2003, 152-165.

[7] FISCHER, V.—DRUTAROVSKY, M. : True Random Number Generator Embedded in Reconfigurable Hardware, In B. S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, Springer, Berlin, 2003, 415-430.

[8] KOHLBRENNER, P.—GAJ, K. : An Embedded True Random Number Generator for FPGAs, In Proceeding of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM Press, 2004, 71-78.

[9] DRUTAROVSKY, M.—FISCHER, V.—SIMKA, M.—CELLE, F. : A Simple PLL-based True Random Number Generator for Embedded Digital Systems, Computing and Informatics **23** No. 5-6 (2004), 501-516.

[10] Open Random Bit Generator, available at: http://mywebpages. comcast.net/orb/index.html.

[11] KENNEDY, M. P.—ROVATTI, R.—SETTI, G. (Eds.) : Chaotic Electronics in Telecommunications", CRC International Press, Boca Raton, 2000.

[12] KENNEDY, M. P.—ROVATTI, R.—SETTI, G. : Embeddable ADC-Based True Random Number Generator for Cryptographic Applications Exploiting Nonlinear Signal Processing and Chaos, IEEE Trans. On Signal Processing **53** No. 2, Feb.2005, 793-805.

[13] BERNSTEIN, G. M.—LIEBERMAN, M. A. : Secure Random Number Generation Using Chaotic Circuits, IEEE Transactions on Circuits and Systems **37** No. 9, September 1990, 1157-1164.

[14] STOJANOVSKI, T. — PIHL,J. — KONCAREV, L. : Chaos-Based Random Number Generators-Part II: Practical Realization, IEEE Transactions on Circuit and Systems-I: Fundamental Theory and Applications **48** No. 3, March 2001, 382-385.

[15] RESTITUTO, M. D.—VÁZQUES, A. R. : Integrated Chaos Generators, Proceedings of the IEEE **90** No. 5, May 2002, 747-767.

[16] CALLEGARI, S.—ROVATTI, R.—SETTI, G. : First Direct Implementation of a True Random Source on Programmable Hardware, Int. J. Circ. Theor. Appl. 2005; 33:1-16.

[17] DRUTAROVSKY, M.—BACA, M.—GALAJDA, P. : Chaos Based True Random Number Generator, design entry to the International PSoC Design Contest, Cypress MicroSystems Inc., February 2004.

[18] PSoC$^{TM}$ Mixed Signal Array Final Data Sheet, Document No. 38-12012 Rev.C, August 28, 2003, 1-332, available at: http://www.cypressmicro.com.

[19] ESS, D. V. : Understanding Switched Capacitor Analog Blocks, Cypress Microsystems Application Note AN2041, 8/22/2002, 1-16, available at: http://www.cypressmicro.com.

[20] DAVIES, R. B. : Exclusive OR (XOR) and hardware random number generators, February 28, 2002, 1-11, available at: http://www.robertnz.net/pdf/xor2.pdf.

[21] RUKHIN, A.— SOTO, J.— NECHVATAL, J.— SMID, M.— BARKER, E.— LEIGH, S.— LEVENSON, M.— VANGEL, M.— BANKS, D.— HECKERT, A.— DRAY, J.— VO, S. : A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22, May 15, 2001, 1-153, http://csrc.nist.gov/rng/.

[22] KIM, S. J.—UMENO, K.—HASEGAWA, A. : Corrections of the NIST Statistical Test Suite for Randomness, Cryptology ePrint Archive, Report 2004/018, January 26, 2004, http://eprint.iacr.org/.

[23] NIST Statistical Test Suite, http:// csrc.nist.gov/rng/rng2.html.

Received 3 November 2005

**Miloš Drutarovský** was born in Prešov, Slovakia, in 1965. He received the Ing (MSc) degree in radioelectronics and CSc (PhD) degree in electronics from the Faculty of Electrical Engineering, Technical University of Košice, Slovakia, in 1988 and 1995, respectively. He defended his habilitation work in digital signal processing in 2000. He is currently working as an Associated Professor at the Department of Electronics and Multimedia Communications, Technical University of Košice. His current research interests include applied cryptography, digital signal processing, DSP and FPGA devices and algorithms for embedded cryptographic architectures.

**Pavol Galajda** was born in Košice, Slovakia, in 1963. He received the Ing (MSc) degree in radioelectronics and CSc (PhD) degree in electronics from the Faculty of Electrical Engineering, Technical University of Košice, Slovakia, in 1986 and 1995, respectively. He is currently working as an Associated Professor at the Department of Electronics and Multimedia Communications, Technical University of Košice. His research interest is in nonlinear circuits theory, spread-spectrum communication via chaotic synchronizations and modulations, software defined radio and electronics systems based on FPGA devices.